Quantum Error Correction Thresholds for the Universal Fibonacci Turaev-Viro Code

Alexis Schotte,[1,2,*] Guanyu Zhu,[2,†] Lander Burgelman,[1] and Frank Verstraete[1]

[1]*Department of Physics and Astronomy, Ghent University, Krijgslaan 281, 9000 Gent, Belgium*
[2]*IBM Quantum, IBM T. J. Watson Research Center, Yorktown Heights, New York 10598, USA, and IBM Almaden Research Center, San Jose, California 95120, USA*

(Received 27 April 2021; revised 8 December 2021; accepted 15 February 2022; published 15 April 2022)

We consider a two-dimensional quantum memory of qubits on a torus which encode the extended Fibonacci string-net code and devise strategies for error correction when those qubits are subjected to depolarizing noise. Building on the concept of tube algebras, we construct a set of measurements and of quantum gates which map arbitrary qubit errors to the string-net subspace and allow for the characterization of the resulting error syndrome in terms of doubled Fibonacci anyons. Tensor network techniques then allow us to quantitatively study the action of Pauli noise on the string-net subspace. We perform Monte Carlo simulations of error correction in this Fibonacci code and compare the performance of several decoders. For the case of a fixed-rate sampling depolarizing noise model, we find an error correction threshold of 4.7% using a clustering decoder.

DOI: 10.1103/PhysRevX.12.021012 Subject Areas: Quantum Physics, Quantum Information Strongly Correlated Materials

I. INTRODUCTION

The biggest theoretical challenge in achieving scalable quantum computation is the construction of more efficient schemes for quantum error correction and fault tolerance [1,2]. Topological quantum error correcting codes, with the most famous representative being the surface code [3–7], are among the most promising candidates for near-term implementation due to their geometric locality which makes them well suited for practical implementation using state-of-the-art hardware technology such as superconducting or semiconducting qubits, ion traps, silicon photonics, nitrogen-vacancy centers, and cold atoms, just to name a few. Surface codes allow for high quantum error correction thresholds but have the drawback that one needs a very large overhead of magic states to make the scheme universal for quantum computation [2,7,8].

In order to overcome this limitation, one has to consider topological codes which allow for non-Clifford logical gates. One approach in this direction is to consider stabilizer codes in three and higher dimensions, which allow for non-Clifford transversal gates [9–15]. This approach also includes schemes based on 3D color codes or 3D surface codes which simulate the third spatial dimension using time within a 2D measurement-based quantum computing architecture [14,15]. A perpendicular direction is to look beyond the stabilizer formalism and consider non-Abelian codes in 2D which are universal for quantum computation without the need for magic-state distillation [16]. In this paper, we follow this second path, as we believe that this is a more natural setting for hardware platforms currently being pursued.

Most conventional topological error correcting codes fall within the framework of the stabilizer formalism [17] and admit quasiparticle excitations which can be characterized as Abelian anyons. However, braiding of these excitations does not allow for a universal gate set. Similarly, double semion topological codes [18,19], while going beyond the stabilizer code class, exhibit Abelian topological order and are not universal. In order to achieve universal topological quantum computation, a necessary condition is to use systems with a more intricate topological order allowing for non-Abelian anyonic excitations, which have the property that the fusion of two anyons can yield several outcomes [16]. In particular, braiding of Fibonacci anyons can be used to realize a fault-tolerant universal gate set [20]. Several lattice models supporting this non-Abelian topological order have been proposed, such as Kitaev's quantum double models [3] or the string-net models of Levin and Wen [21]. While their ground space is still defined as the simultaneous eigenspace of a set of mutually commuting local check operators, their description falls beyond the stabilizer formalism.

In recent years, there has been significant progress in the study of quantum error correction and decoding for

[*]alexis.schotte@ugent.be
[†]guanyu.zhu@ibm.com

non-Abelian topological codes with a nonstabilizer structure, including numerical estimates of their error thresholds [22–26]. These works, however, assume the existence of a protected anyonic fusion space and consider only phenomenological noise models on this space while not specifying a concrete underlying microscopic quantum mechanical spin model suitable for the description of realistic quantum computer implementations.

In this work, we remedy this shortcoming by considering a non-Abelian error correcting code consisting of generic qubits subjected to depolarizing noise. We focus on one of the simplest of those codes, the Turaev-Viro code constructed from the Fibonacci string-net model [21]. Building on the pioneering work of König, Kuperberg, and Reichardt [27] and of Bonesteel and DiVincenzo [28], we construct a set of measurements and quantum gates which map arbitrary qubit errors to the Turaev-Viro subspace and make crucial use of the framework of tensor networks for simulating the error correction process, giving rise to surprisingly high quantum error correction thresholds. Using a clustering decoder and a fixed-rate sampling noise model, we obtain a 4.7% threshold for the code subjected to depolarizing noise and a 7.3% threshold for pure dephasing noise. These numbers are comparable to the code-capacity error threshold of the surface code, which is around 10% for independent identically distributed (IID) bit-flip or phase-flip noise [4,29,30].

Before giving a summary of the paper, let us provide a brief review of the history and current status of the Turaev-Viro codes. Following the pioneering work by Jones in 1984 discovering the Jones polynomials [31], Reshetikhin and Turaev generalized the Jones polynomials of links by introducing the concept of ribbon graphs and the corresponding invariants derived from quantum groups in their influential work in 1990 [32]. Around the same time, Witten [33] and Atiyah [34] introduced the formalism of topological quantum field theory (TQFT). Turaev and Viro introduced a path integral in terms of a discrete state sum describing a wide class of $2+1$D topological quantum field theories in 1992, leading to new quantum invariants of 3-manifolds [35]. Around 1997, Kitaev had the crucial insight that the problem of constructing quantum error correcting codes is effectively equivalent to the one of constructing quantum spin systems providing a Hamiltonian realization of such topological field theories. He introduced a class of quantum double models [3], of which the simplest Abelian version $\mathcal{D}(\mathbb{Z}_2)$ is the well-known toric code. The non-Abelian codes in the quantum double family can be used to implement universal fault-tolerant logical gate sets without magic-state distillation. In 2005, Levin and Wen generalized Kitaev's quantum doubles by introducing string-net models [21], which provide a local Hamiltonian realization of all the unitary Turaev-Viro TQFTs. Their original motivation of introducing the string-net condensation picture was to provide a microscopic mechanism for spin liquid in the context of condensed matter physics. The string nets can be viewed as a planar special case of ribbon graphs introduced by Reshetikhin and Turaev. The string-net models capture all nonchiral topological orders (Abelian and non-Abelian) in 2D, including topological orders of the toric code and Kitaev's quantum double model as specific examples. In 2010, König, Kuperberg, and Reichardt [27] studied a class of Levin-Wen models with a modular input category from the point of view of error correction and called them *Turaev-Viro codes*. They developed the tube algebra for this modular case and defined a complete basis of the anyonic excitations. In 2012, Bonesteel and DiVincenzo proposed the quantum circuits to measure the vertex and plaquette projectors in the Fibonacci Levin-Wen model [28] and, hence, made the first step toward practical implementation of Turaev-Viro codes with ordinary qubits by devising an error detection scheme.

Several technical difficulties had to be solved, however, to turn their error detecting scheme into an error correcting one. First, the original string-net model proposed by Levin and Wen [36] does not admit an easy microscopic description of all types of anyonic excitations in the corresponding topological phases, but only the fluxons (plaquette excitations). As we see, a single vertex error in this model can bring the system out of the string-net subspace such that the created excitations are no longer anyons as in the case of phenomenological anyon models [22,25,26], making error correction and decoding quite challenging. For that purpose, an extended string-net model was defined on a tailed lattice [37,38] (see also Ref. [39], where tail qubits were introduced for the purpose of incorporating charged and dyonic excitations in topological phases). In the same work [37,38], a scheme to trap vertex errors and a tadpole swapping scheme to trap plaquette errors were introduced.

In this work, we adopt this tailed-lattice construction and use a similar strategy to trap local vertex errors. We introduce a measurement scheme in terms of tube algebras or *tube operators* [40,41] whose outcomes contain more syndrome information than the ones reported in Refs. [37,38]. This tube algebra enables the definition of an anyonic fusion basis, which can be used to effectively describe the system evolution in the simpler language of an anyon model. We then use the tensor network description of those tube algebras to convert microscopic noise processes such as Pauli errors into anyon-creation processes. The last step needed to calculate error correcting thresholds then consists of simulating the error correcting process.

It is tempting to think that an efficient classical simulation of the error correction process for the Fibonacci Turaev-Viro code is impossible, since braiding Fibonacci anyons is universal for quantum computation. This issue is addressed in Ref. [26] for a phenomenological Fibonacci anyon model (in which the physical degrees of freedom are anyons as opposed to qubits), and it is demonstrated that it is possible to simulate the error correction threshold with a polynomial complexity. This is because, unlike the quantum computation process where the computational anyons are braided along topologically nontrivial worldlines, the worldlines of noise-created anyons in the error correction process are topologically trivial most of the time. As long as the system is below the percolation threshold corresponding to anyon generation, this classical simulation can be performed in an efficient way. Our work extends the applicability of that result to the case where the physical degrees of freedom are qubits subject to arbitrary noise processes and, hence, allows us to determine error thresholds through classical simulations.

A. Summary

Before proceeding with the main exposition, we first dedicate some space to convey the central ideas presented in this paper, free of superfluous technical details. There are, in essence, two main achievements detailed in this work. The first is the construction of a non-Abelian topological quantum error correcting code consisting of regular qubits and the design of a complete protocol for error detection and correction in this code. The second is the classical simulation of this error correction procedure using tensor network techniques resulting in an estimate for the associated error correction threshold for a microscopic noise model of Pauli errors.

We begin by introducing the central object in our discussion, the extended string-net code. Our starting point is the Fibonacci Levin-Wen string-net model [21] of qubits arranged on a hexagonal lattice, defined from the algebraic data of the Fibonacci unitary fusion category. More specifically, we build on the work of König, Kuperberg, and Reichardt [27], who illustrate that this model may be used as a scheme for universal topological quantum computation, giving rise to the concept of Turaev-Viro codes. By adopting a continuum formulation of the model in terms of trivalent ribbon graphs whose properties are defined by the input category, it is shown that the excitations in a Turaev-Viro code can be identified with the central idempotents of the tube algebra [40,41], which correspond to topological sectors labeled by the doubled category. This tube algebra and the associated characterization of excitations as doubled anyons form the guiding principle throughout the remainder of our discussion. In this work, we adopt an extension of the string-net model that serves as the basis for an error correcting code. This extension has a twofold motivation. On the one hand, we need a way of correcting violations of the ribbon graph branching rules that can be induced by generic errors at the level of the lattice qubits. Moreover, we also require a concise way of characterizing the excitation spectrum in terms of anyonic charges, by defining the action of the tube algebra idempotents in the bulk of the lattice model. Both of these requirements can be met by adding an additional "tail edge" to each plaquette, inspired by the constructions introduced in Refs. [37–39] for similar reasons. These considerations then lead to a model of qubits arranged on the edges of a tailed hexagonal lattice on the torus whose fourfold degenerate ground space serves as a topological quantum memory and effectively encodes two logical qubits and whose excited states can be interpreted as fusion states of doubled Fibonacci anyons. By generalizing the torus setup (genus $= 1$) to a higher-genus surface, one can scale up the number of logical qubits, which grows approximately linearly with the genus. One can, hence, perform universal quantum computation via topological operations corresponding to the elements of the mapping class group of the high-genus surface, which can be generated by Dehn twists [20,27,42]. Alternatively, one can encode the logical information in the fusion basis of computational anyons and perform universal quantum computation via braiding these computational anyons [20,27,43]. Both Dehn twists and braiding can be implemented by local quantum circuit via F moves [27] or via code deformation.

With this code definition, we proceed with defining the protocols for error detection and correction. Generic errors on the lattice qubits can cause violations of the string-net (ribbon graph) branching rules. Such a violation can be interpreted as a string ending in a vertex of the lattice, resulting in a qubit state that lies outside of the string-net subspace, which can, therefore, not be captured as an anyonic fusion state. Adapting the circuits for detecting these vertex violations first introduced in Ref. [28], we define local unitary circuits that can correct an arbitrary combination of vertex errors, by pulling the corresponding string end onto the tail edge of the associated plaquette. After returning the system to the string-net subspace in this way, we then define circuits for syndrome extraction, again guided by the concept of the tube algebra. Measuring the idempotents of the tube algebra in each plaquette reveals the location and charge of all anyonic excitations in the system, yielding the error syndrome. Utilizing the ribbon graph formalism, we naturally arrive at a local unitary circuit which can perform these charge measurements. Equipped with this protocol for syndrome extraction, we are left with the task of recovery, which consists of moving excitations on the lattice and fusing them to back to the anyonic vacuum, thereby returning the system to the code space. Similar to the Abelian case, a logical error occurs when an anyon is wound along a nontrivial cycle of the torus in this process. Building on and extending previous works [27,37,38,42–44], we design protocols for the necessary recovery operations at the level of the qubits. For the decoding procedure itself, which entails deciding which recovery operations should be carried out given an error syndrome, we rely on recent advances in error correction for non-Abelian anyon models [22,23,26] and tailor the decoders introduced there to our specific code, as well as further design new decoders for our purpose. The main difficulty that arises for the non-Abelian case is the fact that error correction has to proceed in an iterative fashion because of indeterminacy of fusion outcomes for non-Abelian anyons. Specifically, we adapt a clustering decoder to our setting and further design a fusion-aware iterative matching decoder.

These considerations conclude our discussion of the code definition and associated error correction protocol, giving a complete scheme for error correction in an extended Fibonacci Turaev-Viro code. We then move on to our second main result: the classical simulation of the error correction procedure and the estimate of the error correction threshold.

The main problem to be tackled here is the question of how to determine what distribution of anyonic excitations is generated by Pauli errors acting at the level of the lattice qubits. The complex description of the excited anyonic fusion states in terms of qubit states makes this a highly nontrivial task, however. Again relying on the concept of the tube algebra, we extend the tensor network anyon ansatz known from the matrix operator description of topological order [41] to construct a projected entangled pair state (PEPS) representation for the anyonic fusion states that appear as excited states in our model. Armed with these PEPSs, we utilize tensor network methods to analyze the effect of Pauli noise on the code, which effectively allows us to translate a physical error rate at the level of the qubits to an anyon generation rate. We can, hence, simplify the classical simulation of the decoding problem to the simulation of noise-driven dynamics of anyonic fusion states, which is infinitely more feasible than directly simulating the full microscopic model itself in the qubit basis. Having overcome this main difficulty, we adapt recently developed techniques for simulating non-Abelian error correction [26] to the hexagonal geometry and doubled Fibonacci excitations relevant to our model. We proceed to illustrate a scheme for the classical simulation of error correction in our code and conclude with an estimate of the error correction threshold using different decoders.

B. Overview

The remainder of this paper is divided into six sections. The first three deal with defining the error correcting code, the necessary circuits for syndrome measurement, and recovery operations and decoding algorithms to correct detected errors. The second half of the paper deals with the classical simulation of the code in order to obtain an estimate for the error correction threshold. The structure is as follows.

Section II contains a complete description of the extended string-net model, which is the microscopic model for the quantum error correcting code studied in this work.

In Sec. III, the error correction scheme is discussed in detail, including all quantum circuits for the required measurements and recovery operations.

Section IV describes the clustering decoder, which we find to be the best performing decoder in this work. Two more decoding algorithms, both based on minimum weight perfect matching, are studied. Their descriptions are given in Supplemental Sec. V [45].

Section V covers all technical details of the classical simulation of the extended Levin-Wen code to obtain an error threshold. This includes general remarks on simulating (universal) non-Abelian codes, the precise definition of the noise model, a discussion on the qualitative description of said noise model in terms of anyon creation and hopping processes through the use of tensor network methods, and an extensive overview of the Monte Carlo simulation.

The numerical results of the classical simulation of the code with the decoders described in Sec. IV and Supplemental Sec. V [45] are presented and discussed in Sec. VI.

Finally, in Sec. VII, we summarize our results and briefly compare them to related works. We then conclude this paper with an outlook on future research directions on non-Abelian codes such as codes based on quantum double models and the experimental realization of the Fibonacci Turaev-Viro code.

Throughout this work, we regularly refer to sections, equations, and figures appearing in Supplemental Material [45]. References to equations and figures appearing in the Supplemental Material are denoted with the letter S [e.g., Eq. (SI.1)].

II. THE EXTENDED STRING-NET CODE

In this section, we introduce the error correcting code which is studied in this paper. We start by giving the technical definition of the code as the ground state of a slightly modified Levin-Wen Hamiltonian in Sec. II A. Then, we introduce a continuum picture of the model, known as the *fattened lattice picture*, in Sec. II B. This alternative viewpoint proves useful for understanding certain operators and to describe excited states of the underlying Hamiltonian. We proceed by briefly discussing code deformation in Sec. II C and then discuss the relation between excitations and anyons in Sec. II D. Specifically, we introduce a family of projectors which define a projective measurement of the anyonic charge of a single plaquette. Finally, in Sec. II E, we introduce a basis for the subspace where all vertex conditions are satisfied in terms of fusion states of anyonic excitations residing in individual plaquettes. The ability to describe certain excited states as fusion states of anyons is essential throughout the rest of this work, in particular, for the classical simulation of the code in Sec. V.

The extended string-net model, which we introduce below, is intimately related to topological quantum field theory. To improve readability of the main text, we omit many details on this relation. As we believe some readers might find it enlightening, we include a more extensive discussion on this topic in Supplemental Sec. I [45].

A. Definition of the code

The extended string-net code is a microscopic realization of a Turaev-Viro code [27]. Its code space is defined as the ground space of the *extended* Levin-Wen string-net model [21,39]. This is a microscopic model of qudits situated on the edges of a tailed trivalent lattice Λ, obtained by modifying the Levin-Wen Hamiltonian [21] to accommodate one additional "tail edge" in every plaquette as shown in Fig. 1. The code space is, hence, denoted by $\mathcal{H}_\Lambda$. Such a modification was first proposed in Ref. [39], with the original goal of incorporating charged and dyonic excitations in (doubled) topological order. Below, we give a brief summary of its definition and of its most important properties.

The model is defined starting from the algebraic data of a unitary fusion category $\mathcal{C}$. For simplicity, we limit ourselves to multiplicity-free self-dual categories. The generalization to generic unitary fusion categories is straightforward but quite tedious. Since we work with the Fibonacci category later (which is self-dual), we do not need the general case. The algebraic data of such an object consist of the following.

(1) *String types.*—A set of all possible string types $\{\mathbf{1}, i_2, \ldots, i_N\}$. The label $\mathbf{1}$ is referred to as the *vacuum label* and represents the absence of a string on a particular edge.

(2) *Branching rules.*—The set of all triplets $\{i, j, k\}$ that are allowed to meet at a vertex (also known as *fusion rules*). We introduce the symbol δ_{ijk} defined by the branching rules as

$$\delta_{ijk} = \begin{cases} 1, & \text{if the triplet}\{i,j,k\}\text{is allowed,} \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

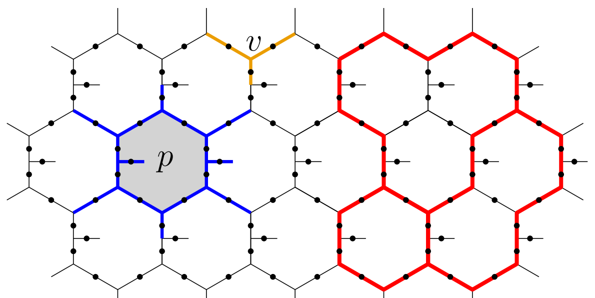


FIG. 1. Qudits arranged on the tailed honeycomb lattice. The support of B_p and Q_v are indicated in blue and orange, respectively. The red edges represent a valid string-net configuration of qubits in the $|1\rangle$ state when using the Fibonacci input category.

For every label i, there is unique dual label i^* satisfying $\delta_{ii^*\mathbf{1}} = 1$, and $(i^*)^* = i$. Note that we are considering self-dual categories, [46] which satisfy $i^* = i$ for every string type i.

(3) *Numerical data.*—For each string type i, a real constant d_i, called the *quantum dimension*, satisfying

$$d_i d_j = \sum_k \delta_{ijk} d_k, \quad d_1 = 1, \quad \text{and} \quad d_{i^*} = d_i, \tag{2}$$

and a six-index symbol F^{ijm}_{kln}, which is a complex constant dependent on six string types i, j, k, l, m, and n. These quantities are required to satisfy the following consistency conditions:

$$\text{physicality:}\ F^{ijm}_{kln}\delta_{ijm}\delta_{klm^*} = F^{ijm}_{kln}\delta_{iln}\delta_{jkn^*}, \tag{3}$$

$$\text{pentagon equation:} \sum_{n=1}^{N} F^{mlq}_{kpn} F^{jip^*}_{mns} F^{jsn}_{lkr} = F^{jip^*}_{q^*kr} F^{r^*iq^*}_{mls}, \tag{4}$$

$$\text{unitarity:}\ (F^{ijm}_{kln})^* = F^{lin}_{jkm^*}, \tag{5}$$

$$\begin{aligned}\text{tetrahedral symmetry:}\ F^{ijm}_{kln} &= F^{jim}_{lkn^*} = F^{lkm^*}_{jin} \\ &= F^{imj}_{k^*nl}\frac{v_m v_n}{v_j v_l},\end{aligned} \tag{6}$$

$$\text{normalization:}\ F^{ii^*1}_{j^*jk} = \frac{v_k}{v_i v_j}\delta_{ijk}, \tag{7}$$

where $v_i = \sqrt{d_i}$.

Note that, for self-dual categories, the F symbols are real valued.

We associate the string types to the elements of an orthonormal basis of the qudit Hilbert space $\mathbb{C}^N$ at each edge. The extended Levin-Wen Hamiltonian is then defined as

$$H_\Lambda = -\sum_v Q_v - \sum_p B_p, \tag{8}$$

where v and p label the vertices and plaquettes of the trivalent lattice Λ, respectively, and $\{Q_v, B_p\}$ are a set of commuting projectors whose support is shown in Fig. 1.

For every vertex v, the three-body projector Q_v imposes the branching rules

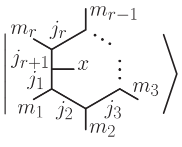
$$Q_v \Big| \overset{j}{\underset{i\quad k}{\curlywedge}} \Big\rangle = \delta_{ijk} \Big| \overset{j}{\underset{i\quad k}{\curlywedge}} \Big\rangle . \tag{9}$$

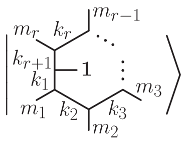
The subspace $\mathcal{H}_{\text{sn}}$ of states that satisfy all vertex projectors is known as the *string-net subspace*. States in $\mathcal{H}_{\text{sn}}$ can be understood as superpositions of string nets, which are defined as string configurations that obey the branching rules.

We work with the tailed honeycomb lattice shown in Fig. 1, for which the plaquette projector B_p is a 16-body operator. On a generic trivalent tailed lattice, it is defined as

$$B_p = \frac{1}{\mathcal{D}^2}\sum_s d_s O_p^s, \tag{10}$$

where $\mathcal{D} = \sqrt{\sum_i d_i^2}$, and

$$O_p^s \Big|$$


$$\Big\rangle = \delta_{x,\mathbf{1}} \sum_{k_1,\ldots,k_{r+1}} \delta_{k_1,k_{r+1}} \cdot \left(\prod_{\nu=1}^{r} F^{m_\nu j_\nu j_{\nu+1}}_{s k_{\nu+1} k_\nu}\right) \Big| \tag{11}$$


$$\Big\rangle .$$

The error correction scheme and numerical simulations described in Secs. III and V are designed specifically for the Fibonacci input category ($\mathcal{C} = \mathrm{FIB}$), which contains only two string types, $\mathbf{1}$ and τ. Hence, the model we consider in the remainder of this paper is a system of qubits. We choose to relate the string types to the standard computational basis states: $\mathbf{1} \to |0\rangle$ and $\tau \to |1\rangle$. The nontrivial fusion rule is $\tau \times \tau = \mathbf{1} + \tau$, which leads to the following branching rules:

$$\delta_{ijk} = \begin{cases} 1, & \text{if } (ijk) \in \{\mathbf{111}, \tau\tau\mathbf{1}, \mathbf{1}\tau\tau, \tau\mathbf{1}\tau, \tau\tau\tau\}, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

The quantum dimensions are

$$d_{\mathbf{1}} = 1, \qquad d_\tau = \phi, \tag{13}$$

where $\phi = \frac{1+\sqrt{5}}{2}$ is the golden ratio. The only nontrivial F matrix is

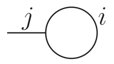
$$[F^{\tau\tau\tau}_{\tau\tau}] = \begin{pmatrix} \phi^{-1} & \phi^{-1/2} \\ \phi^{-1/2} & -\phi^{-1} \end{pmatrix}. \tag{14}$$

For all other combinations of indices, F^{ijm}_{kln} is either 1 or 0, depending on whether or not the corresponding indices in Eq. (3) satisfy the branching rules.

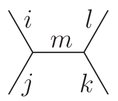
The ground space of Hamiltonian (8) has a degeneracy that depends on the genus of the surface on which the model is defined. On a torus, and with the Fibonacci input category, the code space is four-dimensional, which enables one to encode the state of two logical qubits.

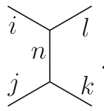
B. The fattened lattice picture

It is convenient to think about the string-net Hilbert space as the lattice realization of the ribbon graph Hilbert space on a punctured surface (see Supplemental Sec. I [45]). For our purpose, it is sufficient to state that this is the space of formal linear combinations of labeled trivalent graphs which satisfy the branching rules in Eq. (1), modulo continuous deformations and the following relations:



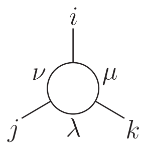
$$= \delta_{j1} d_i \,, \tag{15}$$

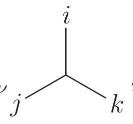


$$= \sum_n F^{ijm}_{kln} \tag{16}$$


The second relation is known as an F move or 2-2 Pachner move.

These *ribbon graphs* are defined on a compact, orientable, surface Δ containing one puncture for every plaquette in the lattice. Each boundary component has a unique marked boundary point, and ribbons are allowed to end only on these marked boundary points. We can relate ribbon graphs on the surface Δ to string nets using the "fattened lattice" picture, which represents an embedding of the lattice Λ in the surface Δ as shown in Fig. 2(a). Whenever ribbons have a more complicated shape that cannot be smoothly deformed to the shape of the embedded lattice, one can first deform them using 2-2 Pachner moves and 1-3 Pachner moves. The latter are defined as



$$= v_\lambda v_\mu v_\nu G^{ijk}_{\lambda\mu\nu} \tag{17}$$


where

$$G^{ijk}_{\lambda\mu\nu} = \frac{1}{v_i v_\lambda} F^{\nu j\lambda}_{k\mu i} = \frac{1}{v_\nu v_k} F^{ijk}_{\lambda\mu\nu}. \tag{18}$$

The action of O_p^s, defined in Eq. (11), can be represented in the fattened lattice picture as the inclusion of a loop with label s around the puncture in plaquette p. Hence, the

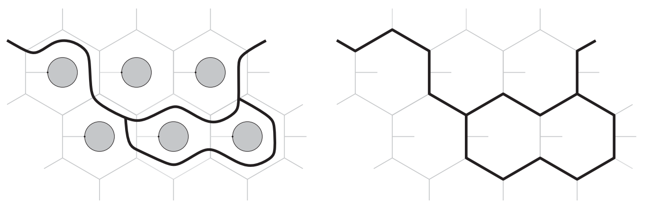


FIG. 2. (a) A ribbon graph on the fattened lattice. (b) The corresponding string-net configuration on the lattice. The gray edges correspond to qudits in the $|0\rangle$ state, while the state of the black edges is given by the string type of the corresponding piece of the ribbon graph in (a).

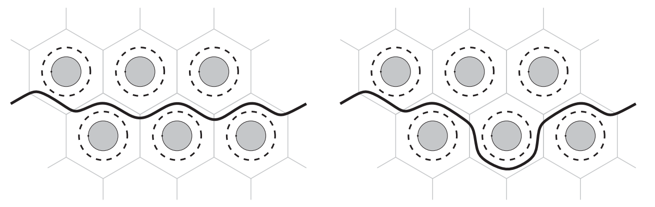
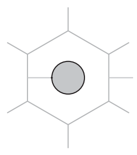
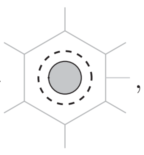


FIG. 3. Two ribbon graphs on the fattened lattice representing the same string-net ground state.

action of the plaquette projector B_p can be represented on the fattened lattice as follows:

B_p :  $\mapsto \frac{1}{\mathcal{D}}$  , (19)

where

$$\vdots \;= \frac{1}{\mathcal{D}} \sum_i d_i \Big|_i . \tag{20}$$

The dashed loop is referred to as a *vacuum loop*. Note that we omit the tail edge on the right-hand side in Eq. (19). After resolving the loop into the lattice using a sequence of F moves, a trivial tail edge should be included. Keep in mind that $B_p = 0$ whenever there is a (nontrivial) ribbon ending in the puncture p, which corresponds to a nontrivial tail edge.

Ground states (up to a normalization factor) then correspond to ribbon graph configurations without any ribbons ending in punctures and where a vacuum loop is added around every puncture as shown in Fig. 3. A short calculation shows that ribbons can be "pulled across" vacuum loops without changing the corresponding string-net state, meaning that one can obtain the same ground state by applying the $\prod_p B_p$ to different string-net states.

C. Code deformation using Pachner moves

The Pachner moves introduced above can be used to relate string-net states defined on different lattice geometries. In particular, when transforming the lattice Λ to Λ', Eqs. (16) and (17) give the appropriate map between the corresponding code spaces $\mathcal{H}_\Lambda$ and $\mathcal{H}_{\Lambda'}$, defined as the ground spaces of Hamiltonians H_Λ and $H_{\Lambda'}$, respectively. For instance, applying the unitary F move (see Fig. 11) to the qudits on certain edges of a ground state transforms this state to a ground state of the string-net Hamiltonian defined on a new lattice obtained by recoupling these edges in the original lattice. The recoupling of lattice edges by a 2-2 Pachner move is shown in Fig. 4. The lattice deformation corresponding to a 1-3 Pachner move is shown in Fig. 5, where one adds a triangular loop on the

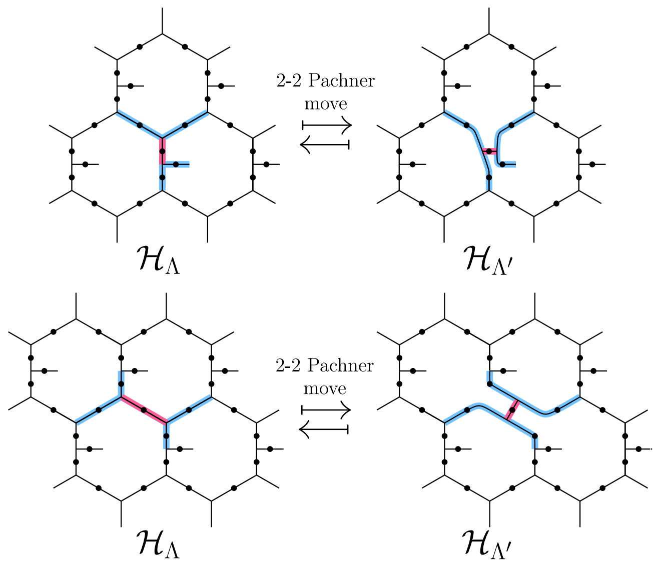


FIG. 4. Lattice deformations by 2-2 Pachner moves on different edges. The affected edge is highlighted in pink; the other edges contained in the F matrix are highlighted in blue.

original vertex. The 1-3 Pachner move can be thought as a fine- and coarse-graining process. In Fig. 5, one entangles three ancilla qudits (white dots) from left to right and adds them into the code space, which effectively fine-grains the lattice. The inverse process from right to left disentangles the qudits in the center out of the code space, which effectively coarse-grains the lattice. Both 2-2 and 1-3 Pachner moves can be implemented via unitary circuits as shown in Sec. III B.

D. Anyonic excitations

Localized excitations in the string-net model exhibit anyonic statistics, described by the quantum double $\mathcal{DC}$ of the input category $\mathcal{C}$ [21]. It is important to note that the categorical double of a unitary fusion category is always braided. Hence, the input category $\mathcal{C}$ does not need to include any braiding structure for the excitations to have well-defined anyonic statistics. When the input category $\mathcal{C}$ is modular (implying it is braided), such as for $\mathcal{C} = \text{FIB}$, the doubled category has a special structure $\mathcal{DC} \cong \mathcal{C} \otimes \bar{\mathcal{C}}$, and its string

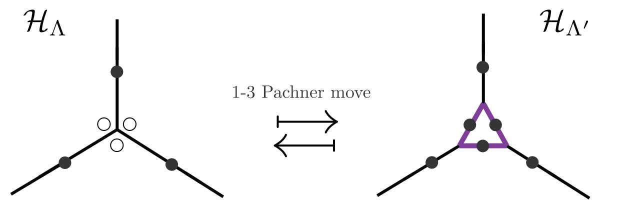
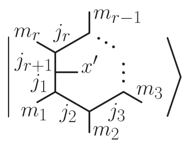
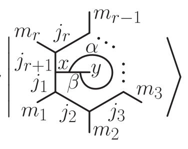


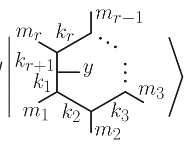
FIG. 5. Lattice deformation by 1-3 Pachner move. The white dots on the left represent ancilla qudits. The fine-graining process (left to right) entangles the three ancilla qudits into the code space. The coarse-graining process (right to left) disentangles the three central qudits out of the code space.

types can be labeled by pairs $a_+\overline{a_-}$ with $a_+, a_- \in \mathcal{C}$ (see Supplemental Sec. I [45] for more details). For notational simplicity, we drop the bar notation for labels in $\bar{\mathcal{C}}$, and we indicate the string types of the doubled category with bold labels: $\boldsymbol{a} = a_+a_- \in \mathcal{DC}$.

The motivation for introducing the tail qudit in every plaquette is that they allow us to define the action of an operator algebra known as Ocneanu's tube algebra [40] in TQFT (see Supplemental Sec. I E [45]), at the level of the lattice qudits. The central idempotents of the tube algebra form projectors onto the different superselection sectors of the theory. By defining their action on an individual plaquette, one obtains a set of projectors corresponding to the different possible values of the doubled anyonic charge contained in that plaquette. The generators of this algebra act on an individual plaquette as

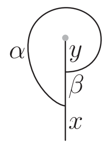
$O_{xy\alpha\beta}$


$= \delta_{x,x'}$


$$= \delta_{x,x'}\frac{v_\alpha v_\beta}{v_y}\sum_{k_1,\dots,k_{r+1}} F^{j_1 j_{r+1} x}_{\alpha\beta k_{r+1}}\left(\prod_{\nu=1}^{r} F^{m_\nu j_\nu j_{\nu+1}}_{\alpha k_{\nu+1} k_\nu}\right) F^{k_{r+1} k_1 y}_{\alpha j_1 \beta}$$


. (21)

This corresponds to gluing the "tube"



(22)

onto the tail edge and resolving it into the lattice using a sequence of 2-2 Pachner moves followed by a 1-3 Pachner move, as shown in Fig. 6.

For the Fibonacci input category, the doubled category (DFIB) contains the string types $\{\mathbf{11}, \mathbf{1}\tau, \tau\mathbf{1}, \tau\tau\}$, which label the different corresponding projectors:

$$\mathcal{P}^{\mathbf{11}} = \frac{1}{\mathcal{D}^2}(O_{\mathbf{1111}} + \phi O_{\mathbf{11}\tau\tau}), \tag{23}$$

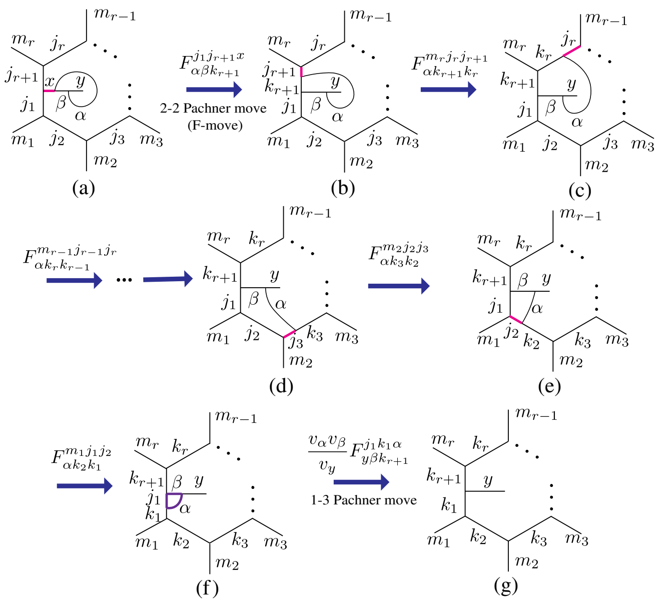

$F^{j_1 j_{r+1} x}_{\alpha\beta k_{r+1}}$
2-2 Pachner move (F-move)
$F^{m_r j_r j_{r+1}}_{\alpha k_{r+1} k_r}$
(a)
(b)
(c)
$F^{m_{r-1} j_{r-1} j_r}_{\alpha k_r k_{r-1}}$
$F^{m_2 j_2 j_3}_{\alpha k_3 k_2}$
(d)
(e)
$F^{m_1 j_1 j_2}_{\alpha k_2 k_1}$
$\frac{v_\alpha v_\beta}{v_y} F^{j_1 k_1 \alpha}_{y\beta k_{r+1}}$
1-3 Pachner move
(f)
(g)

FIG. 6. Derivation of the expression for the tube operator.

$$\mathcal{P}^{\mathbf{1}\tau} = \frac{1}{\mathcal{D}^2}(O_{\tau\tau\mathbf{1}\tau} + e^{4\pi i/5}O_{\tau\tau\tau\mathbf{1}} + \sqrt{\phi}e^{-3\pi i/5}O_{\tau\tau\tau\tau}), \tag{24}$$

$$\mathcal{P}^{\tau\mathbf{1}} = \frac{1}{\mathcal{D}^2}(O_{\tau\tau\mathbf{1}\tau} + e^{-4\pi i/5}O_{\tau\tau\tau\mathbf{1}} + \sqrt{\phi}e^{3\pi i/5}O_{\tau\tau\tau\tau}), \tag{25}$$

$$\mathcal{P}^{\tau\tau} = \frac{1}{\mathcal{D}^2}\bigg(\phi^2 O_{\mathbf{1111}} - \phi O_{\mathbf{11}\tau\tau} + \phi O_{\tau\tau\mathbf{1}\tau} + \phi O_{\tau\tau\tau\mathbf{1}} + \frac{1}{\sqrt{\phi}}O_{\tau\tau\tau\tau}\bigg). \tag{26}$$

Note that $B_p = \mathcal{P}^{\mathbf{11}}$; i.e., the ground space is precisely the anyonic vacuum. The entry in Eq. (26) decomposes into two simple idempotents: $\mathcal{P}^{\tau\tau} = \mathcal{P}^{\tau\tau}_{\mathbf{1}} + \mathcal{P}^{\tau\tau}_{\tau}$, with

$$\mathcal{P}^{\tau\tau}_{\mathbf{1}} = \frac{1}{\mathcal{D}^2}(\phi^2 O_{\mathbf{1111}} - \phi O_{\mathbf{11}\tau\tau}), \tag{27}$$

$$\mathcal{P}^{\tau\tau}_{\tau} = \frac{1}{\mathcal{D}^2}\bigg(\phi O_{\tau\tau\mathbf{1}\tau} + \phi O_{\tau\tau\tau\mathbf{1}} + \frac{1}{\sqrt{\phi}}O_{\tau\tau\tau\tau}\bigg). \tag{28}$$

This decomposition of the central $\tau\tau$ idempotent into two simple idempotents should be interpreted as the fact that a $\tau\tau$ anyon charge in a plaquette does not fix the state of its tail qubit. The corresponding projector has a block-diagonal form with the two blocks corresponding to a $|0\rangle$ or a $|1\rangle$ state for the tail qubit. We label these two cases as $\tau\tau_{\mathbf{1}}$ and $\tau\tau_\tau$, respectively. Both are in the $\tau\tau$ anyon sector, and their respective $+1$ eigenspaces are related as follows:

$$\mathcal{P}^{\tau\bar{\tau}}_{\mathbf{1}\tau}\mathcal{P}^{\tau\bar{\tau}}_{\mathbf{1}} = \mathcal{P}^{\tau\bar{\tau}}_{\mathbf{1}\tau}, \qquad \mathcal{P}^{\tau\bar{\tau}}_{\tau}\mathcal{P}^{\tau\bar{\tau}}_{\mathbf{1}\tau} = \mathcal{P}^{\tau\bar{\tau}}_{\mathbf{1}\tau}, \tag{29}$$

$$\mathcal{P}^{\tau\bar{\tau}}_{\tau\mathbf{1}}\mathcal{P}^{\tau\bar{\tau}}_{\tau} = \mathcal{P}^{\tau\bar{\tau}}_{\tau\mathbf{1}}, \qquad \mathcal{P}^{\tau\bar{\tau}}_{\mathbf{1}}\mathcal{P}^{\tau\bar{\tau}}_{\tau\mathbf{1}} = \mathcal{P}^{\tau\bar{\tau}}_{\tau\mathbf{1}}, \tag{30}$$

where

$$\mathcal{P}_{\mathbf{1}\tau}^{\tau\tau} = e^{-3\pi i/10}\frac{\phi}{\mathcal{D}}O_{\mathbf{1}\tau\tau\tau}, \tag{31}$$

$$\mathcal{P}_{\tau\mathbf{1}}^{\tau\tau} = e^{3\pi i/10}\frac{\sqrt{\phi}}{\mathcal{D}}O_{\tau\mathbf{1}\tau\tau} \tag{32}$$

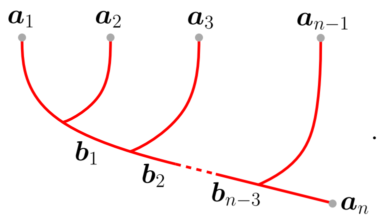
are nilpotent operators.

E. The anyonic fusion basis

An important property of the extended Levin-Wen model is the fact that one can construct a basis of the string-net subspace $\mathcal{H}_{\mathrm{sn}}$, whose elements are labeled by fusion states of $|P|$ anyons (of the doubled category $\mathcal{DC}$), where $|P|$ is the number of plaquettes. This basis is called the *anyonic fusion basis*. Below, we give the expressions for these bases in terms of ribbon configurations in the fattened lattice picture, for modular input categories. More details are given in Supplemental Secs. I D and I H [45].

For simplicity, we first show anyonic fusion basis states on a sphere. Since we are considering the fusion space of anyonic excitations in plaquettes, the corresponding fusion diagram are labeled by the string types of the doubled category $\mathcal{DC}$ (indicated by bold labels):

$$|\vec{\boldsymbol{a}},\vec{\boldsymbol{b}}\rangle = \tag{33}$$



As mentioned above, the anyon label of a plaquette alone does not always fix the state of the tail qudit. Hence, to fully specify the state, we must also fix the tail labels:

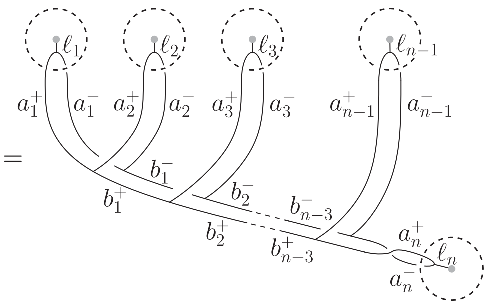
$$\vec{\ell} = [\ell_1, \ell_2, \ldots, \ell_{|P|}],$$

where $\vec{\ell}$ must be consistent with the anyon labels of all plaquettes. For $\mathcal{C} = \mathrm{FIB}$, the allowed combinations of plaquette anyon (DFIB) labels $\boldsymbol{a} = a_+a_-$ and tail labels ℓ are $(a_+a_-)_\ell \in \{\mathbf{1}\mathbf{1}_\mathbf{1}, \mathbf{1}\tau_\tau, \tau\mathbf{1}_\tau, \tau\mathbf{1}, \tau\tau_\tau\}$, which are simply all combinations satisfying $\delta_{a_+a_-\ell} = 1$.

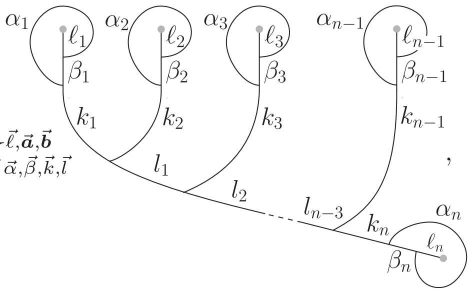
Throughout the remainder of this work, we often adopt a slight abuse of notation by also using a bold label to indicate the joined labels of the plaquette anyon and tail labels: $\boldsymbol{a} = (a_+a_-)_\ell$. It is always clear from the context whether or not a bold label includes the tail label. In particular, only leaf labels can contain a tail label. Internal branch labels of a doubled anyonic fusion tree never include them, since they do not correspond to plaquettes.

An anyonic fusion basis is determined by fixing the branching structure of the corresponding fusion tree and its embedding in the fattened lattice. The basis states are then labeled as $|\vec{\ell},\vec{\boldsymbol{a}},\vec{\boldsymbol{b}}\rangle$, where $\vec{\ell}$ are the tail labels, $\vec{\boldsymbol{a}}$ are the leaf labels (corresponding to the anyon charge of each plaquette), and $\vec{\boldsymbol{b}}$ are the internal branch labels. Before embedding them into the fattened lattice, the corresponding ribbon configurations are

$$|\vec{\ell},\vec{\boldsymbol{a}},\vec{\boldsymbol{b}}\rangle = \tag{34}$$

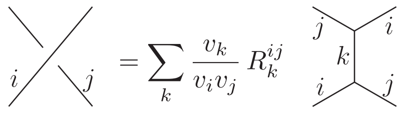


$$= \sum_{\vec{\alpha},\vec{\beta},\vec{k}} X^{\vec{\ell},\vec{\boldsymbol{a}},\vec{\boldsymbol{b}}}_{\vec{\alpha},\vec{\beta},\vec{k},\vec{l}} \tag{35}$$



where the coefficients $X^{\vec{\ell},\vec{\boldsymbol{a}},\vec{\boldsymbol{b}}}_{\vec{\alpha},\vec{\beta},\vec{k},\vec{l}}$ are found by resolving the crossings in Eq. (34) using

$$\text{[crossing } i,j\text{]} = \sum_k \frac{v_k}{v_i v_j} R_k^{ij} \tag{36}$$



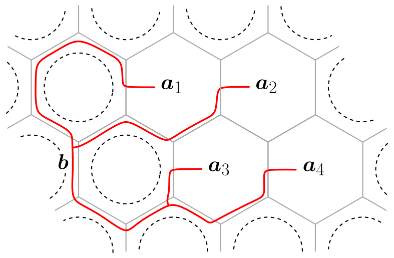
followed by a sequence of F moves and 1-3 Pachner moves. The object R_k^{ij} above is known as the R matrix of the input category $\mathcal{C}$ and defines its braiding properties. It must satisfy certain consistency equations, which are listed in Supplemental Sec. I A [45]. For the Fibonacci category, the only nonzero entries are

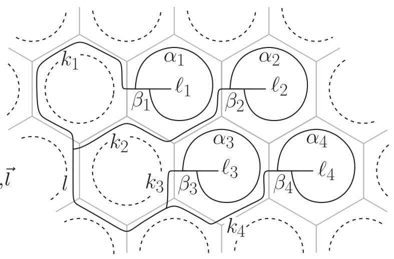
$$R_{\mathbf{1}}^{\tau\tau} = e^{4\pi i/5}, \qquad R_{\tau}^{\tau\tau} = e^{-3\pi i/5}, \qquad R_a^{\mathbf{1}a} = R_a^{a\mathbf{1}} = 1, \tag{37}$$

where $a \in \{\mathbf{1}, \tau\}$. The necessary calculations to obtain $X^{\vec{\ell},\vec{\boldsymbol{a}},\vec{\boldsymbol{b}}}_{\vec{\alpha},\vec{\beta},\vec{k},\vec{l}}$ are performed explicitly in Supplemental Sec. I D [45].

After picking some embedding in the fattened lattice, we find

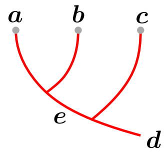
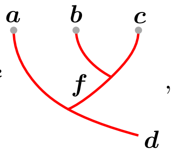
$$|\vec{\ell},\vec{\boldsymbol{a}},\vec{\boldsymbol{b}}\rangle = \tag{38}$$

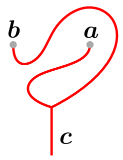
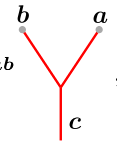


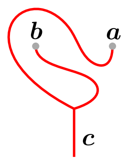
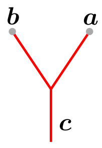
$$= \sum_{\vec{\alpha},\vec{\beta},\vec{k},\vec{l}} X^{\vec{\ell},\vec{a},\vec{b}}_{\vec{\alpha},\vec{\beta},\vec{k},\vec{l}}$$  ,

where we choose not to draw the leaves with vacuum labels explicitly but include vacuum loops in the corresponding plaquettes instead. The fattened lattice state above corresponds to the case where only four plaquettes carry a nontrivial anyonic charge; other cases are analogous. The final expression for the anyonic fusion basis states (as a state of qudits) can then be found by resolving these ribbon graph configurations into the lattice.

Different anyonic fusion bases (that is, bases corresponding to trees with different branching structures, or different embeddings in the fattened lattice) can be related using F moves, braid moves, and Dehn twists defined by the categorical data of $\mathcal{DC}$:

 $= \sum_{\boldsymbol{f}} F^{\boldsymbol{abe}}_{\boldsymbol{cdf}}$  , (39)

 $= R^{\boldsymbol{ab}}_{\boldsymbol{c}}$  , (40)

 $= \left(R^{\boldsymbol{ba}}_{\boldsymbol{c}}\right)^*$  , (41)

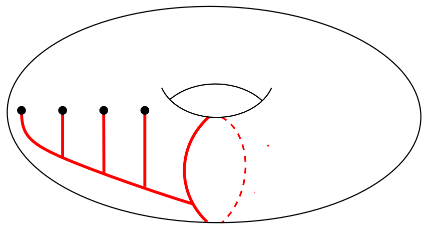
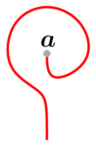
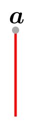
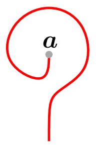
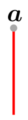


FIG. 7. Fusion diagram of a system of anyons defined on a torus. Note the line wrapping around the torus.

 $= \theta_{\boldsymbol{a}}$  , (42)

 $= (\theta_{\boldsymbol{a}})^*$  . (43)

Because of the particular structure of the doubled category, $\mathcal{DC} \simeq \mathcal{C} \otimes \bar{\mathcal{C}}$, its numerical data can be deducted from that of the input category $\mathcal{C}$. In particular, one has

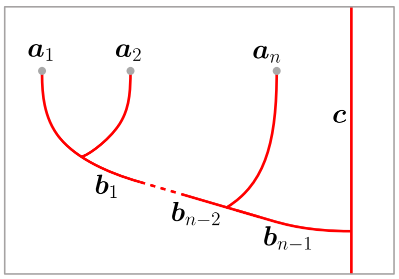
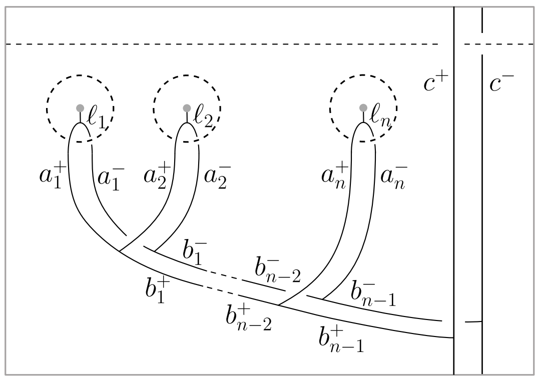
$$F^{\boldsymbol{abe}}_{\boldsymbol{cdf}} = F^{a_+b_+e_+}_{c_+d_+f_+} F^{a_-b_-e_-}_{c_-d_-f_-}, \tag{44}$$

$$R^{\boldsymbol{ab}}_{\boldsymbol{c}} = R^{a_+b_+}_{c_+} (R^{b_-a_-}_{c_-})^*, \tag{45}$$

$$\theta_{\boldsymbol{a}} = \theta_{a_+} (\theta_{a_-})^*. \tag{46}$$

We discussed how these are obtained for modular input categories in more detail in Supplemental Sec. I D [45].

The construction of anyonic fusion basis states on a torus is similar. An important difference is that fusion states of anyons on a torus require us to specify a *handle label* (see Supplemental Sec. I D 2 [45] for more details), which determines how the state transforms when an anyon is moved along a noncontractible loop [47]. An example of such a fusion state is shown in Fig. 7. Anyonic fusion basis states are then labeled as $|\vec{\ell}, \vec{\boldsymbol{a}}, \vec{\boldsymbol{b}}, \boldsymbol{c}\rangle$, where $\vec{\ell}$, $\vec{\boldsymbol{a}}$, and $\vec{\boldsymbol{b}}$ are again the tail, leaf, and internal branch labels, respectively, and $\boldsymbol{c}$ is the handle label. The corresponding ribbon configurations are

$|\vec{\ell}, \vec{\boldsymbol{a}}, \vec{\boldsymbol{b}}, \boldsymbol{c}\rangle =$  $=$  , (47) where the gray box represents the periodic boundary conditions of a torus. Note that the handle label must satisfy $\delta_{\boldsymbol{b}_{n-1}\boldsymbol{c}\boldsymbol{c}}$ or, equivalently, $\delta_{b^+_{n-1}c^+c^+}$ and $\delta_{b^-_{n-1}c^-c^-}$ for the corresponding ribbon graph to obey the branching rules. The crossings on the right-hand side in Eq. (47) must again be resolved using Eq. (36), which leads to superposition of ribbon configurations similar to Eq. (35). These ribbons must then be embedded in the fattened lattice like in Eq. (38) and resolved into the lattice using F moves.

Ground states of the model correspond to (linear combinations of) configurations in which all plaquettes carry a trivial charge ($\boldsymbol{a}_i = \mathbf{11}$, $\boldsymbol{b}_j = \mathbf{11}$, $\forall\, i, j$). On a torus, the degenerate ground space is spanned by the states $|\vec{\mathbf{1}}, \vec{\mathbf{11}}, \vec{\mathbf{11}}, \boldsymbol{c}\rangle$, where $\vec{\mathbf{1}}$ and $\vec{\mathbf{11}}$ represent arrays containing only trivial entries. One can show that these states are indeed orthonormal. Hence, when storing the state of two logical qubits (for $\mathcal{C} = \mathrm{FIB}$), this information is encoded in the handle label. The operations that affect the handle are precisely those in which anyons interact along a noncontractible path such as the process depicted in Fig. 8. As long as no such operations are performed, the encoded information is preserved. Local qudit errors create pairs, triplets, or quadruplets of nontrivial anyons in neighboring plaquettes. The initial ground state can then be recovered by fusing these nontrivial anyons pairwise until none are left, without creating any noncontractible loops in the process.

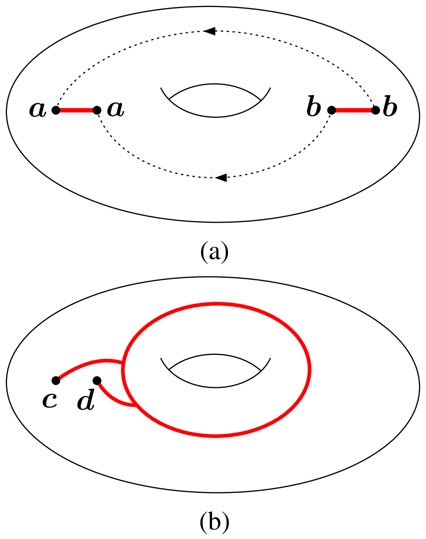


FIG. 8. (a) Topologically nontrivial process which results in a logical error: Two pairs of anyons $(\boldsymbol{a}, \boldsymbol{a})$ and $(\boldsymbol{b}, \boldsymbol{b})$ are created by local qudit errors. The anyons are then fused along the dotted black paths with outcomes $\boldsymbol{c}$ and $\boldsymbol{d}$. (b) The fusion diagram of the resulting state winds around the torus.

Clearly, all anyonic fusion basis states correspond to exceedingly complicated superpositions of qudit states. It would seem that this makes them highly impractical for any actual computation. Fortunately, however, one can formulate a tensor network representation for such states, which enables us to use them for practical applications (in particular, see Sec. V E). This tensor network representation is derived in Supplemental Sec. III.

III. ERROR CORRECTION SCHEME

After defining the microscopic model used to define the code and discussing its most important properties, we now discuss the error correction scheme required to implement the extended string-net code. Specifically, we present the circuits to measure and fix arbitrary local errors.

Our overall error correction procedure is composed of two major steps.

(1) Measure all the vertex operators Q_v in the extended Levin-Wen model Eq. (8) and apply a correction which fixes the vertex errors through unitaries U_V conditioned by a measurement projection P_V. This measurement and correction processes projects the many-body state onto the string-net subspace $\mathcal{H}_{\mathrm{sn}}$.

(2) After projecting to the string-net subspace $\mathcal{H}_{\mathrm{sn}}$, we apply additional measurement circuits to measure the simple idempotents of the tube algebra [Eqs. (23), (24), (25), (27), and (28)] and extract the error syndromes, i.e., the anyon charges and tail labels of all plaquettes. Based on these syndromes, we use our decoders to identify the error location (up to equivalence classes) and apply the corresponding recovery maps to project the state back to the code (ground) space $\mathcal{H}_\Lambda$. We note that the code space is a subspace of the string-net subspace: $\mathcal{H}_\Lambda \subset \mathcal{H}_{\mathrm{sn}}$.

In this section, we discuss all measurement and recovery operators required to implement these steps. In particular, we discuss the vertex measurement and correction processes in Sec. III A and the anyon charge measurements in Sec. III B. The recovery operations are discussed in Supplemental Sec. II [45].

From here on, we work exclusively with the Fibonacci input category; hence, we are working with a system of qubits on a lattice.

A. Vertex measurements and correction

Certain types of errors, such as a single bit-flip error σ_x^e or a coherent error generated by the Pauli-X operator,

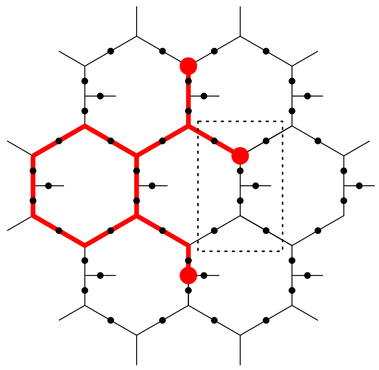
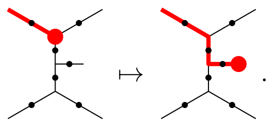


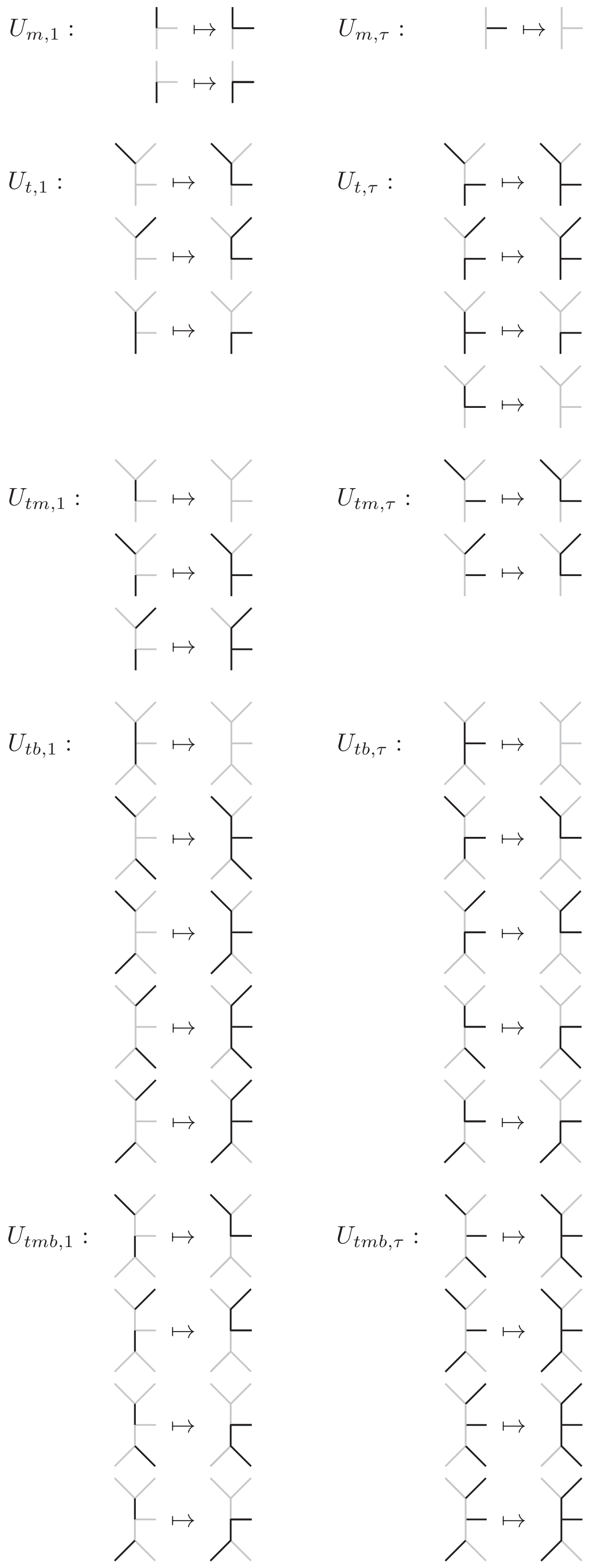
FIG. 9. A configuration that violates the branching rules by having string-endings in three vertices, indicated by the red dots.

i.e., $e^{i\theta\sigma_x^e}$, can cause a violation of the vertex projector Q_v for a vertex v adjacent to edge e. As illustrated in Fig. 9, a vertex error corresponds to a broken string ending at that vertex (in the string-net configurations of the system wave function). Note that a generic error such as $e^{i\theta\sigma_x^e}$ puts the system wave function in a superposition of violating and not violating the vertex condition at any adjacent vertex v. When we measure the vertex operator Q_v, we project to either of the two situations. Vertex violations can be resolved by local unitary operators, which take the system back to the string-net subspace $\mathcal{H}_{\mathrm{sn}}$. Intuitively, one can think of the action of these operators as pulling string ends into the tail edge of a neighboring plaquette:


(48)

Previously, the circuit for measuring the vertex errors has been discovered in Ref. [28]. Here, we adopt this circuit to measure the top, middle, and bottom vertices, with three ancilla qubits (white dots) denoted by t, m, and b, respectively, as shown in Figs. 10(a)–10(c). The circuits for measuring the top and bottom are symmetric, so we show only the top one in Fig. 10(a). For the middle vertex m, we also apply a measurement of the tail qubit at the end of the circuit. To respect the usual convention of quantum circuits, we represent the unoccupied edge as $|0\rangle$, which corresponds to the vacuum string label **1**, and the occupied edge as $|1\rangle$, corresponding to the string label τ. The ancilla qubits are all initialized in the state $|0\rangle$.

We apply a correction U_V, conditioned by the measurement results of the three vertex operators and the tail qubit, to fix the vertex error. This correction is selected out of 14 possible unitaries:



Note that we omit the mappings which are mirror symmetric ($t \leftrightarrow b$) to the listed ones.

The corresponding quantum circuits of the above unitaries are listed in Figs. 10(d)–10(l). For gates which do not have overlap in qubit support, we can parallelize them in a single time step, as indicated by the dashed boxes. As we can see, most of the unitary circuits have depth 1 or 2, while only one of them, $U_{tb,\tau}$ in Fig. 10(i), has depth 4. The overall measurement and correction circuit is summarized in Fig. 10(m), where we parallelize the measurement of the three vertex operators into a depth-5 circuit (in terms of the unitary gates). When taking into account the readout of the ancilla qubits before applying U_V, the depth is 6. Note that we neglect the step of state preparation of the ancilla qubits in the beginning, because, in the situation of repetitive syndrome measurements, the ancilla qubits can always be prepared during the application of the correction unitary U_V. Overall, the depth of the measurement circuits ranges from 5 to 9, or from 6 to 10 when taking into account the measurement step.

A different scheme of fixing vertex errors has been previously proposed in Refs. [37,38], which also uses tail qubits and, hence, has a similar spirit.

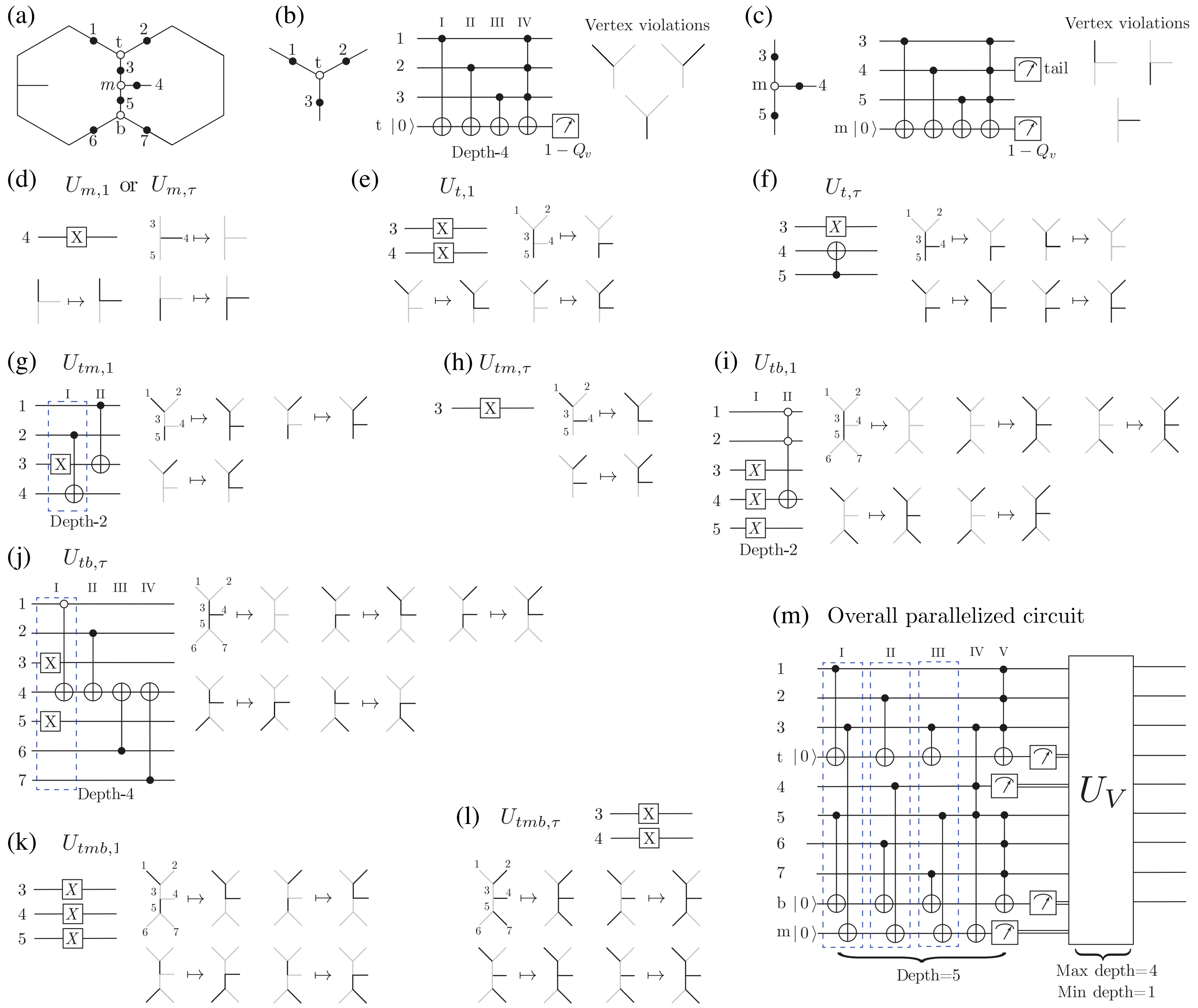


FIG. 10. Measurement and correction circuit for the vertex errors. (a) The qubit labeling. The black dots represent data qubits, while the white dots represent ancilla qubits for measurements. (b),(c) The measurement circuit for the top and middle vertex projectors. The circuit for measuring the bottom vertex can be inferred by symmetry. (d)–(l) The circuits for pulling an open string into the tail edge when certain vertices are violated. (m) The overall circuit for measuring and correcting vertex errors. The first part of the circuit measures the three vertex projectors. The second part is the unitary U_V conditioned on the measurement results which is summarized in (d)–(l).

B. Anyon charge measurements

After measuring the vertex operators and applying the corresponding corrections on the extended string-net code, we transform the many-body state to the string-net subspace $\mathcal{H}_{\mathrm{sn}}$, where it can be described in terms of anyonic fusion states. The local qubit errors, including both the Pauli-X and Z types, now result in the creation of anyonic excitations inside $\mathcal{H}_{\mathrm{sn}}$.

As explained in Sec. II D, one can measure the anyonic charge of a plaquette using the central idempotents of the tube algebra. To fully characterize an excitation, we must also measure its tail label. A joint measurement of the anyonic charge and the tail label is achieved by measuring the *irreducible* idempotents of the tube algebra, listed in Eqs. (23)–(25), (27), and (28). Measuring these irreducible idempotents is done in three steps.

(1) Grow a tube inside the plaquette by introducing ancilla qubits and performing the appropriate quantum circuit, as shown in Fig. 13.
(2) Measure the tube qubits in the appropriate basis.
(3) Either trace out the tube qubits immediately or first resolve the tube back into the lattice before tracing out the ancillas.

As a basic ingredient, the quantum circuit to implement the F-move (2-2 Pachner move) operation F^{abe}_{cdf} in the Fibonacci Turaev-Viro code is shown in Fig. 11. This circuit was first proposed in Ref. [28]. The F-move operation can be viewed as a controlled unitary operation, where the external legs a, b, c, and d are control qubits determining the resulting unitary F^{ab}_{cd}, with the matrix elements being $[F^{ab}_{cd}]_{ef}$. For the Fibonacci Turaev-Viro code, the F matrix is given in Eq. (14). The circuit inside the red dashed box, composed of a five-qubit Toffoli gate in between two single-qubit rotations, applies the conditional unitary corresponding to the F matrix $F^{\tau\tau}_{\tau\tau}$, where $R_y(\pm\theta) = e^{\pm i\theta\sigma_y/2}$ are single-qubit rotations about the y axis with angle $\theta = \tan^{-1}(\phi^{-1/2})$. Note that this conditional unitary is activated only if the control qubits a, b, c, and d are all in the $|1\rangle$ state corresponding to the string label τ. All the other conditional unitaries are implemented by the rest of the quantum circuit.

Based on the circuit for the 2-2 Pachner move (F move), one can also implement the 1-3 Pachner move with a

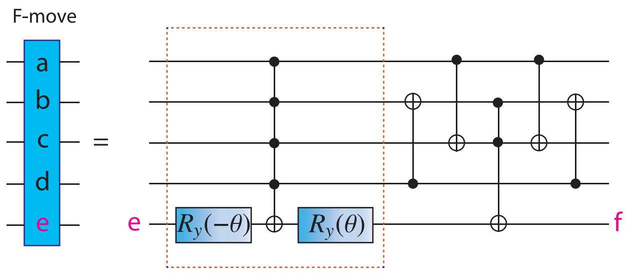


FIG. 11. Quantum circuit to implement the F-move (2-2 Pachner move) operation F^{abe}_{cdf} for the Fibonacci Turaev-Viro code.

unitary circuit, as shown in Fig. 12. The protocol consists of the following steps: (i) Initialize three ancilla qubits (white dots) in state $|0\rangle$. (ii) Apply a CNOT gate which entangles the data qubit labeled j to the new ancilla qubit on the same edge, CNOT: $|j\rangle|0\rangle \mapsto |j\rangle|j\rangle$, as shown in Figs. 12(a) and 12(b). (iii) Apply a modular-$\mathcal{S}$ gate on one ancilla to create a tadpole diagram, as shown in Fig. 12(b). The modular $\mathcal{S}$ does the following transformation: $\mathcal{S}\colon|0\rangle \mapsto \sum_\lambda (d_\lambda/D)|\lambda\rangle$. For the Fibonacci Turaev-Viro code, the modular $\mathcal{S}$ matrix is

$$\mathcal{S} = \frac{1}{\sqrt{2+\phi}}\begin{pmatrix} 1 & \phi \\ \phi & -1 \end{pmatrix}. \tag{49}$$

(iv) Apply an F move to absorb the tadpole onto the edge, as shown in Figs. 12(b) and 12(c). (v Apply another F move to sweep edge λ to attach the right leg (with label k), as shown in Figs. 12(c) and 12(d). From left to right, the circuit effectively fine grain the lattice by entangling the three ancilla qubits into the code space. One can also reverse the circuit (from right to left) which corresponds to a coarse-graining process disentangling three qubits out of the code space.

The "growing" of a tube onto the tailed lattice can then be implemented by a quantum circuit, as shown in Figs. 13 and 14. We start with the tailed lattice in Fig. 13(a) with data qubits (black dots) residing on every edge. We then introduce three ancilla qubits (white dots) in Fig. 13(b) initialized at $|0\rangle$. From Fig. 13(b) to Fig. 13(e), we apply a series of operations to achieve a 1-3 Pachner move to add a triangle loop below the tail: (i) Apply a CNOT gate which entangles the data qubit on the tail to the new ancilla qubit

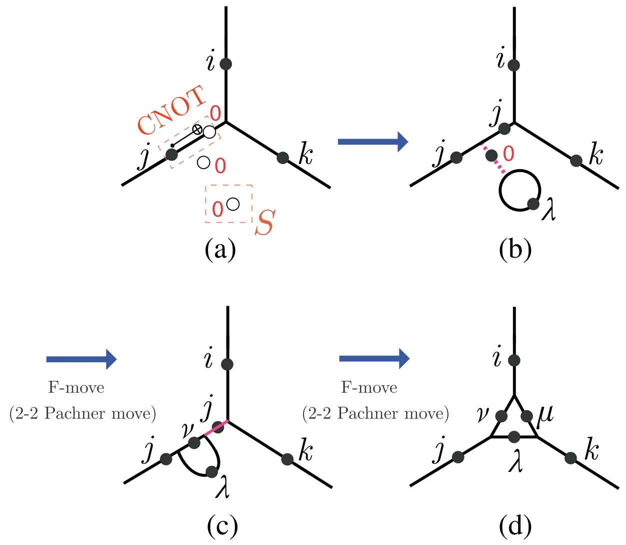


FIG. 12. Quantum circuit to implement the 1-3 Pachner move for the Fibonacci Turaev-Viro code. (a)-(b) A tadpole is created by initializing three ancilla qubits in the $|0\rangle$ state, entangling one of them with a qubit on an edge of the lattice using a CNOT gate, and applying the modular $\mathcal{S}$ gate. (c)-(d) The resulting tadpole is absorbed into the lattice using two F moves.

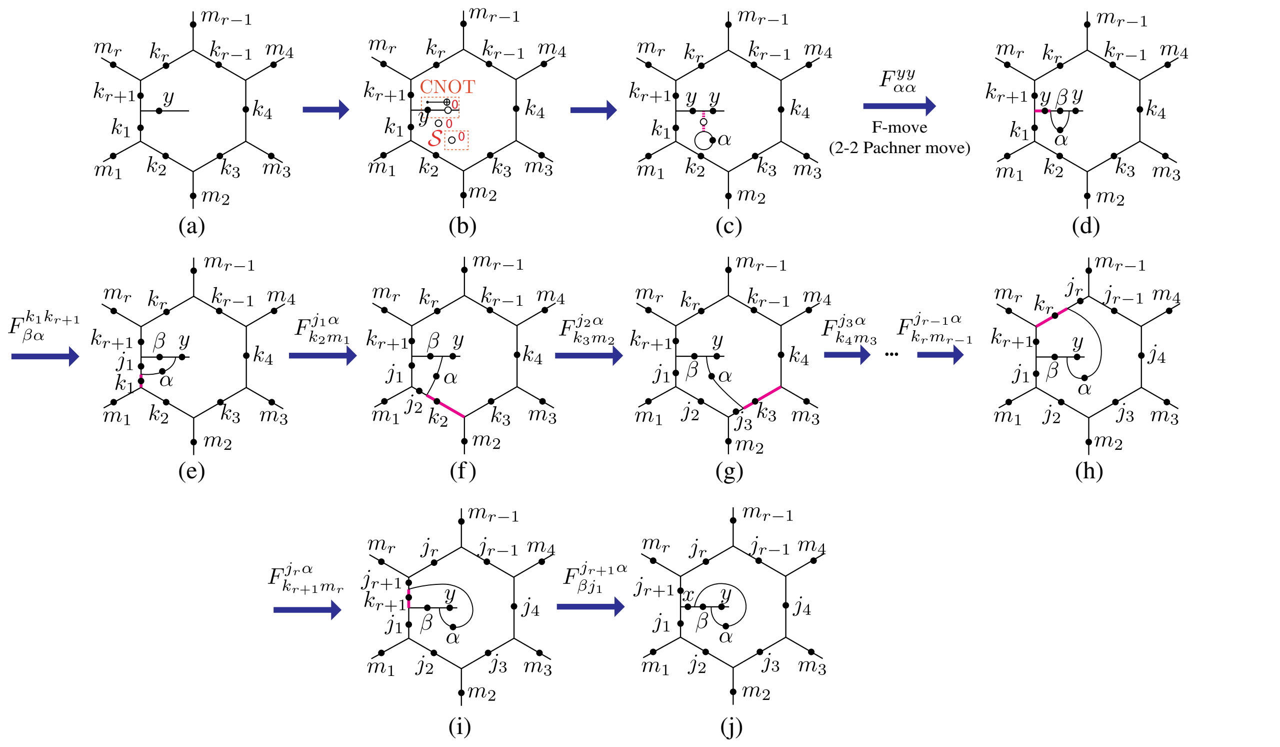


FIG. 13. Protocol and circuit of growing a tube onto a puncture in a plaquette on a tailed lattice via a sequence of local gates and Pachner moves.

on the tail, CNOT: $|y\rangle|0\rangle \mapsto |y\rangle|y\rangle$, as shown in Figs. 13(b) and 13(c). (ii) Apply a modular-$\mathcal{S}$ gate on one ancilla to create a tadpole diagram, as shown in Fig. 13(b), i.e., $\mathcal{S}: |0\rangle \mapsto \sum_\alpha (d_\alpha/D)|\alpha\rangle$. (iii) Apply an F move to absorb the tadpole onto the tail, as shown in Figs. 13(c) and 13(d). (iv) Apply another F move to sweep edge α to attach the left edge, as shown in Figs. 13(d) and 13(e). Afterward, we apply a sequence of F moves to sweep edge α around the whole plaquette, after which it ends up in the upper side of the tail. In this way, we grow a tube. Note that, as a result, the qubits on the plaquette and tail edges [with labels k_i and y in Fig. 13(a)] get rotated one position counterclockwise. The details of the quantum gates in this circuit are shown in Fig. 14. As we can see, in total nine F moves are applied.

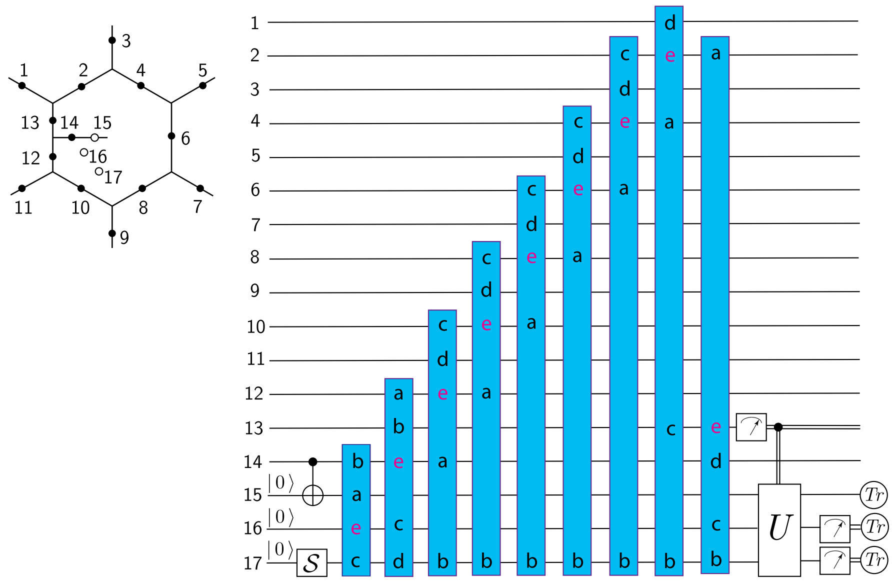
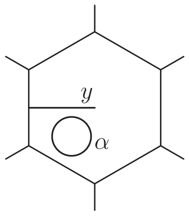


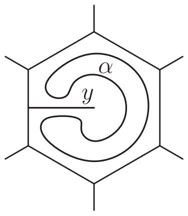
FIG. 14. The complete quantum circuit for the joint measurement of the anyon charge and tail label of a plaquette. Note that, after the grow circuit, qubit 13 corresponds to the tail edge.

After growing the tube, the anyon charge can be inferred by measuring the four qubits on the tube. In order to find the appropriate basis for this measurement, we first note that the growing procedure can be thought of as creating a vacuum bubble [in Fig. 13(c)], stretching it out along the boundary of the plaquette, and finally resolving only half of it into the lattice. The remaining half then constitutes the tube in Fig. 13(j). In terms of ribbon diagrams, the stretched out vacuum bubble inside the plaquette can be written as

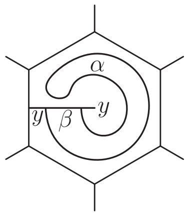
$$\sum_\alpha \frac{d_\alpha}{\mathcal{D}}$$



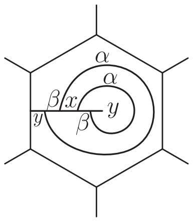
$$= \sum_\alpha \frac{d_\alpha}{\mathcal{D}}$$



$$= \sum_{\alpha,\beta} \frac{d_\alpha}{\mathcal{D}} \frac{v_\beta}{v_\alpha v_y}$$



$$= \sum_{\alpha,\beta,x} \frac{1}{\mathcal{D}} \frac{v_x}{v_y}$$

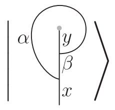


. (50)

The sequence of F moves appearing in the grow circuit corresponds exactly to resolving only the outer tube into the lattice [48]. If we denote the initial state of the lattice qubits as $|\Psi_0\rangle = \sum_y \varepsilon_y |\phi_y\rangle \otimes |y\rangle$ and the ancillas are initially in the $|000\rangle$ state, then the action of the grow circuit is

$$|\Psi_0\rangle \otimes |000\rangle \mapsto \sum_y \varepsilon_y \sum_{\alpha,\beta,x} \frac{1}{\mathcal{D}} \frac{v_x}{v_y} \tilde{O}_{yx\alpha\beta}(|\phi_y\rangle \otimes |y\rangle) \otimes |y\alpha\beta\rangle, \tag{51}$$

where we use the following abbreviation for the tube state vectors:



$$\equiv |x\rangle \otimes |y\alpha\beta\rangle\,,$$

and where

$$\tilde{O}_{yx\alpha\beta} \equiv \sum_\gamma F^{\alpha x\beta}_{\alpha y\gamma} O_{yx\alpha\gamma} \tag{52}$$

is the operator which corresponds to resolving a outer tube [49] into the lattice. If a measurement projects the tube qubits onto the state

$$|\psi\rangle = \sum_{\alpha,\beta} A_{\alpha\beta} |x\rangle \otimes |y\alpha\beta\rangle, \tag{53}$$

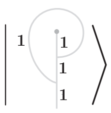
then the full state gets projected onto

$$|\Phi\rangle \otimes |\psi\rangle = \frac{1}{\mathcal{N}} \left(\sum_{\alpha,\beta} \frac{1}{\mathcal{D}} \frac{v_x}{v_y} (A_{\alpha\beta})^* (\mathbb{1} \otimes \langle x|) \tilde{O}_{yx\alpha\beta}(|\phi_y\rangle \otimes |y\rangle) \right) \otimes |\psi\rangle, \tag{54}$$

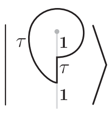
where $\mathcal{N}$ is a normalization factor. Hence, by selecting the basis for the measurement of the tube qubits carefully, we can effectively apply the idempotent projectors Eqs. (23)–(28) or the nilpotent operators Eqs. (31) and (32).

We choose the following basis [50] for the measurement of the tube qubits:

$$|\psi_{\mathbf{11}}\rangle = \frac{1}{\mathcal{D}}$$

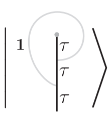


$$+ \frac{\phi}{\mathcal{D}}$$

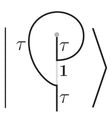


$$= |0\rangle \otimes \frac{1}{\mathcal{D}} \Big(|000\rangle + \phi\, |011\rangle \Big) \equiv |0\rangle \otimes |\tilde{\psi}_{\mathbf{11}}\rangle\,, \tag{55}$$

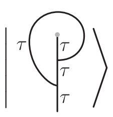
$$|\psi_{\mathbf{1}\tau}\rangle = \frac{1}{\mathcal{D}}$$



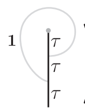
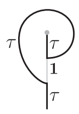
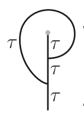
$$+ \frac{e^{4\pi i/5}}{\mathcal{D}}$$



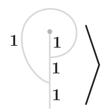
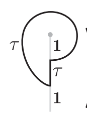
$$+ \sqrt{\phi} \frac{e^{-3\pi i/5}}{\mathcal{D}}$$

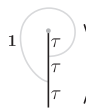
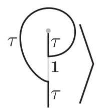
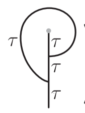


$$= |1\rangle \otimes \frac{1}{\mathcal{D}} \Big(|101\rangle + e^{4\pi i/5}\, |110\rangle + \sqrt{\phi} e^{-3\pi i/5}\, |111\rangle \Big) \equiv |1\rangle \otimes |\tilde{\psi}_{\mathbf{1}\tau}\rangle\,, \tag{56}$$

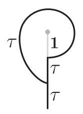
$|\psi_{\tau\mathbf{1}}\rangle = \frac{1}{\mathcal{D}}\Big|$  $\Big\rangle + \frac{e^{-4\pi i/5}}{\mathcal{D}}\Big|$  $\Big\rangle + \sqrt{\phi}\frac{e^{3\pi i/5}}{\mathcal{D}}\Big|$  $\Big\rangle$

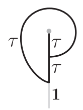
$$= |1\rangle \otimes \frac{1}{\mathcal{D}}\Big(|101\rangle + e^{-4\pi i/5}|110\rangle + \sqrt{\phi}e^{3\pi i/5}|111\rangle\Big) \equiv |1\rangle \otimes |\tilde{\psi}_{\tau\mathbf{1}}\rangle, \tag{57}$$

$|\psi_{\tau\tau,\mathbf{1}}\rangle = \frac{\phi}{\mathcal{D}}\Big|$  $\Big\rangle - \frac{1}{\mathcal{D}}\Big|$  $\Big\rangle = |0\rangle \otimes \frac{1}{\mathcal{D}}\Big(\phi|000\rangle - |011\rangle\Big) \equiv |0\rangle \otimes |\tilde{\psi}_{\tau\tau,\mathbf{1}}\rangle, \quad (58)$

$|\psi_{\tau\tau,\tau}\rangle = \frac{\sqrt{\phi}}{\mathcal{D}}\Big|$  $\Big\rangle + \frac{\sqrt{\phi}}{\mathcal{D}}\Big|$  $\Big\rangle + \frac{1}{\phi\mathcal{D}}\Big|$  $\Big\rangle$

$$= |1\rangle \otimes \frac{1}{\mathcal{D}}\Big(\sqrt{\phi}|101\rangle + \sqrt{\phi}|110\rangle + \frac{1}{\phi}|111\rangle\Big) \equiv |1\rangle \otimes |\tilde{\psi}_{\tau\tau,\tau}\rangle, \tag{59}$$

$|\psi_{\tau\tau,\mathbf{1},\tau}\rangle = \Big|$  $\Big\rangle = |1\rangle \otimes |011\rangle \equiv |1\rangle \otimes |\tilde{\psi}_{\tau\tau,\mathbf{1},\tau}\rangle, \quad (60)$

$|\psi_{\tau\tau,\tau,\mathbf{1}}\rangle = \Big|$  $\Big\rangle = |0\rangle \otimes |111\rangle \equiv |0\rangle \otimes |\tilde{\psi}_{\tau\tau,\tau,\mathbf{1}}\rangle. \quad (61)$

Note that the coefficients appearing in the different states are proportional to those in the corresponding irreducible idempotents in Eqs. (23)–(25), (27), and (28) or nilpotents in Eq. (31), respectively. In fact, these states are precisely the anyonic fusion basis states on Σ_2, as described in Eq. (SI.51) in Supplemental Sec. I D [45].

The measurement of the tube qubits in Fig. 13(j) is done in three steps.

(1) Measure the tail qubit (label x).

(2) Apply one of the following unitaries [51] conditioned on the measurement of the tail qubit:

(a) if the tail qubit is in state $|0\rangle$ (string label $\mathbf{1}$):

$$U_{\mathbf{1}} = |0\rangle(|00\rangle\langle\tilde{\psi}_{\mathbf{11}}| + |11\rangle\langle\tilde{\psi}_{\tau\tau,\mathbf{1}}| + |10\rangle\langle\tilde{\psi}_{\tau\tau,\tau,\mathbf{1}}|);$$

(b) if the tail qubit is in state $|1\rangle$ (string label τ):

$$U_\tau = |0\rangle(|00\rangle\langle\tilde{\psi}_{\tau\tau,\tau}| + |11\rangle\langle\tilde{\psi}_{\tau\tau,\mathbf{1},\tau}| + |01\rangle\langle\tilde{\psi}_{\mathbf{1}\tau}| + |10\rangle\langle\tilde{\psi}_{\tau\mathbf{1}}|).$$

(3) Measure qubits 16 and 17 in Fig. 14 in the Z basis.

Once the tube qubits are measured in the basis Eqs. (55)–(61) using the procedure above, we can trace out the three ancilla qubits [15, 16, and 17 in Fig. 14, constituting the inner three edges of the tube in Fig. 13(j)] to return to the initial tailed lattice layout with a single tail qubit in each plaquette [i.e., the configuration in Fig. 13(a)]. This results in the following positive operator-valued measure (POVM):

$$\left\{\mathcal{P}^{\mathbf{11}}, \mathcal{P}^{\mathbf{1}\tau}, \mathcal{P}^{\tau\mathbf{1}}, \frac{1}{\phi^2}\mathcal{P}^{\tau\tau}_{\mathbf{1}}, \frac{1}{\phi}\mathcal{P}^{\tau\tau}_{\tau}, \frac{1}{\phi}\mathcal{P}^{\tau\tau}_{\mathbf{1}}, \frac{1}{\phi^2}\mathcal{P}^{\tau\tau}_{\tau}\right\}. \tag{62}$$

Note that both a $\tau\tau_{\mathbf{1}}$ and a $\tau\tau_\tau$ excitation correspond to two different measurement outcomes. However, within each of these pairs, the postmeasurement states are not identical. For instance, a $\tau\tau_{\mathbf{1}}$ excitation can result in measurement outcomes $|\psi_{\tau\tau,\mathbf{1}}\rangle$ and $|\psi_{\tau\tau,\mathbf{1},\tau}\rangle$. Obtaining outcome $|\psi_{\tau\tau,\mathbf{1}}\rangle$ effectively applies the $\mathcal{P}^{\tau\tau}_{\mathbf{1}}$ idempotent, meaning the postmeasurement state has trivial tail label $\mathbf{1}$. On the other hand, the outcome $|\psi_{\tau\tau,\mathbf{1},\tau}\rangle$ corresponds to the application of the $\mathcal{P}^{\tau\tau}_{\mathbf{1}\tau}$ nilpotent. Since $\mathcal{P}^{\tau\tau}_{\mathbf{1}\tau} = \mathcal{P}^{\tau\tau}_{\tau}\mathcal{P}^{\tau\tau}_{\mathbf{1}\tau}\mathcal{P}^{\tau\tau}_{\mathbf{1}}$, this means the plaquette initially contains a $\tau\tau_{\mathbf{1}}$ excitation, which gets transformed to a $\tau\tau_\tau$ excitation in the postmeasurement state. The situation for a $\tau\tau_\tau$ excitation is analogous. Hence, these measurements do not preserve the tail label in the case of a $\tau\tau$ anyon, and a subsequent measurement might yield a different outcome. It is important to note that this affects only the tail label *within one anyon sector*; the anyon label itself cannot be altered by subsequent measurements.

In case one wishes to preserve the tail label of excitations in subsequent measurements, [52] an additional step is required before tracing out the three ancillas. This step consists of resolving the tube into the lattice by applying the gates of the grow circuit in reverse, as indicated in

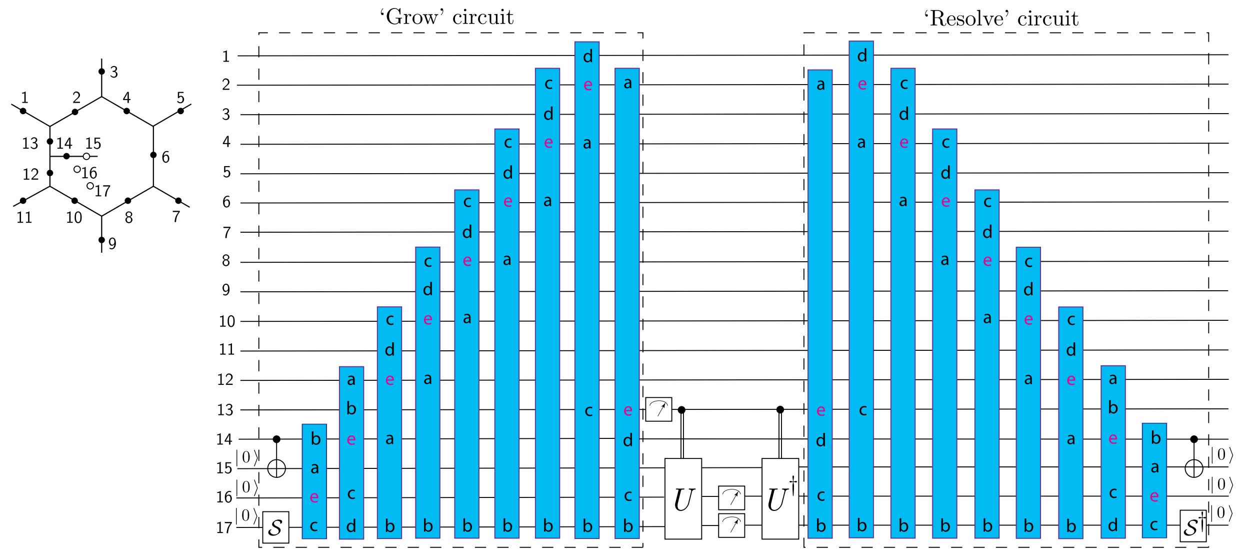


FIG. 15. Alternative quantum circuit for the joint measurement of the anyon charge and tail label of a plaquette, which does preserve the tail label.

Fig. 15. For measurement outcome $|\psi\rangle$ defined Eq. (53), the resolving process is equivalent to the following transformation on the postmeasurement state in Eq. (54):

$$|\Phi\rangle \otimes |\psi\rangle \mapsto \sum_{\alpha,\beta} A_{\alpha\beta} O_{xy\alpha\beta}(|\Phi\rangle \otimes |x\rangle) \otimes |000\rangle. \quad (63)$$

This guarantees that the initial tail label y indeed is recovered. For instance, when obtaining the $|\psi_{\tau\tau,\mathbf{1}\tau}\rangle$ measurement outcome, the resolving process results in the application of the $\mathcal{P}^{\tau\tau}_{\tau\mathbf{1}}$ nilpotent on top of the $\mathcal{P}^{\tau\tau}_{\mathbf{1}\tau}$ nilpotent applied by the measurement, resulting in the combined action $\mathcal{P}^{\tau\tau}_{\mathbf{1}} = \mathcal{P}^{\tau\tau}_{\tau\mathbf{1}}\mathcal{P}^{\tau\tau}_{\mathbf{1}\tau}$, meaning that the tail label is now preserved.

The same result can be achieved by using repeated measurements with the circuit in Fig. 14. In case one measures a $\tau\tau$ excitation but finds that the tail label gets flipped [corresponding to the tube states in Eqs. (60) and (61) and the last two entries in the POVM Eq. (62)], each subsequent measurement [53] has a fixed probability of flipping the tail label back to its initial value (as determined by the first measurement). For a $\tau\tau_\mathbf{1}$ excitation, it follows from Eq. (62) that the probability that n measurements are required before the postmeasurement state has a trivial tail label is $\phi^{-(n+1)}$. Hence, on average, ϕ^2 measurements are required to ensure that the tail label is preserved for a $\tau\tau_\mathbf{1}$ excitation. Likewise, for a $\tau\tau_\tau$ excitation, the probability of needing n measurements to recover the initial tail label is $\phi^{-(2n-1)}$, resulting in an average of ϕ measurements. Whether one should choose for the longer measurement circuit depicted in Fig. 15 or for repeated measurements with the shorter measurement circuit depicted in Fig. 14 depends on what types of excitation are more likely to appear.

The procedure described above determines the anyon charge of a single plaquette. By repeating it for all plaquettes, we obtain the complete error syndrome (in the form of the anyonic content of every plaquette), which must then be passed to a decoding algorithm to determine the appropriate recovery operations to be performed (see Sec. IV). The quantum circuits for performing these recovery operations are given in Supplemental Sec. II [45].

IV. DECODING ALGORITHMS

After the error syndromes (anyon charges) on all the plaquettes are measured using the circuits discussed in the previous section, this syndrome information is passed to a decoder, which, in turn, outputs the recovery operations required to correct the errors. This section contains a detailed description of one of these decoding algorithms.

In this work we study two types of decoders. (A) The first type is the clustering decoder which has been previously applied to decode a phenomenological model of Fibonacci anyons [26]. This decoder is based on a hierarchical clustering algorithm [24] and shares a similar strategy to the hard-decision renormalization-group decoder [54]. The clustering decoder does not use the detailed syndrome information corresponding to the anyon type. Instead, it just uses the limited syndrome information of the presence of absence of anyon, i.e., whether the anyon charge is nontrivial or trivial (in the doubled vacuum sector **11**).

(B) The second type is a fusion-aware iterative minimum-weight perfect matching (MWPM) decoder which modifies the standard MWPM algorithm and incorporates the detailed syndrome information corresponding to anyon type into the decoding strategy. As a comparison, we also show the results of a "*blind*" iterative MWPM decoder which does not use the detailed syndrome information of

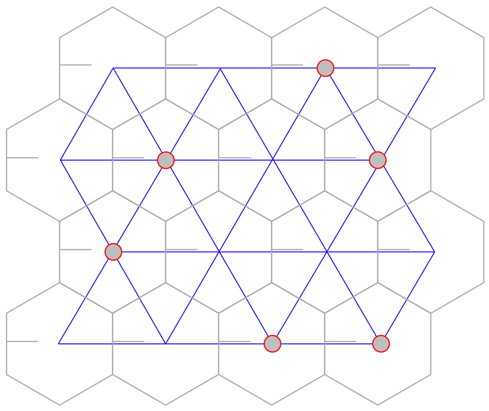


FIG. 16. The decoding graph of the extended string-net code. By connecting the center of the plaquettes of the tailed lattice, we obtain a triangular lattice (blue) as the decoding graph. The anyon charges (gray circles) on the plaquettes of the tailed lattice serve as the syndromes in the error correction scheme and are located on the vertices of the triangular decoding graph.

anyon type but only the presence or absence of an anyon. Because of the use of the detailed syndrome information, the logical error rate of the fusion-aware iterative MWPM decoder is lower than the other one.

Below, we give a detailed description of the clustering decoder. Both the blind and fusion aware MWPM decoders are discussed extensively in Supplemental Sec. V [45]. We indicate the syndrome using a decoding graph, shown in Fig. 16. This is a triangular lattice which is the dual of the original trivalent graph on which the extended string-net code is defined. Anyon charges located in the plaquettes of the trivalent graph, correspond to syndromes on vertices of this triangular decoding graph.

The spirit of the clustering decoder is based on the charge conservation of anyons. As discussed in Sec. V D, local noise can generate certain pairs or, more generally, clusters of anyons. Since these anyon clusters are generated from the vacuum sector **11**, i.e., the ground space of the extended string-net code, the total charge of each anyon cluster should still be trivial (**11**) due to charge conservation. Therefore, if we fuse all the anyons within a particular cluster by moving them toward the same plaquette, we get a total trivial vacuum charge **11**; i.e., all the anyons are annihilated back to the vacuum sector.

However, the decoder does not know which cluster each anyons are generated from based on the syndrome information. Therefore, the decoder may not always apply a correct recovery operation to fuse all the anyons originating from the same cluster. Instead, the decoder may fuse anyons from different clusters, effectively joining the two clusters. Because of the total charge conservation and the fusion rule $\mathbf{11}\times\mathbf{11}=\mathbf{11}$, we know the total anyon charge of the two clusters is still zero. If we fuse all the anyons in these two clusters together, all the charges are still annihilated into the vacuum **11**. The same argument applies to merging multiple clusters. As long as the size of the joined cluster is much smaller than the system size, all the errors can be corrected by merging them into the vacuum **11**. On the other hand, if the joined anyon cluster forms a noncontractible (homologically nontrivial) region, i.e., either wrapping around a cycle of a torus or, more generally, a high-genus surface in the context of a closed manifold or connecting two or more gapped boundaries in the context of an open manifold, the recovery operation with the merging procedure may still annihilate all the anyons but end up applying a nontrivial logical operator along a certain homologically nontrivial cycle, [55] which causes a logical error and the decoder fails. It is also possible that such a noncontractible anyon cluster has a nonzero total charge; therefore, there is some residual anyon after the merging procedure which cannot be annihilated. In both cases, we claim a failure of the clustering decoder. Therefore, in order to apply a successful recovery operation, we need to make sure that the individual clusters are annihilated before growing to a size comparable to the system size, i.e., with a linear dimension comparable to the code distance.

The clustering decoding algorithm for the Fibonacci Turaev-Viro code defined on a torus is summarized below by the pseudocode and is explained in detail with a concrete example:

Algorithm 1. Algorithm (clustering decoder).

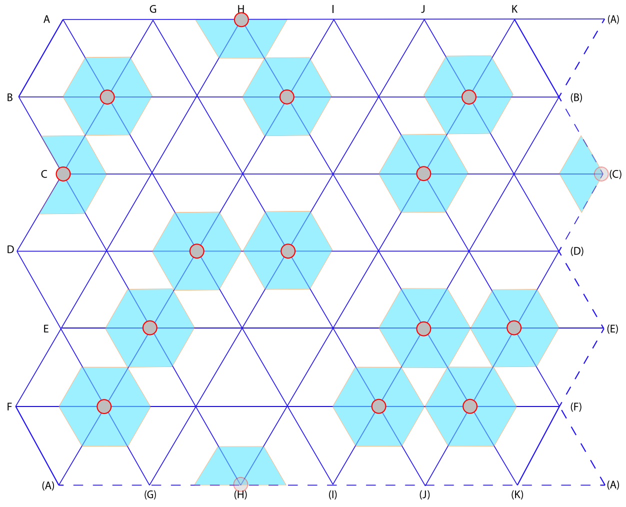
```
# Measure the anyon charge of each plaquette on the tailed trivalent
  lattice; store the nontrivial anyon charge in a list "anyon_charge"
anyon_charge = get_syndrome(state);

# Initialize a cluster for each plaquette with a nontrivial charge
clusters = Cluster(anyon_charge);

# Join any connected clusters
join(clusters, 1);

# While there is more than one nontrivial charge
while size(anyon_charge)>1

# Fuse all anyons within each cluster and measure the resulting
  charge
for cluster in clusters
    fuse_anyons(cluster);

# Check (in the classical simulation) whether any anyon path
  becomes noncontractible, i.e., homologically nontrivial after
  the fusion process
  if anyon_path == noncontractible
    # The decoder claims failure
    return Failure

# Discard any empty cluster with trivial vacuum charge 11;
clusters = non_vac(clusters);
```

(Table continued)

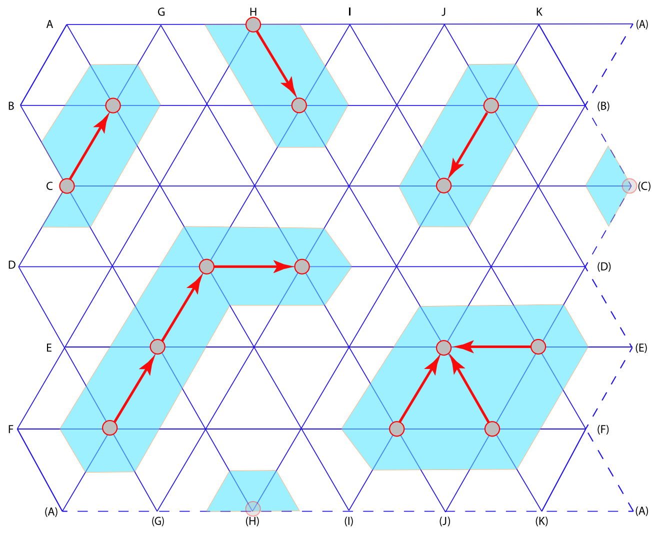
Algorithm 1. *(Continued)*

```
# Grow each cluster by a unit length on the triangular decoding
  graph
grow(clusters, 1);
# Join any overlapping clusters
join(clusters, 0);
end
# If the list of nontrivial anyon charge is empty, i.e., with no
  remaining anyon
if anyon_charge == []
  # The decoder declares success
  return Success
# If there is a single nontrivial remaining charge
else
  # The decoder claims failure
  return Failure
```

As described above, for a given state of the entire system, we first measure the tube operator within each plaquette of the tailed lattice using the circuit in Figs. 13 and 14 and obtain the corresponding anyon charge as the syndromes on the decoding graph. This process is defined as `get_syndrome()` in the decoding algorithm, with the `state` as the input and `anyon_charge` as the output. Now we initialize a cluster on each nonzero anyon charge, defined as `clusters=Cluster(anyon_charge)` in the above algorithm. An example for the decoding graph on a torus with the initial `clusters` (blue shadow) is shown below:

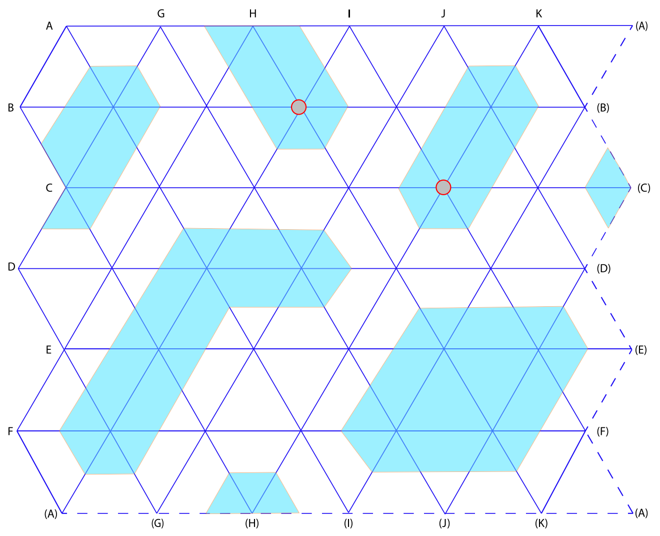


In the next step, we call `join(clusters, 1)` to join neighboring clusters which are separated by only one lattice spacing. Within each cluster, we randomly choose a root anyon, *move* all the other anyons to the root, and finally fuse all of them into a single anyon. The *move operation* is the recovery operation introduced in Supplemental Sec. II B and Fig. S6 [45] and can be simulated classically via modifying the curve diagram (see Sec. V F) as shown in Eqs. (SIV.10) and (SIV.11) in Supplemental Sec. IV F [45]. We note that the order of moving and detailed path do not affect the resulting state after fusing all the anyons within each cluster, and, therefore, we could choose an arbitrary order. The only requirement is that the chosen paths must be inside each cluster. For simplicity, we choose the shortest paths (with shortest graph distance) toward the root anyons from all the remaining anyons (indicated by the red arrows in the figure below) and start moving the anyons in an order with an increasing graph distance between the remaining and root anyons in our numerical simulation:

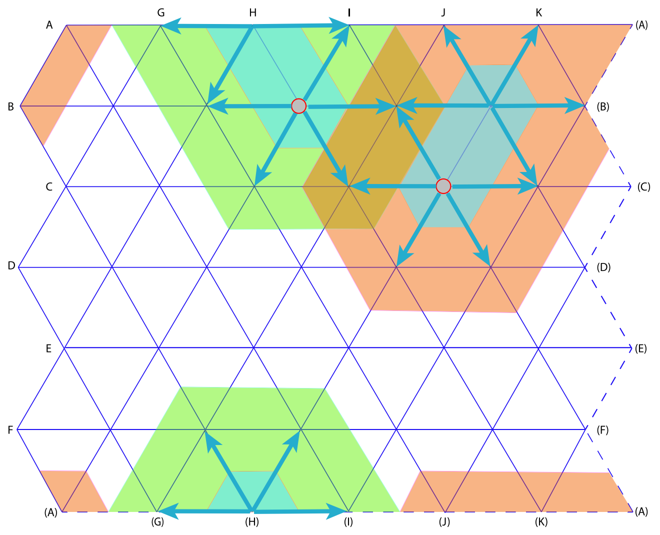


During the classical simulation described below in Sec. V, one needs to monitor whether the decoding process itself induces a logical error. Therefore, after each fusion process, we check (in the classical simulation) whether any anyon path l forms a noncontractible (homologically nontrivial) cycle, which gives rise to a logical error as previously illustrated in Fig. 8. The anyon path here is the sum of the error path and the recovery path: $l = l_e + l_r$. We note that, although the decoder itself does not have access to such information, the classical simulation does. We can, hence, claim failure of the decoder in our Monte Carlo simulation, if such noncontractible cycle occurs.

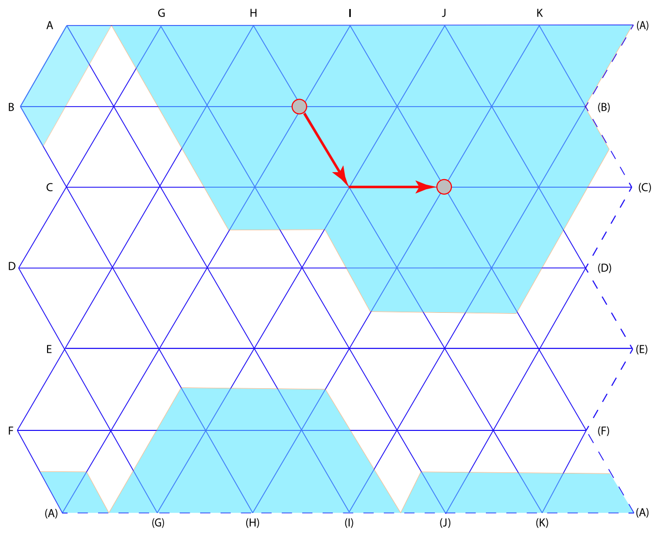
If the decoder does not fail, we continue to measure all the charges of the root anyons. If the measured charge in a particular cluster is zero, i.e., in the vacuum sector 11, we discard the corresponding cluster. The list of clusters, hence, gets updated via `clusters=non_vac(clusters)`:



In the next step, we need to call `grow(clusters, 1)` to grow each remaining cluster by one unit of lattice spacing in all possible directions (six directions for each vertex within the cluster), as indicated by the blue arrows in the figure below. As shown in the figure, the remaining two clusters (blue) grow to the larger clusters (green and orange):



Next, we call `join(clusters, 0)` to join overlapping clusters—i.e., any two clusters sharing a common set of vertices are joined into a single one—and this process is done iteratively until there are no overlapping clusters. In the current example, the green and red clusters in the above figure are joined into a single bigger cluster (blue) in the following figure:



After the merging of clusters, we repeat the above process—i.e., fuse all the anyons within each cluster (indicated by the red arrows in the above figure) and discard the empty cluster with trivial total charge 11 (vacuum)—and then further grow and join the remaining clusters. This iteration is stopped when we have zero or one remaining nontrivial anyon charge.

After the end of the above iteration, we are at the final stage of our decoding algorithm. If there is no any remaining nontrivial anyon charge, i.e., `anyon_charge == []`, we then have successfully corrected all the errors, and the decoder declares success. In the other situation with a single nontrivial remaining anyon, we end up with a logical error and claim failure of the decoder.

V. THRESHOLD SIMULATION

Besides formulating an error correction scheme for the extended Fibonacci string-net code, the main achievement of this work consists of obtaining error correction thresholds for this code with a microscopic noise model. This is achieved through Monte Carlo simulations, in which the quantum state of the system is updated to reflect the application of noise, measurement, and recovery operations until either the initial state is recovered successfully or a logical error occurs. Because of the complicated nature of the extended string-net code and the fact that our simulations require the ability to simulate the dynamics of non-Abelian anyons, this is a very nontrivial task. Below, we discuss the various technical details of these simulations. Note that the numerical simulations presented here are independent of the specific circuits used to realize the projective measurements and unitary transformations required during the error correction process (assuming these circuits can be carried out perfectly).

The structure of this section is as follows: We start by going over some comments on the classical simulation of non-Abelian quantum error correction, made in Refs. [22,23,26] in Secs. V A and V B. We then introduce a microscopic noise model of Pauli errors in Sec. V C and study the effect of individual qubit errors on anyonic fusion states in Secs. V D and V E. Section V F describes the framework of curve diagrams [56], which is used to efficiently characterize anyonic fusion basis bases during the simulation. Finally, in Sec. V G, we provide a detailed outline of all steps performed for a single Monte Carlo sample.

A. Simulating non-Abelian quantum error correction

The defining difference between Abelian and non-Abelian anyon models is the fact that the fusion outcome of a pair of anyons is no longer uniquely determined for non-Abelian models. This fact makes simulating noise, syndrome measurement, and error correction processes for a system exhibiting non-Abelian anyonic excitations considerably more complicated than for Abelian models such as the surface code.

After anyonic excitations are created through the application of noise, their locations and anyon labels can be determined through a syndrome measurement. If we think of the state in terms of fusion trees of anyons, the syndrome measurement projects only onto fixed values for the leaf labels. For non-Abelian anyons, the internal (branch) labels of the fusion tree may still be in a superposition of different configurations. One of the implications is that braiding processes occurring during the recovery phase do not necessarily result in a global phase. Hence, different paths used to physically approach a pair of anyons in order to fuse them might give rise to different fusion outcomes (more precisely, to different probability distributions for the measurement outcome of the total charge of the pair). When simulating the error correction process, we must therefore be very careful in specifying and keeping track of the precise paths followed by the various anyons.

Another implication is that, for non-Abelian models, the decoding process must happen in an iterative fashion. Based on the initial positions and charges of the anyons, a recovery step that consists of a number of fusion processes is suggested. As the result of this recovery cannot be predicted, all fusion processes must be performed in the given order, after which the fusion outcomes must be measured. These measured outcomes then give a new error syndrome that serves as the basis for suggesting a next recovery step, consisting of a new series of fusion processes. This cycle continues until all anyons are fused away and decoding is successful or until some anyon is wound along a nontrivial cycle during a recovery step resulting in a decoding failure. Error correction, thus, proceeds as a dialog between syndrome extraction and decoder, where the new syndrome resulting from a given recovery step is used to determine the next recovery step. This is in stark contrast to Abelian models, where a single syndrome measurement provides all the necessary information to determine all fusion processes that must be carried out in order to return the system to the code space.

B. Classical simulability

The ability to reliably simulate the general dynamics of Fibonacci anyons implies the ability to simulate (and, hence, perform) universal quantum computation. It is, therefore, highly unlikely that such simulations are feasible on a classical computer. However, typical noise and recovery processes such as those that we simulate in the remainder of this work exhibit structure that can be exploited to classically simulate them in regimes where we expect successful error correction to be possible.

Individual local error operations either create a distinct connected group of anyons with vacuum total charge or extend such an existing group (see Sec. V D). These groups can be understood as anyons that have interacted at some point during their lifetime. Since each disconnected group has a trivial total charge, braiding between separate groups is trivial. Hence, the total fusion space factorizes into a tensor product of fusion spaces of individual connected groups, and we are required only to simulate anyon dynamics within each of these groups separately.

With regards to the creation of connected groups of anyons, the noise process behaves as a kind of percolation process. Hence, below the percolation threshold, one expects that the size of the largest connected group scales as $O[\log(L)]$ [with variance $O(1)$], where L is the linear system size [57]. As this is a probabilistic statement, there are instances where the largest connected group has a size larger than $O[\log(L)]$, but the probability of such events is suppressed exponentially with the system size L. This logarithmic scaling of the largest cluster size $s = O[\log(L)]$ counters the exponential scaling of the dimension of the fusion space $d = O[\exp(s)]$ for individual connected groups. Therefore, the fusion spaces of individual connect groups have dimension $\dim = O[\mathrm{poly}(L)]$, meaning that the dynamics within connected groups can be simulated efficiently.

These arguments on the classical simulability of topological quantum error correction with a universal anyon model were first made in Ref. [26]. The behavior of connected clusters of anyons created in the phenomenological model studied there corresponds exactly to the bond percolation model for which the logarithmic scaling mentioned above is verified numerically [57]. To ensure that this still holds for the microscopic model studied here, we explicitly verify the logarithmic scaling of the average size of the largest connected group of anyons after subjecting all qubits to a depolarizing noise model, using Monte Carlo simulations. The results of these simulations are presented and discussed in Supplemental Sec. VII [45].

During the iterative decoding procedure, anyons are fused over increasing length scales, thereby potentially joining together connected groups of anyons through this interaction. As long as large connected groups are sparsely distributed (which is the case on average below the percolation threshold), this should not pose a problem, as the dimension of the fusion space is automatically reduced once the fusion process is actually performed, since this then reduces the number of anyons that must be simulated. We can conclude that we expect efficient simulation of the dynamics relevant to error correction to be possible in the regime where the combined action of noise and recovery does not percolate.

If noise is strong enough, connected groups of anyons percolate and the fusion space no longer factorizes into small disconnected parts. In this case, classical simulation of braiding and fusion becomes intractable. However, this is precisely the regime where we expect regular transport of anyons along nontrivial cycles during recovery, leading to failed error correction. Therefore, the error correction threshold itself lies below this regime, and estimating its value through classical simulations should be possible.

C. Noise model

Our goal is to stimulate the dynamics of a quantum-computing architecture of qubits, which directly implements our error correcting code. We model noise in this system as individual (either depolarizing or dephasing) Pauli errors acting on random qubits, while the system is constantly being monitored [58]. We treat the occurrence of Pauli errors on individual qubits as independent Poisson processes with a fixed rate p, which characterizes the noise strength. The total number of qubit errors, denoted by T, then follows a Poisson distribution with mean $T_0 = |E|p$, where $|E|$ is the total number of edges. (For a tailed honeycomb lattice with periodic boundary conditions and a total of L^2 plaquettes, the total number of edges is $|E| = 5L^2$, resulting in $T_0 = 5L^2 p$.) This fixed-rate sampling noise model is similar to those used for the simulation of non-Abelian quantum error correction with phenomenological models such as in Refs. [22,26]. For simplicity, we assume that all measurements are perfect and are carried out on timescales which are negligible compared to the average time between individual qubit errors [approximately $1/(|E|p)$].

We simulate the dynamics of the system for T time steps, each of which corresponds to the occurrence of a single Pauli qubit error and where T is drawn from a Poisson distribution with mean T_0. Each individual time step consists of the following.

(1) An edge e of the lattice is chosen at random, and a Pauli operator σ_i is picked according to relative probabilities $\{\gamma_x, \gamma_y, \gamma_z\}$. We specifically use depolarizing noise ($\gamma_x = \gamma_y = \gamma_z = 1/3$), dephasing noise ($\gamma_z = 1$), and bit-flip noise ($\gamma_x = 1$). The operator σ_i is then applied on the qubit corresponding to edge e.

(2) All vertex stabilizers Q_v and tail qubits are measured (in the Z basis).

(3) The appropriate unitary operator U_V, conditioned on the measurement outcome V from the measurements above, is applied to fix any violated vertices.

(4) The anyon charge in each plaquette is measured.

The unitary operators used to locally correct violated vertices are introduced in Sec. III; their purpose is to move the system back to the string-net subspace $\mathcal{H}_{\mathrm{sn}}$. Inside this subspace, states can be described in terms of anyonic fusion states, and plaquette anyon charges are well defined. Note that the measurement at the end of each time step means that we never encounter any superpositions of different anyonic charges in individual plaquettes. (In terms of fusion states, this fixes all leaf labels.)

After the T error operations are applied, and if no logical error is induced by the noise process, the syndrome is fed to an iterative (classical) decoding algorithm, which returns a list of anyons to be fused and paths to be followed when doing so. We assume that all recovery operations and additional syndrome measurements are perfect and instantaneous, meaning no additional errors happen during the recovery process.

While the connection between percolation and logical errors in topological codes is not exact [59] (i.e., not all percolation events cause logical errors), all logical errors are the result of events where a pair of anyons are fused along a noncontractible loop. Our Monte Carlo simulations (detailed below in Sec. V G) classify all such percolation events as failures and, therefore, slightly overestimate the logical failure rates. This provides a heuristic argument for the validity of our noise model. The immediate collapse of superpositions in the anyon charge of individual plaquettes does not inhibit the occurrence of percolation processes. We, therefore, do not expect our assumption of constant syndrome measurement to significantly affect the obtained decoder failure rates. Furthermore, in the low noise strength limit, our noise model approximates an IID noise model, as the random qubits affected by Pauli errors are unlikely to be in each other's vicinity. The existence of an error correction threshold for our fixed-rate sampling noise model, therefore, implies the existence of a threshold for an IID noise model as well.

D. Pauli noise in the anyonic fusion basis

We now investigate the effect of the application of noise and subsequent measurement and vertex recovery operations performed in each of the T time steps of the simulation as described above. Because of the syndrome measurement performed at the end of every time step, the quantum state of the system between these steps can always be decomposed as a superposition of anyonic fusion basis states, which all share the same set of leaf labels. It is, therefore, sufficient to understand the action of these different operations on an initial state

$$|\Psi_0\rangle = \sum_t \alpha_t |\psi_t\rangle, \tag{64}$$

where the $|\psi_t\rangle$ represent anyonic fusion basis states Eq. (38) and the states $\{|\psi_t\rangle | \alpha_t \neq 0\}$ all share the same set of leaf labels.

1. Pauli-Z errors

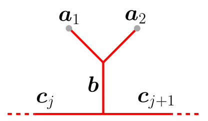
We begin our analysis by studying the case where we act with a σ_z operator on the qubit residing at edge e. One can easily see that, for any edge e, σ_z^e commutes with all vertex operators Q_v defined in Eq. (9), since both operators are diagonal in the same basis. Hence, a σ_z error does not take the system out of the string-net subspace $\mathcal{H}_{\mathrm{sn}}$, and the resulting state can again be decomposed in the anyonic fusion basis:

$$\begin{aligned}|\Psi_1\rangle = \sigma_z^e|\Psi_0\rangle &= \sum_t \alpha_t \sigma_z^e |\psi_t\rangle \\ &= \sum_{t'} |\psi_{t'}\rangle \left(\sum_t \alpha_t \langle \psi_{t'}|\sigma_z^e|\psi_t\rangle\right),\end{aligned} \tag{65}$$

where we use that $\sum_{t'} |\psi_{t'}\rangle\langle\psi_{t'}|$ acts as the resolution of the identity within $\mathcal{H}_{\mathrm{sn}}$. In particular, if we start from a superposition of basis states $|\psi_t\rangle$ that all have the same specific handle label (see Sec. II E), the states $|\psi_{t'}\rangle$ in the resulting superposition also have that same label, unless a logical error occurs through the interaction of the thermal anyons due to the noise operator. Since the simulation of error correction is immediately aborted in the event of such a logical error, it is sufficient to consider only basis states that all have the same handle label as the initial state.

When the edge e is a tail edge, acting with σ_z^e results in a global (± 1) factor, which has no physical consequences. On a general edge e different from a tail edge, the action of σ_z^e commutes with the irreducible idempotents of the tube algebra [Eqs. (23)–(25), (27), and (28)] acting on any plaquette, except for the two that contain the edge e. This means that a σ_z^e error on a general edge can modify the anyon charge of at most two plaquettes. Furthermore, the total charge of these two plaquettes cannot be changed by the action of σ_z^e, since this is a collective property of the pair that is insensitive to the local operator σ_e^z.

With these insights in mind, we pick an anyonic fusion basis of the following form:

$$|\psi^{\vec{a}}_{b,\vec{c}}\rangle = \tag{66}$$


,

where a_1 and a_2 are the charges of the two affected plaquettes, b denotes their total charge, and $\vec{c}$ collectively denotes all other leaf, branch, and handle labels. We can then rewrite the initial state Eq. (64) as

$$|\Psi_0\rangle = \sum_{b,\vec{c}} \alpha_{b,\vec{c}} |\psi^{\vec{a}}_{b,\vec{c}}\rangle. \tag{67}$$

Note that we drop the summation over $\vec{a}$, which is allowed because the initial state contains no superposition in plaquette charges. Since the labels b and $\vec{c}$ are unaffected by σ_z^e, the matrix elements appearing on the right-hand side in Eq. (65) are block diagonal in these labels, and the expression for the state $|\Psi_1\rangle = \sigma_z|\Psi_0\rangle$ can be reduced to

$$|\Psi_1\rangle = \sum_{b,\vec{c}} \sum_{\vec{a}'} |\psi^{\vec{a}'}_{b,\vec{c}}\rangle (\alpha_{b,\vec{c}} \langle \psi^{\vec{a}'}_{b,\vec{c}}|\sigma_z^e|\psi^{\vec{a}}_{b,\vec{c}}\rangle). \tag{68}$$

The probability of finding outcome $\vec{a}' = (a_1', a_2')$ when performing a syndrome measurement in the two affected plaquettes is then given by

$$p(\vec{a}') = \sum_{b,\vec{c}} |\alpha_{b,\vec{c}} \langle \psi^{\vec{a}'}_{b,\vec{c}}|\sigma_z^e|\psi^{\vec{a}}_{b,\vec{c}}\rangle|^2. \tag{69}$$

After performing this measurement, the state of the system collapses to

$$|\Psi_2\rangle = \frac{1}{\sqrt{p(\vec{a}')}} \sum_{b,\vec{c}} |\psi^{\vec{a}'}_{b,\vec{c}}\rangle (\alpha_{b,\vec{c}} \langle \psi^{\vec{a}'}_{b,\vec{c}}|\sigma_z^e|\psi^{\vec{a}}_{b,\vec{c}}\rangle), \tag{70}$$

which is again a superposition of basis states with fixed leaf labels. Note that the matrix elements on the right-hand side in Eqs. (69) and (70) are independent of the unaffected labels $\vec{c}$. Hence, we may replace them with matrix elements of the form $\langle \psi^{\vec{a}'}_b|\sigma_z^e|\psi^{\vec{a}}_b\rangle$, where $|\psi^{\vec{a}}_b\rangle$ are states that contain only nontrivial anyon labels for the two affected plaquettes and for their total charge. Also note that, since the total charge of the affected plaquettes is a collective property, the precise location of the third plaquette which contains this total charge does not affect the value of the matrix elements, and, furthermore, the tail label of that plaquette does not affect the matrix elements.

When calculating the matrix elements, we must pick some convention for the basis elements $|\psi^{\vec{a}}_b\rangle$ [i.e., for the embedding of the fusion tree in Eq. (66) in the lattice], for each of the possible orientations of e. Since σ_z^e has no physical effect when e is a tail edge, we need to consider only four orientations for e. Our choice of basis convention for the σ_e^z matrix elements is given in Fig. 17.

2. Pauli-X and Y errors

The case of a σ_x^e error is similar but comes with some additional complications. Since a σ_x^e operator does not commute with the vertex operators Q_v associated to the vertices bounding e, the state

$$|\Psi_1\rangle = \sum_t \alpha_t \sigma_x^e |\psi_t\rangle \tag{71}$$

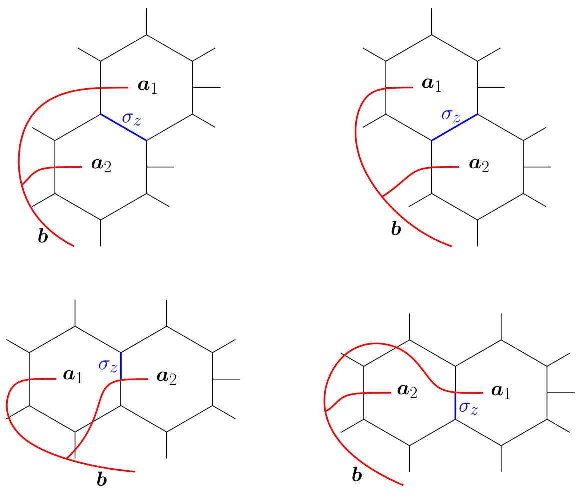
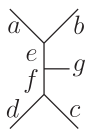


FIG. 17. Basis convention for the affected plaquette charges for all nontrivial orientations of the edge e (highlighted in blue) in the case of a σ_z error.

does not necessarily belong to the string-net subspace $\mathcal{H}_{\rm sn}$ and, hence, cannot be expressed as a superposition of anyonic fusion basis states. In particular, upon measuring the vertex stabilizers for the vertices that bound e, one might find that the ribbon graph branching rules are violated in either of these vertices. Each violated vertex belongs to a set of seven connected qubits on the lattice that we call a *segment*:



The three vertices in each segment are denoted as t, m, and b, corresponding to the top, middle, and bottom vertex, respectively. For each segment with some violated vertices, we also measure the label of the tail qubit (corresponding to edge g in the diagram above). Depending on these measurement outcomes V, a unitary operator U_V is applied to the segment in order to take it back to the $+1$ eigenstate of the three vertex operators Q_t, Q_m, and Q_b associated to the segment. The definitions and corresponding circuits for these unitaries (conditioned on all possible measurement outcomes) are given in Sec. III A.

The probability $p(V)$ of a combined outcome V for the vertex and tail qubit measurements in the relevant segments is given by

$$p(V) = \langle\Psi_1|P_V|\Psi_1\rangle = \sum_{t',t}(\alpha_{t'})^*\alpha_t\langle\psi_{t'}|\sigma_x^e P_V\sigma_x^e|\psi_t\rangle, \tag{72}$$

where P_V is the projector onto the total measurement outcome V. For example, for the case of a σ_x^e acting on the edge e bounded by vertices v_1 and v_2 where only v_1 is violated and the tail qubit q_1 of the segment containing v_1 has label τ, the associated projector P_V is given by

$$P_V = (1-Q_{v_1})(Q_{v_2})(|\tau\rangle_{q_1}\langle\tau|_{q_1}). \tag{73}$$

After performing the vertex and tail qubit measurements with outcome V and applying the appropriate unitary operator U_V to bring the system back to the string-net subspace, the state is given by

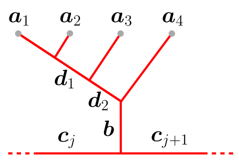
$$|\Psi_2\rangle = \frac{1}{\sqrt{p(V)}}\sum_t \alpha_t U_V P_V\sigma_x^e|\psi_t\rangle = \frac{1}{\sqrt{p(V)}}\sum_{t'}|\psi_{t'}\rangle\left(\sum_t\alpha_t\langle\psi_{t'}|U_V P_V\sigma_x^e|\psi_t\rangle\right), \tag{74}$$

where we again insert the resolution of the identity $\sum_{t'}|\psi_{t'}\rangle\langle\psi_{t'}|$ in $\mathcal{H}_{\rm sn}$ on the right-hand side in order to explicitly express the state as a superposition of anyonic fusion basis states.

As in the case of a σ_e^z error, the expressions (72) and (74) can be simplified by considering which plaquette charges are actually affected by the combined action of the operators U_V, P_V, and σ_x^e. Since a σ_x operator acting on an edge e of the tailed lattice results in at most two violated vertices, the combined action of U_V, P_V, and σ_x^e involves at most two segments. These operators, therefore, commute with any tube operators acting on plaquettes that have no edges in common with these segments. This means that only the charges associated to the plaquettes in the immediate neighborhood of the error can be affected. The number of affected plaquettes depends on the orientation of the edge e.

As before, we pick our anyonic fusion basis to reflect this fact. In the case of four affected plaquettes, the basis states have the form

$$|\psi_{\boldsymbol{b},\vec{\boldsymbol{c}}}^{\vec{\boldsymbol{a}},\vec{\boldsymbol{d}}}\rangle = \tag{75}$$



,

where, again, $\vec{\boldsymbol{a}}$ denotes the charges of the affected plaquettes, $\boldsymbol{b}$ denotes their total charge, and $\vec{\boldsymbol{c}}$ collectively denotes all other unaffected leaf, branch, and handle labels. Note that we now need additional labels $\vec{\boldsymbol{d}}$ to denote the affected internal branch labels. With this choice of basis, the sum over t in Eq. (64) is split into a sum over the unaffected labels $\vec{\boldsymbol{c}}$, the total charge $\boldsymbol{b}$, and the branch labels $\vec{\boldsymbol{d}}$ of the affected part:

$$|\Psi_0\rangle = \sum_{\boldsymbol{b},\vec{\boldsymbol{c}},\vec{\boldsymbol{d}}}\alpha_{\boldsymbol{b},\vec{\boldsymbol{c}}}^{\vec{\boldsymbol{d}}}|\psi_{\boldsymbol{b},\vec{\boldsymbol{c}}}^{\vec{\boldsymbol{a}},\vec{\boldsymbol{d}}}\rangle. \tag{76}$$

As all matrix elements are block diagonal in the unaffected labels, the expression for the vertex measurement outcome probability Eq. (72) reduces to

$$p(V) = \sum_{\boldsymbol{b},\vec{c},\vec{d}} \sum_{\vec{d}'} (\alpha^{\vec{d}'}_{\boldsymbol{b},\vec{c}})^* \alpha^{\vec{d}}_{\boldsymbol{b},\vec{c}} \langle \psi^{\vec{a},\vec{d}'}_{\boldsymbol{b},\vec{c}} | \sigma^e_x P_V \sigma^e_x | \psi^{\vec{a},\vec{d}}_{\boldsymbol{b},\vec{c}} \rangle. \quad (77)$$

The sum over t' in Eq. (74) can again be split into a sum over the unaffected labels $\vec{c}'$, the affected leaf labels $\vec{a}'$, and the branch labels $\vec{d}'$ of the affected part, reducing the expression to

$$|\Psi_2\rangle = \frac{1}{\sqrt{p(V)}} \sum_{\vec{a}',\vec{d}'} \sum_{\boldsymbol{b},\vec{c}} |\psi^{\vec{a}',\vec{d}'}_{\boldsymbol{b},\vec{c}}\rangle \left(\sum_{\vec{d}} \alpha^{\vec{d}}_{\boldsymbol{b},\vec{c}} \langle \psi^{\vec{a}',\vec{d}'}_{\boldsymbol{b},\vec{c}} | U_V P_V \sigma^e_x | \psi^{\vec{a},\vec{d}}_{\boldsymbol{b},\vec{c}} \rangle \right). \quad (78)$$

Measurement of the affected plaquette charges then yields charges $\vec{a}'$ with a probability $p(\vec{a}')$ given by

$$p(\vec{a}') = \frac{1}{p(V)} \sum_{\vec{d}'} \sum_{\boldsymbol{b},\vec{c}} \left| \sum_{\vec{d}} \alpha^{\vec{d}}_{\boldsymbol{b},\vec{c}} \langle \psi^{\vec{a}',\vec{d}'}_{\boldsymbol{b},\vec{c}} | U_V P_V \sigma^e_x | \psi^{\vec{a},\vec{d}}_{\boldsymbol{b},\vec{c}} \rangle \right|^2. \quad (79)$$

The resulting state after this charge measurement is

$$|\Psi_3\rangle = \frac{1}{\sqrt{p(\vec{a}')p(V)}} \sum_{\vec{d}'} \sum_{\boldsymbol{b},\vec{c}} |\psi^{\vec{a}',\vec{d}'}_{\boldsymbol{b},\vec{c}}\rangle \left(\sum_{\vec{d}} \alpha^{\vec{d}}_{\boldsymbol{b},\vec{c}} \langle \psi^{\vec{a}',\vec{d}'}_{\boldsymbol{b},\vec{c}} | U_V P_V \sigma^e_x | \psi^{\vec{a},\vec{d}}_{\boldsymbol{b},\vec{c}} \rangle \right). \quad (80)$$

Once again, the matrix elements on the right-hand side in Eqs. (77), (79), and (80) are independent of the unaffected labels $\vec{c}$, so we require only matrix elements of the form $\langle \psi^{\vec{a}',\vec{d}'}_{\boldsymbol{b}} | O | \psi^{\vec{a},\vec{d}}_{\vec{b}} \rangle$ for fusion basis states involving up to four plaquettes and their total charge.

Our choice of basis convention for the charges of the affected plaquettes for all possible orientations of e is given in Fig. 18. In the case of a σ_x error, all five possible orientations of e are nontrivial. By considering the potentially violated vertices and the definition of the associated unitaries given in Sec. III A for each orientation, it can easily be verified that the depicted plaquettes are indeed the only affected ones and the combined action of the error, measurements, and unitaries commutes with all other tube operators.

The case of a σ_y error is entirely analogous to that of a σ_x. The same operators P_V and U_V appear, and we use the same basis convention as the one depicted in Fig. 18.

In summary, all that is required to capture the effect of Pauli noise on the state of the system are the following matrix elements:

$$\langle \psi^{\vec{a},\vec{d}'}_{\boldsymbol{b}} | \sigma^e_x P_V \sigma^e_x | \psi^{\vec{a},\vec{d}}_{\vec{b}} \rangle, \quad (81)$$

$$\langle \psi^{\vec{a}',\vec{d}'}_{\boldsymbol{b}} | U_V P_V \sigma^e_x | \psi^{\vec{a},\vec{d}}_{\vec{b}} \rangle, \quad (82)$$

$$\langle \psi^{\vec{a},\vec{d}'}_{\boldsymbol{b}} | \sigma^e_y P_V \sigma^e_y | \psi^{\vec{a},\vec{d}}_{\vec{b}} \rangle, \quad (83)$$

$$\langle \psi^{\vec{a}',\vec{d}'}_{\boldsymbol{b}} | U_V P_V \sigma^e_y | \psi^{\vec{a},\vec{d}}_{\vec{b}} \rangle, \quad (84)$$

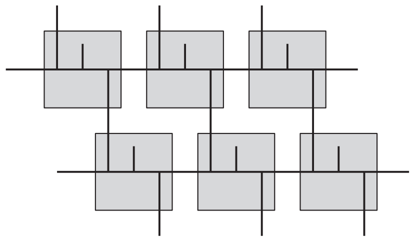
$$\langle \psi^{\vec{a}'}_{\boldsymbol{b}} | \sigma^e_z | \psi^{\vec{a}}_{\boldsymbol{b}} \rangle. \quad (85)$$

These matrix elements must be calculated for all possible orientations of the edge e in Figs. 17 and 18 and for all possible combined outcomes V of the vertex and tail qubit measurements in the case of a σ_x or σ_y error.

E. Computing the matrix elements

At first sight, calculating the matrix elements listed in Eqs. (81)–(85) seems like an intractable job, due to the highly entangled nature of the anyonic fusion basis states on the tailed lattice. Fortunately, as detailed in Supplemental Sec. III [45], this complex entanglement structure can be captured using tensor network representations for anyonic fusion basis states. By using some key insights together with tensor network techniques, we can harness the computational power of tensor networks to compute said matrix elements. Below, we illustrate this procedure for the matrix element Eq. (81), where e is an edge with the orientation depicted in Fig. 18(a). All other matrix elements can be computed in an analogous manner.

For simplicity, we work with square PEPS tensors (see Supplemental Sec. III C [45]), which each correspond to one segment of the lattice as shown in the following diagram:



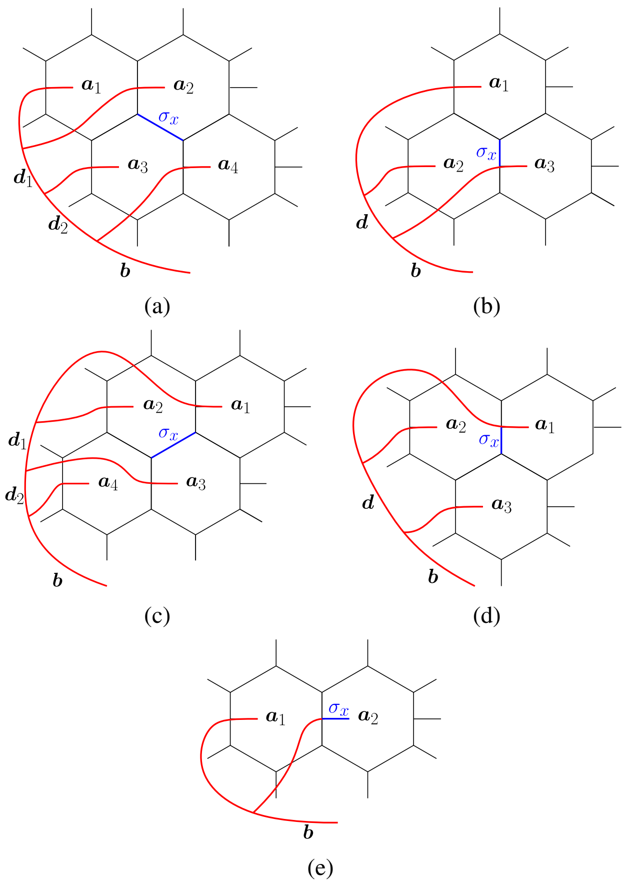
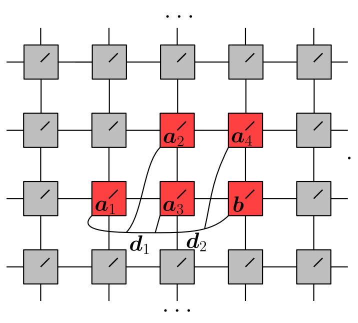


FIG. 18. Basis convention for the affected plaquette charges for the different orientations of edge e (highlighted in blue) in the case of a σ_x or a σ_y error. Note that, depending on the orientation, the charge in either four (a,c), three (b,d), or two (e) plaquettes may be affected by the error.

where we rotate the lattice by 60° counterclockwise. The PEPS tensors are colored gray and red for segments of which the tails end inside plaquettes containing trivial and nontrivial DFIB charges, respectively. With this notation (and after a counterclockwise rotation by 60°), the anyonic fusion state represented in Fig. 18(a) looks as follows:

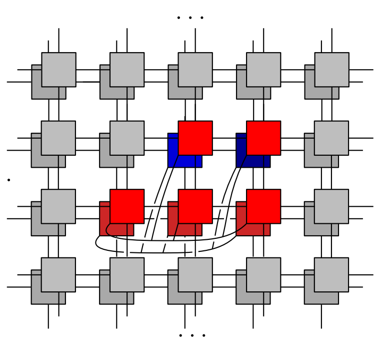
$$|\psi_{\boldsymbol{b}}^{\vec{\boldsymbol{a}},\vec{\boldsymbol{d}}}\rangle = \cdots \qquad\qquad \cdots . \tag{86}$$



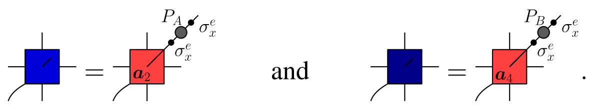
The crossings and branchings in this diagram imply the presence of certain crossing and fusion tensors. Details of these are given in Supplemental Sec. III [45]. The total charge $\boldsymbol{b}$ of the four leaf charges in Fig. 18(a) is assigned to a neighboring segment. As mentioned before, since the total charge of the group of anyons is a collective property, the precise location of the excitation tensor encoding this total charge does not affect the computation of the matrix elements itself. Hence, we choose to place it next to the other charges for convenience.

The matrix elements can then be computed by applying the appropriate operators on the physical indices and then contracting the result with the conjugate PEPS corresponding to the bra vector $\langle\psi_{\vec{b}}^{\vec{a}',\vec{d}'}|$. In this specific case, the operator P_V that projects out the combined vertex and tail qubit measurement outcome V can be decomposed into two operators P_A and P_B that act on the segments of leaf charges $\boldsymbol{a}_2$ and $\boldsymbol{a}_4$ and project out the measurement outcomes for the vertices and tail edge in these segments, respectively. The matrix element is then given by the contraction

$$\langle\psi_{\boldsymbol{b}}^{\vec{\boldsymbol{a}}',\vec{\boldsymbol{d}}'}|\sigma_x^e P_V \sigma_x^e|\psi_{\boldsymbol{b}}^{\vec{\boldsymbol{a}},\vec{\boldsymbol{d}}}\rangle = \cdots \qquad\qquad \cdots , \tag{87}$$



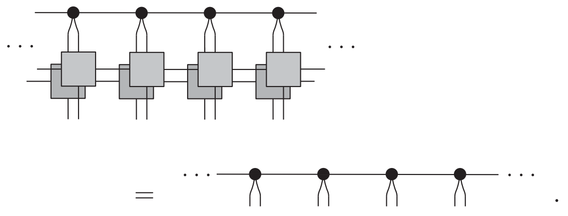
where



(88)

In order to avoid too much clutter, we omit the different labels in the graphical notation. The labels $\vec{a}'$, $\boldsymbol{b}$, $\vec{d}'$ and $\vec{a}$, $\boldsymbol{b}$, $\vec{d}$ are always implied for the top and bottom layer, respectively. Note that the σ_x^e operator is applied on two different tensors. This is nothing more than a side effect of the fact that the physical degrees of freedom are doubled in the PEPS representation.

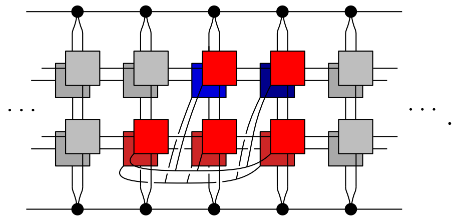
The result of the tensor contraction in Eq. (87) should be independent of the size of the system and must be entirely determined by the nontrivial anyonic charges of the colored segment tensors. We may, therefore, assume the system to be infinite, where all tensors except the colored ones depicted in Eq. (87) correspond to segments with a trivial charge. This infinite contraction can then be simplified by determining the top fixed point matrix product state of the double layer transfer matrix [60]:



(89)

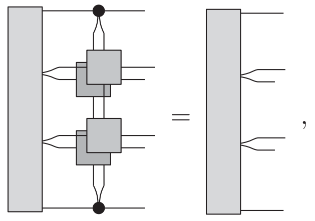
After finding a similar bottom fixed point matrix product state, Eq. (87) can be reduced to

$$\langle\psi_{\boldsymbol{b}}^{\vec{a}',\vec{d}'}|\sigma_x^e P_V \sigma_x^e|\psi_{\boldsymbol{b}}^{\vec{a},\vec{d}}\rangle =$$



(90)

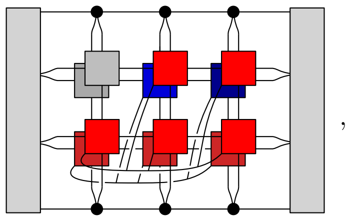
In the same way, left and right fixed points can be determined for this expression:



(91)

and similarly for the right fixed point. This finally results in the finite contraction

$$\langle\psi_{\boldsymbol{b}}^{\vec{a}',\vec{d}'}|\sigma_x^e P_V \sigma_x^e|\psi_{\boldsymbol{b}}^{\vec{a},\vec{d}}\rangle =$$



(92)

which can be computed in the regular way. In Eqs. (89) and (91), we implicitly assume that the PEPS tensors are rescaled such that the leading eigenvalue of the relevant transfer matrices in these expressions is equal to 1. Because of these rescalings, the PEPS for the fusion basis states will not have a unit norm, and the final expression Eq. (92) should, therefore, be divided by $\sqrt{\langle\psi_{\boldsymbol{b}}^{\vec{a},\vec{d}}|\psi_{\boldsymbol{b}}^{\vec{a},\vec{d}}\rangle\langle\psi_{\boldsymbol{b}}^{\vec{a}',\vec{d}'}|\psi_{\boldsymbol{b}}^{\vec{a}',\vec{d}'}\rangle}$ in order to obtain a matrix element that is properly normalized.

F. Storing and manipulating anyonic fusion basis states

As explained in Sec. II E, states in $\mathcal{H}_{\mathrm{sn}}$ can be expressed as linear combinations of anyonic fusion basis states. The anyonic fusion basis itself is determined by picking a pants decomposition of the surface and choosing a basis for each handle if the genus is nonzero. To keep track of the quantum state $|\Psi\rangle$ of the system, one could, in principle, pick a basis $\{|\psi_i\rangle\}$ for the entire string-net subspace and then update all coefficients $\langle\psi_i|\Psi\rangle$ throughout the different steps in the simulation. Such a naive approach is doomed to fail, however, as implementing F moves and braiding in the exponentially large Hilbert space $\mathcal{H}_{\mathrm{sn}}$ quickly becomes intractable as the system size grows. Instead, we need to select a basis that reflects the factorization of the fusion space discussed in Sec. V B. Hence, we require the ability to dynamically introduce a basis for each of the connected groups of anyons separately, in a way that takes into account the specific structure of the relevant noise processes. The framework of *curve diagrams* [26,56] provides exactly that. To improve readability, only a very concise

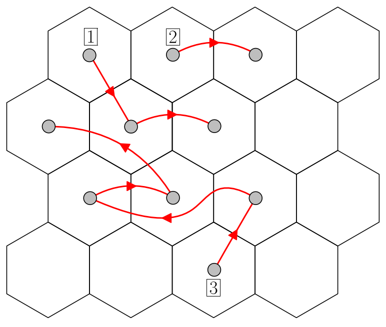
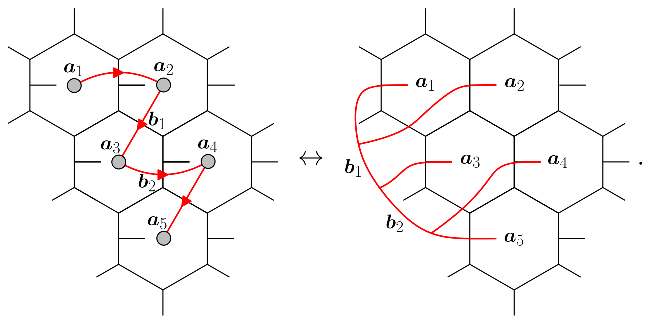


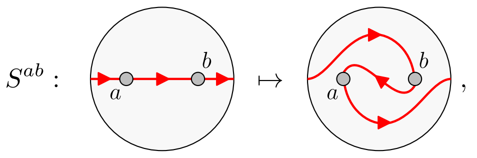
FIG. 19. Example of a configuration of three connected curves on a 4×4 periodic hexagonal lattice.

introduction is given here. A more complete discussion, including technical details of how curve diagrams are manipulated in the numerical simulations, is given in Supplemental Sec. IV [45].

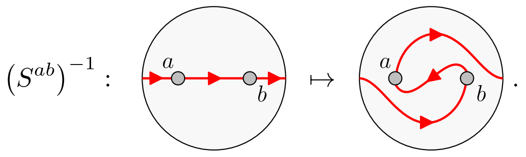
A curve diagram is a set of (nonintersecting) directed curves, each connecting the members of a single connected group of anyons residing at the center of their respective plaquettes. An example of a curve diagram is depicted in Fig. 19. Each curve can be related (up to local Dehn twists in individual plaquettes) to an anyonic fusion basis as depicted below:



During the simulation, the curve diagram representing the anyonic fusion basis in which the state vector is expressed must be updated regularly. Specifically, when neighboring anyons interact (either by an error process or when they are being fused by the decoder), a basis change must be performed in order for these anyons to appear sequentially in their joint fusion tree (in such a way that the corresponding curve follows the shortest path between them; see Supplemental Sec. IV D [45] for details). Such a basis transformation can always be expressed as a combination of two types of actions: *merges*, where two disconnected curves are joined (by connecting the end of one with the beginning of the other), and *swaps*, which are passive exchanges of the form



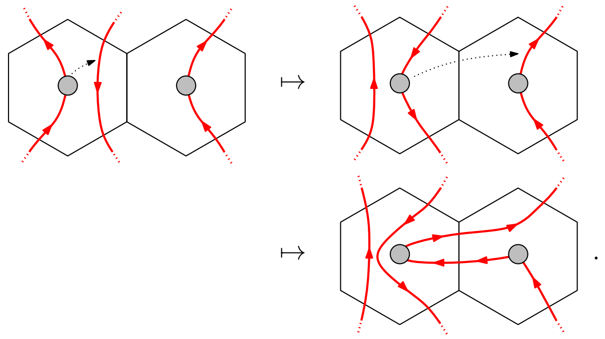
(93)



(94)

The corresponding transformations on the state vector coefficients to reflect this basis change are given in Supplemental Sec. IV C [45].

Since interacting anyons are always neighbors on the lattice, a curve diagram where they appear sequentially on the same curve can be obtained by repeatedly moving one of them (or, rather, its position on the curve) to the next piece of curve encountered while going the boundary of the two tiles in a clockwise fashion (merging the two curves first if they are different), as illustrated below:



We refer to such operations as *refactoring moves*. Each refactoring move is equivalent to a sequence of swaps, which one can determine using the *paperclip algorithm* [26] (see Supplemental Sec. IV D [45] for details).

G. Outline of the simulation

With all the groundwork completed, we are now ready to sketch the outline of the entire error correction threshold simulation, performed with the Fibonacci input category on a tailed hexagonal lattice with periodic boundary conditions in both directions (giving the lattice the topology of a torus). The simulation consists of a fixed number of Monte Carlo samples, each of which simulates the application of noise and recovery processes to an initial ground state. The quantum state of the system is tracked throughout these processes until either a topologically nontrivial process occurs, in which case *failure* is declared, or all anyonic excitations are removed (without any logical errors), in which case *success* is declared.

For a given a system size and noise strength, the logical failure rate P_L is then found by the ratio of failures compared to the total number of Monte Carlo steps. Below, we describe in detail all the important aspects of a single Monte Carlo step, with system size $L \times L$ and a noise strength characterized by p.

1. Cutoff parameters

In addition to logical errors, failure is also declared whenever any of two cutoff parameters are exceeded. The first of these is the maximal allowed tree size $\mathtt{N}_{\max}$. As discussed in Sec. V B, the size of connected groups of anyons can grow very large in some rare occasions. Since the size of the associated fusion space grows exponentially, such situations are extremely costly, both in terms of memory usage and in computation time. Hence, we fix some cutoff size $\mathtt{N}_{\max}$, the simulation is aborted, and a failure is declared, whenever the number of anyons on an individual curve exceeds this value. Note that a similar cutoff is used in Ref. [26].

The second cutoff parameter that we introduce is $\mathtt{V}_{\max}$, which is the maximal number of nonzero coefficients we allow in the vector associated to any individual curve. The motivation behind this cutoff rule is as follows. Since the state vectors appearing during the simulation are generally very sparse, these are stored as sparse arrays. This enables us the keep the value of $\mathtt{N}_{\max}$ higher than one would naively expect (as no memory is allocated to all zero entries, which form the vast majority of the exponentially large state vector). To avoid the extreme time and memory cost of the rare cases where any state vector contains a very large number of nonzero entries, such cases are aborted and a failure is declared.

One must choose these cutoff parameters to be as high as possible, in order to minimize the amount of triggered cutoffs, while still keeping the memory and time cost of the simulations reasonable. Of course, any finite values of these parameters negatively affect the observed logical failure rates, once we leave the regime in which events with very large connected groups are sufficiently rare. However, we argue that their influence can only *lower* the obtained threshold, meaning our results provide a valid lower bound on the actual error correction threshold, independent of the values of $\mathtt{N}_{\max}$ and $\mathtt{V}_{\max}$. For a fixed value of p, larger system sizes (on average) result in higher values for the size of the largest connected group of anyons (see Supplemental Sec. VII [45]). Hence, it is clear that the cutoffs are triggered more often for larger system sizes. Likewise, for a fixed system size L, larger values of p also result in more triggered cutoffs. When displaying the logical failure rate as a function of p, the intersection of the curves corresponding to different system sizes is shifted left compared to its true value (had the cutoff parameters been infinite), meaning that our obtained error correction threshold is indeed a valid lower bound to its unknown true value.

2. Noise phase

The system is initialized in a ground state of the code Hamiltonian Eq. (8), corresponding to the anyonic vacuum with some specific handle labels. However, as explained in Supplemental Sec. IV [45], there is no need to store these handle labels, since they do not affect any relevant processes. Processes in which the handle labels do affect the outcome are precisely those that result in a logical error. Because the Monte Carlo simulation automatically declares a failure in those cases, such processes must never be simulated on the level of state evolution.

The first half of the simulation consists of sequentially applying T noise processes, described in Sec. V C, to this initial ground state, where T is drawn from a Poisson distribution with mean $p5L^2$. For each of these T steps, an edge e is chosen at random, and a Pauli operator σ_i is selected according to the relative probabilities $\{\gamma_x, \gamma_y, \gamma_z\}$. Before the matrix elements computed in Sec. V E can be used to determine the state after the application of error σ_i^e, one must first rewrite the state vector in the appropriate basis.

Given the orientation of e and the type of error σ_i, the affected plaquettes can be read off from Figs. 17 or 18. If none of them contain a nontrivial charge initially, we simply create a new curve diagram with the appropriate shape and with a corresponding trivial fusion state (containing only vacuum charges). If only some of these plaquettes contain a nontrivial charge, we can add vacuum charges to the fusion state of one of the nontrivial charges and modify ("grow") the curve accordingly such that all affected plaquettes are included in the same curve.

The desired basis is the one where the affected anyons appear sequentially along the same curve, in the order depicted in Figs. 17 or 18. This is obtained by applying a series of refactoring moves and merges as described in Sec. V F and performing the corresponding sequences of swaps and merge operations on the affected state vectors. If any of the required refactoring moves is topologically impossible, for instance, the one depicted in Fig. 20, this indicates that a logical error is caused, as the joint path of the curve diagram and the interaction path form a noncontractible loop. Whenever this happens, the simulation declares failure and aborts the current Monte Carlo step.

If all refactoring moves above are legal, a sequence of F moves is performed to isolate the affected anyons from the rest of the fusion tree. This transforms the standard fusion tree shape to the one in Eq. (66), Eq. (75), or a similar shape in the case of three affected plaquettes. In terms of ribbons

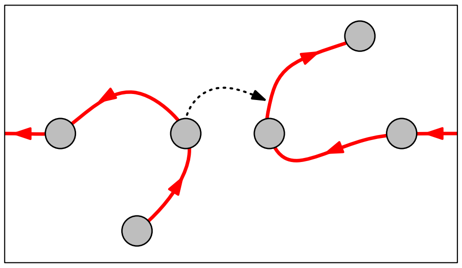


FIG. 20. An illegal refactoring move on a torus, which causes the simulation to declare failure and abort.

in the fattened lattice, the basis then locally looks like the ones depicted in Fig. 17 or Fig. 18, depending on the type of error. Note that such a basis can no longer be represented using curve diagrams. This is not an issue, since we return to a standard basis before the full machinery of curve diagrams is required again.

For the case of a σ_z error, the coefficients in the state vector and the matrix elements Eq. (85) are used to calculate the probabilities Eq. (69) of the different possible charge measurement outcomes. A result is picked according to this probability distribution, after which the state vector is updated using Eq. (70).

In the case of a σ_x error, the matrix elements (81) are used together with the coefficients in the state vector to compute the probabilities Eq. (77) for the outcome of the vertex and tail measurements. A result is then sampled from this distribution. Next, the matrix elements Eq. (82) are used to compute the probabilities Eq. (79) for the various outcomes of the charge measurement (that is performed after the appropriate unitary vertex correction is applied). We again pick a result according these probabilities and update the state vector using Eq. (80). In the case of a σ_y error, we proceed analogously to the σ_x case, using the matrix elements Eqs. (83) or (84).

After the state vector is updated to reflect the collective effect of the noise and the measurements (with the specific outcomes that we pick at random above), we conclude the current "noise step" by transforming the fusion basis back to the one that corresponds to the curve diagram we ended up with earlier (where all affected anyons appear sequentially). This is done by a sequence of F moves that reverts the transformation performed by the first series of F moves.

3. Recovery phase

After completing the process above T times, the error syndrome is given by the locations and charges of all thermal anyons on the lattice. The decoding algorithm (see Sec. IV) is then used to determine an appropriate recovery step. Such a recovery step can be broken down into a sequence of pairwise fusion processes of anyonic excitations. In each such pairwise fusion process, one member of the pair is moved along a specific path on the lattice until it neighbors the other. The moving procedure consists of basic operations where the anyon is moved to a neighboring tile and the curves are continuously deformed accordingly, as described in Supplemental Sec. IV F [45]. This basic moving step is repeated until the two anyons reside in neighboring tiles. A sequence of refactoring moves and merges is then performed to obtain a basis in which they are direct neighbors on the same curve, and the corresponding operations are applied to the affected state vectors. As during the noise process, any illegal refactoring moves cause the simulation to abort the current Monte Carlo step and report a decoding failure. Note that this happens precisely when the intended fusion would create a noncontractible loop in terms of the curve diagrams, which, in turn, corresponds to a logical error.

If necessary, an F move is applied to obtain a fusion tree shape where the anyons are fused directly. Their resulting charge is then projected according to the probabilities dictated by the coefficients in the state superposition, and it is placed in one of the two neighboring plaquettes while the state vector and the curve diagram are updated accordingly.

This basic pairwise fusion process is repeated until the current recovery step is completed, at which point the resulting error syndrome is used to determine the next recovery step. This dialog is iterated until either all anyonic excitations fused away and decoding is successful or a logical error occurs during some fusion process and a decoding failure is declared.

Note that, throughout the entire Monte Carlo step, the system is constantly monitored for violations of the cutoff parameters. If at any point an individual curve contains more than $\mathtt{N}_{\max}$ anyons or a state vector contains more than $\mathtt{V}_{\max}$ nonzero elements, the current Monte Carlo step is automatically reported as a failure.

VI. NUMERICAL RESULTS

The Monte Carlo simulations described in Sec. V are performed for the three different decoders described in Sec. IV and Supplemental Sec. V [45]. Individual Pauli errors are picked using relative probabilities corresponding to the following noise models:

(i) depolarizing noise.—$\gamma_x = \gamma_y = \gamma_z = \frac{1}{3}$;
(ii) dephasing noise.—$\gamma_x = \gamma_y = 0$, $\gamma_z = 1$;
(iii) bit-flip noise.—$\gamma_x = 1$, $\gamma_y = \gamma_z = 0$.

Simulations are performed for a wide range of physical error rates. For depolarizing noise and pure bit-flip noise, we consider the linear system sizes $L = 10, 12, 14, 16, 18$. For dephasing noise, the values $L = 12, 14, 16, 18, 20, 22$ are used.

The logical failure rate for each (p, L) pair is computed by averaging over 10^5 Monte Carlo samples. For the lowest error rates $p = 0.01$ (or $p = 0.02$ in the case of dephasing noise), 10^6 Monte Carlo samples are used in order to improve the accuracy of our results. As visible in the results below, this is ample to guarantee sufficiently small (95%) confidence intervals for the average logical failure rates.

All simulations are done with the following values for the cutoff parameters:

$$\mathtt{N}_{\max} = 27,$$
$$\mathtt{V}_{\max} = 2.5 \times 10^7.$$

For depolarizing noise and bit-flip noise with $L = 18$, the ratios of aborted Monte Carlo samples near the observed thresholds are shown in Table I.

TABLE I. Ratio of aborted Monte Carlo samples with $L = 18$ near the observed thresholds for the various decoders. The corresponding ratios of aborted failures are indicated in parentheses.

	Clustering	Fusion-aware MWPM	Blind MWPM
Depolarizing	6.1%	0.6%	0.8%
Noise	(17.4%)	(2.1%)	(2.9%)
Bit-flip	4.6%	0.3%	0.4%
Noise	(15.1%)	(1.3%)	(1.7%)

We find that the ratio of aborted Monte Carlo samples drops rapidly below the threshold. For instance, at $p = 0.04$ less than 1.1% of Monte Carlo samples (7.4% of reported failures) are aborted for depolarizing noise with $L = 18$. For dephasing noise, the ratio of aborted iterations is negligible for all system sizes and noise strengths we study.

A. Determining the threshold

The error correction threshold is given by the critical value p_c below which the logical failure rate P_L is exponentially suppressed in terms of the system size L. For large system sizes, the threshold manifests itself as the physical error rate for which the logical failure rates of all system sizes coincide. Hence, a rough estimate of the threshold can be obtained by plotting P_L as a function of p for different system sizes and finding the error rate at which the various curves intersect.

A more accurate estimate for the error correction threshold can be obtained using the critical exponent method of Ref. [61]. This method is introduced in the context of the toric code, where an exact mapping to a statistical model is known [4,61]. This model, the two-dimensional random-bond Ising model (RBIM), undergoes a phase transition from an ordered to a disordered phase as the parameter corresponding to the physical error rate increases. This implies a phase transition in the logical failure rate of the toric code. Wang, Harrington, and Preskill demonstrate that in the regime $L \gg |p - p_c|^{-\nu}$, where ν is the critical exponent for the correlation length in the RBIM, the logical failure rate P_L depends only on the dimensionless ratio $L(p - p_c)^{\nu}$.

While the statistical model corresponding to the Fibonacci Turaev-Viro code is not known, it is expected that a similar scale-invariant behavior occurs near the threshold here as well. Specifically, for sufficiently large system sizes, we define the rescaled variable

$$x = (p - p_c)L^{1/\nu}, \tag{95}$$

where ν is some critical exponent, such that the logical failure rate P_L as a function of x is explicitly scale invariant. We can find the correct values for p_c and ν by fitting the values of P_L to the quadratic ansatz

$$P_L(x) = A + Bx + Cx^2, \tag{96}$$

originating from a truncated Taylor expansion in the neighborhood of $x = 0$ ($p = p_c$). We perform this fit explicitly with the data obtained for the clustering decoder with depolarizing noise and dephasing noise.

B. Clustering decoder

The logical failure rate P_L of the clustering decoder as a function of the noise strength p is shown in Figs. 21(a)–21(c) for depolarizing, dephasing noise, and bit-flip noise, respectively. These results clearly manifest threshold behavior. For pure bit-flip noise, the threshold can be estimated from the corresponding plot as $p_c \approx 0.0375 \pm 0.0025$. A more precise estimation, based on the finite-size ansatz discussed above, is made for depolarizing noise and dephasing noise.

For depolarizing noise, the finite-size scaling ansatz Eq. (96) is fitted to the logical failure rates for p ranging from 0.045 to 0.05 in increments of 0.001 25. The following values are found using a nonlinear least squares fit:

$$p_c = 0.0470 \pm 0.0011,$$
$$\nu = 1.62 \pm 0.33.$$

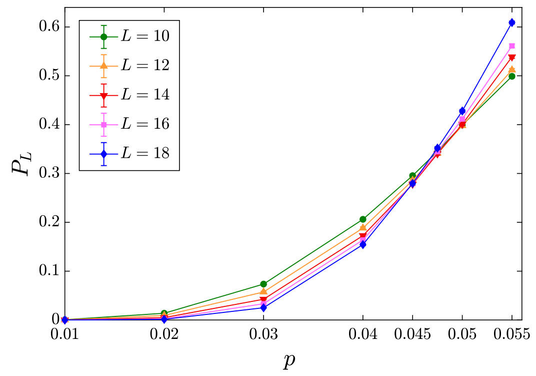
For dephasing noise, the ansatz is fitted to the logical failure rates obtained for p ranging from 0.07 to 0.075 in increments of 0.001 25. With these data, we find

$$p_c = 0.0732 \pm 0.0006, \tag{97}$$

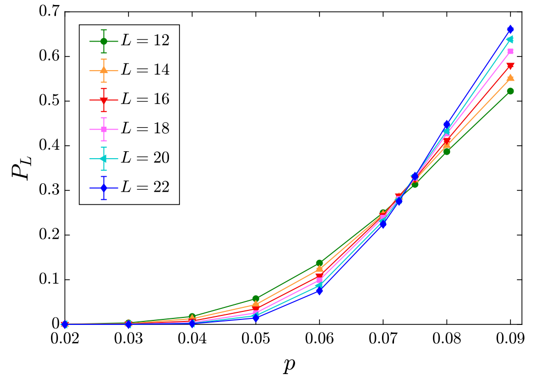
$$\nu = 1.17 \pm 0.08. \tag{98}$$

The confidence intervals are estimated using the jackknife resampling method. In both cases, the obtained threshold is compatible with the rough estimate based on the crossing of the curves in Fig. 21. The logical failure rates in terms of rescaled error rate x defined in Eq. (95) are shown in Figs. 22(a) and 22(b) for depolarizing noise and dephasing noise, respectively. One can see that the obtained parameters do indeed result in a clear “collapse” of the data, as predicted by the finite-size scaling hypothesis.

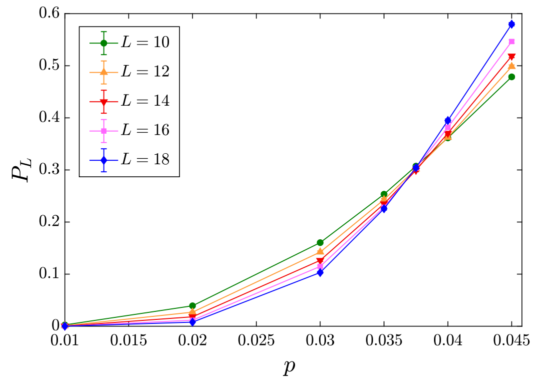
It is no surprise that the highest threshold is found for dephasing noise: No more than two anyonic excitations can be created by a single σ_z error, while up to four anyons can be created by a single σ_x or σ_y error. Hence, the average number of new anyonic excitations created in each time step is the lowest for dephasing noise and the highest for pure bit-flip noise, with the value for depolarizing noise lying somewhere in between these two extremes. The discrepancy between the thresholds for dephasing and bit-flip noise indicates that, for biased noise, there is a preferred choice for the computational basis of the physical qubits.



(a)

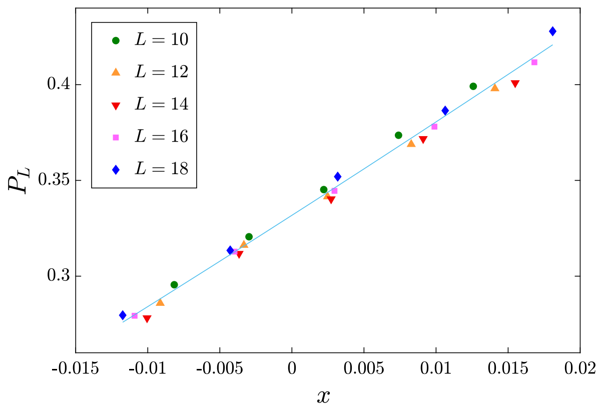


(b)

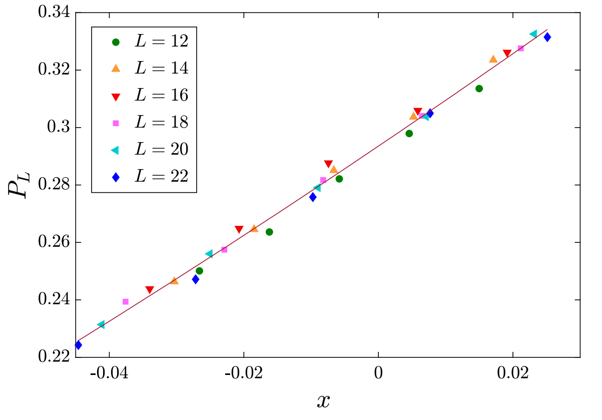


(c)

FIG. 21. Logical failure rate P_L as a function of the physical error rate p for the clustering decoder with (a) depolarizing noise, (b) pure dephasing noise, and (c) pure bit-flip noise.



(a)



(b)

FIG. 22. Logical failure rate P_L as a function of the rescaled error rate $x = (p - p_c)L^{(1/\nu)}$ for (a) depolarizing noise and (b) dephasing noise. The solid line represents the best fit of the model $P_L = A + Bx + Cx^2$.

C. Iterative matching decoders

We consider two types of iterative MWPM decoders: a fusion-aware one and a blind one (see Supplemental Sec. V [45]). The performance of these decoders under both types of noise is shown in Fig. 23. Note that the threshold obtained with these two types of decoders are very close. Under depolarizing noise, both decoders exhibit a threshold around $p_c \approx 0.0300 \pm 0.0025$. Under dephasing noise and bit-flip noise, respectively, we find $p_c \approx 0.0600 \pm 0.0025$ and $p_c \approx 0.0250 \pm 0.0025$ for both decoders. However, when closely comparing the results, as in Fig. 24, one finds a slight overall advantage for the fusion-aware decoder in the sense that its failure rates are lower than those obtained with the blind decoder.

On the other hand, we see that both matching decoders have worse performance and lower thresholds compared to the clustering decoder which does not use the detailed syndrome information of anyon types. This is related to the fact that the clustering decoder seems to be more natural for the Fibonacci code than the matching decoder. It remains an open question whether the optimal decoder for the Fibonacci Turaev-Viro code is fusion aware.

D. Discussion

The results above are expressed in terms of the average qubit error rate p in the fixed-rate sampling noise model described in Sec. V C. It is, therefore, not straightforward to compare these results to those of Abelian models such as the surface code, which are typically expressed in terms of an independent and identically distributed binomial noise

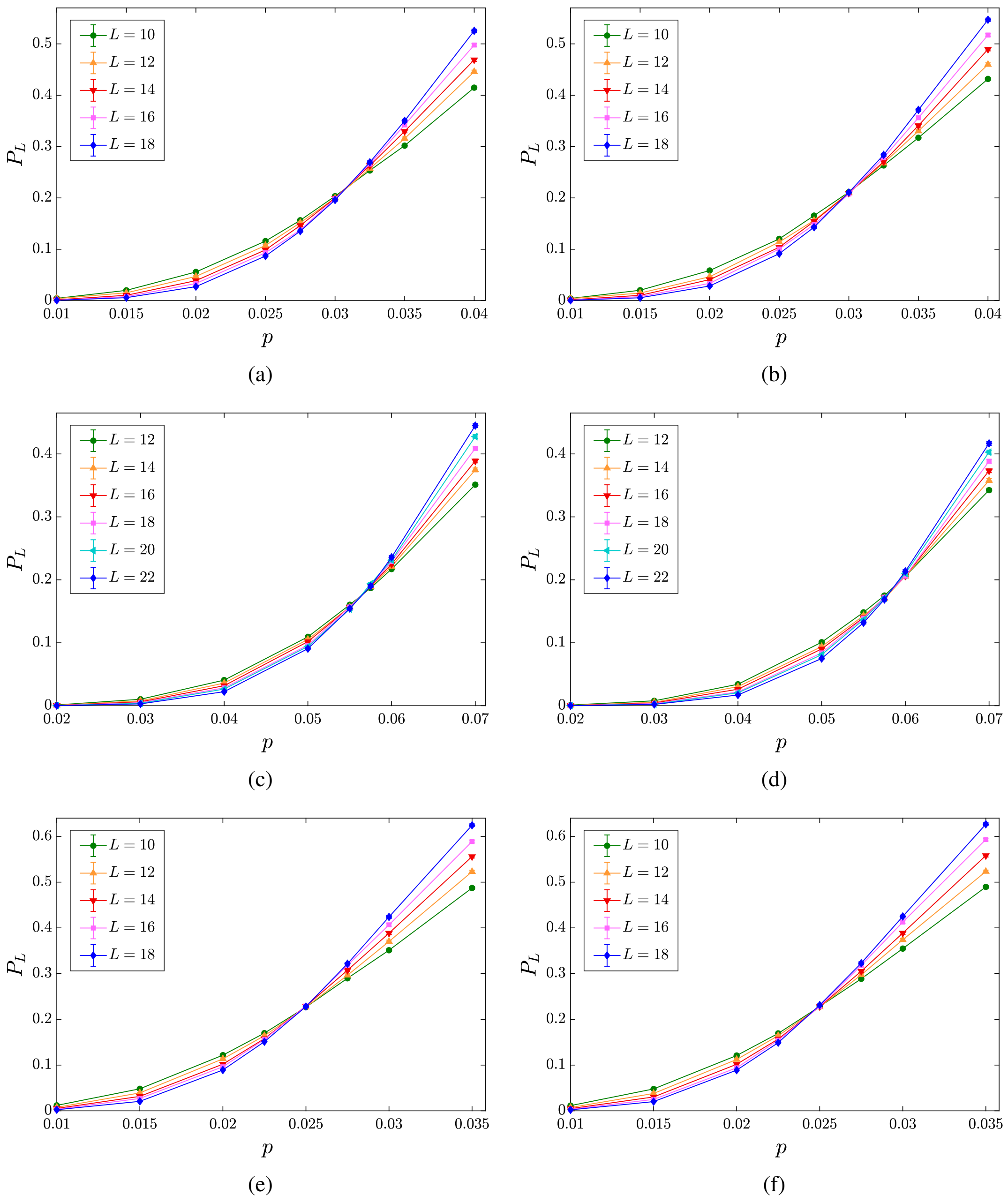


FIG. 23. (a),(c),(e) Logical failure rate P_L as a function of the physical error rate p for the fusion-aware iterative MWPM decoder with (a) depolarizing noise, (c) pure dephasing noise, and (e) pure bit-flip noise. (b),(d),(e) Logical failure rate P_L as a function of the physical error rate p for the blind iterative MWPM decoder with (b) depolarizing noise, (d) pure dephasing noise, and (e) pure bit-flip noise.

strength p_{IID}. However, for Abelian codes, both noise models are equally valid, and their respective noise strengths can be compared using the relation between the IID noise strength p_{IID} and the error rate p of a fixed-rate sampling noise model derived in Supplemental Sec. VIII [45].

Using this relation, one finds that the thresholds obtained for the clustering decoder under depolarizing ($p_c \approx 0.047$) and dephasing ($p_c \approx 0.073$) noise correspond to IID noise strengths of 4.6% and 7.0%, respectively. Remarkably, despite the complexity of the extended string-net code and the fact that it is not known whether or not the clustering decoder is optimal for this code, these values compare very favorably with the optimal thresholds for the surface code [62] (under the assumption of perfect measurements), which are 18.9% for depolarizing noise [64] and around 10% for dephasing noise [4,61].

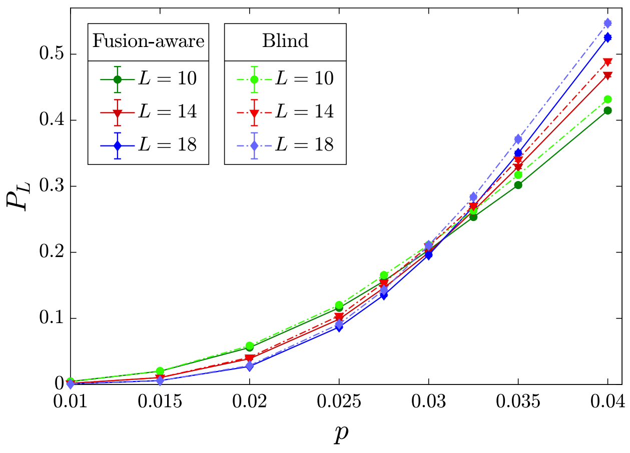


FIG. 24. Comparison between the logical failure rates for the fusion-aware and the blind iterative MWPM decoders under depolarizing noise.

VII. CONCLUSION AND OUTLOOK

In order to estimate the non-Abelian error threshold, we combine concepts and techniques from three seemingly distant fields: quantum error correction, topological quantum field theory, and tensor networks. In particular, we develop a complete error correction scheme and decoding protocols for the Fibonacci Turaev-Viro code, which supports a universal logical gate set via braiding or Dehn twists in two dimension. Making use of the framework of tensor networks and tube algebra, we are able to estimate the code-capacity error correction threshold using a clustering decoder and a fusion-aware iterative matching decoder. The threshold of 4.75% obtained for the clustering decoder is comparable to the code-capacity error threshold of the Abelian surface code in 2D, which is around 10% [4,30].

The main potential advantage of the non-Abelian Turaev-Viro code compared to the surface code is that universal logical gate sets can be implemented just via anyon braiding or Dehn twists without the need for magic-state distillation, which requires significantly more resource overhead. Nevertheless, when directly comparing the resource overhead of the Fibonacci Turaev-Viro code and the surface code, the situation is a bit tricky, since they correspond to different logical gate sets. The surface code uses a Clifford + T logical gate set, where the logical Clifford gates are implemented by lattice surgery or braiding topological defects such as punctures or twists with an $O(d)$ time overhead, and the logical T gate is implemented using the magic-state distillation protocol. On the other hand, the Fibonacci anyon braiding has a completely different generating set of logical gates, which requires a non-negligible circuit depth to approximate the standard Clifford gates. Therefore, the overhead comparison depends on the particular quantum algorithm to be implemented. There already exists a detailed comparison in Ref. [65] for one of the most well-known algorithms, Shor's factoring algorithm, between braiding Fibonacci anyon and the standard Clifford gate set with additional magic-state distillation [66]. In Shor's algorithm, the main computational resource is used for the modular exponentiation. As shown in Ref. [65], factoring a 128-bit number requires approximately 10^3 Fibonacci anyons (logical qubits), while the Clifford + magic-state distillation scheme needs about 10^9 logical qubits. Therefore, when implementing Shor's algorithm, the scheme of braiding Fibonacci anyons actually has a significant advantage. We note that this is just a rough estimate mainly considering the complexity of logical circuit compiling. It does not take into account the detailed threshold values of the Fibonacci and surface codes and assumes the logical gates in both codes have the same logical error rate. Still, such an assumption is reasonable when the physical error rate is sufficiently small compared to the error thresholds of both codes given that these thresholds do not differ by many orders of magnitude.

We further emphasize that the logical gate compiling complexity of braiding Fibonacci anyons should never be the main concern of the practicality of Turaev-Viro codes. Although the present paper focuses on the simplest type of universal Turaev-Viro code hosting Fibonacci anyons, one can simply generalize it to other types of Turaev-Viro codes by changing the numerical data of the fusion category (TQFT) defining the code, i.e., the fusion rules and F matrices, etc. One example of such modification is to implement the Ising Turaev-Viro code, for which the excitations are Ising anyons. When implementing Ising Turaev-Viro codes on a higher-genus surface or bilayer systems, one can implement the T gate via a Dehn twist along a handle [67,68] and, hence, form a universal gate set by combining these with the Clifford gates implemented via braiding and topological charge measurement using Ising anyons. With this example, the comparison with the standard magic-state distillation protocol using surface codes becomes straightforward. In particular, the fault-tolerant T gate can be applied via a linear depth circuit to implement the Dehn twist, i.e., with $O(d)$ time overhead, which requires significantly less space-time overhead than the case of implementing the logical T gate via magic-state distillation in the surface code.

The main conceptual difference of our work and previous works which simulate the third spatial dimension of a 3D color code or 3D surface codes with the time dimension using a just-in-time decoder in a 2D measurement-based quantum architecture [14,15] is that the computational power in our case comes from the 2D code space instead of the 3D code space, and no additional code switching or gauge fixing procedure is needed. Practically, the logical gates in our case can be implemented by braiding via continuous code deformation, and the error threshold and logical error rate for fault-tolerant logical gates is, therefore, expected to be the same as the fault-tolerant threshold for memory storage. In particular, no extra decrease of the error threshold (compared to the storage threshold) due to the implementation of non-Clifford transversal gates, code switching or gauge fixing with 3D surface codes or color codes [10,11,14,15], or just-in-time decoding is expected in our case. Another both fundamental and practical difference is that our scheme can also be implemented in a hybrid approach of active and passive topological protection, where the majority of noise source are passively protected by a 2D Hamiltonian, while only the thermal noise will need to be corrected via active error correction [69]. This hybrid approach may greatly reduce the overhead of active error corrections.

A natural future extension of the work presented here will be the adaptation of our error correction protocols to take into account measurement errors and to determine the error threshold of the code in the presence of both measurement and circuit-level noise. In this setting of full fault-tolerant error correction, our measurement scheme which allows one to distinguish the charges of different anyonic excitations may prove useful, since this information could help in identifying measurement errors when performing repeated syndrome measurements through a consistency check. Thus, while it is still an open question what the optimal decoder for the Fibonacci Turaev-Viro code is and whether it would make use of the detailed charge information in the error syndrome, it is likely that this charge information would yield a notable advantage in the presence of measurement errors. More generally, the tensor network representation used in the current work can also be further used to simulate coherent noise in these non-Abelian codes as well as in the usual surface code.

Another direction to explore is the application of our techniques to different models. A first interesting route would be to investigate other types of Turaev-Viro codes, such as the Ising Turaev-Viro code, which has doubled Ising anyons as excitations. Besides having the standard Clifford $+ T$ logical gate set as mentioned above, this code would have a simpler noncyclic fusion rule structure which could lead to a higher threshold, especially in the presence of measurement noise. In this context, our measurement scheme to extract the specific anyon charges would be particularly useful, since it is shown in Ref. [22] that fusion-aware decoders can yield a significant advantage for Ising-type anyons. A second important direction here will be to investigate non-Abelian codes with a simpler structure, such as lower-weight syndrome operator and lower-depth measurement circuits. The Levin-Wen string-net models are sophisticated in the sense that their plaquette syndrome operator has weight 16. On the other hand, Kitaev's non-Abelian quantum double models [3] can have a weight-4 syndrome operator and could possibly be analyzed using an adaptation of the techniques presented in this work. It would, therefore, be interesting to further explore the error threshold of these alternative models which could be more practical in terms of an experimental implementation.

A different avenue of further research would be the investigation of planar string-net codes with suitable gapped boundary conditions, where information is then encoded in the fusion state of a number of well-separated anyons rather than in the ground state degeneracy associated to a closed manifold with a nontrivial topology (high-genus surface). Alternatively, the logical information can also be encoded in the boundary degeneracy of an open manifold corresponding to a planar geometry in analogy with the Abelian surface code with e and m boundaries. This matter is of significant interest, since a planar geometry is highly attractive regarding experimental realization. The classification of these gapped boundaries has been performed in the language of (bi)module category theory [70], and recent progress has been made in capturing this formalism in terms of tensor network representations [71]. While it does not seem that suitable boundaries for a Fibonacci Turaev-Viro can be constructed in this language, the investigation of these concepts in the context of error correction using different non-Abelian codes would be of great interest.

Besides the study of the quantum memory property of the non-Abelian codes, an important direction is to study and simulate the detailed implementation of a universal set of logical gates in these codes. Besides the approach of doing braiding and Dehn twists [20,27,42,43], one can also perform transversal gates on a folded non-Abelian code equivalent to elements in the mapping class group [72]. An additional advantage of non-Abelian codes appears when they are placed on a hyperbolic surface, which admits both constant-rate encoding [$O(1)$ space overhead] and parallel universal logical gates via constant-depth circuits [44]. A promising direction is to explore non-Abelian codes in higher spatial dimension or on an expander graph. Along this direction, the ultimate goal is to explore and achieve the fundamental limit of space-time overhead. This is because, in higher dimension such as 4D, one can obtain a self-correcting quantum memory. In that case, local errors created when implementing a logical gate with a constant depth circuit [42–44] can be corrected locally in $O(1)$ time. Eventually, this direction could evolve into a flourishing interface between quantum information and quantum topology.

An important aspect in terms of experimental implementation of the Fibonacci Turaev-Viro code is the realization of multi-controlled-Z (multiqubit Toffoli) gates, which are required for an efficient implementation of F moves (see Sec. III B). Therefore, it would be interesting to further explore hardware-efficient implementations of multiqubit gates, instead of always decomposing these into a longer sequence of two-qubit gates. Such multiqubit gates are widely studied in Rydberg-atom [73] and ion-trap systems, and significant progress was recently made in the realization of these gates in superconducting qubits as well [74–76].

Finally, besides the application to quantum error correction, the scheme developed in this paper also paves the way for quantum simulation of topological quantum field theory on a near-term quantum computer. In particular, the measurement and correction schemes will be a crucial ingredient for the state preparation of the TQFT wave functions.

ACKNOWLEDGMENTS

The authors thank Dominic Williamson, Volkher Scholz, and Nick Bultinck for their suggestions during the early stages of this project. Special thanks goes to Bram Vanhecke and Laurens Lootens for their valuable advice and the countless discussions on various aspects of the code. We also appreciate Steve Flammia for the discussion on decoding and classical simulation of Fibonacci anyons. The computational resources (Stevin Supercomputer Infrastructure) and services used in this work were provided by the Flemish Supercomputer Center (VSC), funded by Ghent University, the Research Foundation Flanders (FWO), and the Flemish Government. A. S., L. B., and F. V. were supported by FWO and ERC Grant No. QUTE (647905). A. S. was also supported by a fellowship of the Belgian American Educational Foundation.

[1] B. M. Terhal, *Quantum Error Correction for Quantum Memories*, Rev. Mod. Phys. **87**, 307 (2015).
[2] E. T. Campbell, B. M. Terhal, and C. Vuillot, *Roads towards Fault-Tolerant Universal Quantum Computation*, Nature (London) **549**, 172 (2017).
[3] A. Y. Kitaev, *Fault-Tolerant Quantum Computation by Anyons*, Ann. Phys. (Amsterdam) **303**, 2 (2003).
[4] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, *Topological Quantum Memory*, J. Math. Phys. (N.Y.) **43**, 4452 (2002).
[5] S. B. Bravyi and A. Y. Kitaev, *Quantum Codes on a Lattice with Boundary*, arXiv:quant-ph/9811052.
[6] R. Raussendorf and J. Harrington, *Fault-Tolerant Quantum Computation with High Threshold in Two Dimensions*, Phys. Rev. Lett. **98**, 190504 (2007).
[7] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, *Surface Codes: Towards Practical Large-Scale Quantum Computation*, Phys. Rev. A **86**, 032324 (2012).
[8] S. Bravyi and A. Kitaev, *Universal Quantum Computation with Ideal Clifford Gates and Noisy Ancillas*, Phys. Rev. A **71**, 022316 (2005).
[9] S. Bravyi and R. König, *Classification of Topologically Protected Gates for Local Stabilizer Codes*, Phys. Rev. Lett. **110**, 170503 (2013).
[10] H. Bombin, *Single-Shot Fault-Tolerant Quantum Error Correction*, Phys. Rev. X **5**, 031043 (2015).
[11] M. Vasmer and D. E. Browne, *Three-Dimensional Surface Codes: Transversal Gates and Fault-Tolerant Architectures*, Phys. Rev. A **100**, 012312 (2019).
[12] 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).
[13] T. Jochym-O'Connor and T. J. Yoder, *Four-Dimensional Toric Code with Non-Clifford Transversal Gates*, Phys. Rev. Research **3**, 013118 (2021).
[14] H. Bombin, *2d Quantum Computation with 3d Topological Codes*, arXiv:1810.09571.
[15] B. J. Brown, *A Fault-Tolerant Non-Clifford Gate for the Surface Code in Two Dimensions*, Sci. Adv. **6**, eaay4929 (2020).
[16] C. Nayak, S. H. Simon, A. Stern, M. Freedman, and S. D. Sarma, *Non-Abelian Anyons and Topological Quantum Computation*, Rev. Mod. Phys. **80**, 1083 (2008).
[17] D. E. Gottesman, *Stabilizer Codes and Quantum Error Correction*, Ph.D. thesis, California Institute of Technology, 1997.
[18] M. H. Freedman and M. B. Hastings, *Double Semions in Arbitrary Dimension*, Commun. Math. Phys. **347**, 389 (2016).
[19] G. Dauphinais, L. Ortiz, S. Varona, and M. A. Martin-Delgado, *Quantum Error Correction with the Semion Code*, New J. Phys. **21**, 053035 (2019).
[20] M. H. Freedman, M. Larsen, and Z. Wang, *A Modular Functor which Is Universal for Quantum Computation*, Commun. Math. Phys. **227**, 605 (2002).
[21] M. A. Levin and X.-G. Wen, *String-Net Condensation: A Physical Mechanism for Topological Phases*, Phys. Rev. B **71**, 045110 (2005).
[22] C. G. Brell, S. Burton, G. Dauphinais, S. T. Flammia, and D. Poulin, *Thermalization, Error Correction, and Memory Lifetime for Ising Anyon Systems*, Phys. Rev. X **4**, 031058 (2014).
[23] J. R. Wootton, J. Burri, S. Iblisdir, and D. Loss, *Error Correction for Non-Abelian Topological Quantum Computation*, Phys. Rev. X **4**, 011051 (2014).
[24] J. R. Wootton and A. Hutter, *Active Error Correction for Abelian and Non-Abelian Anyons*, Phys. Rev. A **93**, 022318 (2016).
[25] G. Dauphinais and D. Poulin, *Fault-Tolerant Quantum Error Correction for Non-Abelian Anyons*, Commun. Math. Phys. **355**, 519 (2017).
[26] S. Burton, C. G. Brell, and S. T. Flammia, *Classical Simulation of Quantum Error Correction in a Fibonacci Anyon Code*, Phys. Rev. A **95**, 022309 (2017).
[27] R. König, G. Kuperberg, and B. W. Reichardt, *Quantum Computation with Turaev-Viro Codes*, Ann. Phys. (Amsterdam) **325**, 2707 (2010).
[28] N. E. Bonesteel and D. P. DiVincenzo, *Quantum Circuits for Measuring Levin-Wen Operators*, Phys. Rev. B **86**, 165113 (2012).
[29] S. Bravyi, M. Suchara, and A. Vargo, *Efficient Algorithms for Maximum Likelihood Decoding in the Surface Code*, Phys. Rev. A **90**, 032326 (2014).
[30] T. J. Yoder and I. H. Kim, *The Surface Code with a Twist*, Quantum **1**, 2 (2017).

[31] V. F. R. Jones, *A Polynomial Invariant for Knots via von Neumann Algebras*, Bull. Am. Math. Soc. **12**, 103 (1985).

[32] N. Y. Reshetikhin and V. G. Turaev, *Ribbon Graphs and Their Invariants Derived from Quantum Groups*, Commun. Math. Phys. **127**, 1 (1990).

[33] E. Witten, *Quantum Field Theory and the Jones Polynomial*, Commun. Math. Phys. **121**, 351 (1989).

[34] M. F. Atiyah, *Topological Quantum Field Theory*, Publ. Math. IHÉS **68**, 175 (1988).

[35] V. Turaev and O. Viro, *State Sum Invariants of 3-Manifolds and Quantum 6j-Symbols*, Topology **31**, 865 (1992).

[36] M. Levin and X.-G. Wen, *Detecting Topological Order in a Ground State Wave Function*, Phys. Rev. Lett. **96**, 110405 (2006).

[37] W. Feng, *Non-Abelian Quantum Error Correction*, Ph. D. thesis, Florida State University, 2015.

[38] W. Feng, N. Bonesteel, and D. DiVincenzo (to be published).

[39] Y. Hu, N. Geer, and Y.-S. Wu, *Full Dyon Excitation Spectrum in Extended Levin-Wen Models*, Phys. Rev. B **97**, 195154 (2018).

[40] A. Ocneanu *et al.*, in *Proceedings of the Taniguchi Conference on Mathematics Nara'98* (Mathematical Society of Japan, Tokyo, 2001), pp. 235–263.

[41] N. Bultinck, M. Mariën, D. J. Williamson, M. B. Şahinoğlu, J. Haegeman, and F. Verstraete, *Anyons and Matrix Product Operator Algebras*, Ann. Phys. (Amsterdam) **378**, 183 (2017).

[42] G. Zhu, A. Lavasani, and M. Barkeshli, *Instantaneous Braids and Dehn Twists in Topologically Ordered States*, Phys. Rev. B **102**, 075105 (2020).

[43] G. Zhu, A. Lavasani, and M. Barkeshli, *Universal Logical Gates on Topologically Encoded Qubits via Constant-Depth Unitary Circuits*, Phys. Rev. Lett. **125**, 050502 (2020).

[44] A. Lavasani, G. Zhu, and M. Barkeshli, *Universal Logical Gates with Constant Overhead: Instantaneous Dehn Twists for Hyperbolic Quantum Codes*, Quantum **3**, 180 (2019).

[45] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevX.12.021012 for technical details on TQFT, recovery operations, tensor network representations, curve diagrams, and the fusion-aware iterative matching decoder.

[46] For generic unitary fusion categories, one must pick an orientation for every edge. An edge with label i^* is equivalent to an edge with label i and the opposite orientation.

[47] R. N. C. Pfeifer, O. Buerschaper, S. Trebst, A. W. W. Ludwig, M. Troyer, and G. Vidal, *Translation Invariance, Topology, and Protection of Criticality in Chains of Interacting Anyons*, Phys. Rev. B **86**, 155111 (2012).

[48] Note that resolving both the inner and the outer tube into the lattice would yield a trivial operation, since they constitute the vacuum bubble together.

[49] Note that it has a different shape than our convention Eq. (22), hence the F matrix in Eq. (52).

[50] There are seven ways to label a tube according to the Fibonacci fusion rules. Hence, the seven vectors below span the string-net subspace for the four qubits on the tube.

[51] The operators below must, of course, be completed to true unitary operators. The missing terms are left out to improve readability.

[52] Possibly, this could improve the performance of certain decoders. Furthermore, we assume this for the numerical simulations discussed in Sec. V.

[53] Provided that no errors happen on qubits in or adjacent to the plaquette between these repeated measurements.

[54] S. Bravyi and J. Haah, *Quantum Self-Correction in the 3D Cubic Code Model*, Phys. Rev. Lett. **111**, 200501 (2013).

[55] In the case of an open manifold with gapped boundaries, the logical operator corresponds to a nontrivial relative homology cycle.

[56] S. Burton, *A Short Guide to Anyons and Modular Functors*, arXiv:1610.05384.

[57] M. Z. Bazant, *Largest Cluster in Subcritical Percolation*, Phys. Rev. E **62**, 1660 (2000).

[58] Note that, under these assumptions, it is not appropriate to characterize the noise in terms of an IID noise strength per qubit, since the non-Abelian nature of our code implies that the combined action of individual error and measurement operators does not commute for consecutive errors. We discuss the relation with IID noise in Supplemental Sec. VIII [45].

[59] M. B. Hastings, G. H. Watson, and R. G. Melko, *Self-Correcting Quantum Memories beyond the Percolation Threshold*, Phys. Rev. Lett. **112**, 070501 (2014).

[60] V. Zauner-Stauber, L. Vanderstraeten, M. T. Fishman, F. Verstraete, and J. Haegeman, *Variational Optimization Algorithms for Uniform Matrix Product States*, Phys. Rev. B **97**, 045145 (2018).

[61] C. Wang, J. Harrington, and J. Preskill, *Confinement-Higgs Transition in a Disordered Gauge Theory and the Accuracy Threshold for Quantum Memory*, Ann. Phys. (Amsterdam) **303**, 31 (2003).

[62] The results mentioned here are, in fact, obtained for the toric code (i.e., with periodic boundary conditions). It is likely that the true thresholds for surface codes are slightly lower [63].

[63] A. G. Fowler, *Accurate Simulations of Planar Topological Codes Cannot Use Cyclic Boundaries*, Phys. Rev. A **87**, 062320 (2013).

[64] H. Bombin, R. S. Andrist, M. Ohzeki, H. G. Katzgraber, and M. A. Martín-Delgado, *Strong Resilience of Topological Codes to Depolarization*, Phys. Rev. X **2**, 021004 (2012).

[65] M. Baraban, N. E. Bonesteel, and S. H. Simon, *Resources Required for Topological Quantum Factoring*, Phys. Rev. A **81**, 062317 (2010).

[66] For the latter case, Ref. [65] uses the example of braiding Ising anyons or Majorana fermions to implement Clifford gates, with additional magic-state distillation protocol. However, the logical gate complexity in this case is equivalent to using the surface code with magic-state distillation.

[67] M. Freedman, C. Nayak, and K. Walker, *Towards Universal Topological Quantum Computation in the $\nu = 5\ 2$ Fractional Quantum Hall State*, Phys. Rev. B **73**, 245307 (2006).

[68] M. Barkeshli and J. D. Sau, *Physical Architecture for a Universal Topological Quantum Computer Based on a Network of Majorana Nanowires*, arXiv:1509.07135.

[69] E. Kapit, J. T. Chalker, and S. H. Simon, *Passive Correction of Quantum Logical Errors in a Driven, Dissipative System: A Blueprint for an Analog Quantum Code Fabric*, Phys. Rev. A **91**, 062324 (2015).

[70] A. Kitaev and L. Kong, *Models for Gapped Boundaries and Domain Walls*, Commun. Math. Phys. **313**, 351 (2012).

[71] L. Lootens, J. Fuchs, J. Haegeman, C. Schweigert, and F. Verstraete, *Matrix Product Operator Symmetries and Intertwiners in String-Nets with Domain Walls*, SciPost Phys. **10**, 053 (2021).

[72] G. Zhu, M. Hafezi, and M. Barkeshli, *Quantum Origami: Transversal Gates for Quantum Computation and Measurement of Topological Order*, Phys. Rev. Research **2**, 013285 (2020).

[73] M. Khazali and K. Mølmer, *Fast Multiqubit Gates by Adiabatic Evolution in Interacting Excited-State Manifolds of Rydberg Atoms and Superconducting Circuits*, Phys. Rev. X **10**, 021054 (2020).

[74] A. D. Hill, M. J. Hodson, N. Didier, and M. J. Reagor, *Realization of Arbitrary Doubly-Controlled Quantum Phase Gates*, arXiv:2108.01652.

[75] Y. Kim, A. Morvan, L. B. Nguyen, R. K. Naik, C. Jünger, L. Chen, J. M. Kreikebaum, D. I. Santiago, and I. Siddiqi, *High-Fidelity iToffoli Gate for Fixed-Frequency Superconducting Qubits*, arXiv:2108.10288.

[76] A. J. Baker, G. B. P. Huber, N. J. Glaser, F. Roy, I. Tsitsilin, S. Filipp, and M. J. Hartmann, *Single Shot i-Toffoli Gate in Dispersively Coupled Superconducting Qubits*, Appl. Phys. Lett. **120**, 054002 (2022).