

Quantum error correction thresholds for the universal Fibonacci Turaev-Viro code

Alexis Schotte,^{1,*} Guanyu Zhu,^{2,†} Lander Burgelman,¹ and Frank Verstraete¹

¹*Department of Physics and Astronomy, Ghent University, Krijgslaan 281, 9000 Gent, Belgium*

²*IBM Quantum, IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, and IBM Almaden Research Center, San Jose, CA 95120, USA*

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 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. To the best of our knowledge, this is the first time that a threshold has been estimated for a two-dimensional error correcting code for which universal quantum computation can be performed within its code space via braiding anyons.

CONTENTS			
I. Introduction	2	C. Discussion	45
Summary	3	VII. Conclusion and outlook	46
II. The extended string-net code	4	Acknowledgment	48
A. Definition of the code	4	A. The Turaev-Viro TQFT and the extended Levin-Wen string-net model	48
B. The fattened lattice picture	6	1. Category theory primer	48
C. Code deformation using Pachner moves	6	2. The ribbon graph Hilbert space	50
D. Anyonic excitations	7	3. Dehn twists and braid moves on $\mathcal{H}_\Sigma$	51
E. The anyonic fusion basis	8	4. The anyonic fusion basis	51
III. Error correction scheme	10	5. The tube algebra	57
A. Vertex measurements and correction	11	6. The Levin-Wen string-net model as a lattice realization of $\mathcal{H}_\Sigma$	58
B. Anyon charge measurements	12	7. The extended string-net model	59
C. Recovery operations	18	8. Realizing the anyonic fusion basis on the tailed lattice	60
IV. Threshold simulation	22	B. Tensor network representations	61
A. Simulating non-Abelian quantum error correction	22	1. A tensor network representation for the string-net ground states	61
B. Classical simulability	22	2. Tensor network representations for anyonic fusion basis states	64
C. Noise model	24	3. Square PEPS tensors	66
D. Pauli-noise in the anyonic fusion basis	25	4. Consistency of the tensor network representation	66
E. Computing the relevant matrix elements	28	C. Scaling of largest connected group of anyons	69
F. Storing and manipulating anyonic fusion basis states	30	D. Relating fixed-rate sampling and i.i.d. noise	69
G. Outline of the simulation	34	E. Paperclip configurations	70
H. Recovery phase	36	1. Refactoring along the curve	71
V. Decoding algorithms	36	2. Refactoring against the curve	72
A. Clustering decoder	36	References	73
B. Fusion-aware iterative matching decoder	39		
VI. Numerical results	43		
A. Clustering decoder	44		
B. Iterative matching decoders	44		

* alexis.schotte@ugent.be

† guanyu.zhu@ibm.com

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 (QECC), 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 semi-conducting qubits, ion traps, silicon photonics, NV centers, 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 non-Abelian topological codes with a non-stabilizer structure, including numerical estimates of their error thresholds [22–26]. These works however assume the existence of a protected anyonic fusion space and only consider 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 have obtained 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 i.i.d. 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+1D topological quantum field theories in 1992, leading to new quantum invariants of 3-manifold [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 planar special case of ribbon graphs introduced by Reshetikhin and Turaev. The string-net models capture all non-chiral 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 towards 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 will 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] and [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 has been 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 was 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.

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 (UFC). More specifically, we build on the work of König et al. [27], who illustrated 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 was 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], [38] and [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 PEPS, we utilize tensor network methods to analyze the effect of Pauli noise on the code, which effectively allows to translate a physical error rate at the level of the qubits to an anyon generation rate. We can hence sim-

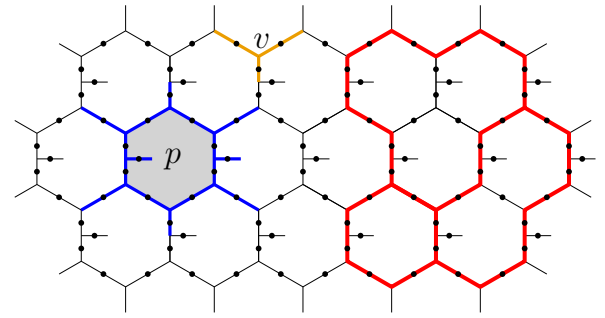


Figure 1: Qudits arranged on the tailed honeycomb lattice. The support of B_p and Q_v are indicated in blue and orange, respectively. The red edges represent a valid string-net configuration of qubits in the $|1\rangle$ state when using the Fibonacci input category.

plify 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.

To the best of our knowledge, our results provide the first threshold estimate for a two-dimensional error correcting code of qubits which is universal for topological quantum computation, without the need for additional non-topological operations.

II. THE EXTENDED STRING-NET CODE

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. We discuss this model and its relation to topological quantum field theory in much greater detail in App. A.

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 will work with the Fibonacci category later (which is self-dual), we will not be needing the general case. The algebraic data of such an object consists of:

1. **String types:** A set of all possible string types $\{1, i_2, \dots, i_N\}$. The label **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

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

For every label i , there is unique dual label i^* satisfying $\delta_{ii^*1} = 1$, and $(i^*)^* = i$. Note that we are considering self-dual categories¹, 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

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

and a six-index symbol F_{klm}^{ijm} , which is a complex constant dependent on 6 string types i, j, k, l, m, n . These quantities are required to satisfy the following consistency conditions:

$$\text{physicality : } F_{klm}^{ijm} \delta_{ijm} \delta_{klm^*} = F_{klm}^{ijm} \delta_{ilm} \delta_{jkn^*} \quad (3)$$

$$\text{pentagon equation : } \sum_{n=1}^N F_{kpn}^{mlq} F_{mns}^{jip^*} F_{lkr}^{jsn} = F_{q^*kr}^{jip^*} F_{mns}^{r^*iq^*} \quad (4)$$

$$\text{unitarity : } (F_{klm}^{ijm})^* = F_{jkm}^{lin} \quad (5)$$

$$\text{tetrahedral symmetry : } F_{klm}^{ijm} = F_{lkn^*}^{jim} = F_{jin}^{lkm^*} = F_{k^*nl}^{imj} \frac{v_m v_n}{v_j v_l} \quad (6)$$

$$\text{normalization : } F_{j^*jk}^{ii^*1} = \frac{v_k}{v_i v_j} \delta_{ijk} \quad (7)$$

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:

$$H_\Lambda = - \sum_v Q_v - \sum_p B_p, \quad (8)$$

where v and p label the vertices and plaquettes of the trivalent lattice Λ , 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:

$$Q_v |i \rangle \left| \begin{array}{c} j \\ k \end{array} \right\rangle = \delta_{ijk} |i \rangle \left| \begin{array}{c} j \\ k \end{array} \right\rangle. \quad (9)$$

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

We will 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, is defined as

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

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

$$O_p^s \left| \begin{array}{c} m_r \quad j_r \quad m_{r-1} \\ j_{r+1} \quad x \quad m_3 \\ m_1 \quad j_2 \quad j_3 \\ m_2 \end{array} \right\rangle = \delta_{x,1} \sum_{k_1, \dots, k_{r+1}} \delta_{k_1, k_{r+1}} \cdot \left(\prod_{\nu=1}^r F_{sk_{\nu+1}k_\nu}^{m_\nu j_\nu j_{\nu+1}} \right) \left| \begin{array}{c} m_r \quad k_r \quad m_{r-1} \\ k_{r+1} \quad 1 \quad m_3 \\ m_1 \quad k_2 \quad k_3 \\ m_2 \end{array} \right\rangle. \quad (11)$$

The error correction scheme and numerical simulations described in Sects. III and IV were designed specifically for the Fibonacci input category ($\mathcal{C} = \text{FIB}$), which contains only two string types, **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: $\mathbf{1} \rightarrow |0\rangle$ and $\tau \rightarrow |1\rangle$. The nontrivial fusion rule is $\tau \times \tau = \mathbf{1} + \tau$, which leads to the following branching rules:

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

The quantum dimensions are

$$d_{\mathbf{1}} = 1, \quad d_\tau = \phi, \quad (13)$$

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

$$[F_{\tau\tau}^{\tau\tau}] = \begin{pmatrix} \phi^{-1} & \phi^{-\frac{1}{2}} \\ \phi^{-\frac{1}{2}} & -\phi^{-1} \end{pmatrix}. \quad (14)$$

¹ 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.

For all other combinations of indices, F_{klm}^{ij} 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. We discuss the ribbon graph Hilbert space, and its relation to the Levin-Wen model extensively in App. A. 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:

$$\text{---} \bigcirc \text{---}^i = \delta_{j1} d_i, \quad (15)$$

$$\begin{array}{c} i \quad l \\ \diagdown \quad \diagup \\ m \\ \diagup \quad \diagdown \\ j \quad k \end{array} = \sum_n F_{klm}^{ijn} \begin{array}{c} i \quad l \\ \diagdown \quad \diagup \\ n \\ \diagup \quad \diagdown \\ j \quad k \end{array}. \quad (16)$$

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 only allowed to end 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 can’t 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

$$\begin{array}{c} i \\ \diagup \quad \diagdown \\ \nu \quad \mu \\ \bigcirc \\ \diagdown \quad \diagup \\ j \quad k \end{array} = v_\lambda v_\mu v_\nu G_{\lambda\mu\nu}^{ijk} \begin{array}{c} i \\ \diagdown \quad \diagup \\ j \quad k \end{array}, \quad (17)$$

where

$$G_{\lambda\mu\nu}^{ijk} = \frac{1}{v_i v_\lambda} F_{k\mu i}^{\nu j \lambda} = \frac{1}{v_\nu v_k} F_{\lambda\mu\nu}^{ijk}. \quad (18)$$

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 action of the plaquette projector B_p can be represented on the fattened

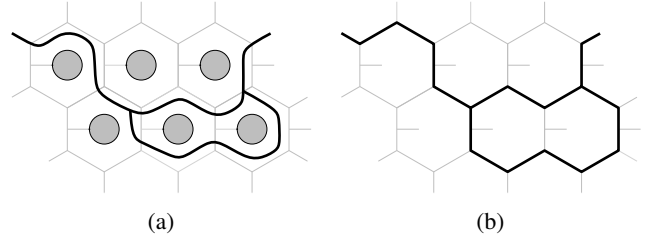


Figure 2: (a) A ribbon graph on the fattened lattice. (b) The corresponding string-net configuration on the lattice. The grey edges correspond to qudits in the $|0\rangle$ state, while the state of the black edges is given by the string type of the corresponding piece of the ribbon graph in (a).

lattice as follows:

$$B_p : \begin{array}{c} \text{---} \bigcirc \text{---}^i \\ \diagdown \quad \diagup \\ \text{---} \end{array} \mapsto \frac{1}{D} \begin{array}{c} \text{---} \bigcirc \text{---}^i \\ \diagdown \quad \diagup \\ \text{---} \end{array}, \quad (19)$$

where

$$\begin{array}{c} \text{---} \bigcirc \text{---}^i \\ \diagdown \quad \diagup \\ \text{---} \end{array} = \frac{1}{D} \sum_i d_i \begin{array}{c} \text{---} \bigcirc \text{---}^i \\ \diagdown \quad \diagup \\ \text{---} \end{array}. \quad (20)$$

The dashed loop is referred to as a *vacuum loop*. Note that we have omitted the tail edge on the right hand side of 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.

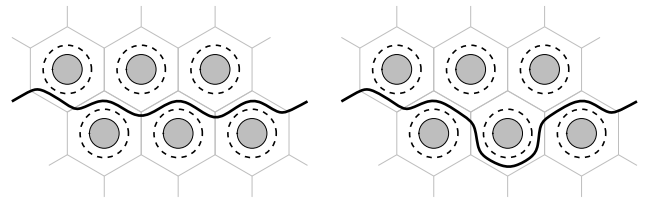


Figure 3: Two ribbon graphs on the fattened lattice representing the same string-net ground state.

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

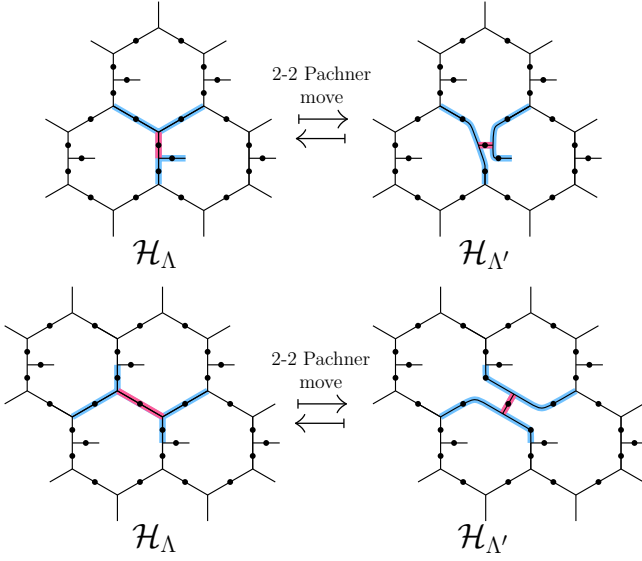


Figure 4: Lattice deformations by 2-2 Pachner moves on different edges. The affected edge is highlighted in pink, the other edges contained in the F -matrix is highlighted in blue.

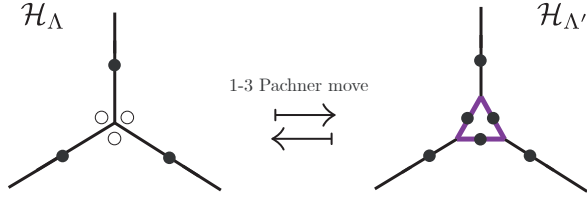


Figure 5: Lattice deformation by 1-3 Pachner move. The white dots on the left represent ancilla qudits. The fine-graining process (left to right) entangles the three ancilla qudits into the code space. The coarse-graining process (right to left) disentangles the three central qudits out of the code space.

(17) give the appropriate map between the corresponding code

$$O_{xy\alpha\beta} \left| \begin{matrix} m_r & j_r & m_{r-1} \\ j_{r+1} & x' & \vdots \\ j_1 & & m_3 \end{matrix} \right\rangle = \delta_{x,x'} \left| \begin{matrix} m_r & j_r & m_{r-1} \\ j_{r+1} & x & \alpha \\ j_1 & \beta & \vdots \\ m_1 & j_2 & j_3 \\ & m_2 & \end{matrix} \right\rangle = \delta_{x,x'} \frac{v_\alpha v_\beta}{v_y} \sum_{k_1, \dots, k_{r+1}} F_{\alpha\beta k_{r+1}}^{j_1 j_{r+1} x} \left(\prod_{\nu=1}^r F_{\alpha k_{\nu+1} k_\nu}^{m_\nu j_\nu j_{\nu+1}} \right) F_{\alpha j_1 \beta}^{k_{r+1} k_1 y} \left| \begin{matrix} m_r & k_r & m_{r-1} \\ k_{r+1} & y & \vdots \\ m_1 & k_2 & k_3 \\ & m_2 & \end{matrix} \right\rangle. \quad (21)$$

This corresponds to gluing the “tube”

$$\quad (22)$$

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 original vertex. The 1-3 Pachner move can be thought as a fine/coarse-graining process. In Fig. 5 one entangles three ancilla qudits (white dots) from left to right and add them into the code space, which effectively fine-grain 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 will be 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 types can be labeled by pairs $a_+ \bar{a}_-$ with $a_+, a_- \in \mathcal{C}$ (see App. A for more details). For notational simplicity, we will drop the bar notation for labels in $\bar{\mathcal{C}}$, and we will indicate the string-types of the doubled category with bold labels: $\mathbf{a} = a_+ a_- \in \mathcal{DC}$.

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 Sec. A 5 in App. A), 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

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 $\{11, 1\tau, \tau 1, \tau\tau\}$, which la-

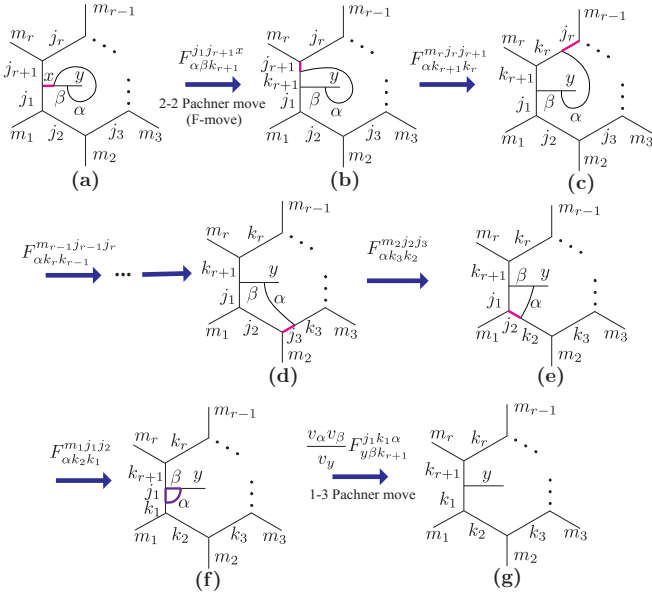


Figure 6: Derivation of the expression for the tube operator.

bel the different corresponding projectors

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

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

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

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

Note that $B_p = \mathcal{P}^{11}$ i.e., the ground space is precisely the anyonic vacuum. The entry in Eq. 26 decomposes into two simple idempotents: $\mathcal{P}^{\tau\tau} = \mathcal{P}_1^{\tau\tau} + \mathcal{P}_\tau^{\tau\tau}$, with

$$\mathcal{P}_1^{\tau\tau} = \frac{1}{\mathcal{D}^2} (\phi^2 O_{1111} - \phi O_{11\tau\tau}), \quad (27)$$

$$\mathcal{P}_\tau^{\tau\tau} = \frac{1}{\mathcal{D}^2} \left(\phi O_{\tau\tau 1\tau} + \phi O_{\tau\tau\tau 1} + \frac{1}{\sqrt{\phi}} O_{\tau\tau\tau\tau} \right). \quad (28)$$

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 $\tau\tau_1$ and $\tau\tau_\tau$, respectively. Both are in the $\tau\tau$ anyon sector and their respective +1 eigenspaces are related as follows:

$$\mathcal{P}_1^{\tau\tau} \mathcal{P}_1^{\tau\tau} = \mathcal{P}_\tau^{\tau\tau}, \quad \mathcal{P}_\tau^{\tau\tau} \mathcal{P}_\tau^{\tau\tau} = \mathcal{P}_1^{\tau\tau}, \quad (29)$$

where

$$\mathcal{P}_{1\tau}^{\tau\tau} = e^{-3\pi i/10} \frac{\phi}{\mathcal{D}} O_{1\tau\tau\tau}, \quad (30)$$

$$\mathcal{P}_{\tau 1}^{\tau\tau} = e^{3\pi i/10} \frac{\sqrt{\phi}}{\mathcal{D}} O_{\tau 1\tau\tau}, \quad (31)$$

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 $\mathcal{H}_{\text{s.n.}}$, 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 will 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 Apps. A 4 and A 8.

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):

$$|\vec{a}, \vec{b}\rangle = \begin{array}{c} \text{Diagram showing a fusion tree with leaf labels } a_1, a_2, a_3, \dots, a_{n-1} \text{ and internal branch labels } b_1, b_2, \dots, b_{n-3}. \end{array} \quad (32)$$

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

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

where $\vec{\ell}$ must be consistent with the anyon labels of all plaquettes. For $\mathcal{C} = \text{FIB}$, the allowed combinations of plaquette anyon (DFIB) labels $\mathbf{a} = a_+ a_-$ and tail labels ℓ are $(a_+ a_-)_\ell \in \{11_1, 1\tau_\tau, \tau 1_\tau, \tau\tau_1, \tau\tau_\tau\}$, which are simply all combinations satisfying $\delta_{a_+ a_- \ell} = 1$.

Throughout the remainder of this work, we will 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: $\mathbf{a} = (a_+ a_-)_\ell$. It will always be 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 $|\vec{\ell}, \vec{a}, \vec{b}\rangle$, where $\vec{\ell}$ are the tail labels, $\vec{a}$ are the leaf labels (corresponding to the anyon charge of each plaquette), and $\vec{b}$ are

the internal branch labels. Before embedding them into the fattened lattice, the corresponding ribbon configurations are

$$|\vec{\ell}, \vec{a}, \vec{b}\rangle = \text{Diagram (33)} \quad (33)$$

$$= \sum_{\vec{\alpha}, \vec{\beta}, \vec{k}} X_{\vec{\alpha}, \vec{\beta}, \vec{k}, \vec{l}}^{\vec{\ell}, \vec{a}, \vec{b}} \text{Diagram (34)} \quad (34)$$

where the coefficients $X_{\vec{\alpha}, \vec{\beta}, \vec{k}, \vec{l}}^{\vec{\ell}, \vec{a}, \vec{b}}$ are found by resolving the crossings in Eq. 33 using

$$i \times j = \sum_k \frac{v_k}{v_i v_j} R_k^{ij} \text{Diagram (35)} \quad (35)$$

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 App. A 1. For the Fibonacci category, the only nonzero entries are

$$R_1^{\tau\tau} = e^{\frac{4\pi i}{5}}, \quad R_\tau^{\tau\tau} = e^{-\frac{3\pi i}{5}}, \quad R_a^{1a} = R_a^{a1} = 1, \quad (36)$$

where $a \in \{1, \tau\}$. The necessary calculations to obtain $X_{\vec{\alpha}, \vec{\beta}, \vec{k}, \vec{l}}^{\vec{\ell}, \vec{a}, \vec{b}}$ were performed explicitly in App. A 4.

After picking some embedding in the fattened lattice, we

find

$$|\vec{\ell}, \vec{a}, \vec{b}\rangle = \text{Diagram (37)} \quad (37)$$

$$= \sum_{\vec{\alpha}, \vec{\beta}, \vec{k}, \vec{l}} X_{\vec{\alpha}, \vec{\beta}, \vec{k}, \vec{l}}^{\vec{\ell}, \vec{a}, \vec{b}} \text{Diagram (38)} \quad (38)$$

where we chose not to draw the leaves with vacuum labels explicitly, but included vacuum loops in the corresponding plaquettes instead. The fattened lattice state above corresponds to the case where only 4 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 $\mathcal{DC}$:

$$\text{Diagram (39)} = \sum_f F_{cdf}^{abe} \text{Diagram (39)} \quad (39)$$

$$\text{Diagram (40)} = R_c^{ab} \text{Diagram (40)} \quad (40)$$

$$\text{Diagram (41)} = (R_c^{ba})^* \text{Diagram (41)} \quad (41)$$

$$\text{Diagram (42)} = \theta_a \text{Diagram (42)} \quad (42)$$

$$\text{Diagram (43)} = (\theta_a)^* \text{Diagram (43)} \quad (43)$$

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

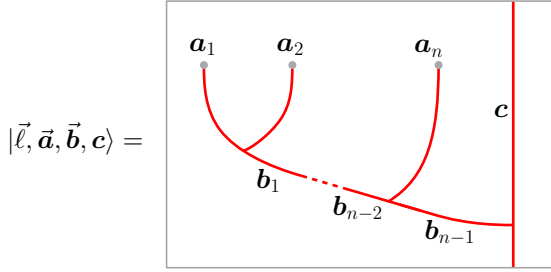
$$F_{cdf}^{abe} = F_{c_+d_+f_+}^{a_+b_+e_+} F_{c_-d_-f_-}^{a_-b_-e_-}, \quad (44)$$

$$R_c^{ab} = R_{c_+}^{a_+b_+} (R_{c_-}^{b_-a_-})^*, \quad (45)$$

$$\theta_a = \theta_{a_+} (\theta_{a_-})^*. \quad (46)$$

We discussed how these are obtained for modular input categories in more detail in App. A 4.

The construction of anyonic fusion basis states on a torus is similar. An important difference is that fusion states of anyons on a torus requires us to specify a *handle label* (see Sec. A 4 b for more details), which determines how the state transforms when an anyon is moved along a non-contractible loop [45].



where the gray box represents the periodic boundary conditions of a torus. Note that the handle label must satisfy $\delta_{b_{n-1}cc}$ 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 of Eq. (47) must again be resolved using Eq. 35, which will lead to superposition of ribbon configurations similar to Eq. (34). These ribbons must then be embedded in the fattened lattice like in Eq. (37), 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 ($a_i = \mathbf{11}$, $b_j = \mathbf{11}$, $\forall i, j$). On a torus, the degenerate ground space is spanned by the states $|\vec{1}, \vec{11}, \vec{11}, c\rangle$, where $\vec{1}$ and $\vec{11}$ 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 $\mathcal{C} = \text{FIB}$), this information is encoded in the handle label. The operations that affect the handle, are precisely those in which anyons interact along a non-contractible 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 will create pairs, triplets or quadruplets of nontrivial anyons in neighboring plaquettes. The initial ground state can

An example of such a fusion state is shown in Fig. 7. Anyonic fusion basis states are then labeled as $|\vec{\ell}, \vec{a}, \vec{b}, c\rangle$, where $\vec{\ell}$, $\vec{a}$ and $\vec{b}$ are again the tail, leaf and internal branch labels, respectively, and c is the handle label. The corresponding ribbon configurations are

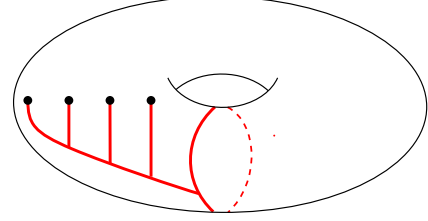


Figure 7: Fusion diagram of a system of anyons defined on a torus. Note the line wrapping around the torus.

$$= \text{[Diagram of resolved crossings]} , \quad (47)$$

then be recovered by fusing these nontrivial anyons pairwise until none are left, without creating any non-contractible loops in the process.

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. IV E). This tensor network representation is derived in App. B.

III. ERROR CORRECTION SCHEME

We now present the circuits to measure and fix arbitrary local errors. From here on, we will work exclusively with the Fibonacci input category, hence we are working with a system of qubits on a lattice. 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

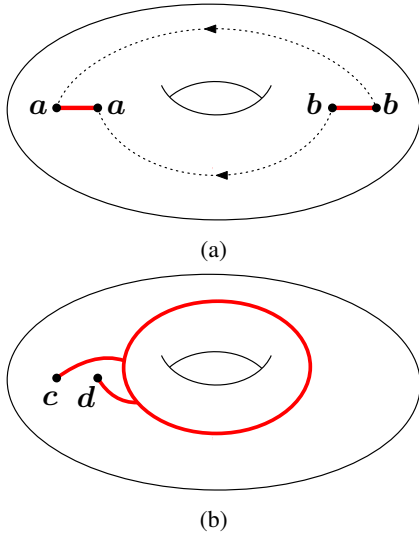


Figure 8: (a) Topologically nontrivial process which results in a logical error: two pairs of anyons (a, a) and (b, b) are created by local qudit errors. The anyons are then fused along the dotted black paths with outcomes c and d . (b) The fusion diagram of the resulting state winds around the torus.

by a measurement projection P_V . This measurement and correction processes projects the many-body state onto the string-net subspace $\mathcal{H}_{s,n}$.

2. After projecting to the string-net subspace $\mathcal{H}_{s,n}$, 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: $\mathcal{H}_\Lambda \subset \mathcal{H}_{s,n}$.

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, the anyon charge measurements in Sec. III B, and the recovery operations in Sec. III C.

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, 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 $e^{i\theta\sigma_x^e}$ will put the system wave function in a superposition of violating and not violating the vertex condition at any adjacent vertex v . When we measure

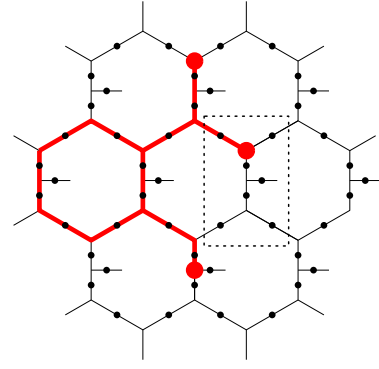
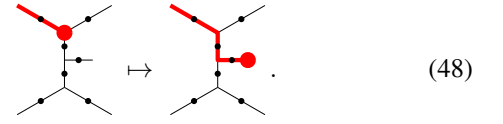


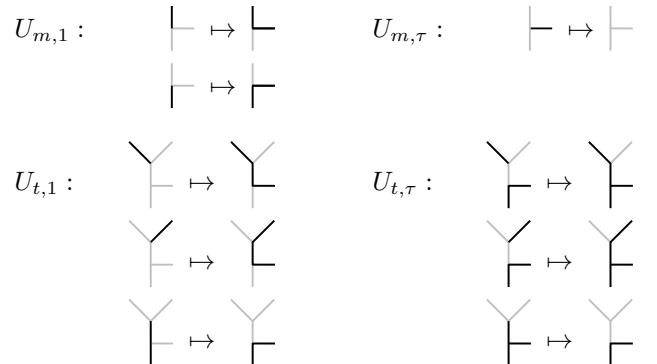
Figure 9: A configuration that violates the branching rules by having string-endings in 3 vertices, indicated the red dots.

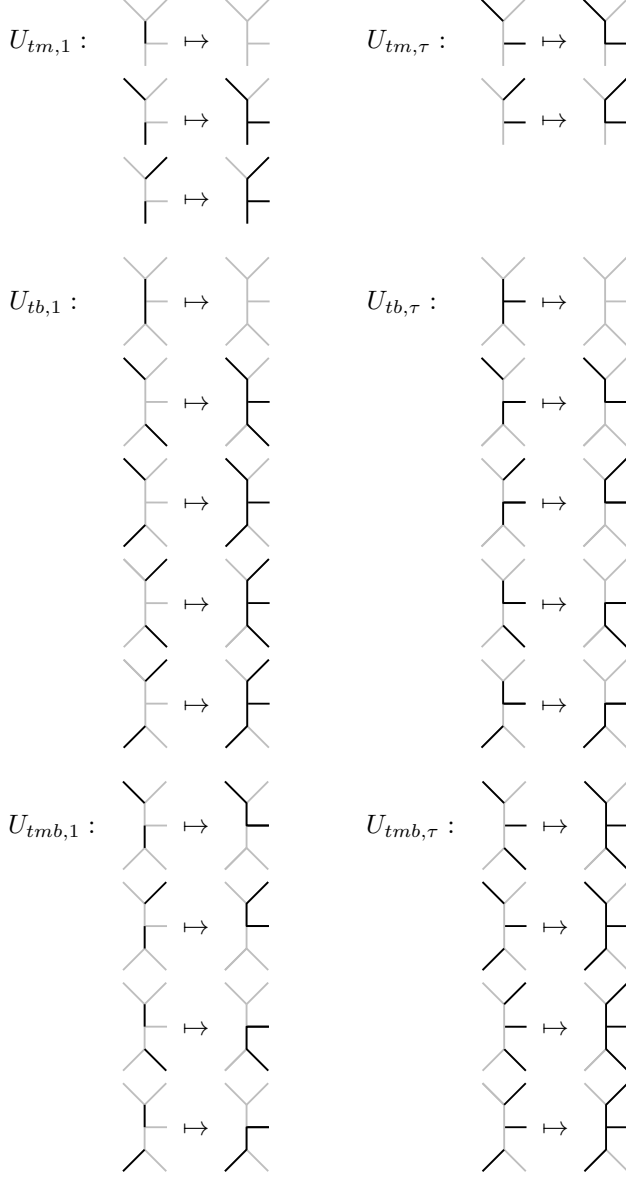
the vertex operator Q_v , we project to either of the two situations. Vertex violations can be resolved by local unitary operators, which take the system back to the string-net subspace $\mathcal{H}_{s,n}$. Intuitively, one can think of the action of these operators as pulling string ends into the tail edge of a neighboring plaquette:



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 Fig. 10(a-c). The circuit for measuring the top and bottom are symmetric, so we only show 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 have omitted the mappings which are mirror symmetric ($t \leftrightarrow b$) to the listed ones.

The corresponding quantum circuits of the above unitaries are listed in Fig. 10(d-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 have parallelized 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 have neglected 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.

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

B. Anyon charge measurements

After measuring the vertex operators and applying the corresponding corrections on the extended string-net code, we have transformed the many-body state to the string-net subspace $\mathcal{H}_{s.n.}$, 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 $\mathcal{H}_{s.n.}$.

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 3 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_{cdf}^{abe} 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, d are control qubits determining the resulting unitary F_{cd}^{ab} , with the matrix elements being $[F_{cd}^{ab}]_{ef}$. For the Fibonacci Turaev-Viro code, the F -matrix is given in Eq. (14). The circuit inside the red dashed box, composed of a 5-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 $\theta = \tan^{-1}(\phi^{-\frac{1}{2}})$. Note that this conditional unitary is only activated 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 2-2 Pachner move (F -move), one can also implement the 1-3 Pachner move with unitary circuit, as shown in Fig. 12. The protocol consists of the following steps: (1) Initialize three ancilla qubits (white dots) in state $|0\rangle$. (2) 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 (a) and (b). (3) Apply

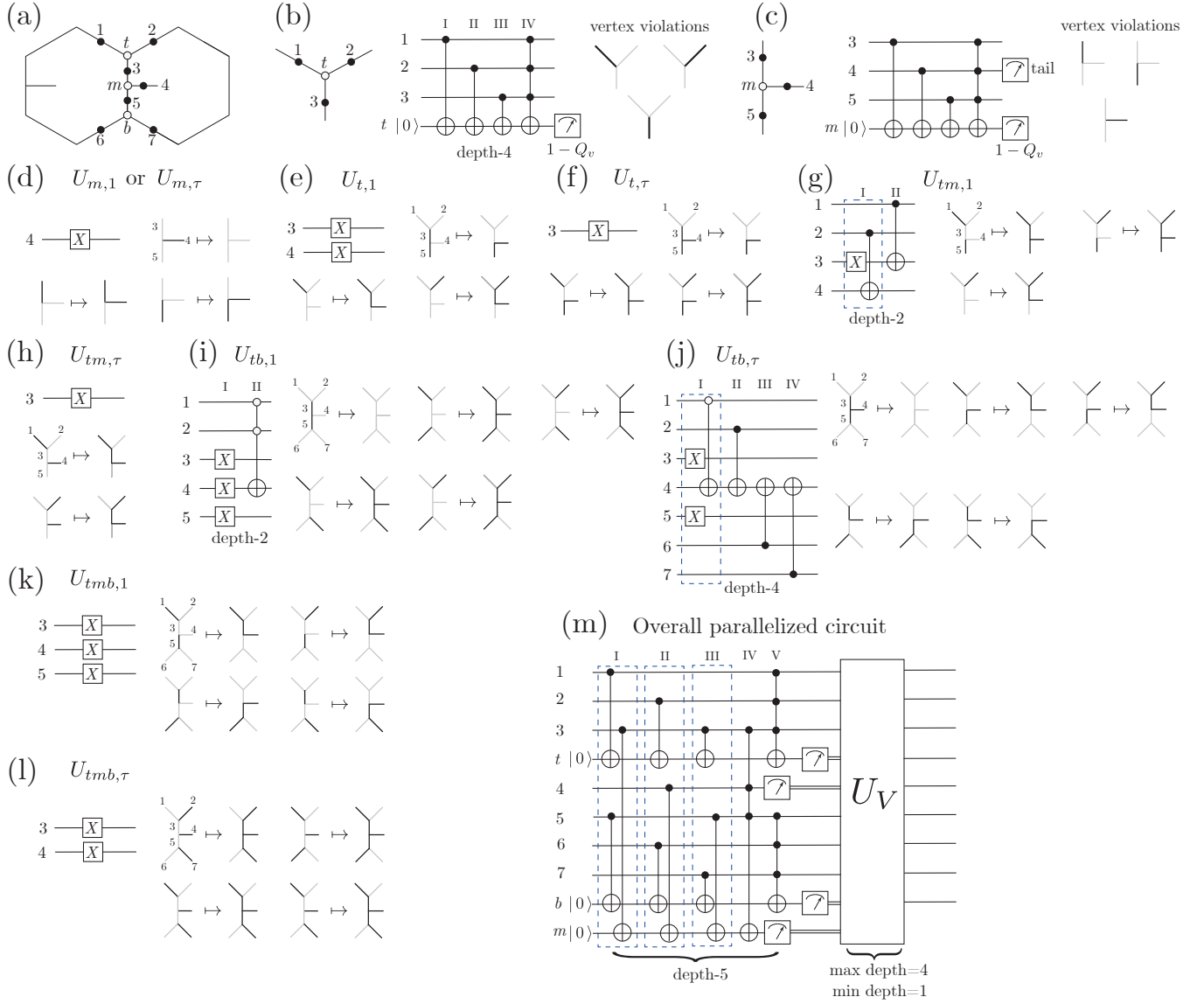


Figure 10: Measurement and correction circuit for the vertex errors. (a) The qubit labeling. The black dots represent data qubits, while the white dots represent ancilla qubits for measurements. (b,c) The measurement circuit for the top and middle vertex projectors. The circuit for measuring the bottom 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).

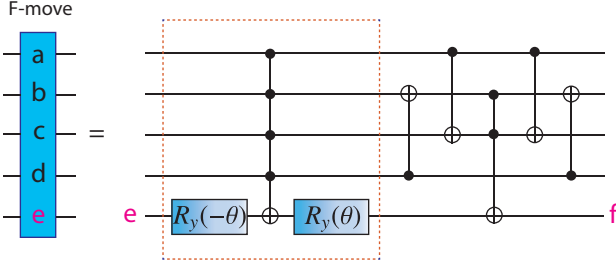


Figure 11: Quantum circuit to implement the F-move (2-2 Pachner move) operation F_{cdf}^{abe} for the Fibonacci Turaev-Viro code.

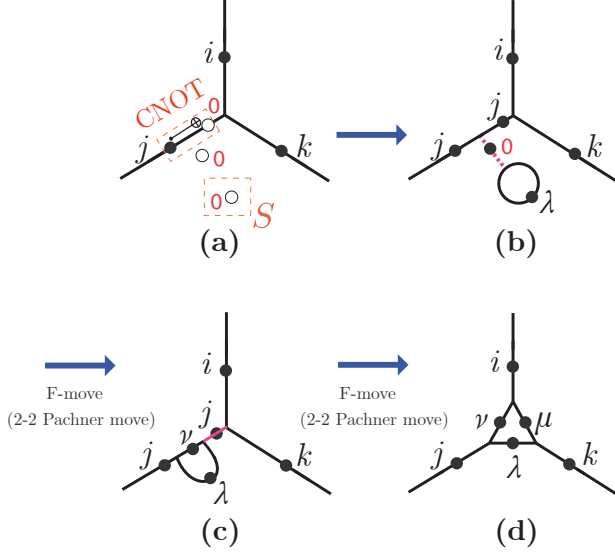


Figure 12: Quantum circuit to implement the 1-3 Pachner move for the Fibonacci Turaev-Viro code.

a modular- $\mathcal{S}$ gate on one ancilla to create a tadpole diagram, as shown in (b). The modular- $\mathcal{S}$ does the following transformation: $\mathcal{S} : |0\rangle \mapsto \sum_{\lambda} \frac{d_{\lambda}}{\mathcal{D}} |\lambda\rangle$. For the Fibonacci Turaev-Viro code, the modular $\mathcal{S}$ -matrix is

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

(4) Apply an F -move to absorb the tadpole onto the edge, as shown in (b) and (c). (5) Apply another F -move to sweep edge λ to attach the right leg (with label k), as shown in (c) and (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 panel (b) to (e), we apply a series of

operations to achieve a 1-3 Pachner move to add a triangle loop below the tail: (1) Apply a CNOT gate which entangles the data qubit on the tail to the new ancilla qubit on the tail, CNOT: $|y\rangle |0\rangle \mapsto |y\rangle |y\rangle$, as shown in (b) and (c). (2) Apply a modular- $\mathcal{S}$ gate on one ancilla to create a tadpole diagram, as shown in (b), i.e., $\mathcal{S} : |0\rangle \mapsto \sum_{\alpha} \frac{d_{\alpha}}{\mathcal{D}} |\alpha\rangle$. (3) Apply an F -move to absorb the tadpole onto the tail, as shown in (c) and (d). (4) Apply another F -move to sweep edge α to attach the left edge, as shown in (d) and (e). Afterwards, 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 have grown 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 9 F -moves have been applied.

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

$$\begin{aligned} \sum_{\alpha} \frac{d_{\alpha}}{\mathcal{D}} \text{ (diagram)} &= \sum_{\alpha} \frac{d_{\alpha}}{\mathcal{D}} \text{ (diagram)} \\ &= \sum_{\alpha, \beta} \frac{d_{\alpha}}{\mathcal{D}} \frac{v_{\beta}}{v_{\alpha} v_y} \text{ (diagram)} \\ &= \sum_{\alpha, \beta, x} \frac{1}{\mathcal{D}} \frac{v_x}{v_y} \text{ (diagram)}. \end{aligned} \quad (50)$$

The sequence of F -moves appearing in the grow circuit corresponds exactly to resolving only the outer tube into the lattice². 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

² 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.

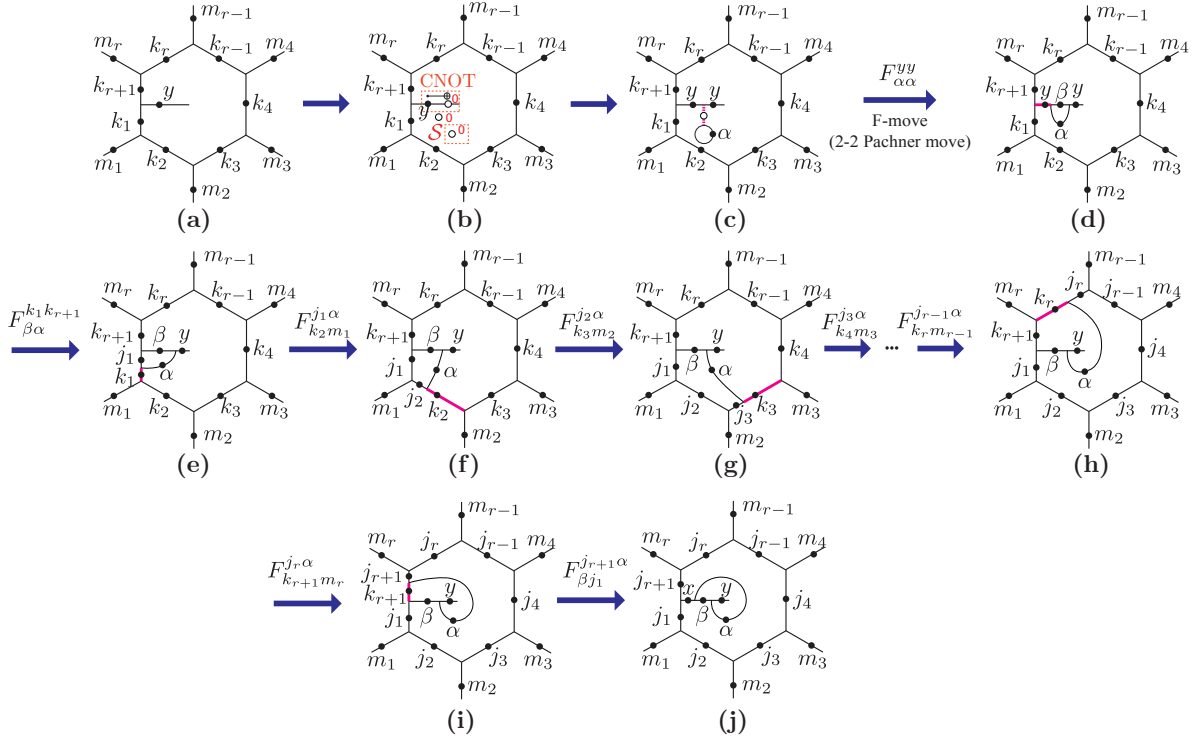


Figure 13: Protocol and circuit of growing a tube onto a puncture in a plaquette on a tailed lattice via a sequence of local gates and Pachner moves.

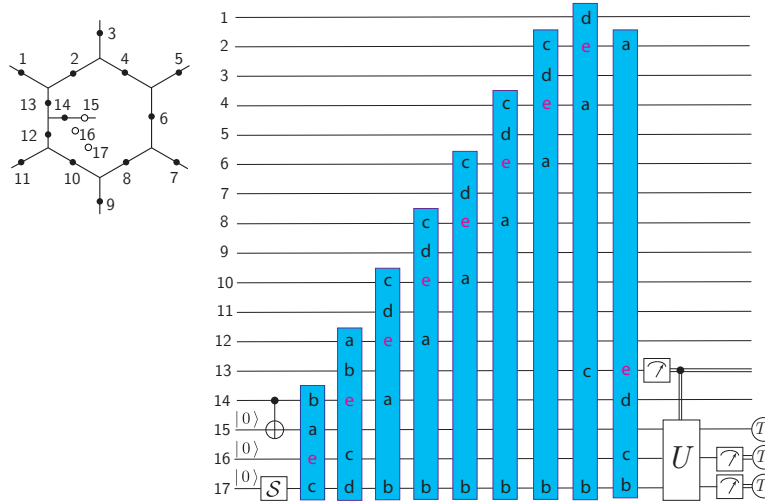


Figure 14: The complete quantum circuit for the joint measurement of the anyon charge and tail label of a plaquette. Note that after the grow circuit, qubit 13 corresponds to the tail edge.

$|000\rangle$ state, then the action of the grow circuit is

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

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

$$\left| \alpha \begin{array}{c} \text{y} \\ \text{y} \\ \beta \\ x \end{array} \right\rangle \equiv |x\rangle \otimes |y\alpha\beta\rangle,$$

and where

$$\tilde{O}_{yx\alpha\beta} \equiv \sum_{\gamma} F_{\alpha y \gamma}^{\alpha x \beta} O_{yx\alpha\gamma} \quad (52)$$

is the operator which corresponds to resolving a outer tube³ into the lattice. If a measurement projects the tube qubits onto the state

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

then the full state gets projected onto

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

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. (30) and (31).

We choose the following basis⁴ for the measurement of the tube qubits:

$$|\psi_{11}\rangle = \frac{1}{\mathcal{D}} \left| \begin{array}{c} 1 \\ 1 \\ 1 \\ 1 \end{array} \right\rangle + \frac{\phi}{\mathcal{D}} \left| \begin{array}{c} 1 \\ \tau \\ 1 \\ 1 \end{array} \right\rangle = |0\rangle \otimes \frac{1}{\mathcal{D}} (|000\rangle + \phi |011\rangle) \equiv |0\rangle \otimes |\tilde{\psi}_{11}\rangle, \quad (55)$$

$$\begin{aligned} |\psi_{1\tau}\rangle &= \frac{1}{\mathcal{D}} \left| \begin{array}{c} 1 \\ \tau \\ \tau \\ \tau \end{array} \right\rangle + \frac{e^{4\pi i/5}}{\mathcal{D}} \left| \begin{array}{c} \tau \\ 1 \\ 1 \\ \tau \end{array} \right\rangle + \sqrt{\phi} \frac{e^{-3\pi i/5}}{\mathcal{D}} \left| \begin{array}{c} \tau \\ \tau \\ 1 \\ \tau \end{array} \right\rangle \\ &= |1\rangle \otimes \frac{1}{\mathcal{D}} (|101\rangle + e^{4\pi i/5} |110\rangle + \sqrt{\phi} e^{-3\pi i/5} |111\rangle) \equiv |1\rangle \otimes |\tilde{\psi}_{1\tau}\rangle, \end{aligned} \quad (56)$$

$$\begin{aligned} |\psi_{\tau 1}\rangle &= \frac{1}{\mathcal{D}} \left| \begin{array}{c} 1 \\ \tau \\ \tau \\ \tau \end{array} \right\rangle + \frac{e^{-4\pi i/5}}{\mathcal{D}} \left| \begin{array}{c} \tau \\ 1 \\ 1 \\ \tau \end{array} \right\rangle + \sqrt{\phi} \frac{e^{3\pi i/5}}{\mathcal{D}} \left| \begin{array}{c} \tau \\ \tau \\ 1 \\ \tau \end{array} \right\rangle \\ &= |1\rangle \otimes \frac{1}{\mathcal{D}} (|101\rangle + e^{-4\pi i/5} |110\rangle + \sqrt{\phi} e^{3\pi i/5} |111\rangle) \equiv |1\rangle \otimes |\tilde{\psi}_{\tau 1}\rangle, \end{aligned} \quad (57)$$

$$|\psi_{\tau\tau,1}\rangle = \frac{\phi}{\mathcal{D}} \left| \begin{array}{c} 1 \\ 1 \\ 1 \\ 1 \end{array} \right\rangle - \frac{1}{\mathcal{D}} \left| \begin{array}{c} \tau \\ 1 \\ \tau \\ 1 \end{array} \right\rangle = |0\rangle \otimes \frac{1}{\mathcal{D}} (\phi |000\rangle - |011\rangle) \equiv |0\rangle \otimes |\tilde{\psi}_{\tau\tau,1}\rangle, \quad (58)$$

$$\begin{aligned} |\psi_{\tau\tau,\tau}\rangle &= \frac{\sqrt{\phi}}{\mathcal{D}} \left| \begin{array}{c} \tau \\ \tau \\ \tau \\ \tau \end{array} \right\rangle + \frac{\sqrt{\phi}}{\mathcal{D}} \left| \begin{array}{c} \tau \\ 1 \\ 1 \\ \tau \end{array} \right\rangle + \frac{1}{\phi \mathcal{D}} \left| \begin{array}{c} \tau \\ \tau \\ \tau \\ \tau \end{array} \right\rangle \\ &= |1\rangle \otimes \frac{1}{\mathcal{D}} (\sqrt{\phi} |101\rangle + \sqrt{\phi} |110\rangle + \frac{1}{\phi} |111\rangle) \equiv |1\rangle \otimes |\tilde{\psi}_{\tau\tau,\tau}\rangle, \end{aligned} \quad (59)$$

³ Note that it has a different shape than our convention Eq. (22), hence the F-matrix in Eq. (52)

⁴ There are seven ways to label a tube according to the Fibonacci fusion rules. Hence, the 7 vectors below span the string-net subspace for the 4 qubits on the tube.

$$|\psi_{\tau\tau,1,\tau}\rangle = \left| \tau \begin{array}{c} \text{1} \\ \tau \\ \tau \end{array} \right\rangle = |1\rangle \otimes |011\rangle \equiv |1\rangle \otimes |\tilde{\psi}_{\tau\tau,1,\tau}\rangle, \quad (60)$$

$$|\psi_{\tau\tau,\tau,1}\rangle = \left| \tau \begin{array}{c} \tau \\ \tau \\ \text{1} \end{array} \right\rangle = |0\rangle \otimes |111\rangle \equiv |0\rangle \otimes |\tilde{\psi}_{\tau\tau,\tau,1}\rangle. \quad (61)$$

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. (30) respectively. In fact, these states are precisely the anyonic fusion basis states on Σ_2 , as described in Eq. (A51).

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 unitaries⁵ conditioned on the measurement of the tail qubit:
 - (a) if the tail qubit is in state $|0\rangle$ (string label **1**):

$$U_1 = |0\rangle \left(|00\rangle \langle \tilde{\psi}_{11}| + |11\rangle \langle \tilde{\psi}_{\tau\tau,1}| + |10\rangle \langle \tilde{\psi}_{\tau\tau,\tau,1}| \right),$$

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

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

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

Once the tube qubits have been 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 POVM:

$$\left\{ \mathcal{P}^{11}, \mathcal{P}^{1\tau}, \mathcal{P}^{\tau 1}, \frac{1}{\phi^2} \mathcal{P}_1^{\tau\tau}, \frac{1}{\phi} \mathcal{P}_\tau^{\tau\tau}, \frac{1}{\phi} \mathcal{P}_1^{\tau\tau}, \frac{1}{\phi^2} \mathcal{P}_\tau^{\tau\tau} \right\}. \quad (62)$$

Note that both a $\tau\tau_1$ and a $\tau\tau_\tau$ excitation corresponds to two different measurement outcomes. However, within each of these pairs, the post-measurement states are not identical. For instance, a $\tau\tau_1$ excitation can result in measurement

outcomes $|\psi_{\tau\tau,1}\rangle$ and $|\psi_{\tau\tau,1,\tau}\rangle$. Obtaining outcome $|\psi_{\tau\tau,1}\rangle$, effectively applies the $\mathcal{P}_1^{\tau\tau}$ idempotent, meaning the post-measurement state will have trivial tail label **1**. On the other hand, the outcome $|\psi_{\tau\tau,1,\tau}\rangle$ corresponds to the application of the $\mathcal{P}_\tau^{\tau\tau}$ nilpotent. Since $\mathcal{P}_1^{\tau\tau} = \mathcal{P}_\tau^{\tau\tau} \mathcal{P}_1^{\tau\tau} \mathcal{P}_1^{\tau\tau}$, this means the plaquette initially contained a $\tau\tau_1$ excitation, which gets transformed to a $\tau\tau_\tau$ excitation in the post-measurement state. The situation for a $\tau\tau_\tau$ excitation is analogous. Hence these measurements do not preserve the tail label in case of a $\tau\tau$ anyon, and a subsequent measurement might yield a different outcome. It is important to note that this only affects the tail label *within one anyon sector*, the anyon label itself cannot be altered by subsequent measurements.

In case one prefers to preserve the tail label of excitations in subsequent measurements⁶, 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. For measurement outcome $|\psi\rangle$ defined Eq. (53), the resolving process is equivalent to the following transformation on the post-measurement 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 will indeed be recovered. For instance, when obtaining the $|\psi_{\tau\tau,1,\tau}\rangle$ measurement outcome, the resolving process result in the application of the $\mathcal{P}_1^{\tau\tau}$ nilpotent on top of the $\mathcal{P}_\tau^{\tau\tau}$ nilpotent applied by the measurement, resulting in the combined action $\mathcal{P}_1^{\tau\tau} = \mathcal{P}_\tau^{\tau\tau} \mathcal{P}_1^{\tau\tau}$, 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 measurement⁷ has a fixed probability of flipping the tail label back to its initial value (as determined by the first measurement). For a $\tau\tau_1$ excitation, it follows from Eq. (62) that the probability that n measurements are required before the post-measurement 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 $\tau\tau_1$ 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 excitations are more likely to appear.

⁵ The operators below must of course be completed to true unitary operators. The missing terms were left out to improve readability.

⁶ Possibly, this could improve the performance of certain decoders. Furthermore, we will assume this for the numerical simulations discussed in Sec IV.

⁷ Provided that no errors happen on qubits in or adjacent to the plaquette between these repeated measurements.

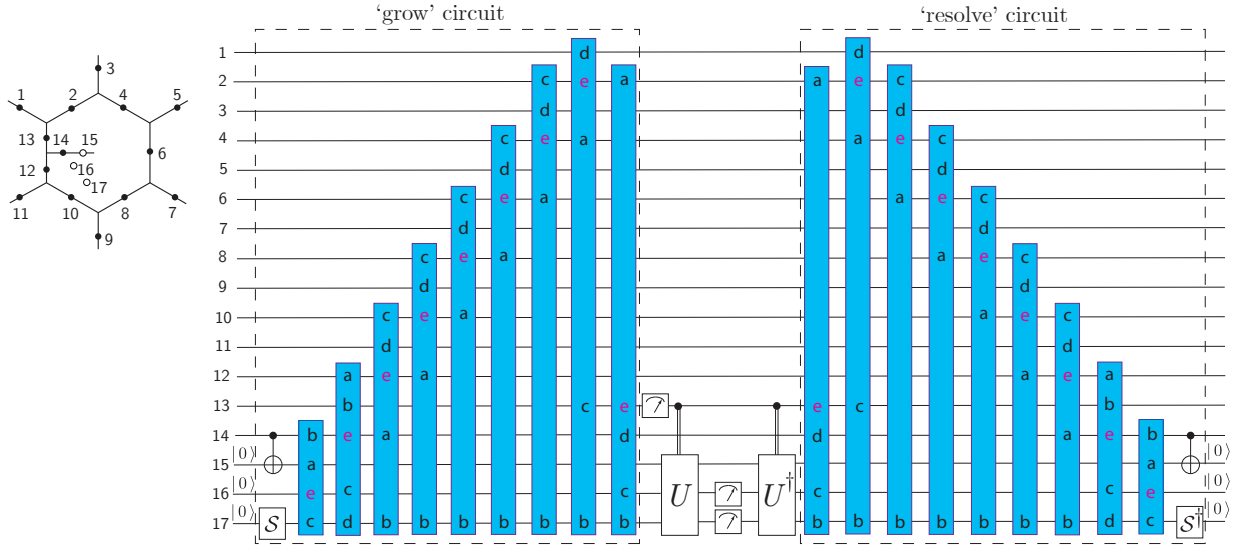


Figure 15: Alternative quantum circuit for the joint measurement of the anyon charge and tail label of a plaquette, which does preserve the tail label.

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 operation to be performed (see Sec. V).

C. Recovery operations

After extracting the error syndrome as described above, the decoding algorithm will suggest a sequence of actions to take to fuse pairs of anyonic excitations along specific paths. There are two types of fundamental recovery operations: *Fuse*, and *Exchange*. Here we will introduce the quantum circuits that implement these recovery operations. The key components of these circuits include the 2-2 Pachner moves and the 1-3 Pachner moves as introduced in Sec. II and the corresponding unitary circuits introduced in Sec. III B.

1. Fuse

We start by defining *fuse* as a recovery operation that fuses anyons in neighboring plaquettes. The fuse protocol is shown in Fig. 16 for the case where an anyon a_2 inside a plaquette is fused with the anyon a_1 inside its left neighboring plaquette. Other fusing directions will have a very similar process and we omit showing all of them. The application of this protocol to a particular example of a ribbon graph state is illustrated in Fig. 17 with further parallelization of the steps in the protocols shown in Fig. 16.

Before detailing the quantum circuit to fuse a pair of neighboring anyons, we consider what such an operation means on the level of the ribbon graphs introduced in Sec. II. We con-

sider two plaquettes p_1 and p_2 , and wish to fuse the anyons contained within them. For this purpose, we pick an anyonic fusion basis which explicitly shows the total charge $b = b^+ b^-$ of the anyons $a_1 = (a_1^+ a_1^-)_{\ell_1}$ and $a_2 = (a_2^+ a_2^-)_{\ell_2}$ contained in the two plaquettes:

$$|\psi_{\vec{b}, \vec{c}}\rangle = \begin{array}{c} a_1 \quad a_2 \\ \diagdown \quad \diagup \\ b \end{array} \begin{array}{c} c_j \quad c_{j+1} \end{array} \equiv \begin{array}{c} \ell_1 \quad \ell_2 \\ \diagdown \quad \diagup \\ b^+ \quad b^- \\ \diagdown \quad \diagup \\ c_j^- \quad c_{j+1}^- \end{array}, \quad (64)$$

where we have denoted all other branch labels on the fusion tree collectively by $\vec{c}$, and have omitted showing all other leaf labels $\{a_3, a_4, \dots\}$. We will refer to these other leaf labels collectively as $\vec{a}_{\text{ot}}$, writing $\vec{a} = [a_1, a_2, \vec{a}_{\text{ot}}]$. For simplicity, we only show the corresponding ribbon graphs, and not their embedding in the fattened lattice as described in Sec. II E.

After the syndrome measurement detailed in Sec. III B, the system is in definite charge eigenstate for all plaquettes, which means that the leaf labels $\vec{a}$ are fixed in the state superposition, while the internal labels $\{b, \vec{c}\}$ are not. In general we then have

$$|\Psi_0\rangle = \sum_{\vec{b}, \vec{c}} \alpha_{\vec{b}, \vec{c}} |\psi_{\vec{b}, \vec{c}}\rangle. \quad (65)$$

The fusion basis state $|\psi_{\vec{b}, \vec{c}}\rangle$ appearing in this decomposition,

can be rewritten as follows:

$$|\psi_{\mathbf{b}, \vec{c}}^{\vec{a}}\rangle = \frac{1}{\mathcal{D}} \dots = \frac{1}{\mathcal{D}} \sum_{\ell=1, \tau} F_{b^+ b^- \ell}^{b^- b^+ 1} \dots \quad (66)$$

where we have used the property that vacuum loops can be “doubled” (see Eq. A46 in Sec. A 4), and have applied an F -move on the double ribbon $b^+ b^-$ of corresponding to the total charge of plaquettes p_1 and p_2 . The fusion of the anyons in these plaquettes corresponds replacing the upper part of the diagram in Eq. 66 by a single puncture as follows

$$\dots \mapsto \dots \quad (67)$$

The fusion can then be represented by the following map on anyonic fusion basis states:

$$\text{fuse} : |\psi_{\mathbf{b}, \vec{c}}^{[\mathbf{a}_1, \mathbf{a}_2, \vec{a}_{\text{ot}}]}\rangle \mapsto \sum_{\ell=1, \tau} F_{b^+ b^- \ell}^{b^- b^+ 1} |\psi_{\vec{c}}^{[\mathbf{b}_\ell, \vec{a}_{\text{ot}}]}\rangle, \quad (68)$$

where⁸

$$|\psi_{\vec{c}}^{[\mathbf{b}_\ell, \vec{a}_{\text{ot}}]}\rangle = \dots \quad (69)$$

⁸ To be precise, we should have written $|\psi_{\mathbf{b}, \vec{c}}^{[\mathbf{b}_\ell, \mathbf{1}\mathbf{1}_1, \vec{a}_{\text{ot}}]}\rangle$, since plaquette p_2 now has a trivial anyonic charge, and the total charge of plaquettes p_1 and p_2 remains unchanged.

The post-measurement state $|\Psi_0\rangle$ then gets mapped to

$$|\Psi_0\rangle \mapsto |\Psi_1\rangle = \sum_{\mathbf{b}, \vec{c}} \alpha_{\mathbf{b}, \vec{c}} \sum_{\ell=1, \tau} F_{b^+ b^- \ell}^{b^- b^+ 1} |\psi_{\vec{c}}^{[\mathbf{b}_\ell, \vec{a}_{\text{ot}}]}\rangle. \quad (70)$$

Measuring the tail label and the anyonic charge of plaquette p_1 after fusing the two anyons, will yield outcomes $\mathbf{b}_\ell$ with the probabilities

$$p(\mathbf{b}_\ell) = \sum_{\vec{c}} \sum_{\ell=1, \tau} \left| \alpha_{\mathbf{b}, \vec{c}} F_{b^+ b^- \ell}^{b^- b^+ 1} \right|^2. \quad (71)$$

Note that the probabilities for the outcomes of a tail measurement, depend only on the total charge $\mathbf{b}$. Furthermore, in any measurement scheme where the anyonic charge of a plaquette is measured independently of its tail label, these measurements will commute.

At the end of the charge measurement cycle, we normally trace out the tube qubits or resolve the tube into the lattice. However, before fusing a pair of anyons, we can keep the tube on the left plaquette (instead of growing it again) in the beginning of the fusion protocol, as shown in Fig. 16(b). The purpose is to carry out the subsequent F -moves for an edge to climb around the tube without encountering any broken string as in the case with a single tail qubit on the left plaquette.

The central idea of the protocol is to merge the two plaquettes and hence the corresponding plaquette excitations, while also combining the tail qubits such that the vertex excitations also get merged. Overall, this process fuses the two anyon charges inside the two plaquettes. The protocol starts at the end of measurement cycle t . One first applies an F -move (2-2 Pachner move) that sweeps the central edge between the two plaquettes towards the left plaquette, as shown in Fig. 16(b). After several F -moves, the central edge now climbs onto the tail of the left plaquette as shown in Fig. 16(d). Now we apply an F -move to move the tail on the right plaquette onto the left tail in Fig. 16(e). In Fig. 16(e), we further sweep the central edge onto the tube. After several steps of further moves, the central edge now forms a triangular bubble with the other two edges in the bottom of the left plaquette, as shown in Fig. 16(h). One now applies a 1-3 Pachner move to absorb the bubble into the vertex in Fig. 16(i), and now the left and right plaquettes are merged into a single large plaquette. The corresponding procedures with the example of the ribbon graph state is also illustrated in Fig. 17(a-h). Now we trace out all the qubits residing on the left tail and only a single tail qubit remains in the lattice, as shown in Fig. 16(j). We are now left with two ribbons going into the same tail and which correspond to the fused anyon charge $\mathbf{a}_1 \times \mathbf{a}_2$. Note that due to the non-Abelian nature, the fused anyon can be in a superposition of multiple anyon charge eigenstates, and hence may not have a definite charge yet. For example, if $\mathbf{a}_1 = \mathbf{a}_2 = \tau \mathbf{1}_\tau \equiv \tau \mathbf{1}$, then one has $\mathbf{a}_1 \times \mathbf{a}_2 = \mathbf{1}\mathbf{1} + \tau \mathbf{1}$.

Since the two initial plaquettes are now merged into a single one, we need to grow another plaquette to recover the original lattice, which is accomplished in Fig. 16(k-p). The growing is achieved by first applying a 1-3 Pachner move in Fig. 16(k) to grow a triangular bubble on the top vertex, followed by a sequence of F -moves that sweep the newly created edge to

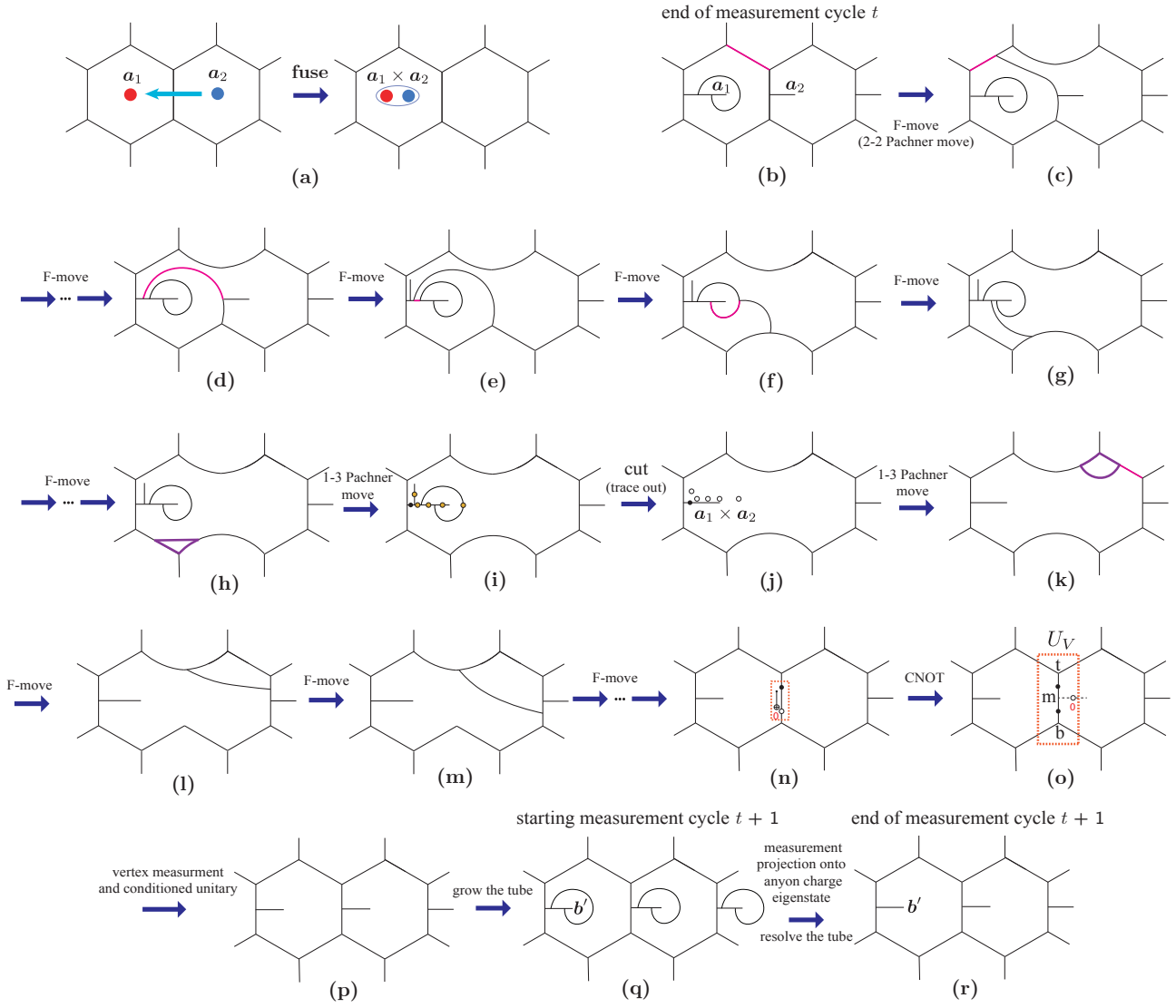


Figure 16: (a) The basic idea of the *fuse* protocol: fuse anyons a_1 and a_2 by moving a_2 on the right plaquette to the left plaquette and then fuse them into a single anyon with charge $a_1 \times a_2$. (b) Starting the fuse protocol in the end of measurement cycle t . The tube on the left plaquette is preserved for the convenience of the fuse protocol. (c) Use an F -move to sweep the middle edge towards the left plaquette. (d) Sweep the central edge onto the left tail. (e) Use an F -move to shuttle the right tail onto the left tail. (f-h) Keep sweeping the middle edge until it reaches the bottom vertex of the left plaquette. (i) Apply a 1-3 Pachner move to shrink the triangular bubble on the bottom vertex. (j) Cut the tubes and extra tail by tracing out all the qubits except the one on the root. Now the two plaquettes have been merged into one and the two anyons are fused into one with charge $a_1 \times a_2$, which is in general in a superposition state. (k) Apply a 1-3 Pachner move to grow a triangular bubble on the top vertex of the right plaquette. (l-n) Use F -moves to keep sweeping the new edge towards the bottom edge and hence grow a new plaquette on the right. Apply a CNOT from the qubit residing on the middle edge to another ancilla qubit initialized at $|0\rangle$, which splits the edge into two. (o, p) Prepare an ancilla qubit at $|0\rangle$ on the new tail. Measure the vertices t , m and b , and apply the unitary U_V conditioned on the measurement result to pull the broken string in and fix all the vertex errors. We have now rebuilt the tailed lattice in (p). (q,r) In the next measurement cycle $t + 1$, start by growing and measuring the tube to project the fused anyon charge $a_1 \times a_2$ to a definite anyon charge b' , and then resolve all the tubes back to the tailed lattice.

form the original middle edge dividing the two plaquettes, illustrated in Fig. 16(n). Now we need to regrow the tail on that middle edge. We first apply a CNOT from the qubit (black dot) residing on the edge on an ancilla qubit (white dot) initialized in the state $|0\rangle$ in order to split the edge in two, as shown

in Fig. 16(n). We further prepare another ancilla qubit (white dot) in the state $|0\rangle$ corresponding to the tail qubit (dashed line) in Fig. 16(o). We then measure the vertex projectors Q_v on the three vertices t , m and b and apply the corresponding unitary U_V to pull the potentially broken string into the tail

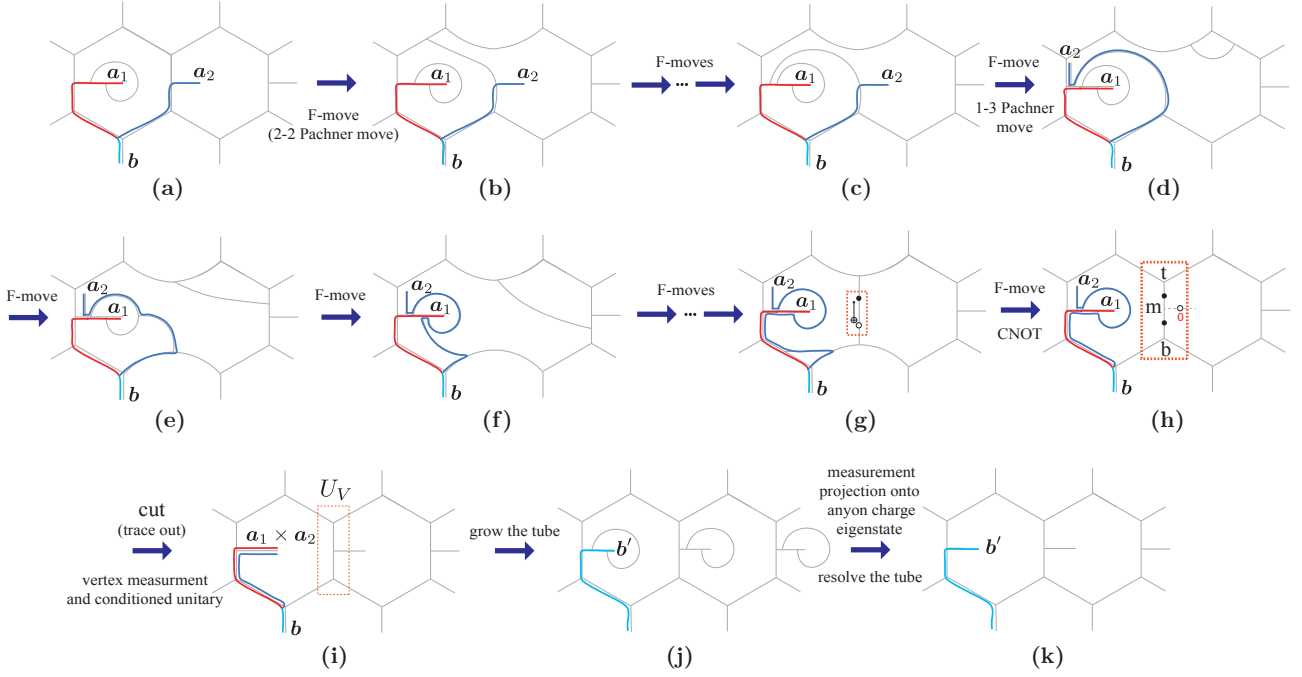


Figure 17: Illustration of the *fuse* protocol with an example of a ribbon graph state with two anyons a_1 and a_2 and their total branch charge b . Note that we have further parallelized the fuse protocol in Fig. 16 by simultaneously doing the plaquette merging and growing a new plaquette on the right.

based on the measurement result V , using the measurement and unitary circuits previously shown in Fig. 10 in Sec. III A. We note that if there is no error introduced in the recovery process, the vertices t , m and b should not have any error and the strings on them should not be broken. However, in reality, additional errors will be introduced during the recovery, and one hence needs to apply the vertex measurements and unitary U_V to pull the string in. Although we have introduced the plaquette growing process separately, we note it can actually be parallelized with the merging of the two original plaquettes, as illustrated in Fig. 17(d-i).

After performing the fusion protocol described here, one obtains a state where the second plaquette has a vacuum charge ($B_p = 1$), while the first plaquette contains the total charge of the two excitations that were fused. Note that the first plaquette is not in a definite charge state, instead, it contains a superposition of all possible values of the total charge of the two fused excitations. Likewise, its tail label is also in a superposition of 1 and τ . In principle, no more actions are required, since the excitations have been fused as expected. However, in the numerical simulation described in Sec. IV, we will assume that after every fusion process, the fusion outcome is projected to some definite anyon label and tail label. This can simply be achieved by including the charge measurement circuit of Sec. III B in the first plaquette at the end of the fusion procedure.

2. Exchange and move

The second type of recovery operation is a counterclockwise/clockwise *exchange* (braiding) of anyons in neighboring plaquettes. The need for this operation arises when one wants to transport an anyon along a path (which might include plaquettes containing nontrivial excitations). In Fig. 18, we show the counterclockwise exchange protocol along with the example of the same ribbon graph state shown above in Fig. 17. The central idea is to rotate the central edge dividing the two plaquette such that the two plaquettes undergo a counterclockwise exchange. One also needs to shuttle the tail accordingly. The final ribbon graph state after the exchange operation is shown in Fig. 17(g), and a counterclockwise exchange/braid can be clearly seen on the ribbon graph. Note that, when applied to excitations with a definite anyon and tail labels, these labels are preserved by the exchange procedure.

We define *move* as a process of shuttling anyon a from plaquette A to plaquette B along some specified path, as illustrated in Fig. 19 with a path $A \rightarrow C \rightarrow B$. This is achieved through a sequence of clockwise or counterclockwise exchanges with all plaquettes along this path. Note that if any other anyons are encountered on the path, the results of the move procedures with clockwise and counterclockwise exchanges are not identical, as they differ by a braid in the anyonic fusion space.

In case all plaquettes along the path have a vacuum charge, one could also implement the movement procure as a se-

quence of fusions, since fusion with a 11 anyon is trivial. However, note that this does not guarantee that the tail label of the excitation that is being moved, is preserved. In particular, when fusing any $\tau\tau$ anyon ($\tau\tau_1$ or $\tau\tau_\tau$) with a neighboring trivial (vacuum) anyon, the probabilities for the measurement outcomes of the tail qubit after the fusion are $1/\phi^2$ and $1/\phi$ for the outcomes $|0\rangle$ and $|1\rangle$, respectively, independent of the tail label of the excitation prior to the fusion. This is due to the fact that *only* the doubled anyon label of an excitation is topologically protected, its tail label is not and can thus be changed by local interactions if one is not careful. Furthermore, the exchange procedure is considerably simpler than the fusion protocol.

IV. 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 was 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. Due to 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 QEC, made in Refs. [22, 23, 26] in Sects. IV A and IV B. We then introduce a microscopic noise model of Pauli errors in Sec. IV C, and study the effect of individual qubit errors on anyonic fusion states in Secs. IV D and IV E. Sec. IV F describes the framework of curve diagrams [46], which is used to efficiently characterize anyonic fusion basis bases during the simulation. Finally, in Sec. IV 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 have been created through the application of noise, their locations and anyon labels can be de-

termined through a syndrome measurement. If we think of the state in terms of fusion trees of anyons, the syndrome measurement only projects 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 dialogue 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 will 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. IV 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 only required to simulate anyon dynamics within each of these groups separately.

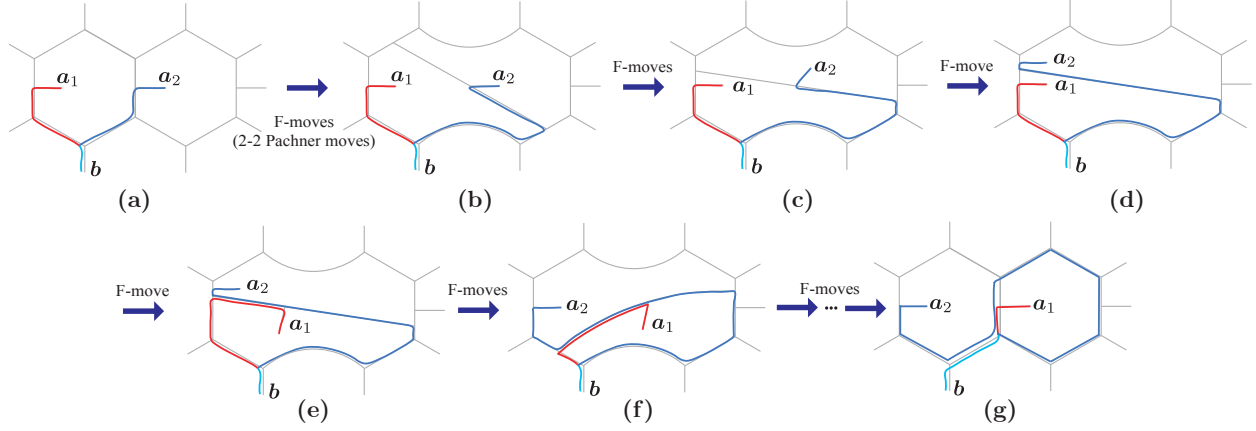


Figure 18: Illustration of the counter-clockwise exchange protocol with an example of ribbon graph state. (a-f) A sequence of F-moves gradually rotates the two plaquettes and the tails accordingly. (g) In the end of the protocol, the position of the two anyons have been exchanged with the ribbons attached to them being braided.

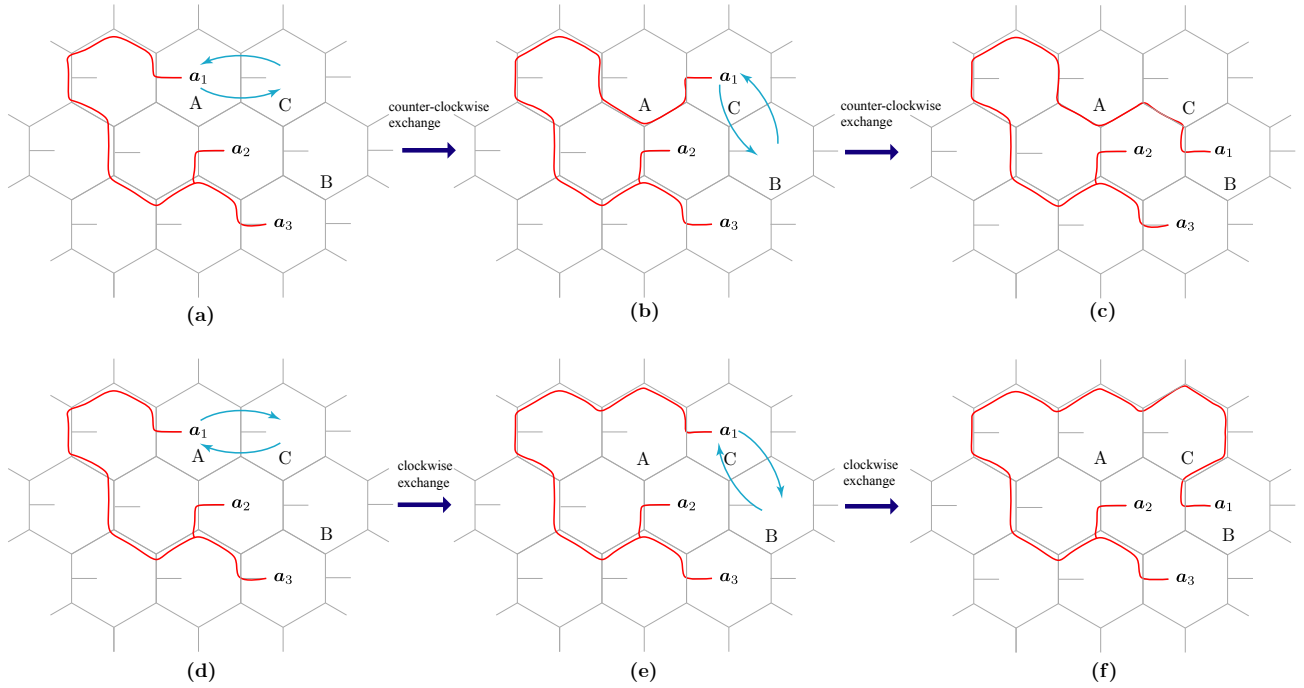


Figure 19: Two equivalent processes to move anyon a_1 from plaquette A to plaquette B through the path $A \rightarrow C \rightarrow B$. (a-c) Implement the move through two steps of counterclockwise exchange between plaquettes A and C, and then plaquettes C and B. The anyon reaches plaquette B in (C). (d-f) Implement the move through two steps of clockwise exchange of neighboring plaquettes. Note that the ribbon graph states (b) and (e) are equivalent because the exchanges happen with vacuum plaquettes and since ribbons can be pulled across those (see Eq. (A45) in Sec. A 4, and also Fig. 3 in Sec. II B). In the same way, the state in (c) is equivalent to the one in (f). However, if any plaquette on the movement path contains a nontrivial anyon, the resulting states are not equivalent, since they correspond to different sequences of braid-moves on the fusion state.

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 [47]. As this is a probabilistic statement, there will be 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 will have dimension $\dim = O(\text{poly}(L))$, 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 where 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 was verified numerically [47]. To ensure that this still holds for the microscopic model studied here, we explicitly verified 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 App. C.

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 will percolate and the fusion space will no longer factorize into small disconnected parts. In this case classical simulation of braiding and fusion will become 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 will lie 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

monitored⁹. 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 tiled honeycomb lattice with periodic boundary conditions and a total of L^2 plaquettes, the total number of edges (and hence qubits) is $|E| = 5L^2$, resulting in $T_0 = 5L^2p$.) 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 [$\sim 1/(|E|p)$].

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 were introduced in Sec III, their purpose is to move the system back to the string-net subspace $\mathcal{H}_{\text{s.n.}}$. 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 will 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 have been applied, and if no logical error was 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

⁹ Note that under these assumptions, it is not appropriate to characterize the noise in terms of an i.i.d. noise strength per qubit, since the non-Abelian nature of our code implies that the combined action of individual error and measurement operators do not commute for consecutive errors. We discuss the relation with i.i.d. noise in App. D

the recovery process.

While the connection between percolation and logical errors in topological codes is not exact [48] (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 non-contractible loop. Our Monte Carlo simulations (detailed below in Sec. IV 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 i.i.d. noise model, as the random qubits affected by Pauli errors are unlikely to be in each others vicinity. The existence of an error correction threshold for our fixed-rate sampling noise model therefore implies the existence of a threshold for an i.i.d. noise model as well.

D. Pauli-noise in the anyonic fusion basis

We will 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

$$|\Psi_0\rangle = \sum_t \alpha_t |\psi_t\rangle, \quad (72)$$

where the $|\psi_t\rangle$ represent anyonic fusion basis states Eq. (37), 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 $\mathcal{H}_{\text{s.n.}}$, and the resulting state can again be decomposed in the anyonic fusion basis:

$$|\Psi_1\rangle = \sigma_z^e |\Psi_0\rangle = \sum_t \alpha_t \sigma_z^e |\psi_t\rangle \quad (73)$$

$$= \sum_{t'} |\psi_{t'}\rangle \left(\sum_t \alpha_t \langle \psi_{t'} | \sigma_z^e | \psi_t \rangle \right), \quad (74)$$

where we have used that $\sum_{t'} |\psi_{t'}\rangle \langle \psi_{t'}|$ acts as the resolution of the identity within $\mathcal{H}_{\text{s.n.}}$. 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 will also have that same label, unless a logical error has occurred 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 only consider 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) (24), (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_z^e .

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

$$|\psi_{\vec{b}, \vec{c}}^{\vec{a}}\rangle = \begin{array}{c} a_1 \quad a_2 \\ \diagdown \quad \diagup \\ \text{---} c_j \quad b \quad c_{j+1} \text{---} \end{array}, \quad (75)$$

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. (72) as

$$|\Psi_0\rangle = \sum_{\vec{b}, \vec{c}} \alpha_{\vec{b}, \vec{c}} |\psi_{\vec{b}, \vec{c}}^{\vec{a}}\rangle. \quad (76)$$

Note that we dropped 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 of Eq. (73) are block diagonal in these labels, and the expression for the state $|\Psi_1\rangle = \sigma_z^e |\Psi_0\rangle$ can be reduced to

$$|\Psi_1\rangle = \sum_{\vec{b}, \vec{c}} \sum_{\vec{a}'} |\psi_{\vec{b}, \vec{c}}^{\vec{a}'}\rangle \left(\alpha_{\vec{b}, \vec{c}} \langle \psi_{\vec{b}, \vec{c}}^{\vec{a}'} | \sigma_z^e | \psi_{\vec{b}, \vec{c}}^{\vec{a}} \rangle \right). \quad (77)$$

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

$$p(\vec{a}') = \sum_{\vec{b}, \vec{c}} \left| \alpha_{\vec{b}, \vec{c}} \langle \psi_{\vec{b}, \vec{c}}^{\vec{a}'} | \sigma_z^e | \psi_{\vec{b}, \vec{c}}^{\vec{a}} \rangle \right|^2. \quad (78)$$

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

$$|\Psi_2\rangle = \frac{1}{\sqrt{p(\vec{a}')}} \sum_{\vec{b}, \vec{c}} |\psi_{\vec{b}, \vec{c}}^{\vec{a}'}\rangle \left(\alpha_{\vec{b}, \vec{c}} \langle \psi_{\vec{b}, \vec{c}}^{\vec{a}'} | \sigma_z^e | \psi_{\vec{b}, \vec{c}}^{\vec{a}} \rangle \right), \quad (79)$$

which is again a superposition of basis states with fixed leaf labels. Note that the matrix elements on the right hand side of Eqs. (78) and (79) are independent of the unaffected labels $\vec{c}$. Hence, we may replace them with matrix elements of the form $\langle \psi_{\vec{b}}^{\vec{a}'} | \sigma_z^e | \psi_{\vec{b}}^{\vec{a}} \rangle$, where $|\psi_{\vec{b}}^{\vec{a}}\rangle$ are states that only contain 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 $|\psi_{\vec{b}}^{\vec{a}}\rangle$ [i.e., for the embedding of the fusion tree in Eq. (75) 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 only need to consider four orientations for e . Our choice of basis convention for the σ_z^e matrix elements is given in Fig. 20.

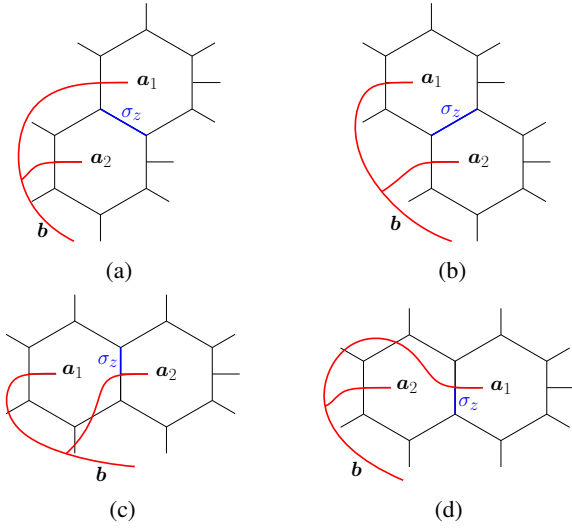


Figure 20: Basis convention for the affected plaquette charges for all nontrivial orientations of the edge e (highlighted in blue) in the case of a σ_z error.

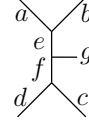
2. Pauli-X and Y errors

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

$$|\Psi_1\rangle = \sum_t \alpha_t \sigma_x^e |\psi_t\rangle \quad (80)$$

does not necessarily belong to the string-net subspace $\mathcal{H}_{\text{s.n.}}$ 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 will belong to a set of 7 connected qubits on the lattice that we will call a *segment*:



The three vertices in each segment will be 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

$$p(V) = \langle \Psi_1 | P_V | \Psi_1 \rangle = \sum_{t', t} \bar{\alpha}_{t'} \alpha_t \langle \psi_{t'} | \sigma_x^e P_V \sigma_x^e | \psi_t \rangle, \quad (81)$$

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

$$P_V = (1 - Q_{v_1})(Q_{v_2})(|\tau\rangle_{q_1} \langle \tau|_{q_1}). \quad (82)$$

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

$$|\Psi_2\rangle = \frac{1}{\sqrt{p(V)}} \sum_t \alpha_t U_V P_V \sigma_x^e |\psi_t\rangle = \frac{1}{\sqrt{p(V)}} \sum_{t'} |\psi_{t'}\rangle \left(\sum_t \alpha_t \langle \psi_{t'} | U_V P_V \sigma_x^e | \psi_t \rangle \right), \quad (83)$$

where we have again inserted the resolution of the identity $\sum_{t'} |\psi_{t'}\rangle \langle \psi_{t'}|$ in $\mathcal{H}_{\text{s.n.}}$ 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_z^e error, the expressions (81) and (83) 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^e operator acting on an edge e of the tiled 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 case of 4 affected plaquettes, the basis states have the form

$$|\psi_{\vec{b}, \vec{c}}^{\vec{a}, \vec{d}}\rangle = \begin{array}{c} \begin{array}{cccc} a_1 & a_2 & a_3 & a_4 \\ & d_1 & & \\ & & d_2 & \\ \dots & c_j & b & c_{j+1} \dots \end{array} \end{array}, \quad (84)$$

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

Eq. (72) is split into a sum over the unaffected labels $\vec{c}$, the total charge $\vec{b}$, and the branch labels $\vec{d}$ of the affected part:

$$|\Psi_0\rangle = \sum_{\vec{b}, \vec{c}, \vec{d}} \alpha_{\vec{b}, \vec{c}}^{\vec{d}} |\psi_{\vec{b}, \vec{c}}^{\vec{a}, \vec{d}}\rangle. \quad (85)$$

As all matrix elements are block diagonal in the unaffected labels, the expression for the vertex measurement outcome probability Eq. (81) reduces to

$$p(V) = \sum_{\vec{b}, \vec{c}, \vec{d}} \sum_{\vec{d}'} \bar{\alpha}_{\vec{b}, \vec{c}}^{\vec{d}'} \alpha_{\vec{b}, \vec{c}}^{\vec{d}} \langle \psi_{\vec{b}, \vec{c}}^{\vec{a}, \vec{d}'} | \sigma_x^e P_V \sigma_x^e | \psi_{\vec{b}, \vec{c}}^{\vec{a}, \vec{d}} \rangle. \quad (86)$$

The sum over t' in Eq. (83) 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

$$|\Psi_2\rangle = \frac{1}{\sqrt{p(V)}} \sum_{\vec{a}', \vec{d}'} \sum_{\vec{b}, \vec{c}} |\psi_{\vec{b}, \vec{c}}^{\vec{a}', \vec{d}'}\rangle \left(\sum_{\vec{d}} \alpha_{\vec{b}, \vec{c}}^{\vec{d}} \langle \psi_{\vec{b}, \vec{c}}^{\vec{a}', \vec{d}'} | U_V P_V \sigma_x^e | \psi_{\vec{b}, \vec{c}}^{\vec{a}, \vec{d}} \rangle \right). \quad (87)$$

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

$$p(\vec{a}') = \frac{1}{p(V)} \sum_{\vec{d}'} \sum_{\vec{b}, \vec{c}} \left| \sum_{\vec{d}} \alpha_{\vec{b}, \vec{c}}^{\vec{d}} \langle \psi_{\vec{b}, \vec{c}}^{\vec{a}', \vec{d}'} | U_V P_V \sigma_x^e | \psi_{\vec{b}, \vec{c}}^{\vec{a}, \vec{d}} \rangle \right|^2. \quad (88)$$

The resulting state after this charge measurement is

$$|\Psi_3\rangle = \frac{1}{\sqrt{p(\vec{a}')p(V)}} \sum_{\vec{d}'} \sum_{\vec{b}, \vec{c}} |\psi_{\vec{b}, \vec{c}}^{\vec{a}', \vec{d}'}\rangle \left(\sum_{\vec{d}} \alpha_{\vec{b}, \vec{c}}^{\vec{d}} \langle \psi_{\vec{b}, \vec{c}}^{\vec{a}', \vec{d}'} | U_V P_V \sigma_x^e | \psi_{\vec{b}, \vec{c}}^{\vec{a}, \vec{d}} \rangle \right). \quad (89)$$

Once again, the matrix elements on the right hand side of Eqs. (86), (88) and (89) are independent of the unaffected labels $\vec{c}$, so we only require matrix elements of the form $\langle \psi_{\vec{b}}^{\vec{a}', \vec{d}'} | O | \psi_{\vec{b}}^{\vec{a}, \vec{d}} \rangle$ 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. 21. 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. 21.

In summary, all that is required to capture the effect of

Pauli-noise on the state of the system are the following matrix elements:

$$\langle \psi_{\vec{b}}^{\vec{a}, \vec{d}} | \sigma_x^e P_V \sigma_x^e | \psi_{\vec{b}}^{\vec{a}, \vec{d}} \rangle, \quad (90)$$

$$\langle \psi_{\vec{b}}^{\vec{a}', \vec{d}'} | U_V P_V \sigma_x^e | \psi_{\vec{b}}^{\vec{a}, \vec{d}} \rangle, \quad (91)$$

$$\langle \psi_{\vec{b}}^{\vec{a}, \vec{d}} | \sigma_y^e P_V \sigma_y^e | \psi_{\vec{b}}^{\vec{a}, \vec{d}} \rangle, \quad (92)$$

$$\langle \psi_{\vec{b}}^{\vec{a}', \vec{d}'} | U_V P_V \sigma_y^e | \psi_{\vec{b}}^{\vec{a}, \vec{d}} \rangle, \quad (93)$$

$$\langle \psi_{\vec{b}}^{\vec{a}', \vec{d}'} | \sigma_z^e | \psi_{\vec{b}}^{\vec{a}, \vec{d}} \rangle. \quad (94)$$

These matrix elements must be calculated for all possible orientations of the edge e in Figs. 20 and 21, 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.

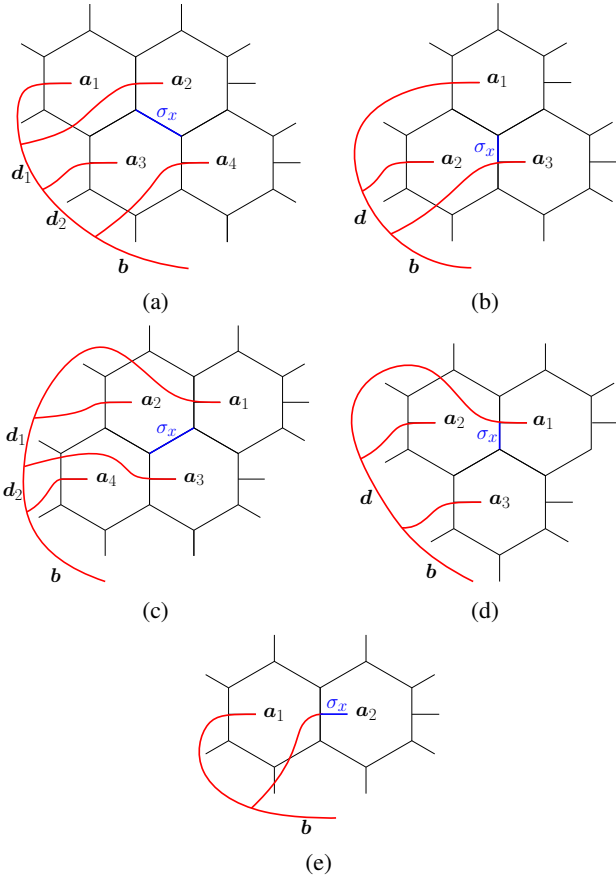
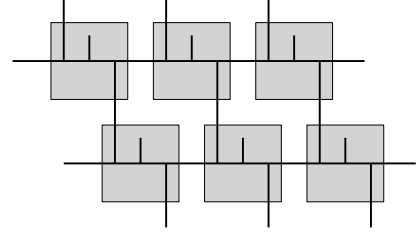


Figure 21: Basis convention for the affected plaquette charges for the different orientations of e (highlighted in blue) in the case of a σ_x or a σ_y error.

E. Computing the relevant matrix elements

At first sight, calculating the matrix elements listed in Eqs. (90)-(94) 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 App. B, 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 will illustrate this procedure for the matrix element Eq. (90), where e is an edge with the orientation depicted in Fig. 21(a). All other matrix elements can be computed in an analogous manner.

For simplicity, we will work with square PEPS tensors (see App. B 3), which each correspond to one segment of the lattice as shown in the following diagram:



where we have rotated the lattice by 60° counterclockwise. The PEPS tensors will be 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. 21(a) looks as follows:

$$|\psi_{\vec{a}, \vec{d}}\rangle = \dots \quad (95)$$

The diagram shows a tensor network contraction of gray and red squares. The gray squares represent segments of the lattice where the tails end inside plaquettes containing trivial DFIB charges, and the red squares represent segments where the tails end inside plaquettes containing nontrivial DFIB charges. The lattice is rotated by 60° counterclockwise. The diagram is labeled with $|\psi_{\vec{a}, \vec{d}}\rangle = \dots$ and (95).

The crossings and branchings in this diagram imply the presence of certain crossing and fusion tensors. Details of these are given in App. B. The total charge b of the four leaf charges in Fig. 21(a) was assigned to a neighboring segment. As was 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 $\langle \psi_{\vec{a}', \vec{d}'} |$. 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 a_2 and 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

$$\langle \psi_{\vec{b}'}^{\vec{a}'} | \sigma_x^e P_V \sigma_x^e | \psi_{\vec{b}}^{\vec{a}} \rangle = \dots \quad (96)$$

where

$$\text{[blue tensor]} = \text{[red tensor]} \quad \text{and} \quad \text{[blue tensor]} = \text{[red tensor]} \quad (97)$$

In order to avoid too much clutter, we have omitted the different labels in the graphical notation. The labels $\vec{a}', \vec{b}, \vec{d}'$ and $\vec{a}, \vec{b}, \vec{d}$ are always implied for the top and bottom layer, respectively. Note that the σ_x^e operator was 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. (96) 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. (96) correspond to segments with a trivial charge. This infinite contraction can then be simplified by determining the top fixed point MPS of the double layer transfer matrix [49]:

$$\dots \text{[grid of tensors]} \dots = \dots \text{[MPS diagram]} \dots \quad (98)$$

After finding a similar bottom fixed point MPS, Eq. (96) can be reduced to

$$\langle \psi_{\vec{b}'}^{\vec{a}'} | \sigma_x^e P_V \sigma_x^e | \psi_{\vec{b}}^{\vec{a}} \rangle = \dots \quad (99)$$

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

$$\text{[MPS diagram]} = \text{[MPS diagram]} \quad (100)$$

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

the finite contraction

$$\langle \psi_{\vec{b}'}^{\vec{a}'} | \sigma_x^e P_V \sigma_x^e | \psi_{\vec{b}}^{\vec{a}} \rangle = \text{[finite contraction diagram]} \quad (101)$$

which can be computed in the regular way. In Eqs. (98) and (100) we have implicitly assumed that the PEPS tensors were 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. (101) should

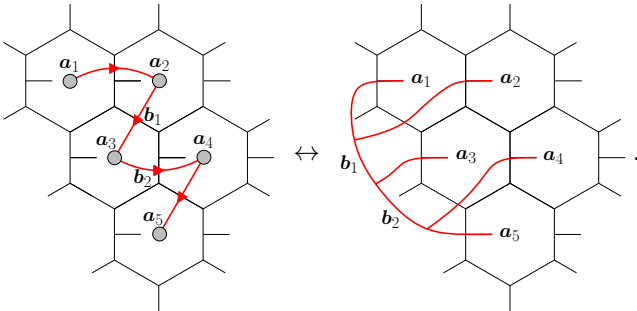
therefore be divided by $\sqrt{\langle \psi_{\vec{b}}^{\vec{a}, \vec{d}} | \psi_{\vec{b}}^{\vec{a}, \vec{d}} \rangle \langle \psi_{\vec{b}}^{\vec{a}', \vec{d}'} | \psi_{\vec{b}}^{\vec{a}', \vec{d}'} \rangle}$ in order to obtain a matrix element that is properly normalized.

F. Storing and manipulating anyonic fusion basis states

As explained in Sec. II E, states in $\mathcal{H}_{\text{s.n.}}$ 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 $\mathcal{H}_{\text{s.n.}}$ 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. IV 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.

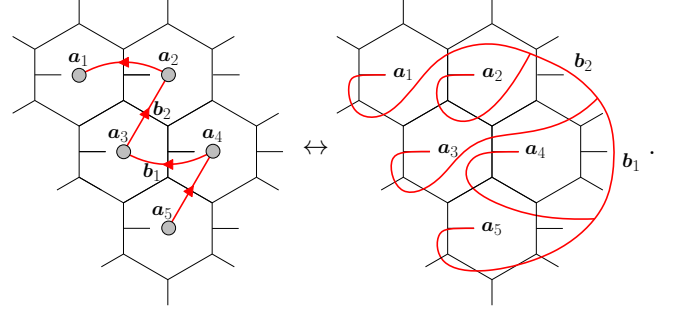
For each connected group of anyons, we can pick out a linear ordering by drawing a directed curve connecting all the anyons (which we locate at the center of their corresponding plaquette) within the group. Directed curves, corresponding to the same linear ordering and only differing by continuous deformations that keep the anyon positions fixed, form a set of equivalence classes that we shall call *curves* or *curve diagrams*. Each configuration of (non-intersecting) curves on the surface corresponds to an equivalence class of anyonic fusion bases that are related up to local Dehn twists in individual plaquettes.

A rigorous definition of curve diagrams, using the language of modular functors, can be found in Ref. [46]. For our purposes however, the following simplified construction will suffice. For a given curve containing n anyons, the corresponding fusion basis (up to local Dehn twists) is found as follows: start by drawing a fusion tree connecting only the first and last anyons, in a way that is homologically equivalent to the curve. Next, following the orientation provided by the curve, add anyons $2, 3, \dots, n-1$ to this fusion tree one by one, by adding a leaf to the tree on the left hand side. This is best illustrated with an example:



A different basis for the same fusion space corresponding to a

different curve diagram would be given by



Note that the way in which the leaf ribbons wrap around the plaquettes containing anyons is not uniquely determined by this construction. It is precisely this freedom that translates to local Dehn twists in individual plaquettes. While the construction could be modified in order to remove this freedom, there is no need for us to bother with such details. Our simulations are set up such that we never need to store any superposition in the anyon labels of individual plaquettes. As such, local Dehn twists give rise to global phases, which have no physical relevance.

The relation between curve diagrams and anyonic fusion bases detailed above, does not specify the handle labels. However, the nature of the simulation does not require the ability to store these values. The handle label (or superposition of such labels), can only be modified by processes in which a pair of anyons on the same curve interact along a path that is *not* homologically equivalent to the piece of curve between them. Since such events are precisely those for which the simulation declares a failure, we do not need the ability to update the handle label. Furthermore, since the total charge of all created excitations is trivial, the value of the handle label has no influence on the outcome of any physical process. Hence, all we need to store is the fusion state of anyonic excitations created *on top of* some initial ground state.

1. Data-structure

In order to simulate the noise and error correction processes, one needs to be able to efficiently store and update the basis in which the quantum state of the system is expressed. The curve diagrams, as defined above, provide a convenient way of doing so. An efficient method for storing the curve diagrams describing the current basis was introduced in Ref. [26]. We slightly modify this construction in order for it to be better suited for the system at hand.

We will store the configuration of curves by storing the shape of the curves running through each individual plaquette separately. Since the presence of the tail edges is irrelevant in this context, we will represent the curve diagram layout on hexagonal *tiles* (each of which corresponds to a plaquette). The total configuration of the curves can be obtained by piecing all tiles together. In order to represent the curves of the connected groups of anyons on the lattice, we assign to each connected curve a unique integer label. An example of three

connected curves on a 4×4 lattice is depicted in Fig. 22(a). Since they represent the layout of the curve diagrams on the lattice, each tile may contain at most one anyon, and we require that the different curves intersect the edges of tiles transversely.

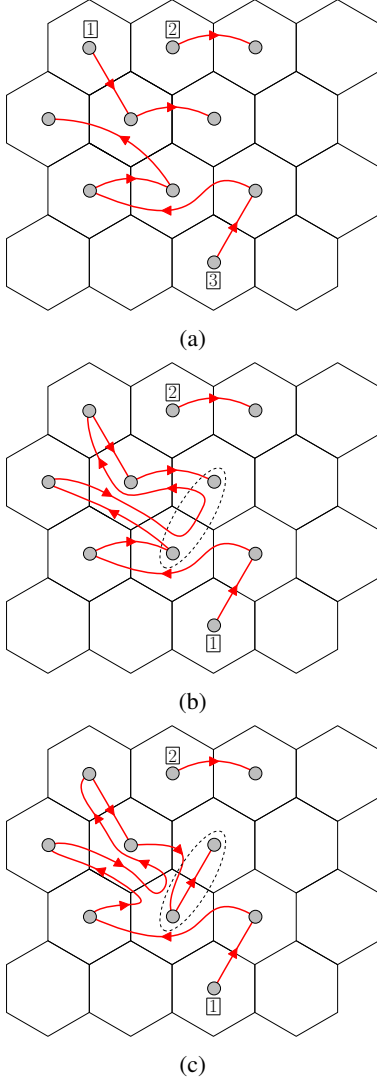


Figure 22: (a): Example of a configuration of three connected curves on a 4×4 periodic hexagonal lattice. Each curve is assigned a unique integer label, depicted here in the square boxes at the start of each curve. (b): Result of a merge procedure in the case the two anyons in the dashed ellipse interact. (c): Result of repeated use of refactoring moves to make the the anyons in the dashed ellipse appear sequentially in the same curve.

In order to save the configuration of curves inside a tile we first assign a unique integer label to each *piece of curve* inside the tile. By a *piece of curve*, we mean a segment of a curve diagram connecting either two edges of the tile, or an edge and an anyon. Note that every piece of curve has an orientation. The edges of a tile are numbered clockwise from 1 to 6 as depicted in Fig. 23(a). For each edge, we record the label and

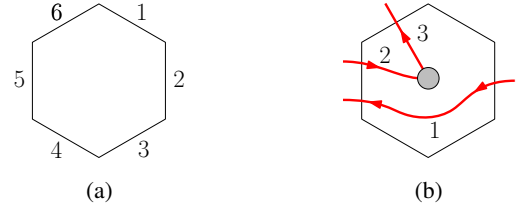


Figure 23: (a): Labeling of the edges of a hexagonal tile in a clockwise direction. (b): Example of a configuration of lines running through tile 10 in Fig. 22(a) (when counting from left to right and from top to bottom). Each line segment is assigned a unique integer label.

orientation (+1 for an incoming line, -1 for an outgoing line) of each piece of curve intersecting it, in the order in which they are encountered when going clockwise around the tile. In addition, when an anyon is present inside the tile, we also store which pieces of curve are connected to it. Finally, for every piece of curve inside the tile, we store the curve label, along with its position inside that curve. If a piece of curve is positioned between the n th and $n + 1$ th anyons on a curve (following the orientation of the curve), its position label is n . An example of such a tile is depicted in Fig. 23(b). The configuration inside this tile would be stored as follows:

$$\begin{bmatrix} 1 & +1 \\ 2 & -1 \\ 3 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \begin{bmatrix} 1 & 3 & 2 \\ 2 & 3 & 3 \\ 3 & 3 & 4 \end{bmatrix}. \quad (102)$$

Here, the first six arrays correspond to the labels and orientations of lines crossing each of the six tile edges, respectively. The next to last array indicates that lines 2 and 3 are connected to the anyon in the tile. The last array indicates the curve and position along the curve of each piece of curve. In this case, all lines belong to curve 3, and the lines with labels 1, 2, 3 appear in the curve after the second, third and fourth anyon along the curve, respectively.

2. Merge

If neighboring anyons on the lattice that lie on different connected curves interact at some point, their corresponding curves must be merged in order to compute the effect of the interaction. In terms of fusion trees, one must think of this procedure as connecting two separate fusion diagrams with a line carrying the trivial label.

The curves are merged by connecting the end of one curve to the start of the other, with the condition that the combined path of the resulting curve and the interaction¹⁰ does *not* contain any homologically nontrivial loop. Other than this requirement, the way in which the curves are merged is arbitrary.

¹⁰ By the path of an interaction we mean the shortest path connecting any pair of plaquettes affected by it that initially belong to two different curves.

trary, one simply has to connect the end of one curve to the start of the other in some suitable way.

In our framework merging is implemented by extending the end of one curve parallel along the curve in its reverse direction until it reaches the tile of one of the interacting anyons. The extended curve is then crossed over to the tile of the other anyon, after which it follows the other curve in its reversed direction and is attached to the start of the latter. An example of this merging procedure is depicted in Fig. 22(b).

Updating the state superposition in the case of a merge operation is trivial, as it simply amounts to taking the tensor product of the state superpositions associated to each curve.

the counterclockwise case, respectively:

$$S^{ab} : \begin{array}{c} \text{Diagram 1: Two horizontal lines with points } a \text{ and } b. \\ \text{Diagram 2: Two curved lines forming a clockwise loop, with } a \text{ and } b \text{ at the ends.} \end{array} \mapsto \quad (103)$$

$$(S^{ab})^{-1} : \begin{array}{c} \text{Diagram 1: Two horizontal lines with points } a \text{ and } b. \\ \text{Diagram 2: Two curved lines forming a counterclockwise loop, with } a \text{ and } b \text{ at the ends.} \end{array} \mapsto \quad (104)$$

It is important to stress that this does *not* represent an active braid move in which two anyons are exchanged. This is a basis transformation: the quantum state of the system remains unchanged, and the coefficients of the state superposition must be updated to express this state in a new basis.

The right transformation on the state vector components is found considering the relation between the anyonic fusion bases corresponding to the left and right hand side of these equations. For a clockwise swap, represented in Eq. (103), this relation is

$$\begin{array}{c} \text{Diagram 1: Two vertical lines labeled } a_j \text{ and } a_{j+1} \text{ on a horizontal line with points } b_1, b_2, b_3. \\ \text{Diagram 2: Two curved lines labeled } a_j \text{ and } a_{j+1} \text{ forming a clockwise loop, on a horizontal line with points } b_1, b'_2, b_3. \end{array} = \sum_{b'_2} B_{a_{j+1}b_3b'_2}^{b_1a_jb_2} \quad (105)$$

where

$$B_{a_{j+1}b_3b'_2}^{b_1a_jb_2} = \sum_c F_{b_3b_1b'_2}^{a_ja_{j+1}c} R_c^{a_ja_{j+1}} F_{a_{j+1}b_3c}^{b_1a_jb_2}. \quad (106)$$

Analogously, for the counter clockwise case, one finds

$$\begin{array}{c} \text{Diagram 1: Two vertical lines labeled } a_j \text{ and } a_{j+1} \text{ on a horizontal line with points } b_1, b_2, b_3. \\ \text{Diagram 2: Two curved lines labeled } a_j \text{ and } a_{j+1} \text{ forming a counterclockwise loop, on a horizontal line with points } b_1, b'_2, b_3. \end{array} = \sum_{b'_2} \left(B_{a_{j+1}b_3b'_2}^{b_1a_jb_2} \right)^* \quad (107)$$

3. Passive exchange

Calculating the outcome of both individual Pauli errors and fusion operations, requires expressing the state in the appropriate anyonic fusion basis. Basis transformations concerning individual curves are performed using passive exchanges or *swaps*, which are represented as follows for the clockwise and

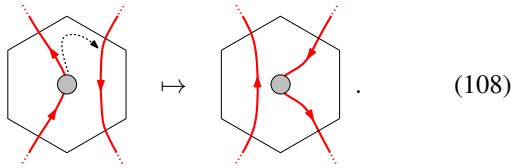
4. The paperclip algorithm

In order to determine the outcome of an interaction on a set of anyons, be it some local noise operator or the fusion of a pair, we must always transform to a basis where these anyons appear sequentially in the linear ordering determined by their curve diagram. Below, we outline how such a basis transformation can be performed using a sequence of swaps for the case where only 2 anyons are involved. The general case for n anyons can then be deduced iteratively.

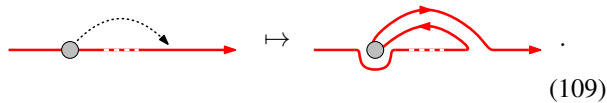
In general, any two curve diagrams f and f' , containing the same anyons, and such that the combined path of f and f'

does not contain any homologically nontrivial loop, can be related to each other by a sequence of swaps. The algorithm for determining this sequence, was introduced in Ref. [46] and dubbed the *refactoring algorithm*. We will describe a different but entirely equivalent formulation of this algorithm, called the *paperclip algorithm* which is more convenient for our purpose. This alternative formulation was introduced in Ref. [26], and determines the sequence of swaps corresponding to a basis transformation where we “move” an anyon (or rather, its position on the curve) to the next piece of curve encountered when moving along the boundary of its tile in a

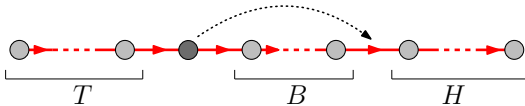
clockwise fashion, as shown in the following diagram:



We will call such transformations *refactoring moves*. Schematically, their action on the curve diagram can be represented as

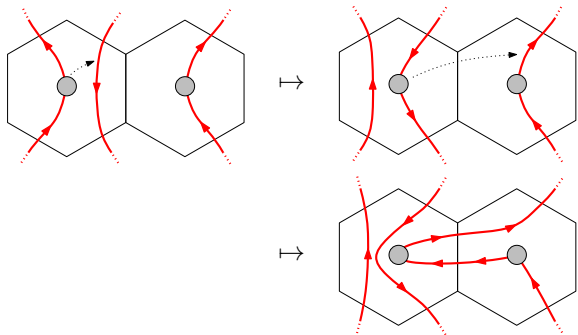


The initial position and destination of a refactoring move divide a curve into three disjoint segments which we name *tail* (T), *body* (B) and *head* (H), respectively (following the orientation of the curve). Their content is defined as follows:



where the dotted line represents the refactoring move. Note that we do not include the anyon which is being moved (indicated in dark above) in any of these segments.

As interacting anyons are always neighbors on the lattice, repeated use of such moves can be used to obtaining a curve diagram where these anyons appear subsequently on the same curve. For example (assuming all encountered lines belong to the same curve):

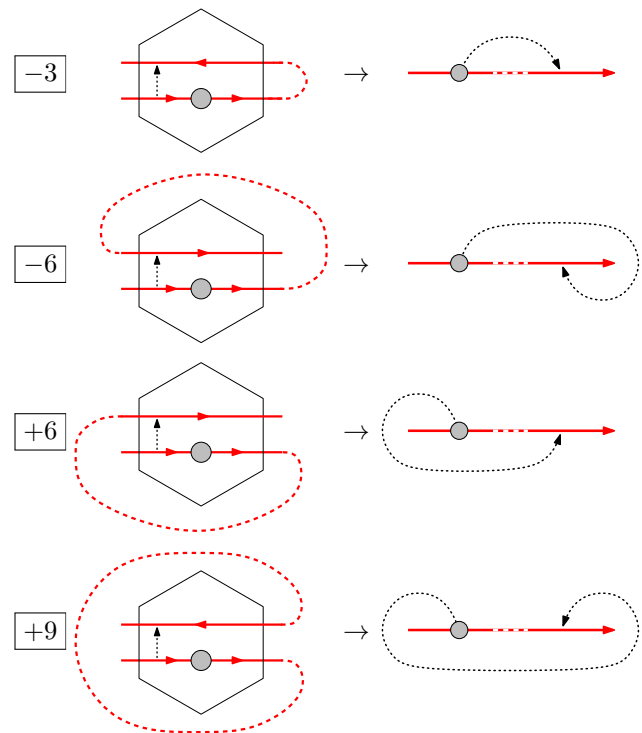


If any lines lie between the two interacting anyons but do not belong to either of their curves, one of the following two actions must be performed: In case the interacting anyons do not belong to the same curve initially, one can attempt to pull such lines through one of these curves to get it out of the way. Since each curve corresponds to a connected group of anyons with trivial total charge, such a deformation does not affect the state vector and is therefore permitted. Whenever this is not possible (i.e., when that does not “remove” the obstructing line), the corresponding curve diagram is essentially trapped between the others, and one is forced to merge it with those containing the interacting anyons, before proceeding with the clockwise “moves”. The latter is of course not desirable, since it increases the size of the associated fusion space, but there

are situations where this is unavoidable. Of course, in case the pair of anyons are members of different curves initially, these must be merged in the process.

The sequence of swaps corresponding to the refactoring moves described above can be determined with the corresponding *turn number*. This number is found by counting the number of right hand 60° turns made by the curve between the initial position of the anyon and its destination (that is, the next piece of curve that intersects the tile boundary). Starting at the anyon’s initial position, every 60° right hand turn contributes $+1$, while every left hand turn contributes¹¹ -1 . In order to ensure that the turn number is independent of where the “destination piece of curve” enters the tile, the total value must then be decreased by 1 for every additional edge of the tile boundary we have to move to (in a clockwise fashion) before encountering it (0 if it appears directly after the initial line, -1 if it appears on the next edge, ...).

Depending on whether the refactoring move is with or against the orientation of the curve, and depending on whether the piece of curve associated with the initial position of the refactoring move has an incoming or outgoing orientation, it is always isotopic to one of 4 different “paperclips”, each of which is associated to a specific turn number. For example, when the anyon is transported along the curve, and the initial piece of curve has an incoming orientation, the possible turn numbers and corresponding “paperclips” are



¹¹ Note that we always count the number of right hand turns while following the curve from the start towards the destination of the refactoring move, independently of whether or not the refactoring is along or against the orientation of the curve itself

The appropriate sequences of swaps for these four different situations are

$$\begin{aligned}
 -3 : & \quad S^{-1}[B], \\
 -6 : & \quad S^{-1}[B] S^{-1}[H] S^{-1}[H^r], \\
 +6 : & \quad S[T^r] S[T] S[B], \\
 +9 : & \quad S[T^r] S[T] S[B] S[H] S[H^r],
 \end{aligned}$$

where the notation $S[H]$ stands for sequentially swapping the anyon with all entries in H in a clockwise fashion. H^r indicates the sequence of anyons in H , in reversed order. The paperclip configurations for the other cases are similar. All of them are listed together with the corresponding sequences of swaps in App. E.

5. Fusion

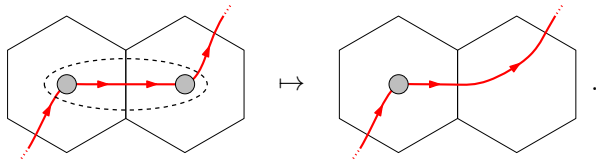
The fundamental operation during recovery is a pairwise fusion process, where one excitation is moved along a specific path until it neighbors another anyon, followed by the fusion of the pair. The physical implementation of these operations is discussed in Sec. III C. Here we discuss how they are implemented on the level of curve diagrams during the simulation.

Before fusing neighboring anyons, one must ensure that they appear sequentially on the same curve diagram, which is by achieved using the paperclip algorithm described above. To simulate the fusion process, we must first isolate the two anyons from the rest of the fusion tree, which is done using an F -move:

$$\begin{array}{c} a_j \quad a_{j+1} \\ \vdots \quad \vdots \\ b_1 \quad b_2 \quad b_3 \end{array} \quad \rightarrow \quad \sum_c F_{a_{j+1} b_3 c}^{b_1 a_j b_2} \quad \begin{array}{c} a_j \quad a_{j+1} \\ \vdots \quad \vdots \\ b_1 \quad b_3 \end{array} \quad (110)$$

The result of the fusion of anyons a_j and a_{j+1} is then picked from the possible values of their total charge c using the probability distribution dictated by the coefficients of the state superposition and the state vector is then projected to the selected outcome. Note that this probability distribution only concerns the resulting anyon charge of the plaquette. In case the outcome $c = \tau\tau$ was selected, the tail label is picked from the probability distribution $\{p(1) = \frac{1}{\phi^2}, p(\tau) = \frac{1}{\phi}\}$, which follows from Eq. (66).

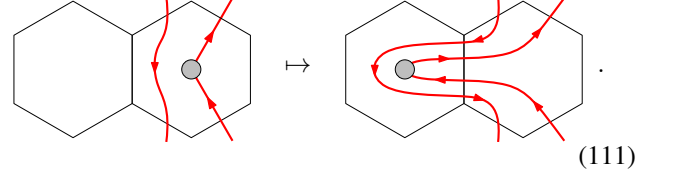
On the level of the curve diagrams, we must then remove one of the anyons as depicted below:



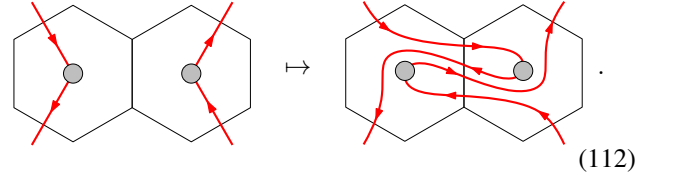
Note that we are required to choose a “target plaquette” in which the fusion outcome is placed.

6. Move and exchange

The moving procedure required to bring a pair of anyons to neighboring tiles can be broken down into a sequence of basic steps where an anyon is moved to a neighboring plaquette. If the neighboring plaquette does not contain an anyon, the curve is updated by moving the excitation to the neighboring tile while continuously deforming the curves in the corresponding tiles:



If the neighboring plaquette contains an anyon, the two anyons are exchanged in a clockwise fashion while deforming the curves in the corresponding tiles accordingly:



Note that the choice for a clockwise exchange over a counter-clockwise one is arbitrary. A different choice would result in different probabilities for the fusion outcomes after the moving procedure, as remarked in Sec. IV A.

Both operations do not affect the coefficients appearing in the state vector. However, they *do* modify the state by changing the corresponding basis elements. Such a transformation can be expressed as

$$|\Psi\rangle = \sum_i \alpha_i |\psi_i\rangle \mapsto |\Psi'\rangle = \sum_i \alpha_i |\psi'_i\rangle, \quad (113)$$

where the basis state $|\psi'_i\rangle$ is obtained from the state $|\psi_i\rangle$ by changing the embedding of the corresponding fusion tree on the surface as dictated by Eq. (111) or Eq. (112).

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 have been 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 $N_{\max}$. As discussed in Sec. IV 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 $N_{\max}$, and 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 was used in Ref. [26].

The second cutoff parameter that we introduce is $V_{\max}$, 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 to keep the value of $N_{\max}$ 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 will 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 will provide a valid lower bound on the actual error correction threshold, independent of the values of $N_{\max}$ and $V_{\max}$. 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 App. C). Hence, it is clear that the cutoffs will be triggered more often for larger system sizes. Likewise, for a fixed system size L , larger values of p will 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, will be 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

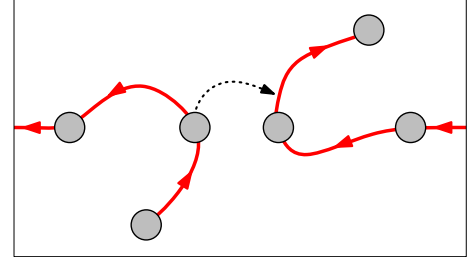


Figure 24: An illegal refactorings move on a torus, which will cause the simulation to declare *failure* and abort.

with some specific handle labels. However, as explained in Sec. IV F, 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 will automatically declare 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. IV 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. IV 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. 20 or 21. 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. 20 or 21. This is obtained by applying a series of refactorings moves as described in Sec. IV F 4, and performing the corresponding sequences of swaps and merge operations on the affected state vectors. If any of the required refactorings moves is topologically impossible, for instance the one depicted in Fig. 24, this indicates that a logical error would be caused, as the joint path of the curve diagram and the interaction path form a non-contractible loop. Whenever this happens the simulation declares *failure* and aborts the current Monte Carlo step.

If all refactorings moves above were 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. (75), Eq. (84), or a similar shape in the case of three affected plaquettes. In terms of ribbons in the fattened lattice, the basis then locally looks like the ones depicted in Fig. 20 or Fig. 21, depending on the type of er-

ror. Note that such a basis can no longer be represented using curve diagrams. This is not an issue, since we will 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. (94) are used to calculate the probabilities Eq. (78) 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. (79).

In the case of a σ_x error, the matrix elements (90) are used together with the coefficients in the state vector to compute the probabilities Eq. (86) for the outcome of the vertex and tail measurements. A result is then sampled from this distribution. Next, the matrix elements Eq. (91) are used to compute the probabilities Eq. (88) for the various outcomes of the charge measurement (that is performed after the appropriate unitary vertex correction was applied). We again pick a result according to these probabilities, and update the state vector using Eq. (89). In case of a σ_y error, we proceed analogously to the σ_x case, using the matrix elements Eq. (92) or Eq. (93).

After the state vector has been updated to reflect the collective effect of the noise and the measurements (with the specific outcomes that we picked 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.

H. 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 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 a 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 Sec. IV F 6. This basic moving step is repeated until the two anyons reside in neighboring tiles. A sequence of refactoring moves (and possibly merges) is then performed to obtain a basis in which they are direct neighbors on the same curve, and the corresponding sequences of swap and merge 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 non-contractible loop in terms of the curve diagrams, which in turn corresponds to a logical error.

If necessary, an F -move is applied to transform to 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 dialogue 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 $N_{\max}$ anyons, or a state vector contains more than $V_{\max}$ nonzero elements, the current Monte Carlo step is automatically reported as a failure.

V. DECODING ALGORITHMS

After the error syndromes (anyon charges) on all the plaquettes are measured using the circuit discussed in Sec. III B, this syndrome information is passed to a decoder, which in turn outputs the recovery operations required to correct the errors.

In this section we discuss 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 [50]. 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 or 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 anyon type but only the presence or absence of anyon. Due to the use of the detailed syndrome information, the logical error rate of the fusion-aware iterative MWPM decoder is lower than the other one.

We will indicate the syndrome using a decoding graph, shown in Fig. 25. 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.

A. Clustering decoder

The spirit of the clustering decoder is based on the charge conservation of anyons. As discussed in Sec. IV D, local

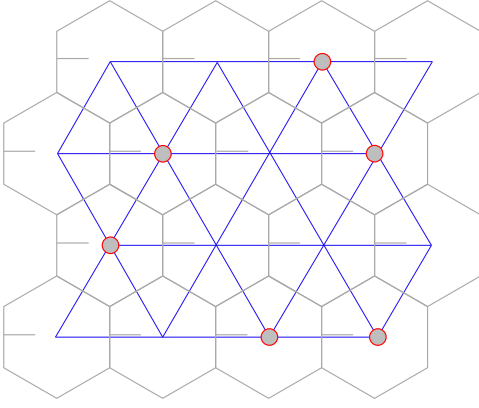


Figure 25: The decoding graph of the extended string-net code. By connecting the center of the plaquettes of the tailed lattice, we obtain a triangular lattice (blue) as the decoding graph. The anyons charge (grey circles) on the plaquettes of the tailed lattice serve as the syndromes in the error correction scheme and are located on the vertices of the triangular decoding graph.

noise can generate certain pairs or, more generally, clusters of anyons. Since these anyon clusters are generated from the vacuum sector $\mathbb{1}$, i.e., the ground space of the extended string-net code, the total charge of each anyon cluster should still be trivial ($\mathbb{1}$) due to charge conservation. Therefore, if we fuse all the anyons within a particular cluster by moving them towards the same plaquette, we will get a total trivial vacuum charge $\mathbb{1}$, i.e., all the anyons are annihilated back to the vacuum sector.

However, the decoder does not know which cluster each anyons were 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. Due to the total charge conservation and the fusion rule $\mathbb{1} \times \mathbb{1} = \mathbb{1}$, 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 will still be annihilated into the vacuum $\mathbb{1}$. 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 $\mathbb{1}$. On the other hand, if the joined anyon cluster forms a non-contractible (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 non-trivial logical operator along certain homologically nontrivial

cycle¹² which will cause a logical error and the decoder fails. It is also possible that such a non-contractible anyon cluster will have a nonzero total charge, therefore there is some residual anyon after the merging procedure which cannot be annihilated. In both cases, we will 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 pseudo-code and will be explained in detail with a concrete example:

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 simulation) whether any anyon path becomes
    # non-contractible, i.e., homologically nontrivial after the
    # fusion process
    if anyon_path == non-contractible
        # The decoder claims failure
        return Failure

    # Discard any empty cluster with trivial vacuum charge  $\mathbb{1}$ ;
    clusters = non_vac(clusters);

    # 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 remain-
# ing anyon
if anyon_charge == []
```

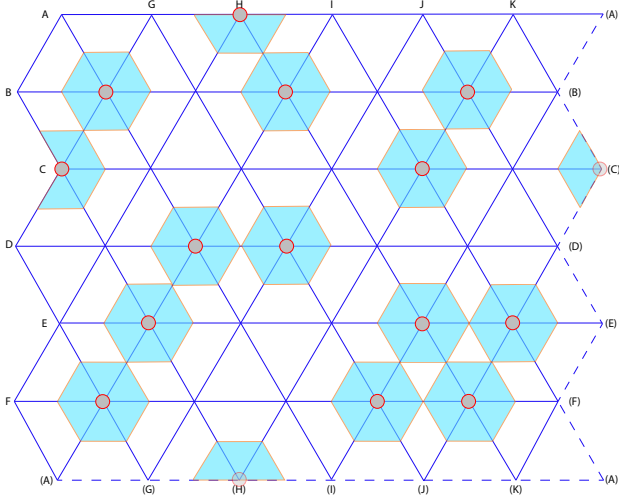
¹² In the case of open manifold with gapped boundaries, the logical operator corresponds to nontrivial relative homology cycle.

```

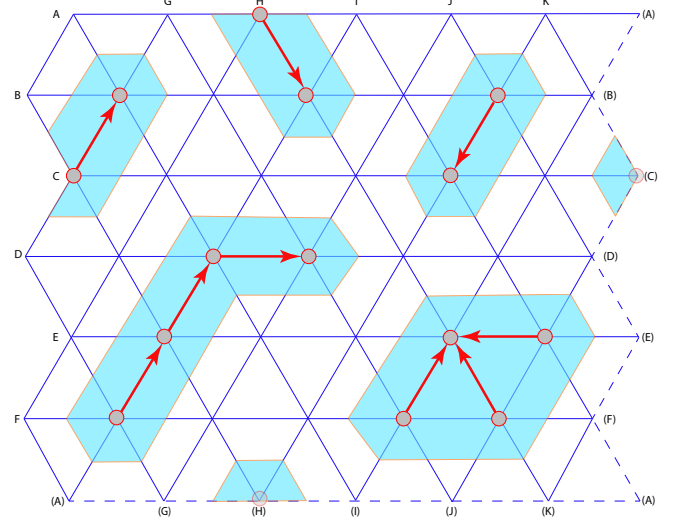
# The decoder declares success
return Success
# If there is a single nontrivial remaining charge
else
    # The decoder claims failure
    return Failure

```

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 non-zero 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:

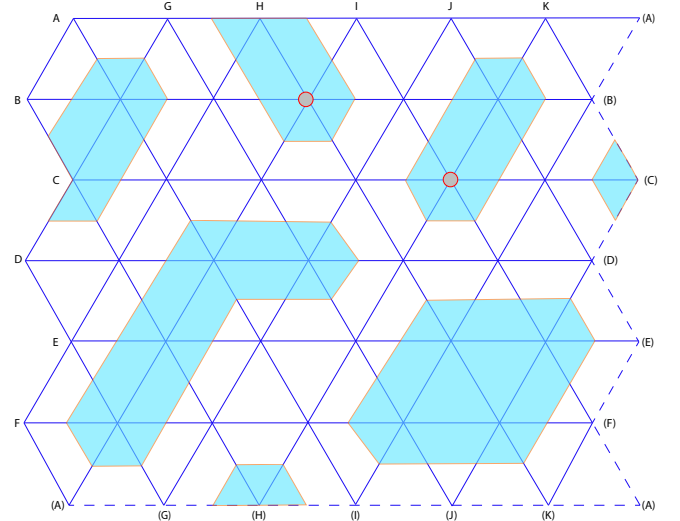


In the next step, we call `join(clusters, 1)` to join neighboring clusters which are separated by only 1 lattice spacing. Within each cluster we randomly choose a root anyon, and *move* all the other anyons to the root and finally fuse all of them into a single anyon. The *move operation* is the recovery operation introduced in Sec. III C 2 and Fig. 19 and can be simulated classically via modifying the curve diagram as shown in Eqs. (111) and (112) in Sec. IV F 6. We note that the order of moving and detailed path do not affect the resulting state after fusing all the anyons within each order, 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) towards 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.

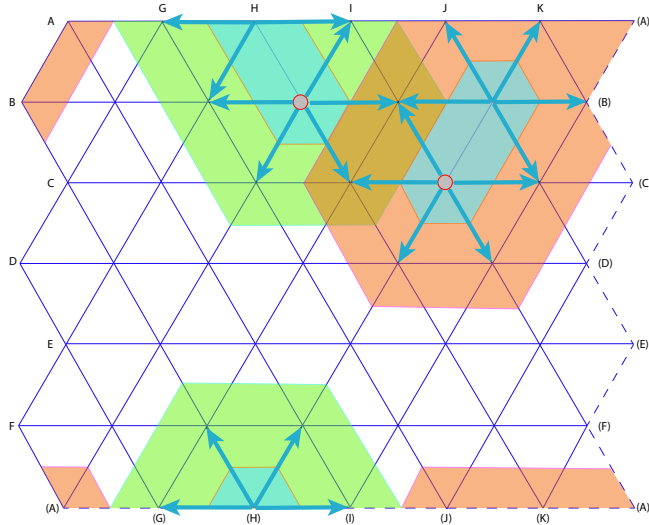


After the fusion process, we check in the classical simulation whether any anyon path l forms a non-contractible (homologically nontrivial) cycle, which will give rise to a logical error as previously illustrated in Fig. 8. The anyon path here is the sum of the error path and the recovery path: $l = l_e + l_r$. We note that, although the decoder itself does not have access to such information, our classical simulation with underlying curve diagrams does. We can hence claim failure of the decoder in our Monte Carlo simulation, if such non-contractible cycle occurs.

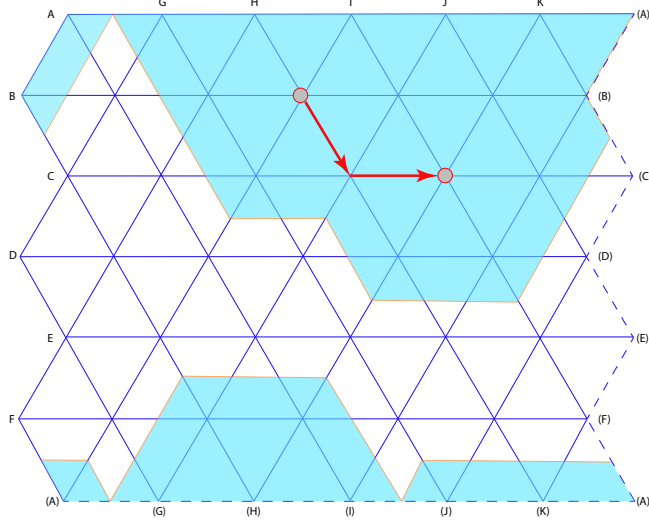
If the decoder does not fail, we continue to measure all the charges of the root anyons. If the measured charge in a particular cluster is zero, i.e., in the vacuum sector **11**, we discard the corresponding cluster. The list of clusters hence gets updated via `clusters=non_vac(clusters)`.



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 will be 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 will claim `failure` of the decoder.

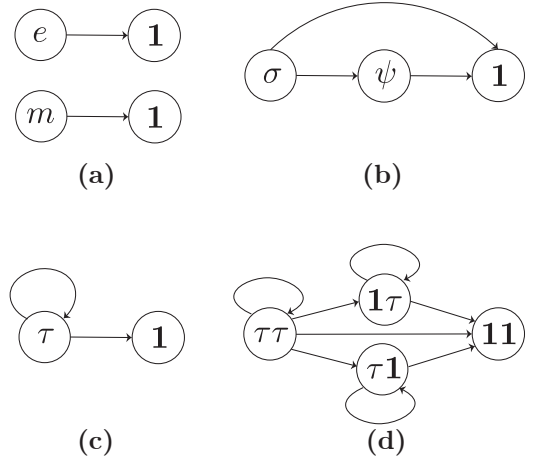


Figure 26: A graphic representation of the fusion rules (fusion graph) of (a) Toric code; (b) Ising category; (c) Fibonacci category (FIB); (d) doubled Fibonacci category (DFIB) corresponding to the Fibonacci Turaev-Viro code. The fusion graph is a directed graph with the directed edge pointing from pair of incoming anyons represented by one vertex to the possible fusion outcome represented by another vertex. (a) and (b) have non-cyclic fusion structure, while (c) and (d) are cyclic. In (d), $\tau\tau$ can refer to either refer to $\tau\tau_1$ or $\tau\tau_\tau$.

B. Fusion-aware iterative matching decoder

The fusion-aware minimum-weight perfect matching (MWPM) decoder applies matching on the anyons with a preferred order of matching certain types of anyons according to the underlying structure in the anyon pair generation under the Pauli noise, as discussed in Sec. IV D.

As we have stated before, the non-Abelian Fibonacci fusion rule $\tau \times \tau = 1 + \tau$ implies that a typical fusion of anyons in DFIB has probabilistic outcome into one of several fusion channels, e.g., $\tau 1 \times \tau \tau = 1\tau + \tau\tau$. This is in stark contract to Abelian anyon models, where fusion processes are always deterministic. This difference can be seen using a graphical representation of the fusion rules (which we will call a *fusion graph* from now on) as shown in Fig. 26(a,c,d). Since standard minimum-weight perfect matching (MWPM) decoder of the Abelian surface code is based on deterministically fusing pairs of Abelian anyons into the vacuum sector, this type of decoder will not work properly when applied to the Fibonacci string-net code.

In Ref. [22], a MWPM decoder has been applied to the non-Abelian phenomenological Ising-anyon model, corresponding to the Ising category $\{1, \sigma, \psi\}$. The corresponding fusion rules are:

$$\psi \times \psi = 1, \quad \psi \times \sigma = \sigma, \quad \sigma \times \sigma = 1 + \psi, \quad (114)$$

as illustrated in the fusion graph Fig. 26(b). The basic strategy there is to apply MWPM to pair up and fuse a particular type of anyons: the σ -anyons. As indicated by the above fusion rule, the matched pair of anyons either fuse into the vacuum

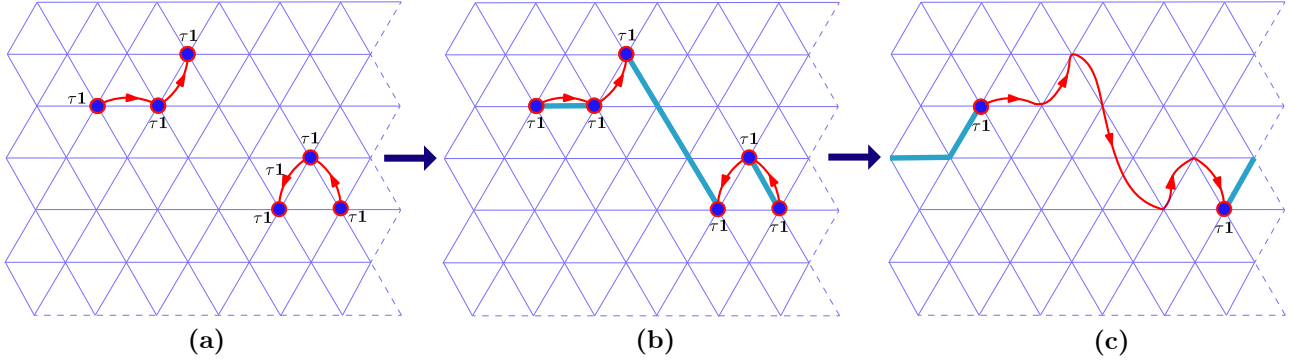


Figure 27: Illustration of the absence of threshold when applying the usual MWPM algorithm to the Fibonacci Turaev-Viro code. (a) Two triplets of 1τ anyons are created from the vacuum by $O(1)$ errors and are spatially separated apart at the order of code distance: $O(d)$. The underlying fusion structure can be seen from the curve diagrams, where the blue circles represent anyons with charge $\tau 1$ and the red directed edges specify the structure of the fusion basis. In particular, the labels on the circles specify the anyon charges. (b) When applying the MWPM algorithm, two of the anyons in each triplet will be matched, while the remaining one in one triplet will be matched with the one in the other triplet. We show the result of matching by highlighted (light blue) edges on the decoding graph. For each pair of the matched anyons, one of the anyons (randomly chosen) will be transported along the highlighted path to fuse with the other. (c) After fusing the matched anyons, we have two remaining $\tau 1$ anyons, while the underlying curve diagram and the corresponding fusion trees of the ribbon graph states are merged. Now one further apply MWPM again to pair up the remaining anyons. A short path (indicated by the highlighted edges) will be chosen by the matching algorithm. One will fuse the anyons along the path selected by the matching algorithm and the anyon path will form a non-contractible cycle corresponding to a logical error.

or the ψ -anyon. When new ψ -anyons are generated from fusion, one further applies MWPM to pair and fuse all the pre-existing and newly generated ψ -anyons. Afterwards, one can clean up all the anyons and hence correct all the errors if the decoder succeeds.

For the Fibonacci Turaev-Viro code, we can adopt a similar strategy. We first apply MWPM to pair up and fuse all the existing anyons, which will either fuse to vacuum or a certain type of anyons. We then apply MWPM again to match the newly-generated anyons. We iterate above process until all the anyons are annihilated into the vacuum and then declare success of the decoder if no logical error occurs. If there is any residual anyon or a logical error corresponding to any anyon-path forming a non-contractible cycle, we claim failure of the decoder. One can further introduce a particular order for merging different types of anyons, similar to the above case in the phenomenological Ising-anyon model in Ref. [22].

However, this type of MWPM decoder does not have a threshold. The underlying reason is related to the particular fusion structure of the Fibonacci and doubled-Fibonacci categories. As shown in Fig. 26, the Ising category has a *non-cyclic* fusion structure [25] such that the fusion graph does not have any cycle (an edge pointing from and to the same vertex, i.e., the same anyon type). In contrast, both the Fibonacci category (FIB) and the doubled Fibonacci category (DFIB) have a *cyclic* structure, i.e., containing at least a cycle in the fusion graph, which complicates the fusion process. As we can see, the Ising fusion rule naturally implies a pair creation process: a pair of anyons, either two σ -anyons or two ψ -anyons are generated from the vacuum 1 . Applying the matching can im-

mediately annihilate these pairs into the vacuum. If one of the σ -anyon might further split into a σ -anyon and a ψ -anyon, we get a triplet (σ, σ, ψ) which still contains a pair of σ . When applying the above MWPM algorithm, two σ 's will be paired up and must generate a ψ -particle if the anyons still stay close by. A further matching with the other ψ can annihilate the pair of ψ 's back to the vacuum 1 .

As we can see, such pair creation mechanism naturally fits the above MWPM algorithm. However, the Fibonacci category (FIB) and its doubled version (DFIB) does not have such simple pair creation structure. In FIB, two τ -anyons can fuse into τ as well due to the cyclic structure, which can further fuse with another τ into the vacuum 1 . This means that not only a pair of τ can be generated from the vacuum 1 , but also a triplet (τ, τ, τ) . In fact, due to the cyclic structure, any number of τ can be created from the vacuum. The absence of a pair generation structure is in contrast to the situation in the Ising category where only pairs of σ can be generated from the vacuum. Similarly, in DFIB, two $\tau 1$ can fuse into $\tau 1$, which means a triplet $(\tau 1, \tau 1, \tau 1)$ or more $\tau 1$ can be generated from the vacuum 11 . Similar argument applies to triplets of 1τ and $\tau\tau$. In the example of Pauli noise, we can see from Fig. 21 that a single Pauli-X error can create 2, 3 or 4 anyons, without a clear pair creation mechanism like the σ -anyons in the Ising category.

Now we can give a proof that the above MWPM decoder does not have a threshold for the Fibonacci Turaev-Viro code (DFIB). Consider a counterexample shown in Fig. 27, where two triplets of $\tau 1$ anyons are created by $O(1)$ errors from the vacuum and spatially separated by $O(d)$, i.e., the order of the code distance. According to the algorithm stated above, we

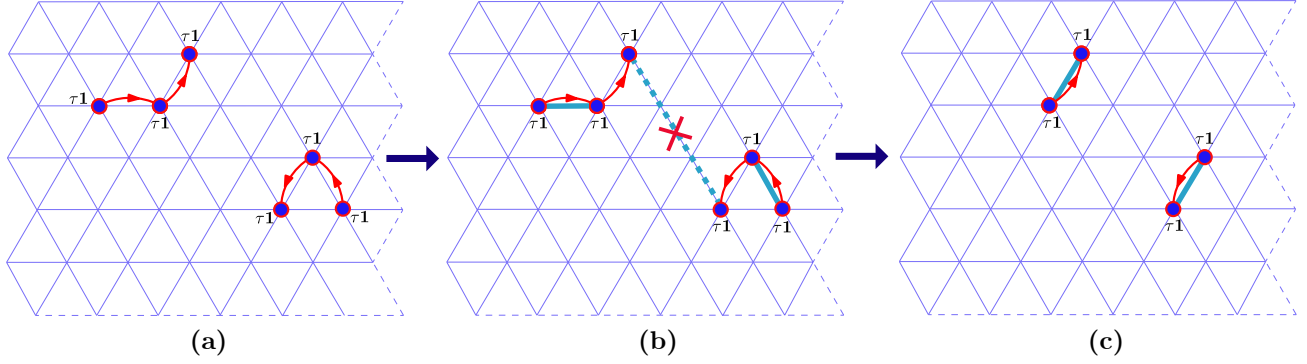


Figure 28: Resolve the issue in the previous example in Fig. 27 by applying the iterative MWPM algorithm. In panel (b), one applies MWPM algorithm to match three pairs of anyons. The matched pair involving anyons from different triplet clusters has a graph distance 3, exceeding the fusing radius $r = 1$ in the first iteration. Therefore, the fusion of this matched pair is rejected. In panel (c), the two previously matched pairs were fused and there are four remaining anyons. Apply another MWPM will match two pairs of anyons within their own cluster and hence successfully correct the errors.

will first apply MWPM to pair up all the present anyons¹³. As indicated by the highlighted edges in Fig. 27(b), two of the anyons within each triplet are matched, while one anyon within one cluster is matched with the one in another triplet. We then fuse the matched anyon pair by transporting one anyon (randomly chosen) along the highlighted path to fuse with the other anyon and ends up with the configuration in Fig. 27(c), where the curve diagrams are merged. We then apply MWPM to pair up the two remaining anyons. A short path is preferred by the matching algorithm as shown by the highlighted edge. When fusing the anyons along this short path, the anyon path forms a non-contractible (homologically nontrivial) cycle, which equivalently applies a chain of errors of length $O(d)$ and hence induces a logical error. In summary, when applying the above MWPM algorithm, even $O(1)$ errors may not be corrected, meaning the effective code distance in this decoding scheme is only $O(1)$ instead of $O(d)$. Therefore, the logical error rate is not exponentially suppressed as a function of code distance d , and the decoder does not have a threshold. As one can easily see, such an issue also exists in the case of single copy of the Fibonacci category. Nevertheless, we note that such problem is not present in the case of the triplet (σ, σ, ψ) in Ising anyon models mentioned above. Using the MWPM algorithm for the Ising-anyon model, one will first apply matching only to the σ -anyons within each triplet, which fuse into a ψ -particle, and then apply matching to the ψ -anyons, which fuse the two remaining ψ -anyons within each cluster. The non-cyclic feature of Ising fusion rule avoids the above issue.

Due to this issue, we need to modify the MWPM algorithm for FIB and DFIB. As we have seen, the main issue is that sometimes the matching algorithm pairs up anyons which are far apart and do not belong to the same anyon cluster generated from the vacuum. Therefore, we propose an iterative

MWPM algorithm. In each iteration, we apply the MWPM algorithm, but only fuse the matched anyon pairs which are within a certain fusing radius r (starting at one lattice spacing in the first iteration). When fusing a certain anyon pair, we may generate some new anyons. Therefore, we match and fuse anyons until all possible anyon pairs within the fusing radius r have been fused and begin the next iteration by increasing the fusing radius by one lattice spacing, i.e., $r \rightarrow r + 1$. We keep doing this until all the anyons are annihilated. During this process, if any anyon path forms a non-contractible cycle, we claim *failure*. If there is a single remaining anyon in the end of the algorithm, we also claim *failure*. Otherwise, we declare *success*. As we will see in the next section, the numerical simulation suggests that such an iterative MWPM algorithm does actually have a threshold.

The above iterative MWPM algorithm is summarized below:

Algorithm (iterative MWPM)

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 the fusing distance at one lattice spacing
 r=1;

While there is more than one anyon left
while size(anyon_charge) > 1

Apply the matching algorithm; only record anyon pairs with maximal distance r and the corresponding path connecting them
 [pairs, paths]=MWPM(anyon_charge, r);

If no anyon pair within the fusing radius is matched
if pairs == []

Increase the fusion radius by one lattice spacing
 r = r+1;

¹³ If the total number of anyons is odd, then there will be a single anyon which is not paired up with any other anyon.

```

else
    # Fuse the anyons along the paths given by MWPM; update
    # the list "anyon_charge" with the fused anyon charge; drop the
    # vacuum charge 11 from the list "anyon_charge"
    anyon_charge = fuse_anyons(pairs,
                               paths);

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

end

# If there is no anyon excitations left
if anyon_charge == []
    # The decoder declares success
    return Success
# If there is a single anyon left
else
    return Failure

```

As we can see, the iterative MWPM algorithm shares a similar hierarchical structure with the clustering algorithm where the cluster is grown by one lattice space in each iteration. When we apply this algorithm to the previous triplet example, we see that the previous issue is resolved, as illustrated in Fig. 28. In the first iteration, we have fusing radius $r = 1$. So when we apply matching, the fusion of the matched pair with a long separation (three lattice spacing) is rejected by the decoder since the separation exceeds the fusing radius. When we do the matching again, these anyons get matched and fused with their neighboring $\tau 1$ anyons generated from the last fusion. Therefore, all the fusion still occur within each triplet cluster and all the anyons are fused back into the vacuum.

In the above iterative MWPM algorithm, we have not used any detailed syndrome information about the anyon types, similar to the case in the clustering algorithm. We call such an algorithm ‘blind’. On the other hand, there are definitely certain features and patterns in the anyon creation process. Therefore, we expect that an iterative MWPM decoder which is fusion-aware, i.e., has access to the detailed information of anyon type, should have a better performance. In particular, in the context of depolarizing noise, a single σ_z error generates a pair of anyons with charge $\tau\tau_1$, as discussed in Sec. IV D 1 and shown in Fig. 20. This is a distinct feature of σ_z error when the system is exposed to all three types of noise ($\sigma_x, \sigma_y, \sigma_z$). In order to exploit such error feature, we require the decoder to have the following behavior: when the fusion-aware MWPM decoder sees a pair of $\tau\tau_1$ anyons, it will prefer to match and fuse them first, rather than to pair any of them with other types of anyons. This choice will tend to clean up the anyon clusters generated by σ_z errors first, and avoids to fuse anyons belonging to different anyon clusters which will join these two clusters.

We hence propose the following modification to obtain the fusion-aware iterative MWPM decoder. In each iteration, we

start matching and fusing specific type of anyons in the following order $[\tau\tau_1, \tau 1, 1\tau, \tau\tau_\tau, \tau\tau_1, \tau 1, 1\tau]$. We only start fusing the next type once the current type of anyon pairs within the fusion radius have all been fused. After we exhausted all the types, we match and fuse the remaining anyons ‘blindly’ within the fusing radius r , i.e., ignoring the detailed anyon charge type. In this case, one is allowed to match and fuse anyons with different anyon types. We then increase the fusing radius by one lattice spacing ($r \rightarrow r + 1$) and continue to the next iteration.

Now we explain the reason of choosing the above order of fusion. According to the fusion graph in Fig. 26(d), one can choose the fusion order from left to right such that the anyon types generated in the fusion process are never on the left of the current type on the fusion graph. In that case, we can choose either the fusion order $[\tau\tau, \tau 1, 1\tau]$ or $[\tau\tau, 1\tau, \tau 1]$. Note that here we do not distinguish $\tau\tau_1$ and $\tau\tau_\tau$ and they make no difference in the fusion graph. However, due to the structure of σ_z -noise as mentioned above, we prefer to match and fuse $\tau\tau_1$ first. On the other hand, $\tau\tau_\tau$ -anyons are less likely to be produced, so we fuse them after $\tau 1$ and 1τ . Then when we start fusing $\tau\tau_\tau$ -anyons, we are again on the very left of the fusion graph and will again produce all types of anyons ($\tau\tau_1, \tau 1, 1\tau$, and $\tau\tau_\tau$). Therefore, after exhausting fusion of $\tau\tau_\tau$, we still need to start fusing $\tau\tau_1, \tau 1$ and 1τ in turn.

We illustrate this algorithm with a specific example in Fig. 29. As we can see that the initial configuration in Fig. 29(a) has several anyon clusters generated from the vacuum sector by local noise, and each cluster has a total vacuum charge 11. In particular, two clusters of $\tau\tau_1$ anyons (green) are created by σ_z errors. The anyon clusters are all connected together, which poses significant challenge for the decoder. Now the decoder starts with fusing radius $r = 1$ and only perform matching on the $\tau\tau_1$ anyons as shown in Fig. 29(b). As we see, the two pairs of $\tau\tau_1$ are immediately matched and fused into the vacuum sector, and we are left with two separate anyon clusters with $\tau 1$ (blue) and 1τ (orange) anyons. Due to the significant separation, decoding becomes much easier now. The decoder continues to match and fuse only the $\tau 1$ anyons, which takes two steps to annihilate all the $\tau 1$ anyons, as shown in Fig. 29(c, d). The decoder now switches to match the 1τ anyons. It performs two steps of matching and fusion with fusing radius $r = 1$ in (e, f), and has exhausted the matching of all possible anyon pairs with a separation of one lattice spacing. Now the decoder increases the fusing distance to $r = 2$, and matches the remaining pair of 1τ anyons separated with two lattice spacing in (g), which are then fused into the vacuum. We can conclude that in this case, the fusion awareness makes the decoding much easier than the blind MWPM decoder which could prefer to join neighboring anyon clusters into a larger cluster and hence increases the chance of failure.

As we will see in the numerical results in the next section, this fusion-aware iterative MWPM decoder does give an overall improvement in logical fidelity over the ‘blind’ iterative MWPM decoder.

We note that Ref. [22] has also adopted such fusion-aware strategies to the clustering decoder in the context of Ising anyon model, i.e., clustering different types of anyons sepa-

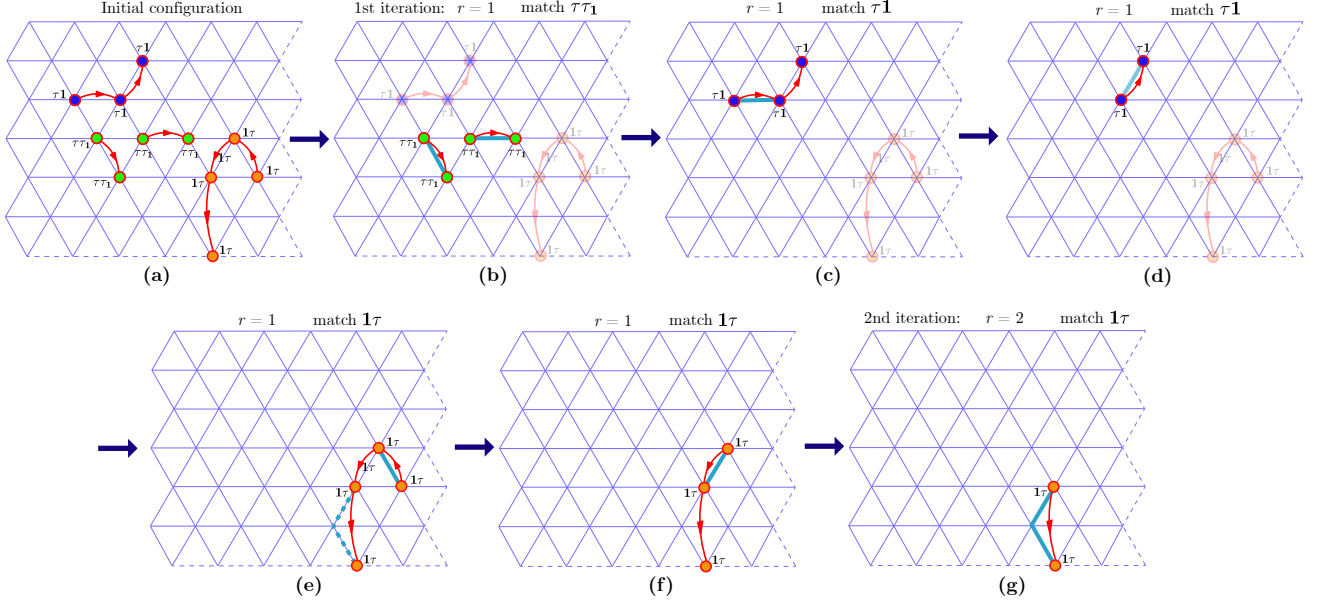


Figure 29: Illustration of the fusion-aware iterative MWPM decoder with an example of initial anyon configuration. (a) Four anyons clusters are generated from the vacuum sector. The anyon charges are shown on the nodes and branches of the curve diagram (see the branch label on the cluster with four 1τ anyons). The two $\tau\tau_1$ clusters are created by σ_z -errors. (b) Match and fuse $\tau\tau_1$ anyons only with fusing distance $r = 1$. (c, d) Match and fuse $\tau 1$ anyons only with fusing distance $r = 1$. (e) Match and fuse 1τ anyons only with fusing distance $r = 1$. The matched path (dashed lines) between the lower two anyons has length 2 and hence exceeds the fusing distance $r = 1$. The fusion between these two anyons is hence rejected. (f) Continue to match and fuse 1τ anyons with fusing distance $r = 2$. (g) In the second iteration, the decoder increases the fusing distance to $r = 2$, and hence is able to match and fuse the remaining pair of 1τ anyons.

rately in a certain chosen order. We have also adopted this strategy to the Fibonacci Turaev-Viro code (DFIB). However, we do not observe any advantage compared to the clustering decoder described in Sec. V A which does not use the detailed syndrome information of anyon types. This is potentially related to the sophisticated cyclic fusion structure of Fibonacci anyon as opposed to the non-cyclic fusion structure of Ising anyons.

VI. NUMERICAL RESULTS

The Monte Carlo simulations described in Chapter IV were performed for the three different decoders described in Chapter V. Individual Pauli errors were picked using relative probabilities corresponding to the following noise models:

- depolarizing noise: $\gamma_x = \gamma_y = \gamma_z = \frac{1}{3}$,
- dephasing noise: $\gamma_x = \gamma_y = 0$, $\gamma_z = 1$,
- bit-flip noise: $\gamma_x = 1$, $\gamma_y = \gamma_z = 0$.

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

The logical failure rate for each (p, L) -pair was computed by averaging over 10^5 Monte Carlo samples. For the lowest error rates $p = 0.01$ (or $p = 0.02$ in case of dephasing noise), 10^6 Monte Carlo samples were 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 were done with the following values for the cutoff parameters:

$$N_{\max} = 27,$$

$$V_{\max} = 2.5 \cdot 10^7.$$

For depolarizing noise and bit-flip noise with $L = 18$ the ratios of aborted Monte Carlo samples near the observed thresholds are shown in the table below. The corresponding ratios of aborted failures are indicated between parentheses.

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

We found 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)

were 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 studied.

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 in 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. [51]. This method was introduced in the context of the toric code, where an exact mapping to a statistical model is known [4, 51]. This model, the 2-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 et al. demonstrated 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

$$x = (p - p_c)L^{1/\nu}, \quad (115)$$

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

$$P_L(x) = A + Bx + Cx^2, \quad (116)$$

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.

A. Clustering decoder

The logical failure rate P_L of the clustering decoder in function of the noise strength p , are shown in Fig. 30(a), Fig. 30(b) and Fig. 30(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, was made for depolarizing noise and dephasing noise.

For depolarizing noise the finite-size scaling ansatz Eq. (116) was fitted to the logical failure rates for p ranging from 0.045 to 0.05 in increments of 0.00125. The following values were found using a non-linear least squares fit:

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

For dephasing noise, the ansatz was fitted to the logical failure rates obtained for p ranging from 0.07 to 0.075 in increments of 0.00125. With this data, we found

$$p_c = 0.0732 \pm 0.0006, \quad (117)$$

$$\nu = 1.17 \pm 0.08. \quad (118)$$

The confidence intervals were 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. 30. The logical failure rates in terms of rescaled error rate x defined in Eq. (115) are shown in Figs. 31(a) and 31(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.

B. Iterative matching decoders

We consider two types of iterative MWPM decoders: a fusion-aware one and a blind one (see Sec. V). The performance of these decoders under both types of noise are shown in Fig. 33. 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. 32, 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 has not used the detailed syndrome

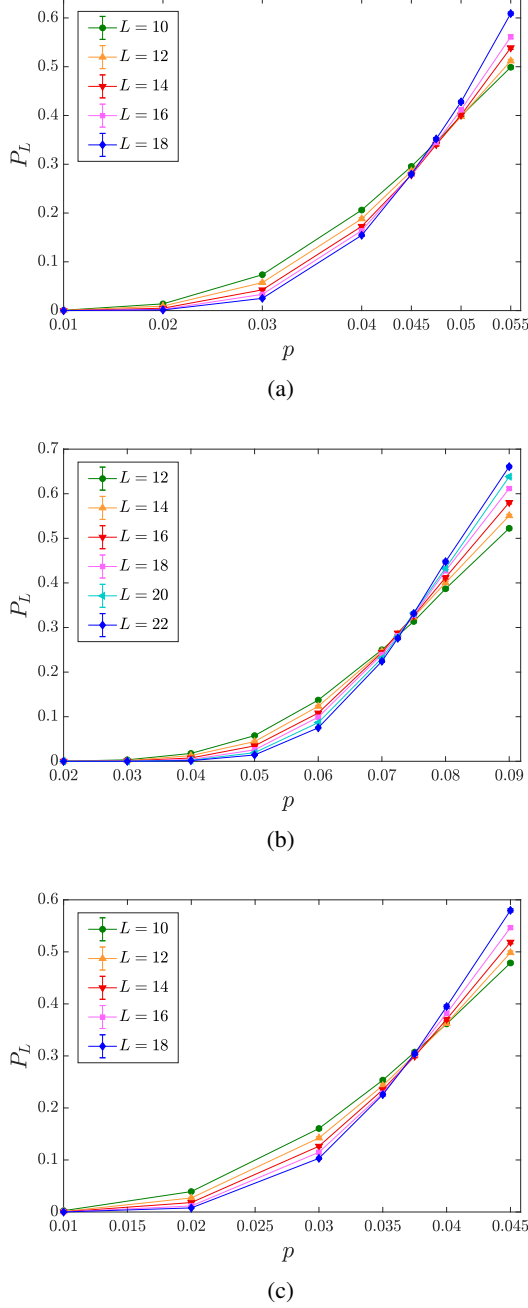


Figure 30: Logical failure rate P_L as a function of the physical error rate p for the clustering decoder with (a) depolarizing noise, (b) pure dephasing noise, and (c) pure bit-flip noise.

information of anyon types. This is related to the fact that 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.

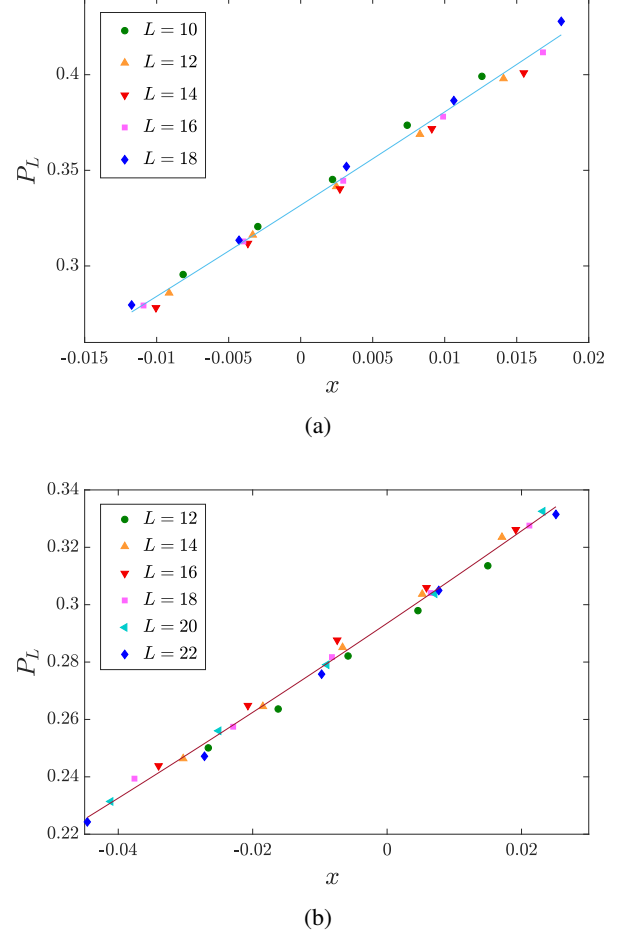


Figure 31: Logical failure rate P_L as a function of the rescaled error rate $x = (p - p_c)L^{(1/\nu)}$ for (a) depolarizing noise and (b) dephasing noise. The solid line represents the best fit of the model $P_L = A + Bx + Cx^2$.

C. 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. IV 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 strength $p_{\text{i.i.d.}}$. However, for Abelian codes, both noise models are equally valid and their respective noise strengths can be compared using the relation between the i.i.d. noise strength $p_{\text{i.i.d.}}$ and the error rate p of a fixed-rate sampling noise model derived in App. D.

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 i.i.d. 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

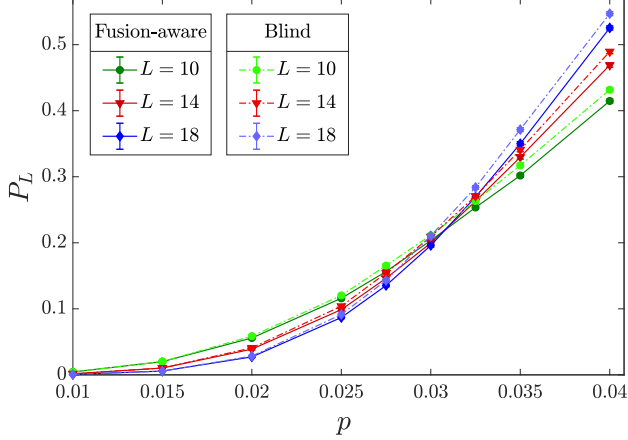


Figure 32: Comparison between the logical failure rates for the fusion-aware and the blind iterative MWPM decoders under depolarizing noise.

is optimal for this code, these values compare very favorably with the optimal thresholds for the surface code¹⁴ (under the assumption of perfect measurements), which are 18.9% for depolarizing noise [53] and around 10% for dephasing noise [4, 51].

VII. CONCLUSION AND OUTLOOK

In order to estimate the non-Abelian error threshold, we have combined concepts and techniques from three seemingly distant fields: quantum error correction, topological quantum field theory, and tensor networks. In particular, we have developed 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 were 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 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, therefore the error threshold and logical error rate for fault-tolerant logical gates is expected to be the same as the fault-tolerant threshold for memory storage. Therefore, 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], as well as the just-in-time decoding is present 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 [54]. This hybrid approach may greatly reduce the overhead of active error correction.

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 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. While not universal for quantum computation by braiding itself, extra topological charge measurement and Dehn twists permit a universal set of fault-tolerant logical gates. More importantly, this code would have a simpler non-cyclic 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 was 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

¹⁴ The results mentioned here were 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 [52].

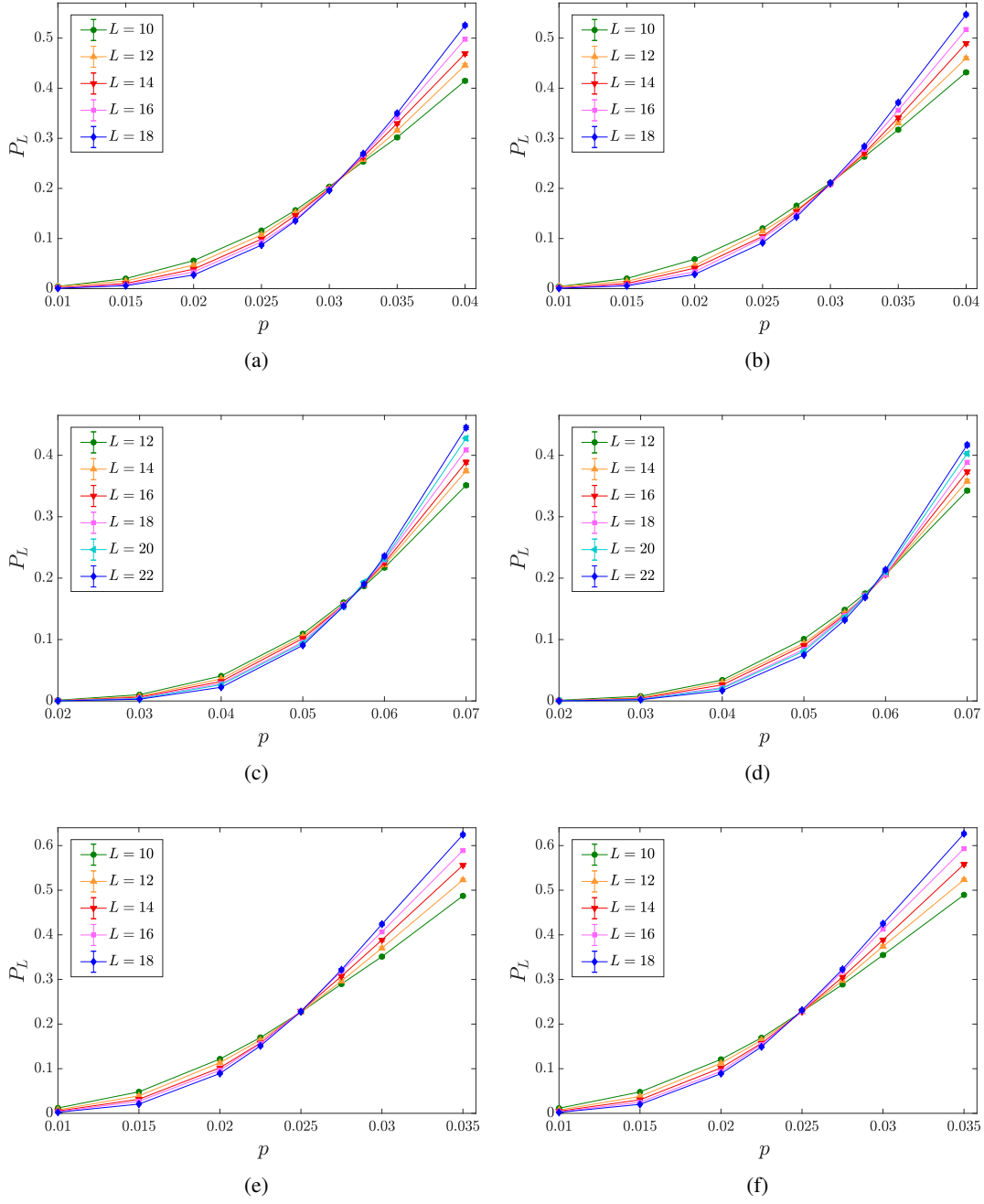


Figure 33: (a,c,e) Logical failure rate P_L as a function of the physical error rate p for the fusion-aware iterative MWPM decoder with (a) depolarizing noise, (c) pure dephasing noise, and (e) pure bit-flip noise. (b,d,f) Logical failure rate P_L as a function of the physical error rate p for the blind iterative MWPM decoder with (b) depolarizing noise, (d) pure dephasing noise, and (f) pure bit-flip noise.

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 [55], and recent progress has been made in capturing this formalism in terms of tensor network representations [56]. 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 [57]. 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 (multi-qubit Toffoli) gates. Therefore, it would be interesting to further explore hardware-efficient implementations of multi-qubit gates, instead of always decomposing these into a longer sequence of two-qubit gates. Such multi-qubit gates are widely studied in Rydberg-atom and ion-trap systems, and it would be useful to further develop these gates in superconducting qubit systems as well.

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.

ACKNOWLEDGMENT

The authors would like to 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. AS, LB and FV were by supported by FWO, and ERC Grant No. QUTE (647905).

Appendix A: The Turaev-Viro TQFT and the extended Levin-Wen string-net model

The extended string-net model defined in Sec. II is best introduced in the context of topological quantum field theory. In particular, it can be understood as a local Hamiltonian realization of the Turaev-Viro TQFT. Hence, the associated error correcting code should be thought of as a microscopic realization of a Turaev-Viro code [27] on a system of qudits. In this appendix, we introduce the ribbon graph Hilbert space originating from the Turaev Viro TQFT. We introduce a basis of this Hilbert space which allows us to interpret ribbon graph configurations in terms of anyonic fusion states. We then discuss the tube algebra which emerges naturally in this context. We conclude by illustrating how the (extended) Levin-Wen string-net model emerges naturally when constructing a lattice realization of the ribbon graph Hilbert space.

1. Category theory primer

The mathematical underpinning of topological order and anyon models, is category theory. Category theory in itself is a broad field of mathematics, which has been studied intensively for several decades. For the purpose of this work however, we do not need the full mathematical machinery of category theory. Instead, we will give a condensed overview of the algebraic data of unitary fusion categories and unitary modular tensor categories, which are needed for defining our model in the remainder of this section.

A *unitary fusion category* (UFC) $\mathcal{C}$ is defined by a finite set of simple objects

$$\{a_1, a_2, \dots, a_N\}, \quad (\text{A1})$$

and a collection of algebraic data for this set. The core of this data is formed by the fusion algebra

$$a \times b = \sum_c N_{ab}^c c, \quad (\text{A2})$$

where $N_{ab}^c \in \mathbb{N}$ are called the fusion coefficients. The fusion algebra satisfies the following associativity condition:

$$\sum_e N_{ab}^e N_{ec}^d = \sum_f N_{bc}^f N_{af}^d. \quad (\text{A3})$$

We define

$$\delta_{ab}^c = \begin{cases} 0 & \text{if } N_{ab}^c = 0, \\ 1 & \text{otherwise.} \end{cases} \quad (\text{A4})$$

Among the simple objects, there must be a unique unit element $\mathbf{1}$, satisfying $N_{1a}^b = N_{a1}^b = \delta_{a,b}$ for all $a, b \in \mathcal{C}$. For each $a \in \mathcal{C}$, there is a unique element $a^* \in \mathcal{C}$ satisfying $(a^*)^* = a$, and $N_{ab}^1 = N_{ba}^1 = \delta_{a^*,b}$.

A UFC associates to every fusion (resp. splitting) vertex, an N_{ab}^c -dimensional vector space V_{ab}^c (resp. V_c^{ab}) over $\mathbb{C}$. For simplicity, we will restrict to the multiplicity-free case, i.e.: $N_{ab}^c = \delta_{ab}^c$.

The fusion associativity condition (A3) then implies that the fusion or splitting vector spaces corresponding to different orderings of fusion or splitting, are isomorphic:

$$\bigoplus_e V_{ab}^e \otimes V_{ec}^d \simeq \bigoplus_f V_{bc}^f \otimes V_{af}^d. \quad (\text{A5})$$

The linear map between those two vector spaces is given by a 6-index object called the F -symbol:

$$\begin{array}{c} a \quad b \quad c \\ \diagdown \quad \diagup \quad \diagup \\ e \quad \bullet \quad f \\ \diagup \quad \diagdown \quad \diagdown \\ d \end{array} = \sum_f F_{cdf}^{abe*} \begin{array}{c} a \quad b \quad c \\ \diagdown \quad \diagup \quad \diagup \\ f \quad \bullet \quad g \\ \diagup \quad \diagdown \quad \diagdown \\ d \end{array}. \quad (\text{A6})$$

The F -symbol is only defined for allowed fusion vertices. We use the convention that it is zero outside this subspace:

$$F_{cdf}^{abe} = \delta_{abe} \delta_{e^*cd} \delta_{adf} \delta_{bcf^*} F_{cdf}^{abe}, \quad (\text{A7})$$

where $\delta_{abc} = \delta_{ab}^{c^*}$. The F -symbol must satisfy the following consistency condition, called the *pentagon equation*:

$$F_{e^*dl}^{c^*fg} F_{e^*lk}^{baf^*} = \sum_h F_{g^*ch}^{baf^*} F_{e^*dk}^{hag^*} F_{k^*dl}^{cbh^*}. \quad (\text{A8})$$

Each simple object $a \in \mathcal{C}$, we define a constant $d_a \in \mathbb{R}$ called the *quantum dimension*:

$$d_a = \frac{1}{|F_{aa^*1}^{aa^*}|}. \quad (\text{A9})$$

The quantum dimensions satisfy

$$d_a d_b = \sum_k N_{ab}^k d_k, \quad d_1 = 1, \quad \text{and} \quad d_{a^*} = d_a. \quad (\text{A10})$$

The *total quantum dimension* $\mathcal{D}$ is defined as

$$\mathcal{D} = \sqrt{\sum_a d_a^2}. \quad (\text{A11})$$

Finally, the F -symbol, when viewed as a matrix $[F_{cd}^{ab}]_{fe} = F_{cdf}^{abe}$, must be unitary on the subspace on which it is defined:

$$[(F_{cd}^{ab})^{-1}]_{ef} = [(F_{cd}^{ab})^\dagger]_{ef} = ([F_{cd}^{ab}]_{fe})^*, \quad (\text{A12})$$

$$\sum_f \left(F_{cdf}^{abe'} \right)^* F_{cdf}^{abe} = \delta_{e,e'} \delta_{abe} \delta_{e^*cd}.$$

A *unitary ribbon category* (URC) is obtained by adding the notion of braiding and twists to a UFC. Braiding is a unitary operation between fusion spaces that corresponds to a counterclockwise exchange:

$$R_c^{ab} : V_{ab}^c \rightarrow V_{ba}^c : \begin{array}{c} a \quad b \\ \diagdown \quad \diagup \\ c \end{array} \mapsto \begin{array}{c} b \quad a \\ \diagdown \quad \diagup \\ c \end{array} = R_c^{ab} \begin{array}{c} b \quad a \\ \diagdown \quad \diagup \\ c \end{array}. \quad (\text{A13})$$

Note that since we are working in the multiplicity-free case, R_c^{ab} is simply a complex phase. Its inverse corresponds to a clockwise exchange:

$$(R_c^{ba})^{-1} : V_{ab}^c \rightarrow V_{ba}^c : \begin{array}{c} a \quad b \\ \diagdown \quad \diagup \\ c \end{array} \mapsto \begin{array}{c} b \quad a \\ \diagdown \quad \diagup \\ c \end{array} = (R_c^{ba})^* \begin{array}{c} b \quad a \\ \diagdown \quad \diagup \\ c \end{array}. \quad (\text{A14})$$

Similar to the pentagon equation (A8) for the F -symbol, the R -symbol must obey consistency conditions called the *hexagon equations*:

$$R_e^{ca} F_{db^*g}^{c^*a^*e} R_g^{cb} = \sum_f F_{db^*f}^{a^*c^*e} R_d^{cf} F_{dc^*g}^{b^*a^*f}, \quad (\text{A15})$$

$$(R_e^{ac})^* F_{db^*g}^{c^*a^*e} (R_g^{bc})^* = \sum_f F_{db^*f}^{a^*c^*e} (R_d^{fc})^* F_{dc^*g}^{b^*a^*f}. \quad (\text{A16})$$

In a URC, every simple object a is assigned a *topological phase* θ_a , used to “untwist” a ribbon:

$$\begin{array}{c} a \\ \diagup \quad \diagdown \\ \bullet \end{array} = \theta_a \begin{array}{c} a \\ \diagup \quad \diagdown \\ \bullet \end{array}, \quad \begin{array}{c} a \\ \diagdown \quad \diagup \\ \bullet \end{array} = (\theta_a)^* \begin{array}{c} a \\ \diagdown \quad \diagup \\ \bullet \end{array}. \quad (\text{A17})$$

The topological phase is related to the R -symbol as follows:

$$\theta_a = \left(R_1^{aa^*} \right)^*. \quad (\text{A18})$$

A *Unitary Modular Tensor Category*, is a unitary ribbon category for which the R -matrices satisfy a certain nondegeneracy condition. We will not need the technical definition of modularity. For more details, we refer the reader to Ref. [58].

a. The Fibonacci category

Throughout this paper, we will place a special focus on the Fibonacci category. This category contains two simple objects:

$$\{\mathbf{1}, \tau\}, \quad (\text{A19})$$

with only one nontrivial fusion rule

$$\tau \times \tau = \mathbf{1} + \tau. \quad (\text{A20})$$

The pentagon equation (A8) with these fusion rules only has one solution that satisfies the unitarity condition (A12). The only nontrivial F -matrix is

$$[F_{\tau\tau}^{\tau\tau}] = \begin{pmatrix} \phi^{-1} & \phi^{-\frac{1}{2}} \\ \phi^{-\frac{1}{2}} & -\phi^{-1} \end{pmatrix}, \quad (\text{A21})$$

where $\phi = \frac{\sqrt{5}+1}{2}$ is the golden ratio. For all other combinations of indices, F_{klm}^{ijm} is either 1 or 0, depending on whether or not the corresponding indices in Eq. (A7) satisfy the branching rules.

The quantum dimensions are

$$d_1 = 1, \quad d_\tau = \phi, \quad (\text{A22})$$

and the total quantum dimension is then given by

$$\mathcal{D} = \sqrt{1 + \phi^2} = \sqrt{2 + \phi} = \sqrt{\sqrt{5}\phi}. \quad (\text{A23})$$

The hexagon equation admits two solutions, which are related by complex conjugation. We will use the following:

$$R_1^{\tau\tau} = e^{\frac{4\pi i}{5}}, \quad R_\tau^{\tau\tau} = e^{\frac{-3\pi i}{5}}, \quad (\text{A24})$$

the other R -symbols allowed by the fusion rules are $R_a^{1a} = R_a^{a1} = 1$ for any a .

Finally, the modular $\mathcal{S}$ -matrix is

$$\mathcal{S} = \frac{1}{\sqrt{2 + \phi}} \begin{pmatrix} 1 & \phi \\ \phi & 1 \end{pmatrix}. \quad (\text{A25})$$

2. The ribbon graph Hilbert space

The *ribbon graph Hilbert space* is the Hilbert space associated to a surface by the Turaev-Viro TQFT defined by a unitary fusion category $\mathcal{C}$ [27]. Even though it is defined for any unitary fusion category, we consider the special case where the input category $\mathcal{C}$ is a unitary modular tensor category. Furthermore, in addition to the conditions (A7), (A8) and (A12) that are satisfied for every UMTC, we impose the following three conditions for the F -symbols:

$$F_{klm}^{ijm} = F_{lkn}^{jim} = F_{jin}^{lkm*} = F_{k*nl}^{imj} \frac{v_m v_n}{v_j v_l}, \quad (\text{A26})$$

$$F_{j*jk}^{ii*1} = \frac{v_k}{v_i v_j} \delta_{ijk}, \quad (\text{A27})$$

$$(F_{klm}^{ijn})^* = F_{jkm}^{lin}. \quad (\text{A28})$$

The first of these conditions is known as tetrahedral symmetry, the second one is a normalization condition and the third condition is an alternative formulation of the unitarity condition Eq. (A12).

Let Σ be a compact orientable surface with a boundary, where we place a single marked point on each connected component of the boundary. A ribbon graph is a labeled, directed graph embedded into Σ , with internal vertices of degree two and three, and a vertex of degree one at each marked boundary point of Σ . The ribbons are labeled with the labels of $\mathcal{C}$. Reversing the direction of an edge in the ribbon graph corresponds to conjugating the edge label. Edges assigned the trivial label 1 can be added to or removed from a given ribbon graph, as the vacuum label denotes the absence of a ribbon. We require that each internal vertex satisfies the fusion rules, i.e.: that $\delta_{ijk} = 1$ for every vertex with incoming lines i, j and k .

The *ribbon graph Hilbert space* $\mathcal{H}_\Sigma$ is the space of formal linear combinations of ribbon graphs, modulo the following relations:

$$\begin{array}{c} i \\ \curvearrowright \end{array} = \begin{array}{c} i \\ \curvearrowleft \end{array}, \quad (\text{A29})$$

$$\begin{array}{c} j \rightarrow \text{circle} \rightarrow i \\ \text{circle} \end{array} = \delta_{j1} d_i, \quad (\text{A30})$$

$$\begin{array}{c} i \quad l \\ \diagdown \quad \diagup \\ j \quad k \end{array} = \sum_n F_{klm}^{ijn} \begin{array}{c} i \quad l \\ \diagdown \quad \diagup \\ j \quad k \end{array}. \quad (\text{A31})$$

A convenient relation, known as the *bubble bursting* equation, follows from Eqs. (A31) and Eq. (A27):

$$\begin{array}{c} k' \\ \downarrow \\ i \quad j \\ \uparrow \\ k \end{array} = \frac{v_i v_j}{v_k} \delta_{kk'} \begin{array}{c} k \\ \downarrow \end{array}. \quad (\text{A32})$$

Another convenient relation that follows from Eq. (A27) is

$$\begin{array}{c} i \quad j \\ \downarrow \quad \downarrow \end{array} = \sum_k \frac{v_k}{v_i v_j} \delta_{ijk*} \begin{array}{c} i \quad j \\ \diagdown \quad \diagup \\ k \\ \diagup \quad \diagdown \\ i \quad j \end{array}. \quad (\text{A33})$$

From here on, in order to simplify the notation and the graphical representations, we will assume that $\mathcal{C}$ is self-dual, meaning $i = i^*$ for all labels $i \in \mathcal{C}$. This allows us to drop the orientations in the ribbon diagrams. The generalization to non-self-dual categories is straight-forward. A much more elaborate treatment of the ribbon graph Hilbert space, including the non-self-dual case, can be found in Ref. [27]. However, for our purpose, this simplified and condensed summary is sufficient.

We now consider the case where Σ is the n -punctured sphere, $\Sigma_n = S^2 \setminus (A_1 \cup \dots \cup A_n)$, where each A_i represents a disk. To each hole we assign a single marked boundary point $p \in \partial A_i$. A labeling ℓ of Σ_n associates to each marked

boundary point p a label $\ell(p) \in \mathcal{C}$. The ribbon graph Hilbert space $\mathcal{H}_{\Sigma_n}$ can then be decomposed as

$$\mathcal{H}_{\Sigma_n} = \bigoplus_{\ell} \mathcal{H}_{\Sigma_n}^{\ell}, \quad (\text{A34})$$

where $\mathcal{H}_{\Sigma_n}^{\ell}$ is the subspace spanned by ribbon graphs with an edge labeled by $\ell(p)$ connected to p , for every marked boundary point p .

We can define a basis for $\mathcal{H}_{\Sigma_n}$ based on a *pants decomposition* of the surface Σ_n . A pants decomposition is associated to a rooted binary tree with $n - 1$ leaves, where the root and leaves each correspond to a hole in Σ_n . Its internal vertices of degree three correspond to *pants segments*, each isomorphic with Σ_3 , while its edges correspond to *cylindrical segments*, each isomorphic with Σ_2 . An example of a pants decomposition of Σ_4 is depicted in Fig. 34. For a given pants decompo-

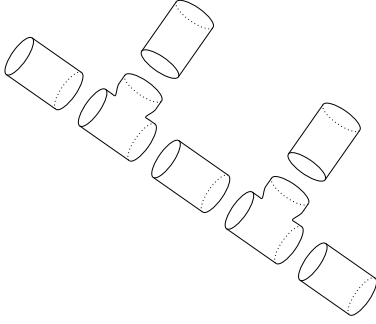


Figure 34: A standard pants decomposition for the sphere with four punctures Σ_4 .

sition, a basis for $\mathcal{H}_{\Sigma_n}$ is obtained by fixing the ribbon graph on each leaf to

$$\text{Cylinder with labels } i, j, k \equiv \text{Pants segment with labels } i, j, k, \quad (\text{A35})$$

on each internal segment and on the root to

$$\text{Cylinder with label } l \equiv \text{Line segment with label } l, \quad (\text{A36})$$

and finally connecting the ribbons in each pant segment. The set of all allowed label assignments of this ribbon graph, constitutes a basis for $\mathcal{H}_{\Sigma_n}$. An inner product on $\mathcal{H}_{\Sigma}$ can now be defined by setting these basis states to be orthonormal. This basis is called a *computational basis*, since it provides a way of encoding $\mathcal{H}_{\Sigma_n}$ into $5n - 6$ qudits. A more general construction, based on specific triangulations of Σ , exists and gives rise to an entire family of computational bases for $\mathcal{H}_{\Sigma_n}$ [27]. These different computational bases can be related using the equivalence rules (A29), (A30) and (A31), ensuring that the inner product they define is unique.

3. Dehn twists and braid moves on $\mathcal{H}_{\Sigma}$

Consider the set $Q \in \Sigma$ of all marked boundary points. Diffeomorphisms $f : \Sigma \rightarrow \Sigma$ that map the Q onto itself, define an action on $\mathcal{H}_{\Sigma}$. Two special types of such transformations, known as Dehn twists and braid moves, are of particular interests to us, because of their relation with the topological properties of anyon models (i.e.: the R -matrices and topological phases appearing in App. A 1).

A *Dehn twist* corresponds to a 2π -counterclockwise twist along a simple closed curve on Σ_n . If we consider the non-contractible curve γ ,

$$\text{Cylinder with curve } \gamma, \quad (\text{A37})$$

then a Dehn twist $D(\gamma)$ along this curve acts as

$$D(\gamma) : \text{Cylinder with label } i \mapsto \text{Twisted cylinder with label } i, \quad (\text{A38})$$

or, schematically

$$D(\gamma) : \text{Line with label } i \mapsto \text{Twisted line with label } i.$$

A *braid move* acts on two holes of a pair of pants in Σ_n and is defined as a π -counterclockwise twist along a simple closed curve on Σ_n enclosing the two holes, followed by a π -twist in the opposite direction on each of the legs corresponding to the two holes,

$$\text{Pair of pants with labels } j, k, i \mapsto \text{Twisted pair of pants with labels } k, j, i, \quad (\text{A39})$$

or, schematically

$$\text{Schematic of braid move: Pair of pants with labels } j, k, i \mapsto \text{Twisted pair of pants with labels } k, j, i.$$

The action of both Dehn twists and braid moves on $\mathcal{H}_{\Sigma}$ follows by linearly extending their action on the computational basis ribbon graphs to the full Hilbert space.

4. The anyonic fusion basis

States described using the computational basis constructed above, do not transform cleanly under the action of the mapping class group generators, making them inconvenient operationally. Below, we will define an orthonormal basis that simultaneously diagonalizes all Dehn twists and behaves nicely

under braid moves. While the construction itself is tedious, this basis will reveal much more of the mathematical structure of $\mathcal{H}_\Sigma$, and will prove invaluable for the rest of this work. In particular, this construction will reveal the relation between states in $\mathcal{H}_\Sigma$ and the Drinfeld center or categorical double $\mathcal{DC}$ of the input category $\mathcal{C}$.

It is important to note that the categorical double of a unitary fusion category is always braided, meaning that one does not need to specify braiding properties for the input category in order to have emergent anyon statistics. In the special case we are considering however, that where the input category $\mathcal{C}$ is itself modular, the Drinfeld center has a special structure. For a modular input category $\mathcal{C}$ the categorical double $\mathcal{DC}$ is isomorphic to the direct product of $\mathcal{C}$ and its reverse category $\bar{\mathcal{C}}$: $\mathcal{DC} \cong \mathcal{C} \otimes \bar{\mathcal{C}}$ [59]. The reverse category $\bar{\mathcal{C}}$ is obtained from $\mathcal{C}$ as follows: for each label $a \in \mathcal{C}$, there is a corresponding label $\bar{a} \in \bar{\mathcal{C}}$ which is the same as a , except for its chirality, which is opposite: $\theta_{\bar{a}} = (\theta_a)^*$. Furthermore the R -matrix is replaced by $R_{\bar{a}}^{\bar{b}\bar{c}} = (R_a^{cb})^*$, which one can interpret as swapping the meaning of clockwise and counterclockwise. The doubled category $\mathcal{DC} \cong \mathcal{C} \otimes \bar{\mathcal{C}}$ then has elements $\{a \otimes \bar{b} \mid a \in \mathcal{C}, \bar{b} \in \bar{\mathcal{C}}\}$, which we will denote as $a\bar{b} \equiv a \otimes \bar{b}$, fusion rules

$$\delta_{a\bar{a}'\bar{b}\bar{b}'c\bar{c}'} = \delta_{abc} \delta_{\bar{a}'\bar{b}'\bar{c}'} = \delta_{abc} \delta_{a'b'c'}, \quad (\text{A40})$$

and numerical data

$$\theta_{a\bar{a}'} = \theta_a \theta_{\bar{a}'} = \theta_a (\theta_{a'})^*, \quad (\text{A41})$$

$$R_{a\bar{a}'}^{b\bar{b}'c\bar{c}'} = R_a^{bc} R_{\bar{a}'}^{\bar{b}'\bar{c}'} = R_a^{bc} (R_{a'}^{c'b'})^*, \quad (\text{A42})$$

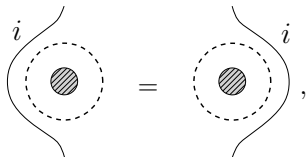
$$F_{c\bar{c}'}^{a\bar{a}'b\bar{b}'e\bar{e}'} = F_{cd\bar{d}f}^{abe} F_{c'd'f'}^{a'b'e'}. \quad (\text{A43})$$

In the following, we will exploit this structure, in order to make the connection between $\mathcal{H}_\Sigma$ and the Drinfeld center of $\mathcal{C}$ explicit.

A *vacuum line*, denoted by a dashed line, is defined as a weighted superposition of ribbons with different labels,

$$\vdots = \frac{1}{\mathcal{D}} \sum_i d_i \Big|_i. \quad (\text{A44})$$

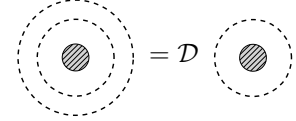
One can easily show that vacuum loops have the property that all other ribbons can freely pass over them:



$$(\text{A45})$$

where the hashed area inside the vacuum loop denotes a generic ribbon graph configuration. This represents a local identity, where the ribbon graphs inside the hashed area and outside the diagram are assumed to be the same for the left- and right-hand side. Intuitively, vacuum loops render the region they contain invisible from ribbon graph configurations

outside. Another useful identity is



$$(\text{A46})$$

which means we can pull one vacuum loop out of another.

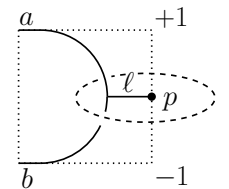
Fix a pants decomposition for Σ_n associated to some rooted binary tree T , and fix a labeling ℓ of the marked boundary points like in Eq. (A34). A labeling of T assigns an anyon label from the input category $\mathcal{C}$ to every edge of T . Such a labeling of T is said to be fusion-consistent if the fusion rules of $\mathcal{C}$ are satisfied at every internal vertex of T . A pair of labelings of T is said to be boundary-consistent with ℓ if for each marked boundary point p , the labels a and b assigned to the corresponding leaf of T by the respective labelings of T satisfy

$$\delta_{ab\ell(p)} = 1. \quad (\text{A47})$$

A pair of fusion-consistent labelings d of T that is boundary-consistent with ℓ is called a ℓ -consistent doubled anyonic fusion diagram.

The *anyonic fusion basis* for $\mathcal{H}_{\Sigma_n}^\ell$ is then an orthonormal basis indexed by ℓ -consistent doubled anyonic fusion diagrams. We still haven't defined the ribbon graph states $|\ell, d\rangle \in \mathcal{H}_{\Sigma_n}^\ell$, that correspond to the ℓ -consistent doubled anyonic fusion diagram d . We will do this by first constructing a three-dimensional ribbon graph on the thickened surface $\Sigma_n \times [-1, 1]$, and then reducing it to a regular ribbon graph on Σ_n .

Think of each of the labelings in d as a ribbon graph, and embed these in the two-dimensional slices $\Sigma_n \times \{1\}$ and $\Sigma_n \times \{-1\}$ of the thickened surface, respectively. Place the marked boundary points of Σ_n in $\Sigma_n \times \{0\}$, and add a vacuum loop in $\Sigma_n \times \{0\}$ around each of the boundary components. Then close off the diagram by adding a ribbon graph edge in $\Sigma_n \times \{0\}$ labeled by $\ell(p)$ connected to every boundary point p , and connect the corresponding edges of the graphs in $\Sigma_n \times \{1\}$ and $\Sigma_n \times \{-1\}$ to this new edge, *inside* the vacuum loop around p ,



$$(\text{A48})$$

where the vertical direction represents the additional coordinate. Boundary consistency of d ensures that the vertex created by this closing off of the diagram is a valid ribbon graph vertex. Finally, in order to reduce the resulting diagram to a state in $\mathcal{H}_{\Sigma_n}^\ell$, visualize it as a two-dimensional ribbon graph with crossings in Σ_n (this requires slightly offsetting $\Sigma_n \times \{1\}$ and $\Sigma_n \times \{-1\}$, the convention for the offset only affects the phase of $|\ell, d\rangle$). These crossings can be resolved using the

R -symbol of the input category by combining (A13), (A31) and (A27) into

$$\begin{array}{c} i \quad j \\ \diagdown \quad \diagup \\ \diagup \quad \diagdown \\ j \quad i \end{array} = \sum_k \frac{v_k}{v_i v_j} R_k^{ij} \begin{array}{c} j \quad i \\ \diagdown \quad \diagup \\ k \\ \diagup \quad \diagdown \\ i \quad j \end{array}. \quad (\text{A49})$$

The state $|\ell, d\rangle \in \mathcal{H}_{\Sigma_n}^\ell$ is defined as the resulting superposition of ribbon graphs in Σ_n that is obtained by resolving all crossings in this way. The three-dimensional ribbon graphs introduced here can be manipulated in the same way as regular ribbon graphs in Σ_n , and it is sometimes useful to perform manipulations in the three-dimensional picture before reduc-

ing the ribbon graph back to two dimensions. It was shown in Appendix A of Ref. [27] that the anyonic fusion basis is a complete orthonormal basis for $\mathcal{H}_{\Sigma_n}^\ell$. A basis for the entire Hilbert space $\mathcal{H}_{\Sigma_n}$ is then obtained by taking the union of the bases for all values of ℓ according to (A34).

The anyonic fusion basis states of $\mathcal{H}_{\Sigma_2}$ take the form

$$|k, \ell, a, b\rangle = \begin{array}{c} a \\ \circlearrowleft \\ b \end{array} \begin{array}{c} \ell \\ \text{---} \end{array} \begin{array}{c} \text{---} \\ \circlearrowright \\ k \end{array} \equiv \begin{array}{c} \ell \\ \text{---} \\ a \quad b \\ \text{---} \\ k \end{array}. \quad (\text{A50})$$

Reducing this to a two-dimensional ribbon graph yields

$$\begin{array}{c} \ell \\ \text{---} \\ a \quad b \\ \text{---} \\ k \end{array} = \frac{1}{D} v_a v_b \sum_{\alpha, \beta} v_\alpha v_\beta \sum_{\gamma, \delta} d_\gamma d_\delta R_\gamma^{a\alpha} R_\delta^{\alpha b} G_{\alpha\beta}^{k\delta\gamma} G_{\beta\alpha}^{\gamma\delta} G_{\alpha\gamma}^{k\beta\delta} \begin{array}{c} \ell \\ \text{---} \\ a \quad b \\ \text{---} \\ k \end{array}, \quad (\text{A51})$$

where the six-index G -symbol is defined as

$$\begin{array}{c} i \\ \nu \\ \circlearrowleft \\ \lambda \end{array} \begin{array}{c} \mu \\ \nu \\ \circlearrowright \\ k \end{array} = v_\lambda v_\mu v_\nu G_{\lambda\mu\nu}^{ijk} \begin{array}{c} i \\ \text{---} \\ j \quad k \end{array}. \quad (\text{A52})$$

From Eqs. (A26), (A28), (A31) and (A32), it then follows that, for an input category describing a self-dual anyon model,

$$G_{\lambda\mu\nu}^{ijk} = \frac{1}{v_k v_\nu} F_{\lambda\mu\nu}^{ijk}. \quad (\text{A53})$$

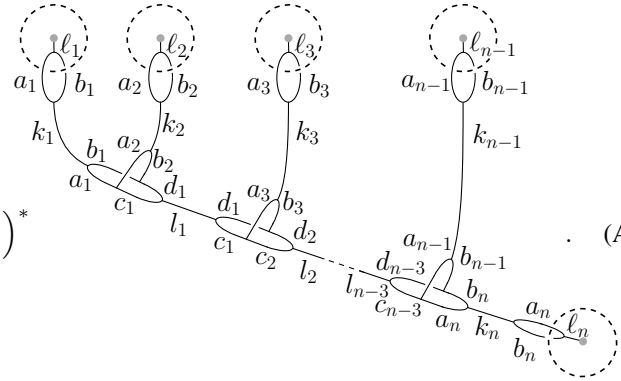
For the general case of $\mathcal{H}_{\Sigma_n}$ with a standard pants decomposition we get anyonic fusion basis states of the form

$$|\vec{\ell}, \vec{a}, \vec{b}, \vec{c}, \vec{d}\rangle = \begin{array}{c} \ell_1 \quad \ell_2 \quad \ell_3 \quad \ell_{n-1} \\ \text{---} \quad \text{---} \quad \text{---} \quad \text{---} \\ a_1 \quad b_1 \quad a_2 \quad b_2 \quad a_3 \quad b_3 \quad a_{n-1} \quad b_{n-1} \\ \text{---} \quad \text{---} \quad \text{---} \quad \text{---} \\ d_1 \quad d_2 \quad d_{n-3} \\ \text{---} \quad \text{---} \quad \text{---} \\ c_1 \quad c_2 \quad c_{n-3} \\ \text{---} \quad \text{---} \quad \text{---} \\ a_n \quad b_n \quad \ell_n \end{array}. \quad (\text{A54})$$

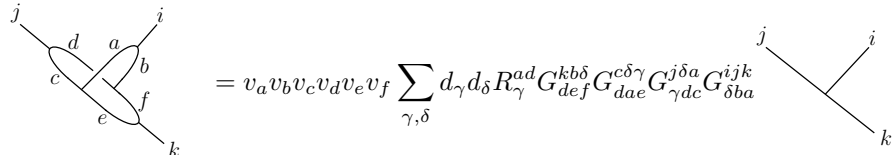
Note that the doubled ribbons were crossed before attaching them to the root ℓ_n . This is a matter of convention, and was done to give it the same geometry as the other leaves in the pants decomposition. The vacuum loop around the root puncture of the pants decomposition is unnecessary in principle, but was added here to further symmetrize the expression between all punctures¹⁵. These choices are particularly convenient for the case of self-dual input categories that we are considering here.

¹⁵ The inclusion of this additional vacuum loop should be accompanied by a factor $1/D$ in order to ensure proper normalization. We choose not to write it explicitly to avoid additional clutter. Unless mentioned otherwise, proper normalization is always assumed for anyonic fusion basis states.

Reducing basis state (A54) to a two-dimensional ribbon graph, is done by first using Eqs. (A33) and (A14) to reduce all doubled interior lines to single lines, which gives

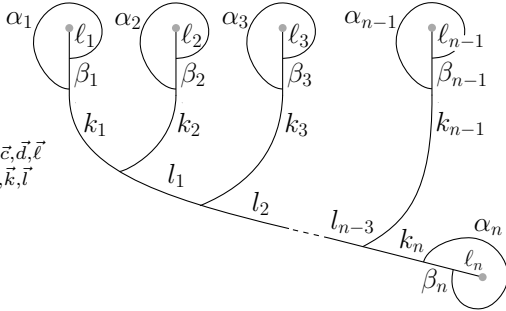
$$|\vec{\ell}, \vec{a}, \vec{b}, \vec{c}, \vec{d}\rangle = \sum_{\vec{k}, \vec{l}} \left(\prod_{x=1}^n \frac{v_{k_x}}{v_{a_x} v_{b_x}} \right) \left(\prod_{y=1}^{n-3} \frac{v_{l_y}}{v_{c_y} v_{d_y}} \right) (R_{k_n}^{b_n a_n})^* \quad (A55)$$


The leaf segments can then be reduced as in (A51), while the pants segments in (A55) can be reduced as follows:



$$= v_a v_b v_c v_d v_e v_f \sum_{\gamma, \delta} d_\gamma d_\delta R_\gamma^{ad} G_{def}^{kbb\delta} G_{dae}^{c\delta\gamma} G_{\gamma dc}^{j\delta a} G_{\delta ba}^{ijk} \quad (A56)$$

By combining all of this, one finds

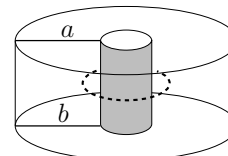
$$|\vec{\ell}, \vec{a}, \vec{b}, \vec{c}, \vec{d}\rangle = \sum_{\vec{\alpha}, \vec{\beta}, \vec{k}} X_{\vec{\alpha}, \vec{\beta}, \vec{k}, \vec{l}}^{\vec{a}, \vec{b}, \vec{c}, \vec{d}, \vec{\ell}} \quad (A57)$$


where the coefficients $X_{\vec{\alpha}, \vec{\beta}, \vec{k}, \vec{l}}^{\vec{a}, \vec{b}, \vec{c}, \vec{d}, \vec{\ell}}$ follow from Eqs. (A55), (A51) and (A56).

The basis states of the anyonic fusion basis for a given pants decomposition, can be interpreted as fusion states of anyons from the doubled category $\mathcal{DC}$. Indeed, one can easily verify that the topological phases and the R -matrix match with (A41) and (A68), respectively, by computing the action of a Dehn twist and of a braid move. Furthermore the bases corresponding to different pants decompositions are related by the correct doubled F -symbol (A43). These calculations are performed in detail in Ref. [27]. For completeness, we include them (for the self-dual case) below.

a. The action of Dehn twists, braiding, and recoupling on the anyonic fusion basis

Consider a cylindrical segment in the pants decomposition (for which we have constructed the anyonic fusion basis), with labels a and b assigned to it by an anyonic fusion basis state $|\ell, d\rangle$. The three-dimensional ribbon graph in $\Sigma_2 \times [-1, 1]$ corresponding to this cylindrical segment, can be represented as the thickened disk



$$, \quad (A58)$$

where the shaded region denotes the inner component of $(\partial\Sigma_2) \times [-1, 1]$. Note that we have applied (A45) in order to pull a vacuum loop from the shaded region. This vacuum loop can be thought of as originating at one of the boundary components inside the shaded region, using (A46). For simplicity we will stop drawing the outer boundary of the cylindrical segment. A Dehn-twist along a non-contractible curve γ around the cylinder acts on this three-dimensional ribbon graph in exactly the same way as it would on a regular ribbon graph (A38), giving

$$D(\gamma) = \theta_a \theta_b^*, \quad (\text{A59})$$

where the resulting state was simplified by moving the ribbons to $\Sigma_n \times \{0\}$ and pulling them through the vacuum loop, and using (A17) to remove the resulting kinks. The above identity shows that the anyonic fusion basis diagonalizes Dehn twists, and that the corresponding eigenvalues are the topological spins of the doubled anyons given in Eq. (A41).

Now consider a pants segment, with corresponding labels a, b and c , and a', b' and c' in $|\ell, d\rangle$. The resulting three-dimensional ribbon graph is

$$(\text{A60})$$

where we have again pulled a vacuum loop from one of the shaded regions. A braid move on the two interior holes acts in the same way it would on a regular ribbon graph (A39), giving

$$= R_a^{bc} (R_{a'}^{c'b'})^*, \quad (\text{A61})$$

where we have again pulled the upper and lower ribbons through the vacuum loop, and have used (A13) and (A14) to remove the resulting kinks. A quick glance at (A42) confirms that this is again consistent with the interpretation of fusion basis state as fusion states of doubled anyons.

Finally we investigate the relation between anyonic fusion basis states associated with different pants decompositions of Σ_n . For this we can simply use Eq. (A31) on both the top and the bottom ribbon graph independently in the three-dimensional picture. For Σ_4 this gives the relation

$$= \sum_{f,f'} F_{cdf}^{abe} F_{c'd'f'}^{a'b'e'}, \quad (\text{A62})$$

which is indeed the correct F -matrix for the doubled theory DC given in Eq. (A43).

b. Higher-genus surfaces

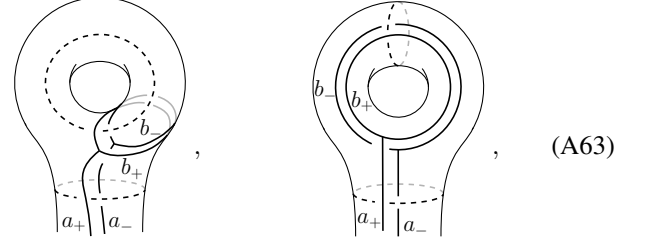
The definition of the anyonic fusion basis can be extended to surfaces of a higher genus. Consider a surface Π_n with genus g , and n punctures. We start by noting that Π_n homeomorphic to the $(n+g)$ -punctured sphere Σ_{n+g} with a handle (a

punctured torus) glued g punctures. This is known as a handle decomposition.

As before, we start by fixing a pants decomposition of Σ_{n+g} corresponding to a rooted binary tree T of our choosing. Anyonic fusion basis states of $\mathcal{H}_{\Pi_n}$ are then labeled with a labeling ℓ of the n marked boundary points, a doubled anyonic fusion diagram d of $n + g$ anyons (including the root), and a set j of g doubled anyon labels which we will call the *handle labels*. As before, the doubled anyonic fusion diagram d must be ℓ -consistent. In addition to this, it must also be consistent with the handle labels j , by which we mean that for a handle $h \in \{1, \dots, g\}$ with handle label $j(h) = b_+ \bar{b}_-$, the doubled anyonic label $a_+ \bar{a}_-$ of the corresponding leaf of T must satisfy $\delta_{a_+ b_+ b_+} = \delta_{a_- b_- b_-} = 1$

The state $|\ell, d, j\rangle \mathcal{H}_{\Pi_n}$ is then constructed similarly as the anyonic fusion basis states on the punctured sphere. Start by constructing the three-dimensional ribbon graph corresponding to the doubled fusion diagram d on $\Sigma_{n+g} \times [-1, 1]$. In the n regular punctures, the diagram is closed off according to the boundary labeling ℓ using (A48). In the remaining g punctures, the diagram is closed off using one of the follow-

ing ribbon configurations:



where a_+ and a_- are the labels of the corresponding edge in the doubled fusion diagram, and b_+ and b_- are the handle labels of the corresponding handle h .

Finally, this three-dimensional ribbon graph in $\Pi_n \times [-1, 1]$ is reduced to a regular ribbon graph in Π_n as before, using Eq. (A49) to remove any crossings. The two possible choices for the handle configuration in (A63) result in two nonequivalent bases. The unitary operator relating them can be found in Ref. [27]. For $a_+ = a_- = 1$, it is given by the S-matrix of the doubled category $\mathcal{C} \otimes \bar{\mathcal{C}}$. Note that Eqs. (A59) and (A61) are still satisfied, meaning that the resulting ribbon graph states indeed transform appropriately under the action of the mapping class group.

On a torus, the anyonic fusion basis states (with the first handle choice and $j = ef$) take the following form:

$$|\vec{\ell}, \vec{a}, \vec{b}, \vec{c}, \vec{d}, e, f\rangle =$$

$$, \quad (A64)$$

where the gray rectangle represents the periodic boundary conditions of a torus.

The subspace of $\mathcal{H}_{\Pi_n}$ where all punctures have the doubled anyon label $1\bar{1}$ is called the *anyonic vacuum*, and is isomorphic to $\mathcal{H}_{\Pi_{n=0}}$. It follows from (A63), that the anyonic vacuum on a surface with nonzero genus is degenerate. For a torus, this degeneracy is precisely the number of anyon labels in $\mathcal{C} \otimes \bar{\mathcal{C}}$, as can be deduced from (A64). The vacuum subspace on a torus is spanned by the different possible handle labels. These states are locally indistinguishable since the loops around the handle can be continuously deformed in an arbitrary way without changing the resulting state.

Note that in the case where the doubled anyons associated to the punctures of the torus have a trivial total charge (corre-

sponding to $c_{n-2} = d_{n-2} = 1$ above), the handle labels in the corresponding anyonic fusion states have no influence on braiding operations on these anyons, since the loops around the handle (the lines labeled with e and f above) can always be pulled through a group of anyons with a trivial total charge without changing the state. In practice, when working with states with a trivial total charge, one can largely ignore the precise value of the handle loop labels, since they can not be affected by local operations on $\mathcal{H}_{\Pi}$. The only operations that can modify the handle labels are those involving paths that are homotopically nonequivalent to the ribbons connected to the punctures. An example of such an operation is a clockwise exchange of two punctures along a path that is homotopically nonequivalent to the ribbons connecting these punctures in the

are given by

$$\mathcal{P}^{1\bar{1}} = \frac{1}{\mathcal{D}^2} (O_{1111} + \phi O_{11\tau\tau}), \quad (\text{A74})$$

$$\mathcal{P}^{1\bar{\tau}} = \frac{1}{\mathcal{D}^2} \left(O_{\tau\tau 1\tau} + e^{4\pi i/5} O_{\tau\tau\tau 1} + \sqrt{\phi} e^{-3\pi i/5} O_{\tau\tau\tau\tau} \right), \quad (\text{A75})$$

$$\mathcal{P}^{\tau\bar{1}} = \frac{1}{\mathcal{D}^2} \left(O_{\tau\tau 1\tau} + e^{-4\pi i/5} O_{\tau\tau\tau 1} + \sqrt{\phi} e^{3\pi i/5} O_{\tau\tau\tau\tau} \right), \quad (\text{A76})$$

$$\mathcal{P}^{\tau\bar{\tau}} = \frac{1}{\mathcal{D}^2} \left(\phi^2 O_{1111} - \phi O_{11\tau\tau} + \phi O_{\tau\tau 1\tau} + \phi O_{\tau\tau\tau 1} + \frac{1}{\sqrt{\phi}} O_{\tau\tau\tau\tau} \right). \quad (\text{A77})$$

where $\phi = (1 + \sqrt{5})/2$ is again the golden ratio. The last entry decomposes into two simple idempotents: $\mathcal{P}^{\tau\bar{\tau}} = \mathcal{P}_1^{\tau\bar{\tau}} + \mathcal{P}_\tau^{\tau\bar{\tau}}$ with

$$\mathcal{P}_1^{\tau\bar{\tau}} = \frac{1}{\mathcal{D}^2} (\phi^2 O_{1111} - \phi O_{11\tau\tau}), \quad (\text{A78})$$

$$\mathcal{P}_\tau^{\tau\bar{\tau}} = \frac{1}{\mathcal{D}^2} \left(\phi O_{\tau\tau 1\tau} + \phi O_{\tau\tau\tau 1} + \frac{1}{\sqrt{\phi}} O_{\tau\tau\tau\tau} \right). \quad (\text{A79})$$

It is important to note that both the +1 eigenstates of $\mathcal{P}_1^{\tau\bar{\tau}}$ and those of $\mathcal{P}_\tau^{\tau\bar{\tau}}$, should be interpreted as containing a $\tau\bar{\tau}$ anyon. It follows from Eq. (A68) that their respective +1 eigenspaces are related through the nilpotent operators

$$\mathcal{P}_{1\tau}^{\tau\bar{\tau}} = e^{-3\pi i/10} \frac{\phi}{\mathcal{D}} O_{1\tau\tau\tau}, \quad (\text{A80})$$

$$\mathcal{P}_{\tau 1}^{\tau\bar{\tau}} = e^{3\pi i/10} \frac{\sqrt{\phi}}{\mathcal{D}} O_{\tau 1\tau\tau}, \quad (\text{A81})$$

as follows

$$\mathcal{P}_{1\tau}^{\tau\bar{\tau}} \mathcal{P}_1^{\tau\bar{\tau}} = \mathcal{P}_\tau^{\tau\bar{\tau}}, \quad \mathcal{P}_{\tau 1}^{\tau\bar{\tau}} \mathcal{P}_\tau^{\tau\bar{\tau}} = \mathcal{P}_1^{\tau\bar{\tau}}. \quad (\text{A82})$$

6. The Levin-Wen string-net model as a lattice realization of $\mathcal{H}_\Sigma$

We now turn to the realization of the ribbon graph Hilbert space $\mathcal{H}_{\Sigma_n}$ as the ground space of a lattice spin model. We will consider surfaces Σ_n containing n punctures and no other boundary components.

Let $\mathcal{T}$ be a triangulation of Σ_n , and denote its dual graph by Λ . Then Λ is a connected graph with vertices of degree three (each corresponding to a triangle in $\mathcal{T}$). Note that we take Λ to contain boundary edges for each hole, corresponding to edges in $\mathcal{T}$ that lie along the boundary of the respective holes. We now modify the Λ as follows: for each hole, remove all but one of the boundary edges, and then remove the resulting

vertices of degree two by identifying the two edges for each of them. The are then left with a lattice containing exactly one boundary edge for each hole, and vertices of degree 3.

A qudit with local Hilbert space $\mathcal{H}_e = \mathbb{C}^N$ is placed on each edge e of Λ , where N is the number of anyon labels in the input category $\mathcal{C}$. We choose an orthonormal basis $\{|i\rangle_e\}$ for this local Hilbert space, where each basis element is labeled by an anyon type of $\mathcal{C}$. This gives a lattice spin system with a Hilbert space $\mathcal{H} \equiv (\mathcal{H}_e)^{\otimes E} = (\mathbb{C}^N)^{\otimes E}$, where E is the number of edges in Λ .

For every vertex v in Λ , we define the following vertex operator:

$$Q_v = \sum_{ijk} \delta_{ijk} |ijk\rangle \langle ijk|, \quad (\text{A83})$$

where i, j and k are the labels of the qubits associated the three edges connected to vertex v . One can easily see that all Q_v operators commute with one another, meaning that they can be simultaneously diagonalized. We denote the simultaneous +1 eigenspace of all Q_v by $\mathcal{H}_{\text{s.n.}} \equiv \{|\psi\rangle \in \mathcal{H} \mid \forall v : Q_v |\psi\rangle = |\psi\rangle\}$, and will refer to it as the *string-net subspace* of $\mathcal{H}$. Let P be the number of plaquettes in Λ , and let $\Delta \simeq \Sigma_{n+P}$ be the surface obtained by placing a puncture at the center of each plaquette in Σ_n . The states in $\mathcal{H}_{\text{s.n.}}$ can then be regarded as (superpositions of) ribbon graphs on Δ by embedding the lattice Λ in surface Δ . More precisely, one can show that the string-net subspace is isomorphic to $\bigoplus_\ell \mathcal{H}_{\Sigma_{n+P}}^{(\ell, 1^P)}$, which is the subspace of $\mathcal{H}_\Delta$ defined by the condition that all P holes corresponding to plaquettes in Λ must have a trivial boundary label.

For each plaquette p of the lattice Λ , we now introduce a plaquette operator which corresponds to adding a vacuum loop (divided by the total quantum dimension) around the puncture in Σ_{n+P} , that corresponds to the said plaquette:

$$\begin{aligned} B_p \left| \begin{array}{c} m_r \quad j_r \quad m_{r-1} \\ j_1 \quad \vdots \quad j_3 \\ m_1 \quad j_2 \quad m_2 \quad m_3 \end{array} \right\rangle &= \frac{1}{\mathcal{D}} \left| \begin{array}{c} m_r \quad j_r \quad m_{r-1} \\ j_1 \quad \text{loop} \quad j_3 \\ m_1 \quad j_2 \quad m_2 \quad m_3 \end{array} \right\rangle \\ &= \frac{1}{\mathcal{D}^2} \sum_s d_s \left| \begin{array}{c} m_r \quad j_r \quad m_{r-1} \\ j_1 \quad s \quad j_3 \\ m_1 \quad j_2 \quad m_2 \quad m_3 \end{array} \right\rangle. \end{aligned} \quad (\text{A84})$$

Its action on $\mathcal{H}_{\text{s.n.}}$ can be determined by repeatedly using Eq. (A31) to pull the interior loop onto the lattice, giving

$$B_p = \frac{1}{\mathcal{D}^2} \sum_s d_s O_p^s, \quad (\text{A85})$$

with

$$O_p^s \left| \begin{array}{c} m_r \quad j_r \quad m_{r-1} \\ j_1 \quad \vdots \quad m_3 \\ m_1 \quad j_2 \quad m_2 \end{array} \right\rangle = \sum_{k_1, \dots, k_r} \left(\prod_{\nu=1}^r F_{sk_\nu k_{\nu+1}}^{m_\nu j_{\nu+1} j_\nu} \right) \left| \begin{array}{c} m_r \quad k_r \quad m_{r-1} \\ k_1 \quad \vdots \quad m_3 \\ m_1 \quad k_2 \quad m_2 \end{array} \right\rangle. \quad (\text{A86})$$

Note that it follows from the F -symbol condition (A7) that $B_p = 0$ outside the subspace where all involved vertices v satisfy the vertex condition imposed by Q_v , meaning B_p is properly defined on the entire Hilbert space $\mathcal{H}$.

Using the identity (A46), one can see that the operators B_p are in fact projectors. They are in fact the lattice realization of the central idempotent $\mathcal{P}^{11}$ of the tube algebra defined in Eq. (A70). Furthermore, because B_p corresponds to adding a vacuum loop around a puncture inside plaquette p , all B_p operators commute. Hence, $\{Q_v\} \cup \{B_p\}$ is a set of commuting projectors.

We can now introduce the following Hamiltonian on $\mathcal{H}$:

$$H_\Lambda = - \sum_v Q_v - \sum_p B_p, \quad (\text{A87})$$

for which the ground space $\mathcal{H}_\Lambda$ is precisely the simultaneous +1 eigenspace of all Q_v and B_p operators. We claim that $\mathcal{H}_\Lambda$ is isomorphic to $\mathcal{H}_{\Sigma_n}$. Indeed, since adding a vacuum loop around a puncture allows any ribbons to be pulled across it (effectively hiding the presence of the said puncture), the +1 eigenspace of B_p inside $\mathcal{H}_{s,n}$ is isomorphic to $\bigoplus_\ell \mathcal{H}_{\Sigma_{n+P-1}}^{(\ell, 1^{P-1})}$. By this reasoning, one finds that the simultaneous +1 eigenspace of all Q_v and B_p operators is isomorphic to $\mathcal{H}_{\Sigma_n}$.

A state in the ribbon graph Hilbert space $\mathcal{H}_{\Sigma_n}$ can be explicitly mapped to the ground space $\mathcal{H}_\Lambda$ of H_Λ with the map

$$\Gamma_\Lambda : \mathcal{H}_{\Sigma_n} \rightarrow \mathcal{H}_\Lambda$$

whose action is defined as follows:

- Given a state in $\mathcal{H}_{\Sigma_n}$, continuously deform the ribbons in an arbitrary way to avoid all (imaginary) punctures at the centers of the plaquettes of Λ . This deformation yields a state in $\bigoplus_\ell \mathcal{H}_{\Delta}^{(\ell, 1^P)}$, which can be identified with a lattice state in the string-net subspace. This lattice state is then an eigenstate with eigenvalue +1 of all the vertex operators Q_v .
- Apply the total projector $B = \prod_p B_p$ to this lattice state to map it to the ground space of the string-net Hamiltonian H_Λ .

It was shown in Ref. [27] that this map defines an isomorphism between the ribbon graph Hilbert space $\mathcal{H}_{\Sigma_n}$ and the string-net ground space $\mathcal{H}_\Lambda$, which confirms our claim that $\mathcal{H}_\Lambda \simeq \mathcal{H}_{\Sigma_n}$.

For the case where $n = 0$, (A87) is precisely the Levin-Wen Hamiltonian introduced in Ref. [21], which is conjectured to describe all doubled topological phases. It's important to note that the braiding properties of the input category $\mathcal{C}$ are not required to define the Levin-Wen model, which

is defined for any unitary fusion category satisfying the additional rules (A27), (A26) and (A28). In fact, the original construction has since been generalized in order to drop these additional requirements, for example see Ref. [60].

7. The extended string-net model

In the lattice realization of the ribbon graph Hilbert space above, we found that the string-net Hilbert space of the model is isomorphic to $\bigoplus_\ell \mathcal{H}_{\Delta}^{(\ell, 1^P)}$. We will now introduce a small modification in Hamiltonian (A87), such that the string-net subspace $\mathcal{H}_{s,n}$ will now be isomorphic to $\mathcal{H}_\Delta$ instead. This is achieved by modifying the lattice on which the model is defined, in order to introduce additional degrees of freedom for each plaquette. Such a modification was first proposed in Ref. [39], with the goal of incorporating charged and dyonic excitations in (doubled) topological phases. Here, we will refer to this modified Levin-Wen model as the *extended Levin-Wen model*, or the *extended string-net model*.

The extended string-net model is defined on a tailed lattice, which is obtained by adding a “tail” edge inside of each plaquette, as is depicted for the honeycomb lattice in Fig. 35. This additional edge will allow for strings to end inside plaquettes, which corresponds to allowing any boundary labels for the P holes corresponding to the plaquettes in Δ , as opposed to fixing them to 1 as before.

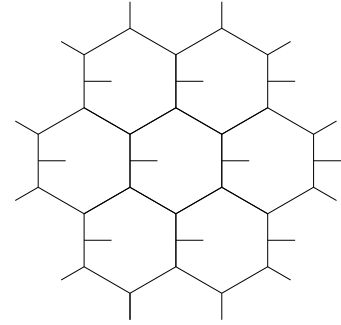
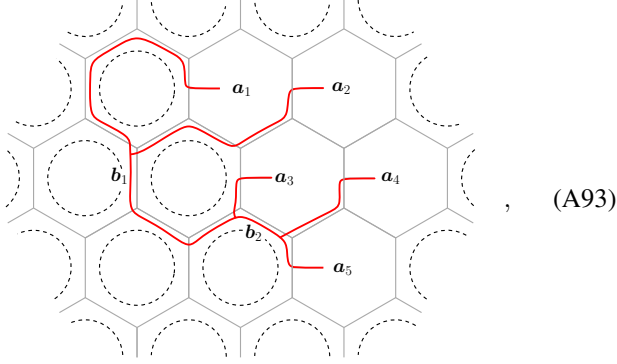


Figure 35: The tailed honeycomb lattice.

The modifications that must be made to (A87) are straightforward: the sum over all vertices is extended to include the additional P vertices connected to the tail edges, and the plaquette operator B_p must be replaced by a modified operator $\tilde{B}_p$ to accommodate the additional qudits:

$$H = - \sum_v Q_v - \sum_p \tilde{B}_p. \quad (\text{A88})$$

anyon label, this embedding could look like



where we chose not to draw the leaves with vacuum labels explicitly, but included vacuum loops in the corresponding plaquettes instead. Using the reduction to two-dimensional ribbon graphs in (A57), (A93) can be rewritten as

$$\sum_{\vec{\alpha}, \vec{\beta}, \vec{k}} X_{\vec{\alpha}, \vec{\beta}, \vec{k}, \vec{l}}^{(\vec{a}, \vec{b})} \quad \text{(A94)}$$

The actual lattice state is then obtained by using the definition (A44) to write out the vacuum loops as superpositions of ribbon loops, repeatedly using Eq. (A31) to pull all ribbons onto the lattice edges, and then finally including the all missing (trivial) tail qudits.

It should be apparent by now that even though we are considering a rather simple fusion state with only 5 nontrivial

charges, this corresponds to an exceedingly complex superposition of lattice qubit states. In App. B we will see that tensor network states provide the perfect framework for describing the anyonic fusion states in terms of local entanglement degrees of freedom.

Appendix B: Tensor network representations

During the previous two decades, tensor networks states [61–63] have emerged as a vital tool for both the theoretical and numerical study of strongly correlated quantum many body systems. Their strength originates in the fact that they can be used to describe low-energy states of interacting systems in terms of local entanglement degrees of freedom, which allows for a very efficient representation of these states despite the exponential scaling of the quantum many-body Hilbert space.

It is well known that string-net ground states allow for a remarkably simple PEPS (projected entangled pair state) representation, constructed from the algebraic data of the input category [64, 65]. Below, we will illustrate how such a tensor network representation is obtained, and then expand on these known results by constructing an explicit PEPS representation of any anyonic fusion basis state on a tailed lattice.

1. A tensor network representation for the string-net ground states

We start by constructing the PEPS representation of the ground state

$$\mathcal{N} \prod_p B_p (\otimes_e |1\rangle_e), \quad \text{(B1)}$$

where $\mathcal{N}$ is a normalization factor. Graphically, (a patch of) this state can be represented as

$$\text{Diagram} = \mathcal{N} \sum_{\{\mu\}} d_{\mu_1} d_{\mu_2} \dots \text{Diagram} \quad \text{(B2)}$$

where qudits in the $|1\rangle$ state are represented by the gray edges. To find the state from this graphical notation, one has to resolve the loops appearing in Eq. (B2) into the lattice. This is done in two steps. First we fuse the neighboring loops along

each edge using Eq. (A33):

$$\text{Diagram} = \sum_k \frac{v_k}{v_\mu v_\nu} \delta_{\mu\nu k} \text{Diagram}$$

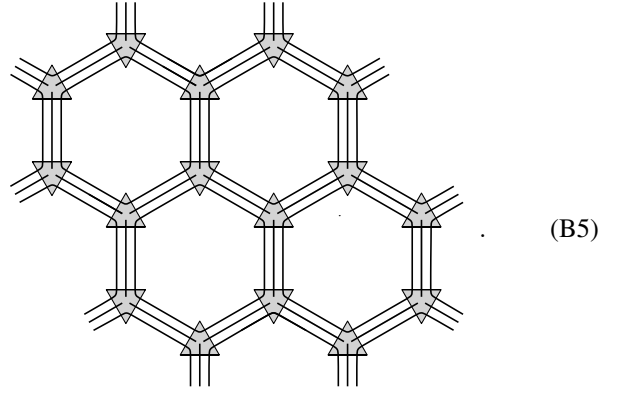
Then, the resulting configuration at every vertex is resolved into the lattice using Eq. (A52):

$$\begin{array}{c} i \\ \nu \quad \mu \\ \circ \\ j \quad \lambda \quad k \end{array} = v_\lambda v_\mu v_\nu G_{\lambda\mu\nu}^{ijk} \begin{array}{c} i \\ j \quad k \end{array}. \quad (\text{B3})$$

The PEPS tensor is obtained by splitting the factor in Eq. (B3) evenly between the two adjacent vertices. The result is

$$\begin{array}{c} \nu' \quad i \quad \mu \\ \nu \quad j \quad \lambda' \quad \lambda \quad k' \end{array} = \delta_{ii'} \delta_{jj'} \delta_{kk'} \delta_{\lambda\lambda'} \delta_{\mu\mu'} \delta_{\nu\nu'} \sqrt{v_i v_j v_k} G_{\lambda\mu\nu}^{ijk}, \quad (\text{B4})$$

where i' , j' and k' represent the physical degrees of freedom associated to the qudits on the 3 edges. Note that the physical degrees of freedom are doubled, since each of them is appearing at the two vertices connected to a given edge. The diagonal structure of Eq. (B4), ensures that the values of the two physical indices representing the same qudit are always equal. We impose the convention that for every closed loop with label μ on the virtual level, a factor d_μ is implied. This convention automatically takes care of the factors d_{μ_i} appearing in Eq. (B2), which can then be represented as



To simplify the notation, we will follow the convention that a pair of legs crossing through a tensor represents a Kronecker delta between the indices on these legs, indicating that the tensor is diagonal in this pair of indices.

The tensor network state constructed above corresponds specifically to the ground state Eq. (B1). It can, however, be modified to represent any string-net ground state. To do this, first note that we can obtain any ground state by acting with the total projector $B = \prod_p B_p$ on some state $|\phi\rangle \in \mathcal{H}_{\text{s.n.}}$ corresponding to a configurations of strings on the lattice. The tensor network state for $B|\phi\rangle$, can then be obtained by modifying our original construction, to account for additional strings running between the vacuum loops in Eq. (B2). Locally, such a string-configuration can take two forms: a single string, or two strings fusing to a third one.

The first configuration looks as follows:

$$\sum_{\{\mu\}} d_{\mu_1} d_{\mu_2} d_{\mu_3} d_{\mu_4} \begin{array}{c} s \\ \mu_1 \quad \mu_2 \\ \mu_3 \quad \mu_4 \end{array} = \sum_{\{\mu\}} \sum_{\mu'_1, \mu'_2, \mu'_4} d_{\mu_3} \frac{v_{\mu_1} v_{\mu'_1}}{v_s} \frac{v_{\mu_2} v_{\mu'_2}}{v_s} \frac{v_{\mu_4} v_{\mu'_4}}{v_s} \begin{array}{c} s \\ \mu'_1 \quad \mu'_2 \\ \mu'_3 \quad \mu'_4 \end{array}, \quad (\text{B6})$$

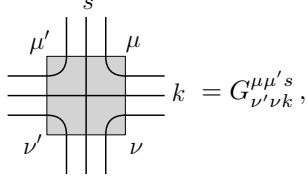
where we have pulled the string s into the different loops along its path using Eq. (A33). Using F -moves, we can again pull ribbons from neighboring plaquettes into each edges:

$$\begin{array}{c} \mu' \\ \nu' \quad \nu \end{array} \begin{array}{c} \mu \\ s \end{array} = \sum_k F_{\nu'\nu k}^{\mu\mu' s} \begin{array}{c} \mu' \\ \nu' \quad k \end{array} \begin{array}{c} \mu \\ \nu \end{array} = \sum_k \sqrt{v_s v_\mu v_{\mu'}} \sqrt{\frac{v_k}{v_\mu v_\nu}} \sqrt{\frac{v_k}{v_{\mu'} v_{\nu'}}} \sqrt{v_s v_\nu v_{\nu'}} G_{\nu'\nu k}^{\mu\mu' s} \begin{array}{c} \mu' \\ \nu' \quad k \end{array} \begin{array}{c} \mu \\ \nu \end{array}. \quad (\text{B7})$$

The first two factors on the right hand side can be recognized as the symmetrized contribution to each vertex from Eq. (B3). The first factor on the right hand side of Eq. (B7) will therefore contribute to the vertex to the left of the edge, while the second factor will contribute to the right vertex, ensuring that we can use the PEPS tensor Eq. (B4) on both vertices. The third and fourth factors on the right hand side will contribute to the upper and lower plaquette of the edge

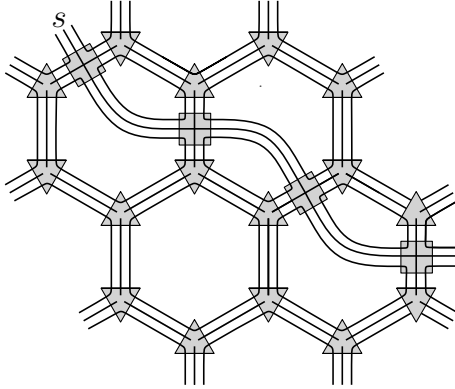
in Eq. (B7), respectively. When combined with the factors in Eq. (B6), these contributions result in a total factor of the form $\frac{v_\mu v_{\mu'}}{v_s} \cdot v_s v_\mu v_{\mu'} = d_\mu d_{\mu'}$ for each plaquette separately [instead of a single quantum dimension factor like in Eq. (B2)].

If we now place the crossing tensor



$$k = G_{\nu' \nu k}^{\mu \mu' s}, \quad (\text{B8})$$

in the PEPS at every crossing of the ribbon with label s with a lattice edge, this gives the correct superposition of qudit states. Note that the quantum dimension factors in each plaquette are again taken care of by the convention for closed loops at the virtual level. Eq. (B6) can then be represented with the following tensor network state:

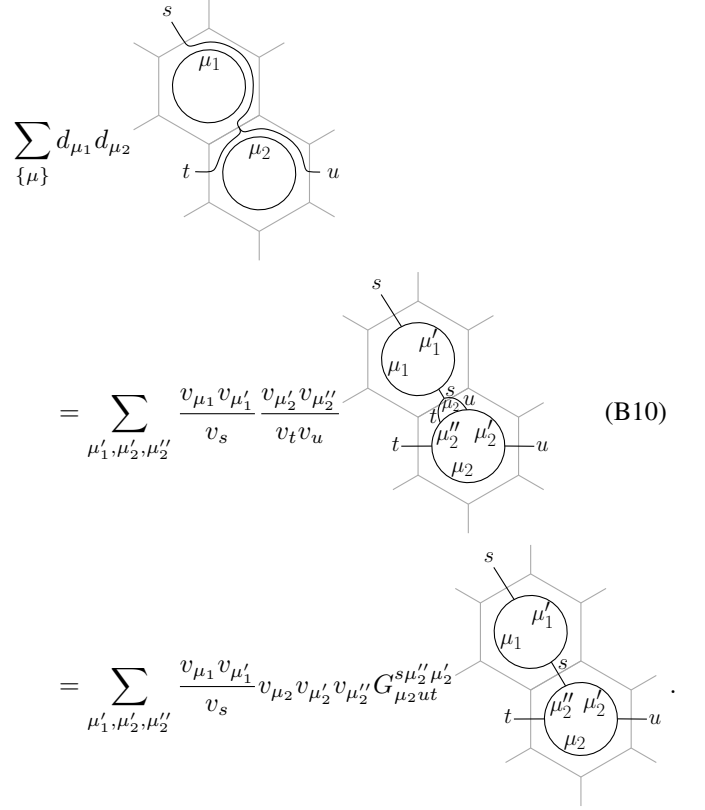


$$, \quad (\text{B9})$$

where the additional loops in the plaquette correspond to the second summation on the right hand side of Eq. (B6). Note that in the tensor network representation above, one should ensure that the string-label is fixed to the correct string-label s . In case one were to use the tensors above to represent a closed string, an additional Kronecker-delta tensor must be included to avoid summing over all string-labels.

The tensor network representation of a ground state obtained from a string-configuration containing fusing strings, is derived in a similar fashion. We again start by pulling the

strings onto the various loops using Eqs. (A33) and (A52):



$$\sum_{\{\mu\}} d_{\mu_1} d_{\mu_2} \quad (\text{B10})$$

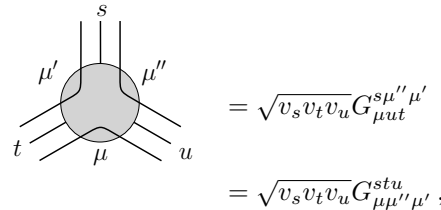
$$= \sum_{\mu_1', \mu_2', \mu_1'', \mu_2''} \frac{v_{\mu_1} v_{\mu_1'}}{v_s} \frac{v_{\mu_2'} v_{\mu_2''}}{v_t v_u} \quad (\text{B11})$$

$$= \sum_{\mu_1', \mu_2', \mu_1'', \mu_2''} \frac{v_{\mu_1} v_{\mu_1'}}{v_s} v_{\mu_2} v_{\mu_2'} v_{\mu_2''} G_{\mu_2 ut}^{s \mu_2'' \mu_2'}$$

The edge crossings can again be resolved as in Eq. (B7), giving exactly the same situation as before for the upper plaquette. For the lower plaquette, the combined contribution from of the last factor in Eq. (B7) for the three edge crossings combines with the right hand side of Eq. (B10) to a total factor of the form

$$\sqrt{v_s v_{\mu'} v_{\mu''}} \sqrt{v_t v_{\mu} v_{\mu''}} \sqrt{v_u v_{\mu} v_{\mu'} v_{\mu''}} G_{\mu ut}^{s \mu'' \mu'} = d_{\mu} d_{\mu'} d_{\mu''} \sqrt{v_s v_t v_u} G_{\mu ut}^{s \mu'' \mu'}. \quad (\text{B11})$$

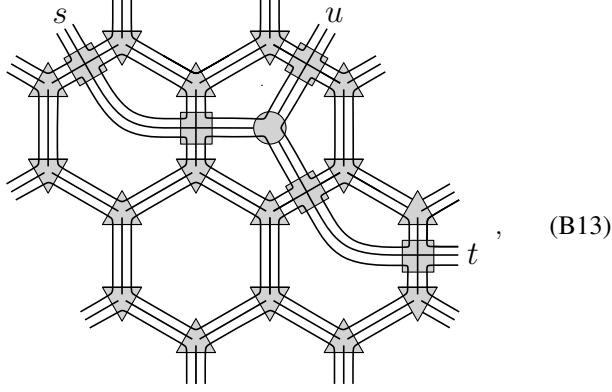
We can now define the tensor



$$= \sqrt{v_s v_t v_u} G_{\mu ut}^{s \mu'' \mu'} = \sqrt{v_s v_t v_u} G_{\mu \mu'' \mu'}^{stu}, \quad (\text{B12})$$

to represent the fusion of strings on the virtual level. This tensor, along with the closed loop convention at the virtual level, takes care of all remaining factors in Eq. (B11), and gives the correct superposition of qudit states. A ground state obtained from a initial configuration containing fusing strings

can then be represented as:



where one must again be cautious to fix the string-labels to the correct values when “closing off” this tensor network on the virtual level.

The tensors in Eqs. (B4), (B8) and (B12) can be used to construct the tensor network representation of any ground state of the Levin-Wen model. Ground states of the extend Levin-Wen model can be constructed by adding the following tensor on every vertical edge:

$$\begin{array}{c} x \\ \mu \quad | \quad \nu \\ \hline \mu \quad | \quad \nu \\ y' \\ y \end{array} t = \delta_{xx'} \delta_{yy'} \delta_{x'y'} \delta_{t1}. \quad (\text{B14})$$

This tensor simply includes a trivial tail qudit and replaces the qudit on the vertical edge by two identical ones. Since its action is trivial, we deem it unnecessary to draw it explicitly. In the tensor network diagrams below, its pretense is implied in any plaquette where the tail qudit is not specified explicitly.

2. Tensor network representations for anyonic fusion basis states

The construction above can be extended to a tensor network representation of any anyonic fusion basis state. An important ingredient that is missing is the tensor network representation of the following ribbon configuration:

$$\sum_{\mu} d_{\mu} \begin{array}{c} k \\ \mu \\ \alpha \\ \beta \\ \ell \end{array} = \sum_{\mu} \sum_{\mu'} \frac{v_{\mu} v_{\mu'}}{v_k} \begin{array}{c} k \\ \mu' \\ \mu \\ \alpha \\ \beta \\ \ell \end{array}. \quad (\text{B15})$$

We again follow the same approach and pull the ribbons onto the physical lattice, which now has a non-trivial tail edge:

$$\begin{array}{c} \mu' \\ \alpha \\ k \\ \beta \\ \ell \\ \mu \end{array} = \sum_{x,y} \sqrt{\frac{v_y}{v_{\mu} v_{\alpha}}} \sqrt{\frac{v_x}{v_{\mu'} v_{\alpha}}} \sqrt{v_k v_{\mu} v_{\mu'}} \sqrt{v_k v_x v_y v_{\alpha} v_{\beta}} G_{\alpha \mu' k}^{\mu \beta x} G_{x \beta \ell}^{\alpha y \mu} \begin{array}{c} \mu' \\ \alpha \\ x \\ y \\ \mu \end{array} \ell, \quad (\text{B16})$$

where the steps in the reduction are completely analogous to those in Eq. (B7). The first two factors on the right hand side will contribute to the bottom and top vertices, respectively, ensuring that we can use the vertex tensor Eq. (B5) at both vertices. The third factor can be recognized from Eq. (B7) as the factor contributing to the left plaquette. We then absorb the remaining factors into the definition of the string-end tensor

$$\begin{array}{c} x \quad \alpha' \\ \mu' \\ k' \\ \mu \end{array} \begin{array}{c} x' \\ y' \\ y \end{array} \ell' = \delta_{xx'} \delta_{yy'} \delta_{kk'} \delta_{\ell \ell'} \delta_{\alpha \alpha'} \sqrt{v_k v_x v_y} \frac{v_{\beta}}{v_{\alpha}} G_{\alpha \mu' k}^{\mu \beta x} G_{x \beta \ell}^{\alpha y \mu}. \quad (\text{B17})$$

This definition differs from the remaining factors on the right hand side of Eq. (B16) by $\frac{1}{d_{\alpha}}$, in order to counter the factor d_{α} that arises from the closed loop convention on the virtual level. This tensor has three physical indices, x' , y' and ℓ' , representing the physical state of the two qudits on the vertical edge and the connected tail qudit. The dotted leg with index $\alpha \beta k \ell$ is included to avoid summing over different values of the string labels, as we are representing a state in which the labels α , β , k and ℓ are fixed.

We now have all the ingredients we need to construct a PEPS realization of an arbitrary anyonic fusion basis state on the tailed lattice. All that remains is to construct PEPS tensors representing the doubled leaf and pants segments appearing in Eq. (A55) by combining the string end and fusion tensors with

the decompositions of the doubled ribbons to two-dimensional ribbon configurations. The factors $\frac{v_k}{v_a v_b}$ in Eq. (A55) that result from the reduction of the doubled ribbons in the interior branches of the fusion basis state are split evenly into both sides of the resulting single ribbon. Each doubled leaf seg-

ment therefore receives an extra factor $\sqrt{\frac{v_k}{v_a v_b}}$, which, when combined with the factors on the right hand side of Eq. (A51) yields the definition of the excitation tensor for a doubled anyon with a leaf label $\mathbf{a} = a\bar{b}_\ell$:

$$\begin{array}{c} x \quad \alpha' \\ \mu' \quad \mu \\ k' \quad \ell' \\ y \end{array} \mathbf{a} = \frac{\sqrt{v_a v_b}}{\mathcal{D}} \sum_{\alpha\beta k} \sqrt{v_k} v_\alpha v_\beta \sum_{\gamma, \delta} d_\gamma d_\delta R_\gamma^{a\alpha} R_\delta^{\alpha b} G_{\alpha ab}^{k\delta\gamma} G_{b\alpha\ell}^{\beta a\delta} G_{a\gamma\delta}^{k\beta\alpha} \begin{array}{c} x \quad \alpha' \\ \mu' \quad \mu \\ k' \quad \ell' \\ y \end{array} \alpha\beta k \ell \quad , \quad (\text{B18})$$

or, equivalently,

$$\begin{array}{c} x \quad \alpha \\ \mu' \quad \mu \\ k \quad \ell' \\ y \end{array} \mathbf{a} = \delta_{xx'} \delta_{yy'} \delta_{\ell\ell'} \frac{\sqrt{v_a v_b}}{\mathcal{D}} \sqrt{v_x v_y} d_k \sum_{\beta} d_\beta \sum_{\gamma, \delta} d_\gamma d_\delta R_\gamma^{a\alpha} R_\delta^{\alpha b} G_{\alpha ab}^{k\delta\gamma} G_{b\alpha\ell}^{\beta a\delta} G_{a\gamma\delta}^{k\beta\alpha} G_{\alpha\mu'k}^{\mu\beta x} G_{x\beta\ell}^{\alpha y\mu} . \quad (\text{B19})$$

For the tensor representing the root of the fusion tree, a factor $(R_{\ell_5}^{b_5 a_5})^*$ must be included, in accordance with Eq. (A55). We will not include it implicitly, but it is implied whenever a tensor is the root of the fusion tree. In a similar way, the doubled pants segment on the left hand side of Eq. (A56) gets an additional factor $\sqrt{\frac{v_i v_j v_k}{v_a v_b v_c v_d v_e v_f}}$ which, when combined with the reduction on the right hand side, leads to a doubled fusion tensor for three doubled Fibonacci anyons $\mathbf{a} = a\bar{b}$, $\mathbf{b} = c\bar{d}$ and $\mathbf{c} = e\bar{f}$ of the form

$$\begin{array}{c} i \\ \mu' \quad \mu'' \\ j \quad k \\ \mu \end{array} \mathbf{abc} = \sqrt{v_i v_j v_k v_a v_b v_c v_d v_e v_f} \sum_{\gamma, \delta} d_\gamma d_\delta R_\gamma^{ad} G_{def}^{kb\delta} G_{dae}^{c\delta\gamma} G_{\gamma dc}^{j\delta a} G_{\delta ba}^{ijk} \begin{array}{c} i \\ \mu' \quad \mu'' \\ j \quad k \\ \mu \end{array} . \quad (\text{B20})$$

A PEPS representation of an arbitrary anyonic fusion basis state of the form (A92) can then be constructed by applying the following correspondence rules:

$$\text{---} \mathbf{a} \rightarrow \begin{array}{c} x \quad \alpha \\ \mu' \quad \mu \\ k \quad \ell' \\ y \end{array} \mathbf{a} , \quad \begin{array}{c} \mathbf{a} \\ \mathbf{b} \quad \mathbf{c} \end{array} \rightarrow \begin{array}{c} i \\ \mu' \quad \mu'' \\ j \quad k \\ \mu \end{array} \mathbf{abc} . \quad (\text{B21})$$

For the specific case of the fusion basis state in Eq. (A93), the resulting PEPS is depicted in Fig. 36.

The PEPS representation for anyonic fusion basis states we have just derived provides a powerful tool for numerical calculations with the relevant states in the physical subspace. It was obtained here through the explicit reduction of the anyonic fusion basis states defined in App. A4. Even though the graphical calculus on these three-dimensional ribbon configurations shows that they indeed behave as fusion states of doubled Fibonacci anyons under twists and braiding, it is not

intuitively clear how this behavior translates once such a configuration has been reduced to a superposition of qudit states. It is therefore useful to explicitly verify that our PEPS ansatz itself has the desired behavior under relevant operations. We perform such checks explicitly in Sec. B4.

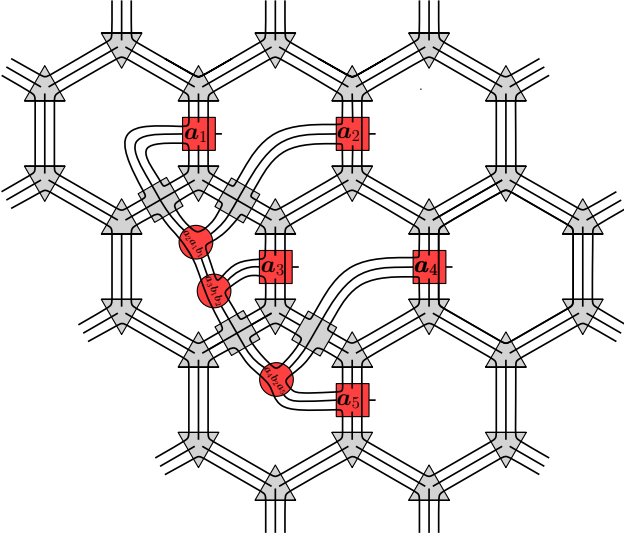


Figure 36: PEPS representation of the anyonic fusion basis state in Eq. (A93).

3. Square PEPS tensors

In practice, it is often much more convenient to work with a square geometry for the PEPS. This can be achieved by contracting the tensors within each segment. In order to simplify the resulting square tensors, we combine each group of triple indices to a single index. Note that, since the definition of the tensor (B4) requires that each set of tripled indices satisfies the Fibonacci fusion rules in order to yield a nonzero value, these regrouped virtual indices only need to have dimension 5, corresponding to the 5 combinations abc satisfying $\delta_{abc} = 1$: $\{111, 1\tau\tau, \tau1\tau, \tau\tau1, \tau\tau\tau\}$. A similar trick can be used when grouping the physical indices, since these come in three sets of 3 labels that must satisfy the fusion condition in each vertex (furthermore, these sets share indices that where doubled by the tensor network construction). The resulting square tensors will be colored gray and red for segments of which the tails end inside plaquettes containing trivial and nontrivial DFIB charges, respectively:

$$\text{Diamond} \equiv \text{Square} , \quad \text{Red Diamond} \equiv \text{Red Square} . \quad (\text{B22})$$

With this simplified notation, the fusion state depicted in Fig. 36 can be written as

$$\dots \text{ (Simplified Tensor Network) } \dots , \quad (\text{B23})$$

where we rotated the lattice counterclockwise by 60° before obtaining the squared lattice. In this simplified notation, we assume that the crossing tensors defined in Eq. (B8) are inserted at every edge crossing of a virtual string, and assume that the strings are fused using the appropriate doubled fusion tensors Eq. (B20). Note that when using the simplified tensors (without the triple indices), one must be careful to correctly enforce the closed loop convention adopted when defining the PEPS tensors on the honeycomb lattice.

4. Consistency of the tensor network representation

From the PMPO description of topological order [41] it is known that an anyon ansatz in PEPS must satisfy certain consistency conditions. It was shown through numerical calculations that our PEPS representation of the anyonic fusion basis states indeed satisfies all properties required of a tensor network description of anyonic excitations, providing an important consistency check for our framework. In particular, the braiding and fusion behavior of the DFIB excitation tensors was explicitly verified. Below we showcase some of these consistency checks, which were performed numerically with the Fibonacci input category, but should hold in general for any modular ribbon category satisfying Eqs. (A26), (A27) and (A28). It is worth noting that these consistency conditions are in fact implied by the construction of the tensors. The numerical checks merely confirm that no mistakes were made in the derivation.

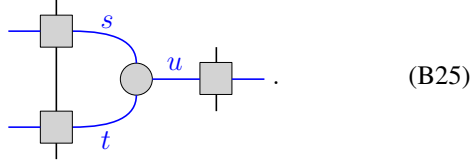
We start by noting that the representation of ribbons on the virtual level as depicted in Eq. (B9), can be understood as a MPO operator acting on the virtual level of a PEPS. The MPO tensor itself is given by what we have previously called the crossing tensor,

$$\text{Crossing Tensor} = G_{\nu\mu s}^{\mu'\nu'k} , \quad (\text{B24})$$

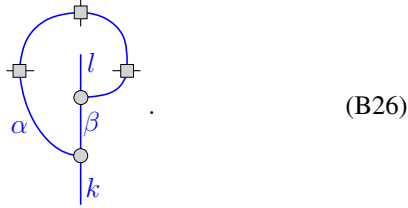
where we have rotated the definition in Eq. (B8) over 90° degrees and we now denote the index that encodes the ribbon

label s in blue. We refer to this label as the *block label* of the MPO tensor. For notational convenience we will often assume that the tripled lines are grouped (as done in Sec. B 3), in which case we only explicitly write the block label for the grouped index. It should be emphasized however that we still maintain the closed loop convention introduced before, even when we depict indices as being grouped. From the pentagon equation for the input category it follows that the MPOs can be moved freely through the ground state PEPS vertex tensors, which is referred to as the *pulling through* property. This corresponds to the freedom to continuously deform the ribbons is App. A 2, and the fact that the string-net ground state inherits this property. Adding the single block MPOs with a weight $w_s = d_s/\mathcal{D}^2$ for each block s and closing the resulting operator into a loop results in a projector MPO (PMPO) that acts trivially on any region of the PEPS with vacuum total charge. This PMPO can be thought of as the virtual representation of a vacuum loop (divided by the total quantum dimension).

The fusion tensors in Eq. (B12) that represent ribbon fusion on the virtual level can be interpreted as tensors that represent the fusion of MPOs of different blocks:



Using these fusion tensors we can build a virtual representation of the tube algebra introduced in App. A 5, which is then generated by elements of the form



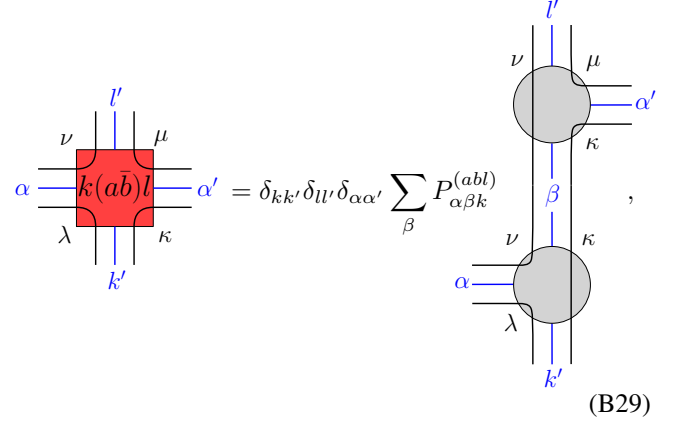
It was shown in Ref. [41] that these objects form a $\mathcal{C}^*$ -algebra whose central idempotents correspond to the topological superselection sectors of the theory, which corresponds to the results presented in App. A 5 that were we was obtained using the graphical calculus of the ribbon graph Hilbert space. In particular, we can derive the expression for the idempotents and nilpotents of the $\mathcal{C}^*$ -algebra by simply realizing Eq. (A69) on the virtual level. The operators of interest are the tube operators of the form $\mathcal{P}_{kl}^{a\bar{b}}$, that represent both the simple idempotents and nilpotents of the tube algebra. Just as in App. A 5 these can be decomposed into a superposition of the basis elements in Eq. (A66) as

$$\mathcal{P}_{kl}^{a\bar{b}} = \sum_{\alpha\beta} P_{\alpha\beta k}^{(abl)} O_{kl\alpha\beta}, \quad (\text{B27})$$

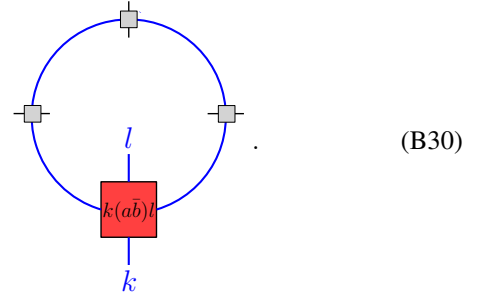
where the coefficients $P_{\alpha\beta k}^{(abl)}$ are given by

$$P_{\alpha\beta k}^{(abl)} = \frac{1}{\mathcal{D}^2} \frac{d_a d_b}{v_k} v_\alpha v_\beta \sum_{\gamma, \delta} d_\gamma d_\delta R_\gamma^{a\alpha} R_\delta^{\alpha b} G_{\alpha ab}^{k\delta\gamma} G_{b\alpha l}^{\beta a\delta} G_{a\gamma\delta}^{k\beta\alpha}. \quad (\text{B28})$$

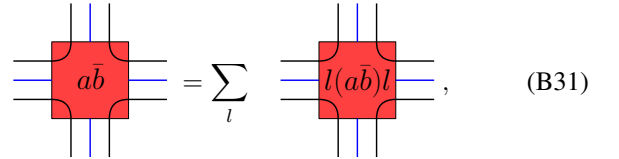
On the virtual level, these operators can be represented by the tensor



giving rise to an MPO of the form



As the central idempotents $\mathcal{P}^{a\bar{b}}$ of the tube algebra can be constructed as $\mathcal{P}^{a\bar{b}} = \sum_l \mathcal{P}_{ll}^{a\bar{b}}$, the tensors in Eq. (B29) can be combined to give the square tensors representing the central idempotents of the $\mathcal{C}^*$ -algebra:



giving a representation for the central idempotents on the virtual level.

Using these explicit expressions for the $\mathcal{P}_{kl}^{a\bar{b}}$ on the virtual level, all properties required of the anyon ansatz in our model can be explicitly verified. As a first check, the stacking behavior of the idempotents and nilpotents on the virtual level was verified by explicitly computing $\mathcal{P}_{kl}^{a\bar{b}} \mathcal{P}_{k'l'}^{a'\bar{b}'}$, giving the ex-

pected results:

$$= \delta_{aa'} \delta_{bb'} \delta_{lk'} \quad , \quad (B32)$$

This identity can be seen as the virtual representation of Eq. (A68). Next it was explicitly verified that the excitation tensors defined in Eq. (B18) behave correctly under the virtual action of the tube algebra idempotent and nilpotent MPO operators in Eq. (B29):

$$= \delta_{aa'} \delta_{bb'} \quad , \quad (B33)$$

$$\propto \quad (B34)$$

In these expressions the excitation tensors (B18) were rotated by 90 degrees counterclockwise. For the specific case of DFIB excitations, (B34) shows that the $\tau_{11}^{\tau\bar{\tau}}$ and $\tau_{\tau\tau}^{\tau\bar{\tau}}$ excitations have *virtual* support in both the $\mathcal{P}_{11}^{\tau\bar{\tau}}$ and $\mathcal{P}_{\tau\tau}^{\tau\bar{\tau}}$ simple idempotents. It should be stressed that, even though we represent the diagrams in single line notation, the closed loop convention for the PEPS representation of string-net states must be used in the actual computations of all contractions.

We conclude this section with the topological properties (fusion rules, braiding properties and topological spin) of our anyon ansatz. To simplify the expressions we again denote a DFIB charge and FIB tail label with a single bold label, $\mathbf{a} = a_+ \bar{a}_- \ell$, or $\mathbf{a} = a_+ \bar{a}_-$ depending on the context. It was verified that fusion of two excitations tensors using the doubled fusion tensor (B20) only has virtual support in the correct MPO central idempotent:

$$= \delta_{cc'} \quad (B35)$$

The braiding properties of the doubled anyons, can be translated to the following relation on the virtual level:

$$= R_c^{ab} \quad (B36)$$

As for all other relations in this section, this was explicitly verified for the Fibonacci input category. This confirms that our ansatz does indeed possess the correct DFIB braiding behavior. Finally, the topological spin of anyonic excitations, emerges on the virtual level as follows:

$$= \theta_a \quad (B37)$$

where $\theta_a = \theta_{a+}(\theta_{a-})^*$ according to Eq. (A41). Alternatively, this can be expressed on the level of the central idempotents as

$$= \theta_a \quad (B38)$$

Both relations were verified for the Fibonacci input category, which ensures that the PEPS ansatz was indeed derived correctly.

Appendix C: Scaling of largest connected group of anyons

Below, we present the results concerning the scaling of the average size of the largest group of connected anyons created by the application of depolarizing noise. These results represent a “worst case scenario” for the noise model (see

Sec. IV C) used in our threshold simulations, in the sense that they correspond to the improbable case where charge measurements of the plaquettes affected by noise operators always yield a nontrivial outcome.

The average size S of the largest connected group of anyons after noise application as a function of the linear system size L , was determined by simulating the (fixed-rate sampling) noise process for a total of 10^4 Monte Carlo samples with depolarizing noise. The results for a range of noise strengths p are shown in Fig. 37.

As expected, we find that S scales logarithmically with the system size L . This confirms that classical simulation of the system subjected to depolarizing noise, is indeed possible for noise strengths up to at least $p = 0.05$.

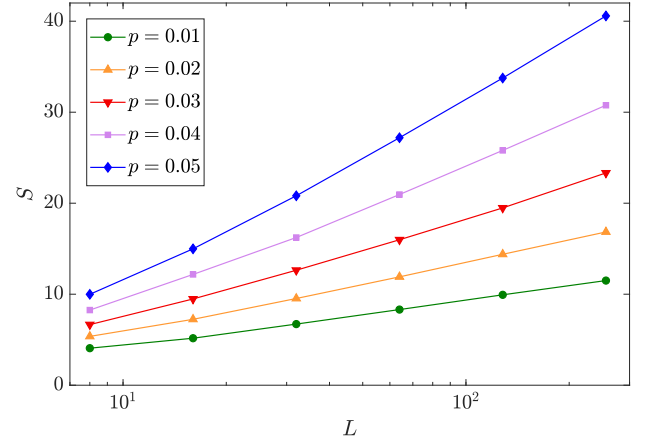


Figure 37: The worst case scenario scaling of the average size S of the largest connected group of anyons with linear system size L , for the depolarizing noise model.

Appendix D: Relating fixed-rate sampling and i.i.d. noise

The results above were presented in terms of the average number of errors per qubit p , which is the parameter char-

acterizing the noise strength in our fixed-rate sampling noise model. This is a natural measure in the context of this work, as the non-Abelian nature of our quantum memory and the fact that it undergoes constant syndrome measurements mean that noise processes cannot be formulated in terms of an independent and identically distributed noise model. However, in order to compare our results to known error threshold results for Abelian codes, we require a way of relating p to the i.i.d. noise strength $p_{\text{i.i.d.}}$ used in such models. To this end, we illustrate the relation between the Poisson and i.i.d. binomial noise models in Abelian stabilizer codes, for which both these noise models are equally valid, following the reasoning presented in [22].

We again consider our Poisson noise model in which a total of T error operations are executed, where T is drawn from a Poisson distribution with mean $T_0 = |E|p$ and $|E|$ is the total number of edges, but now apply it in the context of an Abelian stabilizer code. For each individual error process, an edge e of the lattice is chosen at random and a Pauli operator σ_i is

applied according to relative probabilities $\{\gamma_x, \gamma_y, \gamma_z\}$. If we think of the global fixed-rate sampling noise model as arising from a superposition of fixed-rate sampling noise processes on each edge of the lattice individually, then we may assume that the latter can be approximately characterized as Poisson distributed events themselves. According to this reasoning, the noise model would then be captured by a superposition of three different Poisson distributions at the level of each individual edge, with means $\gamma_x p$, $\gamma_y p$ and $\gamma_z p$ for σ_x , σ_y and σ_z errors, respectively.

Since all Pauli errors acting on a given edge either commute or anticommute, we can compute the i.i.d probability for a net σ_x error on an individual edge, $p_{\text{i.i.d.}}^x$, by adding the probabilities of all Poisson error processes where the number of σ_x errors acting on the edge is odd and the number of σ_y and σ_z errors acting on the edge are both even, or where the number of σ_x errors acting on the edge is even and the number of σ_y and σ_z errors acting on the edge are both odd. We therefore get:

$$\begin{aligned} p_{\text{i.i.d.}}^x &= \left(\sum_{k=0}^{+\infty} e^{-\gamma_x p} \frac{(\gamma_x p)^{(2k+1)}}{(2k+1)!} \right) \left(\sum_{k=0}^{+\infty} e^{-\gamma_y p} \frac{(\gamma_y p)^{(2k)}}{(2k)!} \right) \left(\sum_{k=0}^{+\infty} e^{-\gamma_z p} \frac{(\gamma_z p)^{(2k)}}{(2k)!} \right) \\ &\quad + \left(\sum_{k=0}^{+\infty} e^{-\gamma_x p} \frac{(\gamma_x p)^{(2k)}}{(2k)!} \right) \left(\sum_{k=0}^{+\infty} e^{-\gamma_y p} \frac{(\gamma_y p)^{(2k+1)}}{(2k+1)!} \right) \left(\sum_{k=0}^{+\infty} e^{-\gamma_z p} \frac{(\gamma_z p)^{(2k+1)}}{(2k+1)!} \right) \\ &= \frac{e^{-p}}{4} \left(e^p + e^{(\gamma_x - \gamma_y - \gamma_z)p} - e^{(-\gamma_x + \gamma_y - \gamma_z)p} - e^{(-\gamma_x - \gamma_y + \gamma_z)p} \right). \end{aligned} \quad (\text{D1})$$

Similar expressions can be derived in an analogous fashion for $p_{\text{i.i.d.}}^y$, $p_{\text{i.i.d.}}^z$ and $p_{\text{i.i.d.}}^{\text{none}}$, which denote the i.i.d. error probability for a net σ_y error, net σ_z error or no net error at all on an individual edge, respectively:

$$p_{\text{i.i.d.}}^y = \frac{e^{-p}}{4} \left(e^p + e^{(-\gamma_x + \gamma_y - \gamma_z)p} - e^{(\gamma_x - \gamma_y - \gamma_z)p} - e^{(-\gamma_x - \gamma_y + \gamma_z)p} \right) \quad (\text{D2})$$

$$p_{\text{i.i.d.}}^z = \frac{e^{-p}}{4} \left(e^p + e^{(-\gamma_x - \gamma_y + \gamma_z)p} - e^{(\gamma_x - \gamma_y - \gamma_z)p} - e^{(-\gamma_x + \gamma_y - \gamma_z)p} \right) \quad (\text{D3})$$

$$p_{\text{i.i.d.}}^{\text{none}} = \frac{e^{-p}}{4} \left(e^p + e^{(\gamma_x - \gamma_y - \gamma_z)p} + e^{(-\gamma_x + \gamma_y - \gamma_z)p} + e^{(-\gamma_x - \gamma_y + \gamma_z)p} \right). \quad (\text{D4})$$

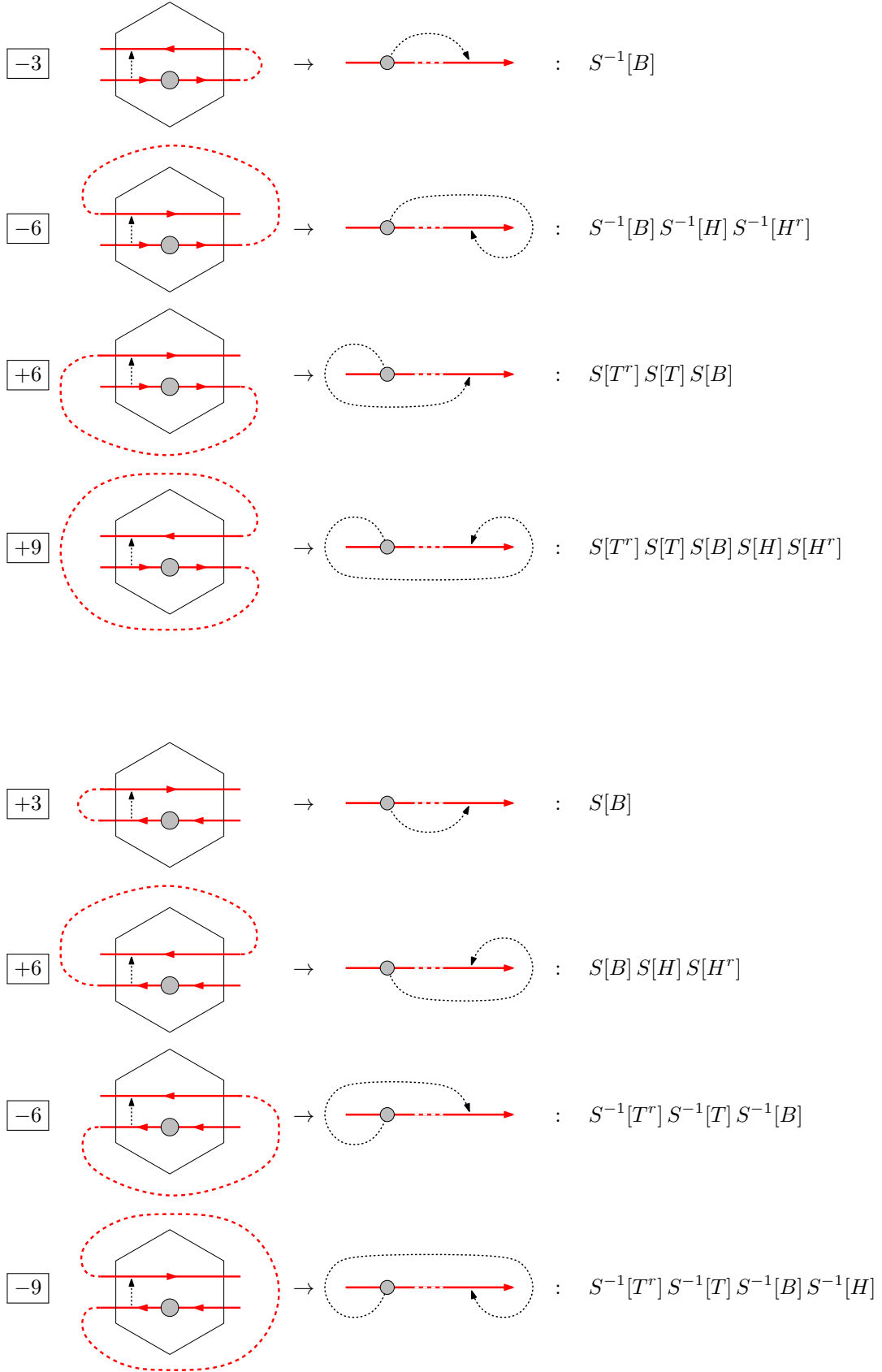
Hence, for $p \ll 1$ we have $p_{\text{i.i.d.}}^x \approx \gamma_x p$, $p_{\text{i.i.d.}}^y \approx \gamma_y p$, $p_{\text{i.i.d.}}^z \approx \gamma_z p$ and $p_{\text{i.i.d.}}^{\text{none}} \approx 1 - p$.

Appendix E: Paperclip configurations

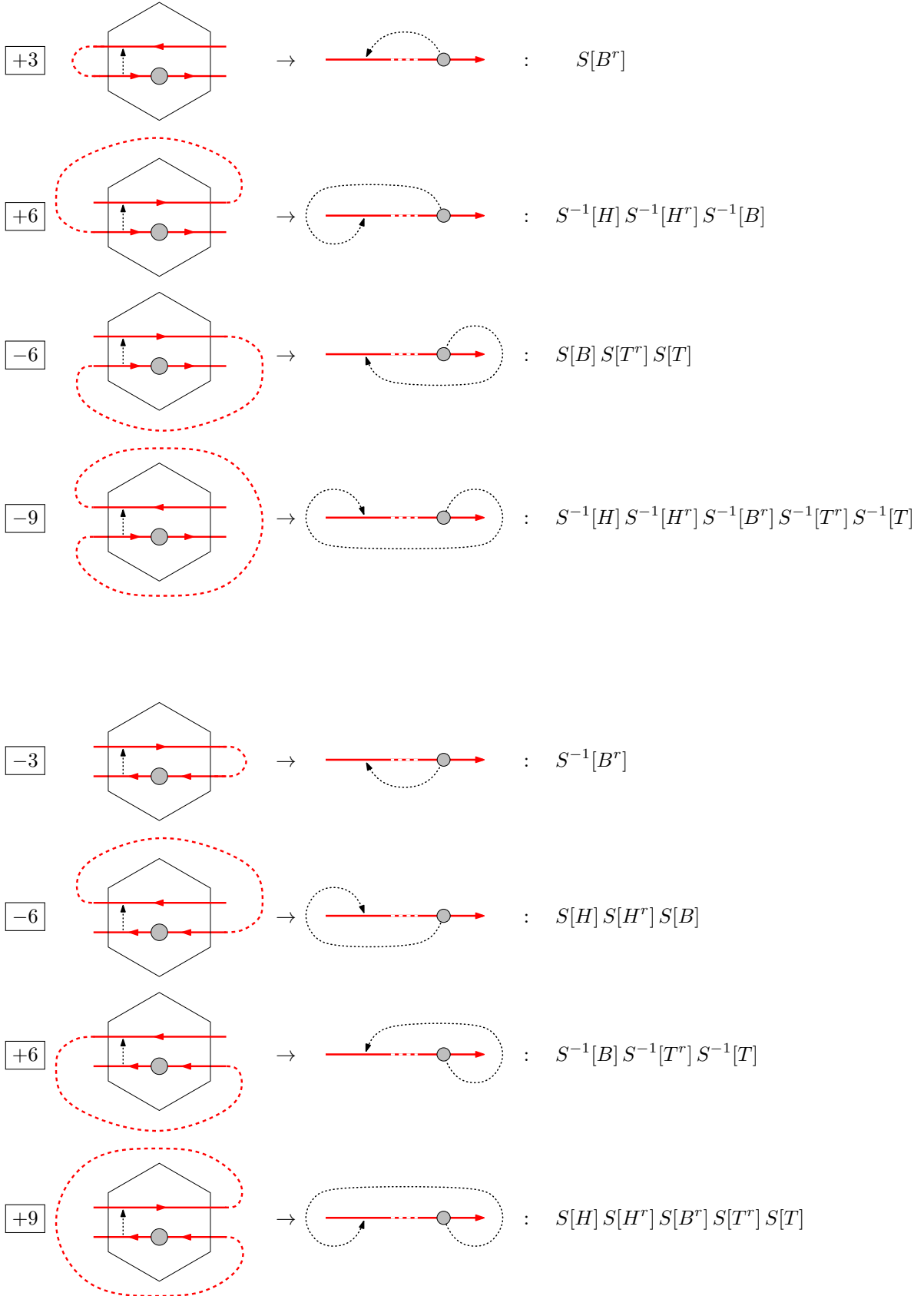
The paperclip algorithm (see Sec. IV F 4) works by exploiting the fact that refactoring moves along the boundary of an individual tile, can be classified into 16 different paperclip configurations, and that these entirely determine the corre-

sponding sequence of swaps. These 16 configurations can be split into two groups corresponding to refactoring moves along or against the orientation of the curve diagram. Each of these groups can itself be split into two groups, depending on whether the initial piece of curve containing the anyon has an incoming or outgoing orientation. Within each of these groups, the different configurations can be distinguished using their respective turn numbers, as defined in Sec. IV F 4. Below, we list all possible paperclip configurations, and the corresponding sequence of swaps for each of them.

1. Refactoring along the curve



2. Refactoring against the curve



- [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* **549**, 172 (2017).
- [3] A. Y. Kitaev, Fault-tolerant quantum computation by anyons, *Ann. Phys. (NY)* **303**, 2 (2003).
- [4] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, *J. Math. Phys.* **43**, 4452 (2002).
- [5] S. B. Bravyi and A. Y. Kitaev, Quantum codes on a lattice with boundary (1998), 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 (2018), arXiv:1810.09571 [quant-ph].
- [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, A. Stern, M. Freedman, and S. Das 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, *Comm. 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, *Comm. 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, *Comm. 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. (NY)* **325**, 2707 (2010).
- [28] N. Bonesteel and D. 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, *Comm. Math. Phys.* **127**, 1 (1990).
- [33] E. Witten, Quantum field theory and the jones polynomial, *Comm. 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, In preparation.
- [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.*, Operator algebras, topology and subgroups of quantum symmetry - construction of subgroups of quantum groups -, in *Taniguchi Conference on Mathematics Nara'98* (Mathematical Society of Japan, 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. (NY)* **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] R. N. Pfeifer, O. Buerschaper, S. Trebst, A. 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).
- [46] S. Burton, A short guide to anyons and modular functors (2016), arxiv:1610.05384 [quant-ph].
- [47] M. Z. Bazant, Largest cluster in subcritical percolation, *Phys. Rev. E* **62**, 1660 (2000).
- [48] M. B. Hastings, G. H. Watson, and R. G. Melko, Self-correcting quantum memories beyond the percolation threshold, *Phys.*

- Rev. Lett. **112**, 070501 (2014).
- [49] 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).
 - [50] S. Bravyi and J. Haah, Quantum self-correction in the 3d cubic code model, Phys. Rev. Lett. **111**, 200501 (2013).
 - [51] C. Wang, J. Harrington, and J. Preskill, Confinement-higgs transition in a disordered gauge theory and the accuracy threshold for quantum memory, Ann. Phys. (NY) **303**, 31 (2003).
 - [52] A. G. Fowler, Accurate simulations of planar topological codes cannot use cyclic boundaries, Phys. Rev. A **87**, 062320 (2013).
 - [53] 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).
 - [54] 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).
 - [55] A. Kitaev and L. Kong, Models for gapped boundaries and domain walls, Comm. Math. Phys. **313**, 351 (2012).
 - [56] L. Lootens, J. Fuchs, J. Haegeman, C. Schweigert, and F. Verstraete, Matrix product operator symmetries and intertwiners in string-nets with domain walls (2020), arxiv:2008.11187 [quant-ph].
 - [57] 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).
 - [58] Z. Wang, *Topological quantum computation*, 112 (American Mathematical Soc., 2010).
 - [59] P. Etingof, S. Gelaki, D. Nikshych, and V. Ostrik, *Tensor categories*, Vol. 205 (American Mathematical Soc., 2016).
 - [60] A. Hahn and R. Wolf, Generalized string-net model for unitary fusion categories without tetrahedral symmetry, Phys. Rev. B **102**, 115154 (2020).
 - [61] F. Verstraete, V. Murg, and J. I. Cirac, Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems, Adv. Phys. **57**, 143 (2008).
 - [62] J. C. Bridgeman and C. T. Chubb, Hand-waving and interpretive dance: an introductory course on tensor networks, J. Phys. A **50**, 223001 (2017).
 - [63] I. Cirac, D. Perez-Garcia, N. Schuch, and F. Verstraete, Matrix product states and projected entangled pair states: Concepts, symmetries, and theorems (2020), arXiv:2011.12127 [quant-ph].
 - [64] Z.-C. Gu, M. Levin, B. Swingle, and X.-G. Wen, Tensor-product representations for string-net condensed states, Phys. Rev. B **79**, 085118 (2009).
 - [65] O. Buerschaper, M. Aguado, and G. Vidal, Explicit tensor network representation for the ground states of string-net models, Phys. Rev. B **79**, 085119 (2009).