Quantum Lego Expansion Pack: Enumerators from Tensor Networks

ChunJun Cao,[1,2,3,*] Michael J. Gullans,[1] Brad Lackey,[4] and Zitao Wang[5]

[1] *Joint Center for Quantum Information and Computer Science, NIST/University of Maryland, College Park, Maryland 20742, USA*

[2] *Institute for Quantum Information and Matter, Caltech, Pasadena, California 91125, USA*

[3] *Department of Physics, Virginia Tech, Blacksburg, Virginia 24060, USA*

[4] *Microsoft Quantum, Redmond, Washington 98052, USA*

[5] *Meta Platforms Inc., Menlo Park, California 94025, USA*

(Received 9 November 2023; revised 1 March 2024; accepted 23 May 2024; published 22 July 2024)

We provide the first tensor-network method for computing quantum weight enumerator polynomials in the most general form. If a quantum code has a known tensor-network construction of its encoding map, our method is far more efficient, and in some cases exponentially faster than the existing approach. As a corollary, it produces decoders and an algorithm that computes the code distance. For non-(Pauli)-stabilizer codes, this constitutes the current best algorithm for computing the code distance. For degenerate stabilizer codes, it can be substantially faster compared to the current methods. We also introduce novel weight enumerators and their applications. In particular, we show that these enumerators can be used to compute logical error rates exactly and thus construct (optimal) decoders for any independent and identically distributed single qubit or qudit error channels. The enumerators also provide a more efficient method for computing nonstabilizerness in quantum many-body states. As the power for these speedups rely on a quantum Lego decomposition of quantum codes, we further provide systematic methods for decomposing quantum codes and graph states into a modular construction for which our technique applies. As a proof of principle, we perform exact analyses of the deformed surface codes, the holographic pentagon code, and the two-dimensional Bacon-Shor code under (biased) Pauli noise and limited instances of coherent error at sizes that are inaccessible by brute force.

DOI: 10.1103/PRXQuantum.5.030313

I. INTRODUCTION

Topological and geometrical insights have led to a number of recent breakthroughs in quantum error correction, e.g., Refs. [1–3]. On the other hand, quantum weight enumerator polynomials [4] provide a complementary, algebraic perspective on quantum error-correcting codes (QECCs). Quantum weight enumerators contain crucial information of the code property. A number of variants and generalizations have also been applied to derive linear programming bounds [5–7], to understand error detection under symmetric [8] and asymmetric [9] Pauli errors, and for generating magic state distillation protocols [10]. In quantum many-body physics, enumerators, also known as sector lengths, have been used to study the entanglement structure [11] of quantum states. The weight distributions of operators have also played an important role in quantum chaos [12]. However, wider applications of the quantum weight enumerators have been relatively limited beyond codes or states of small sizes compared to the other approaches partly to due their prohibitive computational costs.

Building upon the previous framework of quantum Lego (QL) [13] and the recently developed tensor-weight-enumerator formalism [14], we revisit the weight-enumerator perspective of quantum error correction and provide a more efficient method to compute them. We present new results in both formalism and in algorithm that enable a number of novel applications for quantum error correction, measurement-based quantum computation, and quantum many-body physics. On the formalism level, we review abstract weight enumerators and their corresponding MacWilliams identities [14]. We then introduce mixed enumerators, higher genus enumerators, coset enumerators, and generalized enumerators, which are useful for the study of subsystem codes, decoders, and logical error probability under general independent

*Contact author: ccj991@gmail.com

and identically distributed (i.i.d.) single-qubit error channels.

On the algorithmic level, we provide a tensor-network method for computing these quantum weight enumerators in their most abstract forms. Because one can read off the code distance from weight enumerators, the problem of finding them is at least as hard as the minimal distance problem for classical linear codes, which is NP hard [15–18]. We show that quantum weight enumerators also produce optimal decoders, hence the general problem is at least #P complete, which is the hardness of evaluating weight enumerators for classical linear codes [19]. However, more efficient algorithms are possible if additional structures are known. To the best of our knowledge, our work constitutes the best current algorithm for generating quantum weight-enumerator polynomials as long as a good QL construction for the quantum code is known. Compared to the brute-force method, our algorithm provides up to a substantial speedup.

The enumerators immediately induce a protocol to compute quantum code distances. To the best of our knowledge, it provides the first such protocol for general quantum codes beyond (Pauli) stabilizer codes, which can be exponentially faster than brute-force search in many instances. For nondegenerate Pauli stabilizer codes, the complexity scaling is roughly comparable with existing algorithms for classical linear codes under reasonable assumptions, which implies that it scales exponentially with the code distance. For degenerate codes, our method can be exponentially faster in certain instances compared to known methods based on classical linear codes.

Finally, we introduce novel applications and new abstract enumerators that have not been discussed in the literature. We generalize [8] and connect enumerators to logical error probabilities when the code is subjected to any i.i.d. single-qudit error channel. We then provide the optimal decoder for any code that admits a known QL construction and propose a more accurate method to compute effective distances and error thresholds. Our arguments hints at a general connection between the hardness of distance calculation, optimal decoding, and the amount of entanglement present in the system. Because the speedup relies on a tensor-network construction of the quantum code or quantum state in question, we also provide a systematic method for building all stabilizer codes and graph states using quantum Lego. This also includes the cases where the stabilizer group is non-Abelian, such as the quantum double models. These breakthroughs lift a long-standing computational barrier for the exact analyses of quantum codes and resource states in measurement-based quantum computation. Additionally, we show that the higher genus weight enumerator of pure states computes the nonstabilizerness of a state, thus providing an another efficient method for the challenging task of computing quantum many-body magic.

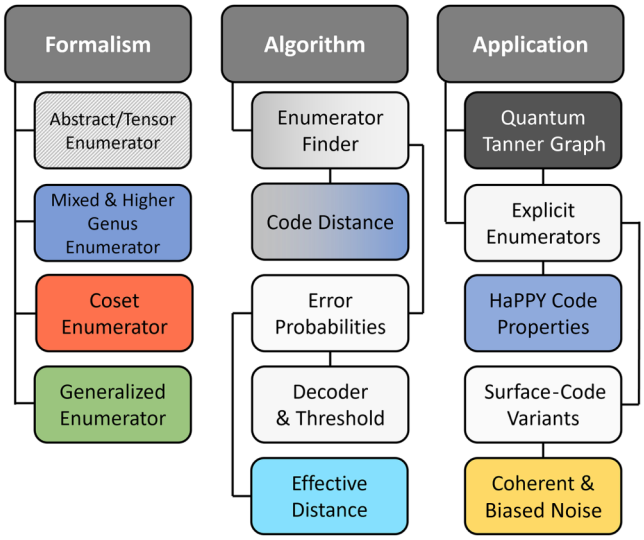

FIG. 1. Summary of contributions. Topic dependencies are red-green-blue color coded. If all three colored topics in the formalism section are used, then the color is white. Cyan indicates green and blue topics. Yellow indicates red and green topics. Black indicates that it does not use any of the new formalism, but is a new tensor-network construction. Half shaded grey and blue indicates it uses grey and blue topics. Half shaded grey and white indicates that it uses all four formalism topics.

As a proof of principle, we derive weight enumerators, compute (biased) distances, and obtain exact analytical expressions for logical error probabilities under depolarizing and coherent noise for a few well-known stabilizer and subsystem codes that are of order a hundred qubits or so. The novel contributions in this paper are summarized in Fig. 1. Overall, the tensor-network-based approach to quantum weight enumerators provides both a unified framework and the practical means for studying code properties, decoding, entanglement, and magic in quantum codes and quantum many-body systems at large.

In Sec. II, we review the basics of weight-enumerator polynomials in the most abstract form and introduce their generalizations. In Sec. III, we discuss their existing applications for computing code distance and extend their applications for error detection under general error channels. We present novel constructions, such as mixed enumerators, higher genus enumerators, and coset enumerators. We introduce new applications of these enumerators in building optimal decoders, in analyzing subsystem codes, and in computing nonstabilizerness in quantum states. We show that the error-detection threshold for a large class of codes have a universal value of $1/6$ and suggest improvements for threshold computations based on existing sampling-based methods when used in conjunction with enumerators. Then we discuss the computational cost of this method and provide some entanglement-based intuition in Sec. V. As a proof of principle, and to provide

novel analysis of existing codes, we study some common examples and explain their significance in Sec. VI. In Sec. VI A we construct various weight enumerators of the (rotated) surface code and its deformations. We compare their performances under biased noise and coherent error channels. In Sec. VI B we provide a new tensor-network construction of the two-dimensional (2D) color code using Steane codes as basic building blocks and compute its enumerators. In Sec. VI C we study different bulk qubits with mixed enumerators in the holographic HaPPY code. We obtain their (biased) distances and performance under (biased) Pauli noise. In Sec. VI D we apply the mixed enumerator technology to the Bacon-Shor code and showcase its computation for subsystem codes. Finally, we make some summarizing comments in Sec. VII and provide insights on the connection with the stat mech model and graph states.

We prove the relevant theorems, discuss technical implementations and clarify practical simplifications in the Appendices. Although not stated explicitly, the distance finding protocol introduced in Ref. [20] effectively computes the Shor-Laflamme enumerators for a subset of stabilizer codes known as local tensor-network codes. Their approach also shares a number of similarities with our own, which we explain in Appendix C 3. For such stabilizer codes, our protocol generally offers a quadratic speedup in the form of reduced bond dimensions. In the regime where the stabilizer code has a high rate and code words are highly entangled, our method can lead to an exponential advantage using the quantum MacWilliams identities.

II. GENERAL FORMALISM

Throughout the paper, we represent multi-indexed objects like vectors and tensors in bold face letters $\mathbf{A}, \mathbf{B}$ to avoid clutter of indices. Scalar objects are written in regular fonts like A, B.

A. Abstract scalar weight enumerator

Abstract scalar weight enumerators introduced in Ref. [14] include common enumerators discussed in the literature [4,6,9]. Let $\mathcal{E}$ be an error basis on Hilbert space $\mathfrak{H}$ with local dimension q. A *weight function* is any function $\mathrm{wt} : \mathcal{E} \to \mathbb{Z}_{\geq 0}^k$. We extend this (without introducing new notation) to $\mathrm{wt} : \mathcal{E}^n \to \mathbb{Z}_{\geq 0}^k$ by

$$\mathrm{wt}(E_1 \otimes \cdots \otimes E_n) = \mathrm{wt}(E_1) + \cdots + \mathrm{wt}(E_n). \tag{2.1}$$

For a k tuple of indeterminates $\mathbf{u} = (u_1, \ldots, u_k)$ we write

$$\mathbf{u}^{\mathrm{wt}(E)} = u_1^{\mathrm{wt}(E)_1} \cdots u_k^{\mathrm{wt}(E)_k}. \tag{2.2}$$

We can then define abstract enumerators of Hermitian operators M_1, M_2 for a weight function wt as

$$\begin{aligned} A(\mathbf{u}; M_1, M_2) &= \sum_{E \in \mathcal{E}^n} \mathrm{Tr}(EM_1)\,\mathrm{Tr}(E^\dagger M_2)\mathbf{u}^{\mathrm{wt}(E)} \\ B(\mathbf{u}; M_1, M_2) &= \sum_{E \in \mathcal{E}^n} \mathrm{Tr}(EM_1E^\dagger M_2)\mathbf{u}^{\mathrm{wt}(E)}. \end{aligned} \tag{2.3}$$

These polynomials satisfy a quantum MacWilliams identity. Let us restrict to the case where our error basis satisfies $EFE^\dagger F^\dagger = \omega(E,F)I$ for a phase $\omega(E,F)$. This includes the Pauli basis (of local dimension q) as well as general Heisenberg representations. Consider the (polynomial-valued) function $f(E) = \mathbf{u}^{\mathrm{wt}(E)}$ for a weight function $\mathrm{wt} : \mathcal{E} \to \mathbb{Z}_{\geq 0}^k$. Then the discrete Wigner transform of this function is

$$\hat{f}(D) = \frac{1}{q}\sum_E \omega(E,D) f(E) = \frac{1}{q}\sum_E \omega(E,D)\mathbf{u}^{\mathrm{wt}(E)}. \tag{2.4}$$

Theorem 1. Suppose there exists an algebraic mapping $\Phi(\mathbf{u}) = (\Phi_1(\mathbf{u}), \ldots, \Phi_k(\mathbf{u}))$ such that

$$\Phi(\mathbf{u})^{\mathrm{wt}(D)} = \hat{f}(D) = \frac{1}{q}\sum_E \omega(E,D)\mathbf{u}^{\mathrm{wt}(E)}. \tag{2.5}$$

Then for any M_1, M_2 we have

$$B(\mathbf{u}; M_1, M_2) = A(\Phi(\mathbf{u}); M_1, M_2). \tag{2.6}$$

Proof. See Ref. [14]. ■

The map Φ is a generalization of the discrete Wigner transform. For the remainder of the work, we take $\mathcal{E}$ to be the Pauli group. By considering different forms of the variable $\mathbf{u}$, abstract weight function wt, and transformation Φ, one can recover existing scalar enumerator polynomials and their MacWilliams identities. For completeness, we review a few common enumerators in Appendix A that are used in this work.

B. Generalized abstract weight enumerators

Slightly extending the form in the previous section, we define a novel generalized weight enumerator.

$$\begin{aligned} \bar{A}(\mathbf{u}; M_1, M_2) &= \sum_{E,F \in \mathcal{E}^n} \mathrm{Tr}[EM_1]\,\mathrm{Tr}[F^\dagger M_2]\mathbf{u}^{\mathrm{wt}(E,F)}, \\ \bar{B}(\mathbf{u}; M_1, M_2) &= \sum_{E,F \in \mathcal{E}^n} \mathrm{Tr}[EM_1F^\dagger M_2]\mathbf{u}^{\mathrm{wt}(E,F)}, \end{aligned} \tag{2.7}$$

where $\mathrm{wt}(E,F)$ is an abstract function of the operators E, F and $\mathbf{u}$ is a set of variables. It has no obvious classical

analogues as far as we know. These types of enumerators are useful in analyzing qudit-wise general error channels. We further elaborate this connection in Sec. III B for coherent noise and other single-qubit errors such as amplitude damping channels. We are not able to identify MacWilliams identities for these types of enumerator polynomials in general.

C. Tensor weight enumerators

One can generalize the above scalar enumerator formalism to vectors and tensors. The reasons for this extension is twofold: (1) the novel vector or tensor enumerators can probe code properties unavailable to their scalar counterparts and (2) the cost for computing scalar enumerators is generally expensive and scales exponentially with $n-k$. However, by contracting suitable tensor weight enumerators, one can break down the computation of scalar enumerators into manageable pieces and render the process far more efficient. In this section, we briefly review the basic definitions of these vectorial and tensorial enumerators and introduce their graphical representations.

From Ref. [14], we define tensor enumerators

$$\mathbf{A}^{(J)}(\mathbf{u};M_1,M_2)=\sum_{E,\bar{E}\in\mathcal{E}^m}\sum_{F\in\mathcal{E}^{n-m}}\mathrm{Tr}((E\otimes_J F)M_1)\times\mathrm{Tr}((\bar{E}^\dagger\otimes_J F^\dagger)M_2)\mathbf{u}^{\mathrm{wt}(F)}e_{E,\bar{E}}, \quad (2.8)$$

$$\mathbf{B}^{(J)}(\mathbf{u};M_1,M_2)=\sum_{E,\bar{E}\in\mathcal{E}^m}\sum_{F\in\mathcal{E}^{n-m}}\mathrm{Tr}((E\otimes_J F)M_1\times(\bar{E}^\dagger\otimes_J F^\dagger)M_2)\mathbf{u}^{\mathrm{wt}(F)}e_{E,\bar{E}}, \quad (2.9)$$

where $\{e_{E,\bar{E}}\}$ are orthonormal basis vectors of a q^4-dimensional vector space. Here, $\mathrm{wt}(F)$ is an abstract weight function we discussed in the previous section and $\mathbf{u}$ can be an n tuple of variables, and $J\subseteq\{1,\ldots,n\}$ is a set of m qudits and locations. We write $\otimes_J$ denotes the tensor product of length m Pauli string E interlaced with Pauli string F of length $n-m$ at the positions marked in the set J. Later we will also use $\mathcal{E}^{n-m}[d]$ as the set of Pauli operators F on $n-m$ sites that have weight d.

To give a more concrete illustration of these objects consider the case of a rank-1 tensors ($m=1$), which we refer to as *vector* enumerators. For simplicity consider the usual (quantum) Hamming weight where $\mathbf{u}=z$ and $\mathrm{wt}(E)$ returns the number of nonidentity tensor factors in the Pauli operator E. For $J=\{j\}$ the vector enumerators along leg j read

$$\mathbf{A}^{(j)}(z;M_1,M_2)=\sum_{E,\bar{E}\in\mathcal{E}}\sum_{d=0}^{n}A_d^{(j)}(E,\bar{E};M_1,M_2)z^d e_{E,\bar{E}}, \quad (2.10)$$

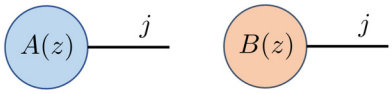

FIG. 2. Vector enumerators.

$$\mathbf{B}^{(j)}(z;M_1,M_2)=\sum_{E,\bar{E}\in\mathcal{E}}\sum_{d=0}^{n}B_d^{(j)}(E,\bar{E};M_1,M_2)z^d e_{E,\bar{E}}, \quad (2.11)$$

with coefficients (weights) here are defined as

$$A_d^{(j)}(E,\bar{E};M_1,M_2)=\sum_{F\in\mathcal{E}^{n-1}[d]}\mathrm{Tr}((E\otimes_j F)M_1)\,\mathrm{Tr}((\bar{E}^\dagger\otimes_j F^\dagger)M_2), \quad (2.12)$$

$$B_d^{(j)}(E,\bar{E};M_1,M_2)=\sum_{F\in\mathcal{E}^{n-1}[d]}\mathrm{Tr}((E\otimes_j F)M_1(\bar{E}^\dagger\otimes_j F^\dagger)M_2). \quad (2.13)$$

The $\mathcal{E}^{n-1}[d]$ here is the set of operators that have weight d on the $n-1$ qubits except the j th one, and $E\otimes_j F$ is a Pauli string that has E inserted on the j th position of the Pauli string:

$$E\otimes_j F=F_1\otimes F_2\otimes\ldots F_{j-1}\otimes E_j\otimes F_{j+1}\otimes\cdots\otimes F_n.$$

Formally, it is also convenient to express these coefficients in coordinates, once we have chosen a standard basis $\{\hat{\mathbf{e}}_j\}$. For example, one can denote

$$A_d^{(j)}(E,\bar{E};M_1,M_2)\to A_d^j, \quad (2.14)$$

$$B_d^{(j)}(E,\bar{E};M_1,M_2)\to B_d^j, \quad (2.15)$$

by identifying $j=0,\ldots,q^4$ where $E,\bar{E}$ each has q^2 distinct values. For simplicity, we abuse notation and use j as an open index that labels the dangling leg that comes from the j th qudit. The corresponding vector enumerator polynomials are $A^j(z;M_1,M_2),B^j(z;M_1,M_2)$, which we represent graphically as rank-1 tensors in Fig. 2.

In the same vein, the coefficients for a tensor enumerator of rank m may be written as

$$\sum_d A_d^{(J)}(E,E;M_1,M_2)z^d\to\sum_d A_d^{j_1\ldots j_m}z^d=A^{j_1,j_2,\ldots,j_m}(z), \quad (2.16)$$

$$\sum_d B_d^{(J)}(E,E;M_1,M_2)z^d\to\sum_d B_d^{j_1\ldots j_m}z^d=B^{j_1,j_2,\ldots,j_m}(z), \quad (2.17)$$

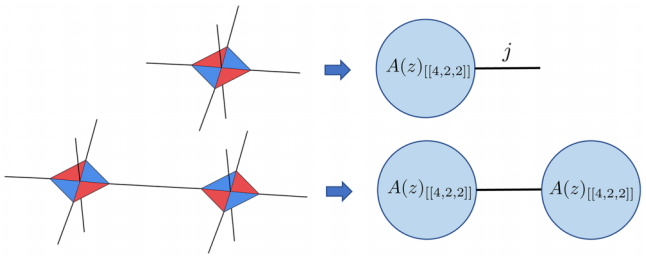

FIG. 3. WEP from tracing $[\![4,2,2]\!]$ codes.

where each tensor coefficient $A^{j_1,j_2,\ldots,j_m}(z)$, $B^{j_1,j_2,\ldots,j_m}(z)$ is a scalar enumerator. A graphical representation of $A^{j_1,j_2,\ldots,j_m}(z)$ is given below in Fig. 4 (top left).

In practice, it is often sufficient to consider reduced versions of these enumerators that only keep the diagonal terms with $E=\bar{E}$, which we represent using the same graphical form, but now with reduced bond dimension $j_\ell = 1,\ldots,q^2$. Such enumerators are known as the *reduced enumerators* and they are sufficient for studying Pauli errors in stabilizer codes. See Ref. [14] and Appendix B 3. In this work, we use the color blue to denote A-type enumerators and orange to denote B-type enumerators. We often drop the variable z or $\mathbf{u}$ to avoid clutter, but it should be understood that the tensor components of these objects are polynomials.

One can also easily define other tensor enumerators such as the double and complete enumerators by choosing different expressions for the abstract forms $\mathbf{u}$ and weight functions $\mathrm{wt}(E)$. An extension to the generalized abstract tensor enumerator is also possible. Details are found in Appendix B.

D. Tracing tensor enumerators

Let us define a trace operation $\wedge_{j,k}$ over the tensor enumerators, which connects any two legs j,k in the tensor network. Graphically, it is represented by a connected edge

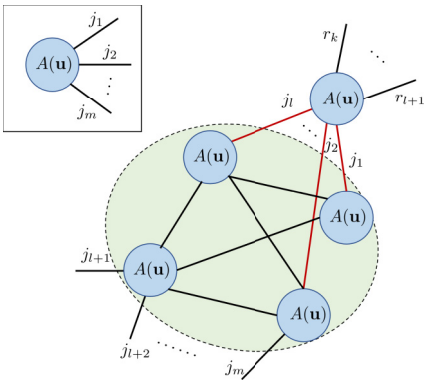

FIG. 4. Graphical representation of a type-A tensor enumerator (box). Tracing the type-A tensors as in Eq. (19). Green region can be seen as $A^{j_1,j_2,\ldots,j_l,\ldots,j_m}(\mathbf{u})$. Traced legs are red.

in the dual enumerator tensor network. Acting on the basis element $e_{E,\bar{E}}$ we define

$$\wedge_{j,k} e_{E,\bar{E}} = e_{E\setminus\{E_j,E_k\},\bar{E}\setminus\{\bar{E}_j,\bar{E}_k\}} \tag{2.18}$$

when $E_j = E_k^*$ and $\bar{E}_j = \bar{E}_k^*$ and zero otherwise.

Each contraction can be understood as tracing together two tensors. However, we can also view the two tensors as a single tensor enumerator (using the tensor product) then performing a self-trace, which is necessary and sufficient to build up any tensor network. Informally, the trace of the tensor enumerator is the tensor enumerator of the traced network, which is formally stated as the following.

Theorem 2. Suppose $j,k \in J \subseteq \{1,\ldots,m\}$. Then

$$\wedge_{jk}\mathbf{A}^{(J)}(\mathbf{u};M_1,M_2) = \mathbf{A}^{(J\setminus\{j,k\})}(\mathbf{u};\wedge_{j,k}M_1,\wedge_{j,k}M_2),$$

and similarly for $\mathbf{B}^{(J)}$.

Proof. See Theorem 7.1 of Ref. [14]. ■

Theorem 2 allows us to compute the weight enumerator of a contracted tensor network by contracting the tensor enumerators of each QL block. For example, to construct a scalar enumerator given the QL representation of an encoding map in Fig. 3, we first lay down its "shadow" that is the tensor enumerator for each $[\![4,2,2]\!]$ atomic code. Then we trace together these blocks following the same network connectivity.

The component form of contracting tensor enumerators can be expressed as the conventional sum over indices for a tensor trace. For reduced enumerators at $q=2$ this reads,

$$A^{j_{l+1},\ldots,j_m,r_{l+1},\ldots,r_k}(\mathbf{u}) = \sum_{j_1,j_2,\ldots,j_l} A^{j_1,j_2,\ldots,j_l,\ldots,j_m}(\mathbf{u})A^{j_1,j_2,\ldots,j_l,r_{l+1}\ldots r_k}(\mathbf{u}) \tag{2.19}$$

and similarly for $\mathbf{B}(\mathbf{u};M_1,M_2)$, where the only difference from a traditional tensor network is the variables $\mathbf{u}$ associated with the polynomial. One can connect these tensors sequentially; at each step an atomic code is glued to the (generically) bigger connected component, Fig. 4. For the full tensor enumerator, or when $q>2$, we need to take more care in raising and lowering the indices to recast them into the proper covariant and contravariant forms before summing over repeated indices.

While it is natural to use symbolic packages to implement this formalism, we will also elaborate in Appendix C how to implement these objects as the usual multilinear function without symbolic packages using conventional tensor-network methods.

III. APPLICATIONS OF WEIGHT ENUMERATORS

A. Code distance from enumerators

The genesis of quantum weight enumerators came from the case $M_1 = M_2 = \Pi$, the projection onto a stabilizer code, and $\mathbf{u}^{\mathrm{wt}(E)} = z^{\mathrm{wt}(E)}$. After an appropriate normalization, the enumerators $A^{\mathrm{norm}}(z) = A(z)/K^2$, $B^{\mathrm{norm}}(z) = B(z)/K$ encode the weight distributions of stabilizers (logical identities) and normalizers (all logical operators) of the code, respectively [4]. The normalized polynomials $A^{\mathrm{norm}}(z)$, $B^{\mathrm{norm}}(z)$ have $B_0 = A_0 = 1$. It follows that $B^{\mathrm{norm}}(z) - A^{\mathrm{norm}}(z)$ yields the weight distributions of nontrivial logical Pauli operators. Therefore, the smallest d for which $B_d \neq A_d$ is thus the (adversarial) code distance. This observation also generalizes to any quantum code [8]. Formally we capture this in the following result for later reference.

Theorem 3. Let $\mathcal{C}$ be a quantum code, $\Pi_{\mathcal{C}}$ be the projection onto its code subspace and

$$A(z; \Pi_{\mathcal{C}}, \Pi_{\mathcal{C}}) = \sum_d A_d z^d, \tag{3.1}$$

$$B(z; \Pi_{\mathcal{C}}, \Pi_{\mathcal{C}}) = \sum_d B_d z^d \tag{3.2}$$

be its weight enumerator polynomials properly normalized. Then

(a) $A_0 = B_0 = 1$,
(b) $B_d \geq A_d \geq 0$ for all d, and
(c) the code distance is $t+1$ where t is the largest integer for which $B_i = A_i$ for all $0 \leq i \leq t$.

A similar version holds for the refined enumerator, as shown by Ref. [9], from which one can determine the biased distances for the code (Theorem 6).

As one can read off the distances from the enumerators, our tensor-network method provides a straightforward way to compute and verify adversarial distances for all quantum codes whose QL description is known. This provides the first viable method to compute distances for a quantum code that need not be a stabilizer code.

B. Error detection

With weight enumerators in hand, we can easily obtain the probability for uncorrectable errors [8]. For any quantum code $\mathcal{C}$, let $\Pi_{\mathcal{C}}$ be the projector onto the code subspace, and write the orthogonal projector onto $\mathcal{C}^{\perp}$ as $\Pi_{\mathcal{C}}^{\perp}$. We say an error E *uncorrectable* if it cannot be detected, that is $\Pi_{\mathcal{C}} E \Pi_{\mathcal{C}} \propto \Pi_{\mathcal{C}}$, and is not proportional to the logical identity. Operationally, one performs a measurement with respect to $(\Pi_{\mathcal{C}}, \Pi_{\mathcal{C}}^{\perp})$. An error is detected if the result is contained in $\mathcal{C}^{\perp}$. For stabilizer codes, this corresponds to errors with trivial error syndrome that perform a nonidentity logical operation.

Consider depolarizing channel with unbiased noise, which acts identically on any single qubit with

$$\rho_j \to (1-3p)\rho_j + pX\rho_j X + pY\rho_j Y + pZ\rho_j Z, \tag{3.3}$$

where ρ_j is the reduced density matrix on site j. For stabilizer codes, it is easy to check that the probability of the random Pauli errors coinciding with a nontrivial logical operator is nothing but $p_{ne} = (B^{\mathrm{norm}} - A^{\mathrm{norm}})(z = p, w = 1-3p)$ because a Pauli error with weight d occurs with probability $p^d(1-3p)^{n-d}$. As in Theorem 3, we have taken the enumerators to be normalized such that $A_0 = B_0 = 1$. In general, Ref. [8] shows that the error probability for any code with $\dim \mathcal{C} = K$ is

$$p_{ne} = \frac{K}{(K+1)}\big(B^{\mathrm{norm}}(p, 1-3p) - A^{\mathrm{norm}}(p, 1-3p)\big). \tag{3.4}$$

Note the overall multiplicative factor compared to our initial estimation using stabilizer code because some logical errors takes the initial codeword to a nonorthogonal state, but only the orthogonal component is counted as nontrivial logical error in this construction.

We can extend the argument of Ref. [8] to more general error models. Suppose the error channel is given by

$$\rho_j \to \sum_{i=1}^{q^2} K_i \rho_j K_i^{\dagger}, \tag{3.5}$$

which acts identically across all physical qudits, then on the whole system, the errors act as

$$\mathcal{E}(\rho) = \sum_{\mathbf{i}} \mathcal{K}_{\mathbf{i}} \rho \mathcal{K}_{\mathbf{i}}^{\dagger}, \tag{3.6}$$

where

$$\mathcal{K}_{\mathbf{i}} = K_{i_1} \otimes K_{i_2} \otimes \cdots \otimes K_{i_n}, \tag{3.7}$$

and $\mathbf{i}$ is summed over all q^2-nary strings of length n. It is important to note that for each $\mathbf{i}$, the Kraus operator and its conjugate are the same, there are no cross terms.

Theorem 4. The nondetectable error probabilities of any error channel with the form of Eq. (25) is given by

$$p_{nd} = \frac{K}{K+1}\Big(\frac{1}{K}\sum_{\mathbf{i}} \mathrm{Tr}[\mathcal{K}_{\mathbf{i}}^{\dagger}\Pi\mathcal{K}_{\mathbf{i}}\Pi] - \frac{1}{K^2}\sum_{\mathbf{i}} \mathrm{Tr}[\mathcal{K}_{\mathbf{i}}^{\dagger}\Pi]\,\mathrm{Tr}[\mathcal{K}_{\mathbf{i}}\Pi]\Big) \tag{3.8}$$

for a quantum code with dimension K with projector Π.

Proof. See Appendix D. ■

For instance, in the depolarizing channel Eq. (22) each $\mathcal{K}_\mathbf{i}$ is simply a Pauli string $E \in \mathcal{E}^n$ weighted by $p^{\mathrm{wt}(E)}(1-3p)^{n-\mathrm{wt}(E)}$. Substituting we find that the two terms in Eq. (27) are simply the enumerator polynomials A and B evaluated at $z = p$ and $w = 1-3p$ as expected.

1. General error channels in the Pauli basis

For each K_a, its Pauli decomposition $K_a = \sum_E c_E^a E$ allows us to re-express the error probability in terms of the generalized weight enumerators in Sec. II B. In such cases, we can reorganize the sum over i by Pauli types. Again, let the noise model be single-qubit errors that are identical across all physical qubits such that

$$\rho \to \sum_i^{q^2} K_i \rho K_i^\dagger = \sum_{P,\bar{P}} k_{P\bar{P}} P \rho \bar{P}^\dagger. \tag{3.9}$$

Let us label each $P\bar{P}$ pair as G so that $|\{G\}| = q^4$ and so write $k_G = k_{P\bar{P}}$. For example, $\{G\} = \{II, IX, XI, IZ, ZI, XX, ZZ \ldots\}$ (all 16 arrangements) for $q = 2$.

Then let wt_G^n be a weight function

$$\mathrm{wt}_G^n(E \otimes F) = \sum_{i=1}^n \mathrm{wt}_G(E_i \otimes F_i^\dagger), \tag{3.10}$$

where

$$\mathrm{wt}_G(E_i \otimes F_i) = \begin{cases} 1 \text{ if } E_i \otimes F_i = G \\ 0 \text{ otherwise,} \end{cases} \tag{3.11}$$

and $E \otimes F = \bigotimes_i E_i \otimes F_i$. Thus wt_G^n counts the number of times $G = P \otimes \bar{P}$ appears in a string $E \otimes F$ where E, F each has length n. The relevant terms can then be expanded in this basis as

$$\begin{aligned} B(\{k_G\}; \Pi, \Pi) &= \sum_\mathbf{i} \mathrm{Tr}[\mathcal{K}_i \Pi \mathcal{K}_i^\dagger \Pi] \\ &= \sum_{E,F \in \mathcal{E}^n} \mathrm{Tr}[E \Pi F^\dagger \Pi] \prod_G k_G^{\mathrm{wt}_G^n(E \otimes F)}, \end{aligned} \tag{3.12}$$

$$\begin{aligned} A(\{k_G\}; \Pi, \Pi) &= \sum_\mathbf{i} \mathrm{Tr}[\mathcal{K}_i \Pi] \mathrm{Tr}[\mathcal{K}_i^\dagger \Pi] \\ &= \sum_{E,F \in \mathcal{E}^n} \mathrm{Tr}[E \Pi] \mathrm{Tr}[F^\dagger \Pi] \prod_G k_G^{\mathrm{wt}_G^n(E \otimes F)}. \end{aligned} \tag{3.13}$$

We can then distill a set of enumerators sufficient in describing the effect of all error channels

$$\bar{A}(\mathbf{u}_G; M_1, M_2) = \sum_{E,F \in \mathcal{E}^n} \mathrm{Tr}[E M_1] \mathrm{Tr}[F^\dagger M_2] \mathbf{u}^{\mathrm{wt}_G^n(E \otimes F)}, \tag{3.14}$$

$$\bar{B}(\mathbf{u}_G; M_1, M_2) = \sum_{E,F \in \mathcal{E}^n} \mathrm{Tr}[E M_1 F^\dagger M_2] \mathbf{u}_G^{\mathrm{wt}_G^n(E \otimes F)}, \tag{3.15}$$

where

$$\mathbf{u}_G^{\mathrm{wt}_G^n(E \otimes F)} = \underbrace{u_{II}^{\mathrm{wt}_{II}^n(E \otimes F)} u_{IP_1}^{\mathrm{wt}_{IP_1}^n(E \otimes F)} \cdots u_{P_q P_q}^{\mathrm{wt}_{P_q P_q}^n(E \otimes F)}}_{\text{all } q^4 \text{ terms}}.$$

We see this is nothing but a specific form of the generalized enumerator we introduced in Sec. II B. Note that we need only to compute the relevant enumerators once. The effects of different error models are now completely captured by the polynomials and can be evaluated by inserting the relevant values of c_G.

By substituting the proper expressions for Kraus operators, we are now in a position to rephrase all identical single qubit error channels in the form of weight enumerators. In practice, computing the generalized enumerator that accommodates arbitrary error channels can be rather expensive. Even for qubits, we would in general require 16 different variables in a polynomial. Fortunately for common channels, the computation simplifies and it is possible to express them with a much smaller set. As the Kraus representations are not unique it may be possible that some representations yield more succinct expressions than others. For pedagogical reasons, let us apply this to a few common error channels on qubits.

2. Biased Pauli errors

For a noise model where bit-flip (X) error and phase (Z) error can occur independently on physical qubits with probability p_x, p_z, respectively. The error channel is

$$\begin{aligned} \rho \to (1 - p_x - p_z + p_x p_z)\rho + (p_x - p_x p_z) X \rho X \\ + p_x p_z Y \rho Y + (p_z - p_x p_z) Z \rho Z. \end{aligned}$$

For stabilizer codes, the probability that the Pauli error coincides with a nontrivial logical operation is given by the normalized double weight enumerator of [9]

$$(D - D^\perp)(x, y, z, w), \tag{3.16}$$

evaluated at $x = 1 - p_x$, $y = p_x$, $z = 1 - p_z$ and $w = p_z$. Applying Theorem 4, we see that the actual noncorrectable logical error probability is given by Eq. (35) but again modified by multiplicative factor $K/(K+1)$ when taken into account the effect of nonorthogonal states.

Similarly, a channel where all Pauli errors have different independent error probabilities

$$\rho \to (1-p_x-p_y-p_z)\rho + p_x X\rho X + p_y Y\rho Y + p_z Z\rho Z$$

have noncorrectable error probability given by the complete enumerators,

$$p_{ne} = \frac{K}{K+1}\Big(F(p_x,p_y,p_z,1-p_x-p_y-p_z) - E(p_x,p_y,p_z,1-p_x-p_y-p_z)\Big). \tag{3.17}$$

For definitions of $D, D^{\perp}, E, F$, see Refs. [9,14] or Appendix A.

3. Coherent error

Pauli errors are in some sense classical; for a coherent quantum device, unitary errors are also relevant. Compared to Pauli errors, studies of the impact of coherent errors are less common [21–23] partly hampered by the computational costs. Nevertheless various methods exist. Here we examine a special case of single-qubit coherent error and express it in terms of weight-enumerator polynomials. Suppose we have single qubit and qudit coherent error applied identically to all physical qubits

$$\rho_i \to U_i \rho_i U_i^{\dagger} \tag{3.18}$$

acting on each qubit i, where each unitary can be decomposed as

$$U_i = aI_i + bX_i + cY_i + dZ_i. \tag{3.19}$$

The logical error probability is

$$\bar{p}_{nd} = \frac{K}{K+1}\Big(\frac{1}{K}\operatorname{Tr}[U^{\dagger}\Pi U\Pi] - \frac{1}{K^2}\operatorname{Tr}[U^{\dagger}\Pi]\operatorname{Tr}[U\Pi]\Big). \tag{3.20}$$

Expanding $U = \bigotimes_i U_i$ in the Pauli basis, we have $U^{\dagger} = \sum_E k_E^* E$ and $U = \sum_F k_F F$ where we sum E, F over all length n Pauli strings. As coefficients k_E, k_F depend only on the number of Paulis that appear in F

$$k_F = a^{n-w(F)} b^{w_x(F)} c^{w_y(F)} d^{w_z(F)}, \tag{3.21}$$

each term in the overall probability is

$$\begin{aligned}\operatorname{Tr}[U^{\dagger}\Pi U\Pi] &= \sum_{E,F\in\mathcal{E}^n} \operatorname{Tr}[E\Pi F\Pi] k_E^* k_F \\ &= \sum_{E,F\in\mathcal{E}^n} \operatorname{Tr}[E\Pi F\Pi] a^{n-w(F)} b^{w_x(F)} \\ &\quad \times c^{w_y(F)} d^{w_z(F)} \bar{a}^{n-w(E)} \bar{b}^{w_x(E)} \bar{c}^{w_y(E)} \bar{d}^{w_z(E)}\end{aligned} \tag{3.22}$$

$$\begin{aligned}\operatorname{Tr}[U^{\dagger}\Pi]\operatorname{Tr}[U\Pi] &= \sum_{E,F\in\mathcal{E}^n} \operatorname{Tr}[E\Pi]\operatorname{Tr}[F\Pi] k_E^* k_F \\ &= \sum_{E,F\in\mathcal{E}^n} \operatorname{Tr}[E\Pi]\operatorname{Tr}[F\Pi] a^{n-w(F)} b^{w_x(F)} \\ &\quad \times c^{w_y(F)} d^{w_z(F)} \bar{a}^{n-w(E)} \bar{b}^{w_x(E)} \bar{c}^{w_y(E)} \bar{d}^{w_z(E)}.\end{aligned} \tag{3.23}$$

These are nothing but the generalized versions of the complete weight enumerators

$$A(\mathbf{u}_1,\mathbf{u}_2;M_1,M_2) = \sum_{E,F\in\mathcal{E}^n} \operatorname{Tr}[E^{\dagger}M_1]\operatorname{Tr}[FM_2]\mathbf{u}_1^{\mathrm{wt}(F)}\mathbf{u}_2^{\mathrm{wt}(E)}, \tag{3.24}$$

$$B(\mathbf{u}_1,\mathbf{u}_2;M_1,M_2) = \sum_{E,F\in\mathcal{E}^n} \operatorname{Tr}[E^{\dagger}M_1FM_2]\mathbf{u}_1^{\mathrm{wt}(F)}\mathbf{u}_2^{\mathrm{wt}(E)} \tag{3.25}$$

evaluated at $M_1 = M_2 = \Pi$, $w_1 = a, x_1 = b, y_1 = c, z_1 = d$ and w_2, x_2, y_2, z_2 at their complex conjugates. To simplify the notation, we absorbed each 4 tuple of variables into abstract variables $\mathbf{u}_i$ and weight functions $\mathrm{wt}(\cdot)$ such that

$$\mathbf{u}_i^{\mathrm{wt}(E)} = x^{\mathrm{wt}_x(E)} y^{\mathrm{wt}_y(E)} z^{\mathrm{wt}_z(E)} w^{n-\mathrm{wt}(E)}. \tag{3.26}$$

4. Amplitude damping and dephasing channels

Amplitude damping channel is relevant for superconducting qubits. Its has a Kraus representation with operators

$$\begin{aligned}K_0 &= \begin{pmatrix}1 & 0\\ 0 & \sqrt{1-\gamma}\end{pmatrix} \\ &= \frac{1}{2}(1+\sqrt{1-\gamma})I + \frac{1}{2}(1-\sqrt{1-\gamma})Z, \\ K_1 &= \begin{pmatrix}0 & \sqrt{\gamma}\\ 0 & 0\end{pmatrix} = \frac{\sqrt{\gamma}}{2}X + \frac{i\sqrt{\gamma}}{2}Y\end{aligned} \tag{3.27}$$

In this case, we only need to keep eight distinct variables $\{u_{II}, u_{IZ}, u_{ZI}, u_{ZZ}, u_{XX}, u_{XY}, u_{YX}, u_{YY}\}$ as the remaining coefficients k_G are 0. In fact, the nonzero coefficients further satisfy $k_{IZ} = k_{ZI} = \lambda_I\lambda_Z, k_{PP} = |\lambda_P|^2, k_{XY} = \bar{k}_{YX} = \lambda_X\bar{\lambda}_Y$ where λ_P are the coefficients in the Pauli expansion of the Kraus operators. Therefore, the end polynomial would only require four independent variables $\{\lambda_I, \lambda_X, \lambda_Y, \lambda_Z\}$. In other words, when summing the polynomial in practice, we only sum over the qudit strings of local dimension 4 where the coefficients for G are nonvanishing. Furthermore, one

can rewrite the nonzero coefficients as

$$\prod_G k_G^{\mathrm{wt}_G(E\otimes F)} = \prod_P \lambda_P^{\mathrm{wt}_P(E)} \prod_{\bar{P}} \bar{\lambda}_{\bar{P}}^{\mathrm{wt}_{\bar{P}}(F)}, \tag{3.28}$$

which depends on four parameters and is no more complicated than the complete enumerator.

For a dephasing channel, K_0 remains the same while

$$K_1 = \sqrt{\gamma}\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} = \frac{\sqrt{\gamma}}{2}(I - Z). \tag{3.29}$$

Expanding and simplifying, we find that it only depends on two nonzero coefficients $c_{II} = (1 + \sqrt{1-\gamma})/2$ and $c_{ZZ} = (1 - \sqrt{1-\gamma})/2$. Thus this is even easier than computing the original weight enumerator! Furthermore, instead of summing over the full Pauli group, we only need to sum over $E \in \mathcal{Z}^n$ where $\mathcal{Z}$ is the set of Pauli strings that only contains I or Z.

C. Effective distance

While adversarial distance is a useful measure of the goodness of a code, it is also informative to devise more refined measures like effective distances [24,25] that serve as useful benchmarks of code performance with respect to different error profiles. For example, recall that Ref. [25] defines an effective distance

$$d' = \mathcal{N}^{-1} \log(p_0(1-p)^{-n}) \tag{3.30}$$

for codes under depolarizing channel, where $p = p_X + p_Y + p_Z$ and $\mathcal{N}$ is some normalization factor that depends on the physical error probabilities. In the original definition, p_0 is the probability where the Pauli noise implements the most likely nontrivial logical operator. Using enumerators, we can also produce more precise effective distances under depolarizing noise, where p_0 is replaced by the probability p_{ne} where Pauli noise implements *any* nontrivial logical operator. Similar measures have been used to quantify effective code performance [24,26]. For example, one can define another effective distance for some c_1, c_2

$$d_{\mathrm{eff}} = c_1 \log(p_L) + c_2 \tag{3.31}$$

such that d_{eff} is higher for lower error rate p_L.

Similar to Ref. [26], we also use the normalized logical error probability

$$p_L^{\mathrm{norm}} = p_L/p_{s=0} \tag{3.32}$$

as a measure of code performance throughout this work. Here $p_{s=0}$ is the probability of error nondetection and better protection corresponds to a smaller normalized error rate. This is not a distance measure and it corresponds to the probability of uncorrectable error where the "error-correction" protocol simply discards the quantum state upon detecting an error.

D. Subsystem codes and mixed enumerator

The above applications are general and can be used for any quantum code. Let us now focus on a few more applications that are most closely tied to stabilizer codes and subsystem codes.

Mixed enumerators are made by tracing together tensor enumerator of both $\mathbf{A}(\mathbf{u})$ and $\mathbf{B}(\mathbf{u})$ types.

Proposition 1. Let $M(\mathbf{u})$ be a mixed enumerator polynomial obtained from tracing tensor enumerators of A and B types. MacWilliams transform on $M(\mathbf{u})$ produces a dual polynomial $M^{\perp}(\mathbf{u})$, which, up to normalization, can be built from the same tensor network where we exchange the A- and B-type tensors.

Proof. The MacWilliams transform commutes with trace as long as the generalized Wigner transform is its own self-inverse up to a constant multiple. This is clearly the case when the tensor enumerators are diagonal, when the generalized Wigner transform reduces to regular Wigner transform. The same must also hold true for the generalized transform, as the MacWilliams transform commutes with trace when the tensor enumerators are not mixed. ■

A key application is finding the distance of subsystem codes where we need to enumerate all gauge-equivalent representations of the logical operators. It is convenient to think of the subsystem code as a stabilizer code encoding multiple logical qubits where some of them are demoted to gauge qubits. To obtain its distance, we first enumerate all logical operators, which is given by $B(\mathbf{u})$ of the stabilizer code. This can again be obtained by $A(\mathbf{u})$ and applying the MacWilliams identity. We then need to enumerate all gauge-equivalent logical identities of this subsystem code $I(\mathbf{u})$. Technical details in obtaining $I(\mathbf{u})$ can depend on the specific tensor network in question. However, it is rather straightforward if all logical legs in the network are independent, i.e., the encoding map defined by the QL tensor network has trivial kernel. For example, this is the case for holographic code, but not for the Bacon-Shor code tensor network.

For encoding tensor networks that have trivial kernel, we can divide the input legs, which we call logical legs in Ref. [13], into two categories: (i) the ones where operator pushing produce logical operators, which we now call logical legs, and (ii) the ones that alter the state of gauge qubits, which we now call gauge legs. Let us first assume that each tensor has only one such an input leg that is either logical or gauge, which is the case for the holographic code. To enumerate the logical identity, we construct a mixed enumerator—for each tensor in the QL network whose input leg is logical, we contract the tensor enumerator $\mathbf{A}(\mathbf{u})$ of the local atomic code (e.g., the $[\![5,1,3]\!]$ code in HaPPY) on the corresponding vertex in the enumerator tensor network. If the tensor in the QL network has a gauge leg,

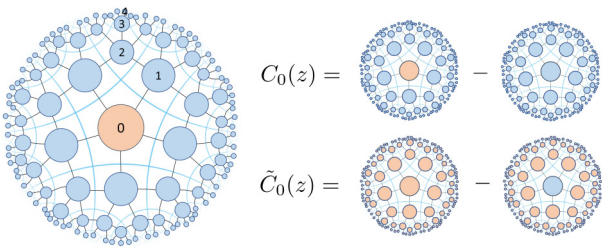

FIG. 5. Left: tensor network for a mixed enumerator of the holographic HaPPY pentagon code, where blue indicates insertion of $\mathbf{A}(z)$ and orange $\mathbf{B}(z)$. Right: different tensor networks compute the different distribution of logical operators. The same exercise can be repeated for logical qubits at different distances from the center (labeled 0,1,2,3,4).

then we contract the $\mathbf{B}(\mathbf{u})$ tensor enumerator of the local atomic code (Fig. 5). The resulting tensor network enumerates the weights of all $g\bar{I}$ for $g \in \mathcal{G}$. Then the difference $\tilde{C}(\mathbf{u}) = B(\mathbf{u}) - I(\mathbf{u})$ between these enumerators only contains the weights of nonidentity logical operators, which informs us about the distance. This is also known as the *word distance* [27,28].

Similarly, if we want to compute the distance of a logical qubit in the stabilizer code (i.e., all logical, no gauge qubits), then we only enumerate the stabilizer equivalent logical operations that act nontrivially on that qubit. For this we insert $\mathbf{B}(\mathbf{u})$ on the vertex containing the logical qubit of interest and $\mathbf{A}(\mathbf{u})$ everywhere else. This enumerator now only counts the stabilizer equivalent of that particular logical qubit, instead of all logical qubits, like $B(\mathbf{u})$. This can be quite relevant in the holographic code, where the central bulk qubit can have a distance that scales with system size, whereas the ones on the peripheral have constant distance [27].

For instance, in the holographic HaPPY code, Fig. 5, one can treat the system as a stabilizer code. Then the *stabilizer distance* can be determined by counting all nonidentity logical operators associated with a particular bulk qubit. In the figure we choose the logical qubit living on the central tile. The stabilizer distance is then the minimum power of z in $C_0(z)$ for which the coefficient is nonzero. If we treat it as a subsystem code, then the distance should instead be counted by including the logical operators of other bulk qubits as gauge qubits using the enumerator $\tilde{C}_0(z)$.

If each tile has multiple input legs, some of which are gauge and others logical, we then need to make slight modifications to the tensor enumerators used in the above prescription. For a word distance computation, we send $\mathbf{A}(\mathbf{u}) \to \mathbf{A}'(\mathbf{u})$ such that $\mathbf{A}'(\mathbf{u})$ enumerates all logical identity operators of the local atomic code, e.g., the $[\![4,2,2]\!]$ atomic code on even columns of a 2D Bacon-Shor code tensor network (Fig. 20). In other words, we enumerate all elements of the non-Abelian gauge group $\mathcal{G}$. For Pauli operators, this modification is rather straightforward as we simply count the number of operators that act as identity on the logical legs. More precisely, let

$$\Pi' = \frac{1}{|\mathcal{G}|}\sum_{g\in\mathcal{G}} g \tag{3.33}$$

and prepare $\mathbf{A}'(\mathbf{u}) = \mathbf{A}^{(J)}(\mathbf{u}; M_1 = \Pi', M_2 = \Pi')$ as a reduced tensor enumerator.

For stabilizer distance computations, we send $\mathbf{B}(\mathbf{u}) \to \mathbf{B}'(\mathbf{u})$, the latter of which enumerates the number of logical operators that act as the identity on gauge qubits. This can be prepared by a similar reduced tensor enumerator such that $\mathbf{B}'(\mathbf{u}) = \mathbf{A}^{(J)}(\mathbf{u}; M_1 = \Pi'', M_2 = \Pi'')$ where

$$\Pi'' \propto \sum_{g\in\mathcal{G}'} g \tag{3.34}$$

and $\mathcal{G}'$ is generated by the center of $\mathcal{G}$ and logical operators $\{\mathcal{L}_{\text{logical}} \otimes I_{\text{gauge}}\}$ that act as identity on the gauge qubits. In other words, we construct a new gauge group $\mathcal{G}'$ where we swapped the roles of the gauge and logical qubits in the original code defined by $\mathcal{G}$.

Finally, for a tensor network whose encoding map has a nontrivial kernel, i.e., the logical legs are interdependent, one should take extra care in applying the above recipe for building a useful mixed enumerator. For instance, in the Bacon-Shor code tensor network (Fig. 20), multiple input legs are interdependent and several of them correspond to the same logical or gauge degree of freedom. One then needs to make sure that the type of the legs (gauge vs logical) is being tracked consistently across different tensors when contracting the tensor network.

Instead of the mixed enumerators introduced above, we can also directly use tensor enumerators to study subsystem codes. This is similar to the approach by Refs. [14,20]. The recipe for building a relevant tensor enumerator of the code is quite similar. For each tensor that contains the logical leg and qubit, we put down a tensor enumerator $\mathbf{B}(z)$ of the *encoding tensor* at that node (e.g., the tensor of the $[\![6,0,4]\!]$ state in the HaPPY code) in the tensor network, except now that we keep the logical index open in addition to the contracted legs of the tensor. The components of the resulting tensor enumerator $\mathbf{B}(z)$ now contains the weight distribution for each logical Pauli operator. This allows us to read off the distances for each logical operator, after subtracting off the part that enumerates the logical identity by fixing certain tensor indices to 0. This can be performed efficiently if the number of open logical legs is not too many although the number of gauge qubits can still be high [29].

E. Higher genus enumerator

There is a well understood link between multiple weight enumerators of classical codes and modular forms in number theory, wherein the degree of the modular form is

the number of codewords whose weight is being enumerated [30,31]. Consequently, that number is now typically referred to as the "genus" of the enumerator. Just as in the classical case, we can extend this to higher genus quantum weight enumerators by introducing weight functions that count the number of factors where tuples of error operators realize specific error patterns. We can also study subsystem codes using genus two enumerators.

For concreteness, consider genus $g = 2$. We introduce q^4 variables $\mathbf{u} = (u_{G_1,G_2} : G_1, G_2 \in \mathcal{E})$, and weight function $\mathrm{wt} : \mathcal{E}^n \to \mathbb{Z}^{q^4}$ that counts factors

$$\mathrm{wt}(E,F) = \#\{i : E_i = G_1,\ F_i = G_2,\ G_1, G_2 \in \mathcal{E}\}. \tag{3.35}$$

The genus-2 weight enumerators of Hermitian operators M_1, M_2 on $\mathfrak{H} \otimes \mathfrak{H}$ are

$$A^{(2)}(\mathbf{u}; M_1, M_2) = \sum_{E,F\in\mathcal{E}^n} \mathrm{Tr}((E\otimes F)M_1)\,\mathrm{Tr}((E\otimes F)^\dagger M_2)\mathbf{u}^{\mathrm{wt}(E,F)} \tag{3.36}$$

$$B^{(2)}(\mathbf{u}; M_1, M_2) = \sum_{E,F\in\mathcal{E}^n} \mathrm{Tr}((E\otimes F)M_1(E\otimes F)^\dagger M_2)\mathbf{u}^{\mathrm{wt}(E,F)}. \tag{3.37}$$

Notice the coefficients of these enumerators are just what would use for a systems with $2n$ factors. The interesting addition is the additional variables to track correlations in the weights of E and F. Indeed if we were to ignore these correlations and evaluate $u_{G_1,G_2} = u_{G_1}u_{G_2}$ then we recover the ordinary enumerators:

$$\begin{aligned} A^{(2)}(\{u_{G_1}u_{G_2}\}; M_1\otimes M_1', M_2\otimes M_2') &= A(\{u_G\}; M_1, M_2)\cdot A(\{u_G\}; M_1', M_2'), \\ &= A(\{u_G\}; M_1\otimes M_1', M_2\otimes M_2'), \end{aligned} \tag{3.38}$$

and similarly for $B^{(2)}$.

To capture new information in the higher genus enumerators, we evaluate their variables in interesting ways. For example, consider the case where $M_1 = M_2 = \Pi_1 \otimes \Pi_2$ where Π_1 and Π_2 are projections that need not commute. Evaluating

$$u_{G_1,G_2} = \begin{cases} u_{G_1} & \text{if } G_1 = G_2 \\ 0 & \text{if } G_1 \neq G_2, \end{cases} \tag{3.39}$$

we have

$$\mathbf{u}^{\mathrm{wt}(E,F)} = \prod_{G_1,G_2} u_{G_1,G_2}^{\mathrm{wt}_{G_1,G_2}(E,F)} = \begin{cases} \prod_G u_G^{\mathrm{wt}_G(E)} & \text{if } E = F \\ 0 & \text{if } E \neq F. \end{cases} \tag{3.40}$$

Thus

$$A^{(2)}(\mathbf{u}; \Pi_1\otimes\Pi_2) = \sum_{E\in\mathcal{E}^n} \mathrm{Tr}(E\Pi_1)^2\,\mathrm{Tr}(E\Pi_2)^2\mathbf{u}^{\mathrm{wt}(E)}, \tag{3.41}$$

$$B^{(2)}(\mathbf{u}; \Pi_1\otimes\Pi_2) = \sum_{E\in\mathcal{E}^n} \mathrm{Tr}(E\Pi_1 E\Pi_1) \times \mathrm{Tr}(E\Pi_2 E\Pi_2)\mathbf{u}^{\mathrm{wt}(E)}. \tag{3.42}$$

In particular, consider a subsystem code whose gauge group decomposes as $\mathcal{G} = \mathcal{G}_1 \cup \mathcal{G}_2$ where each of $\mathcal{G}_1$ and $\mathcal{G}_2$ are maximal Abelian subgroups and $\mathcal{C}(\mathcal{G}) = \mathcal{G}_1 \cap \mathcal{G}_2$. This is the case for generalized Bacon-Shor codes [32] where $\mathcal{G}_1$ consists of the X-type generators of $\mathcal{G}$ and the row operators, while $\mathcal{G}_2$ is the Z-type generators and the column operators [33]. Each of $\mathcal{G}_1$ and $\mathcal{G}_2$ could be considered a stabilizer in its own right, however the weight enumerators of these have little to do with subsystem code of $\mathcal{G}$.

Nonetheless, consider them as stabilizers of codes and write the projections onto their code subspaces as $\Pi_1 = 1/2^{n-k_1}\sum_{S\in\mathcal{G}_1} S$ and $\Pi_2 = 1/2^{n-k_2}\sum_{S\in\mathcal{G}_2} S$ where k_1, k_2 are the dimensions of these codes. Then

$$\mathrm{Tr}(E\Pi_1)^2 = \begin{cases} 4^{k_1} & \text{if } E \in \mathcal{G}_1 \\ 0 & \text{otherwise,} \end{cases} \tag{3.43}$$

and similarly for $\mathrm{Tr}(E\Pi_2)^2$, and therefore

$$\begin{aligned} A^{(2)}(\mathbf{u}; \Pi_1\otimes\Pi_2) &= 4^{k_1+k_2}\sum_{E\in\mathcal{G}_1\cap\mathcal{G}_2} \mathbf{u}^{\mathrm{wt}(E)} \\ &= 4^{k_1+k_2}\sum_{d=0}^{n} \#(\mathcal{E}^n[d]\cap\mathcal{C}(\mathcal{G}))w^{n-d}z^d \end{aligned} \tag{3.44}$$

is the enumerator of the stabilizer of the subsystem code of $\mathcal{G}$. Also

$$\mathrm{Tr}(E\Pi_1 E\Pi_1) = \begin{cases} 2^{k_1} & \text{if } E \in \mathcal{N}(\mathcal{G}_1) \\ 0 & \text{otherwise,} \end{cases} \tag{3.45}$$

and similarly for $\mathrm{Tr}(E\Pi_2 E\Pi_2)$. Hence

$$\begin{aligned} B^{(2)}(\mathbf{u}; \Pi_1\otimes\Pi_2) &= 2^{k_1+k_2}\sum_{E\in\mathcal{N}(\mathcal{G}_1)\cap\mathcal{N}(\mathcal{G}_2)} \mathbf{u}^{\mathrm{wt}(E)} \\ &= 4^{k_1+k_2}\sum_{d=0}^{n} \#(\mathcal{E}^n[d]\cap\mathcal{N}(\mathcal{G}))w^{n-d}z^d \end{aligned} \tag{3.46}$$

is the enumerator for the logical operators of the subsystem code.

Notice that up to taking a logarithm and a constant offset, Eq. (61) is also the stabilizer Renyi-2 entropy when $q = 2$, $\mathbf{u}^{\mathrm{wt}(E)} \to$ constant for $\Pi = |\psi\rangle\langle\psi|$. The stabilizer Renyi entropy is a computable measure of non-stabilizerness.

Definition 1. The α-stabilizer Renyi entropy [34] is given by

$$M_\alpha(|\psi\rangle) = (1-\alpha)^{-1} \log \sum_{E\in\mathcal{P}^n} \Xi_E^\alpha(|\psi\rangle) - n\log 2 \quad (3.47)$$

where

$$\Xi_E(|\psi\rangle) = \frac{1}{2^n}\langle\psi|E|\psi\rangle^2 = \frac{1}{2^n}\mathrm{Tr}[E\Pi]^2. \quad (3.48)$$

We see that $\Xi_E(|\psi\rangle)^\alpha$ is precisely the coefficient of the type A genus-α weight enumerator Eq. (59).

As a corollary, because the higher genus enumerators can be computed using the tensor-network method, this provides a method for computing the stabilizer Renyi entropy also. While there are existing methods based on matrix product state [35,36] that computes stabilizer Renyi entropy, the enumerator tensor network provides an alternative and more general method for computing the magic of a quantum many-body system.

Physically, this also establishes a new connection between weight enumerator and quantum many-body magic, in addition to the entanglement angle that has been explored in the form of sector length.

F. Coset enumerator and errors with nontrivial syndrome

Until now, we have been working with particular instances of weight-enumerator polynomials, that is, $M_1 = M_2 = \Pi$ or related operators. For stabilizer codes, they recover the weight distributions of error operators that have trivial syndrome. However, for the purpose of decoding, it is also useful to learn the probability $P(E|s)$ for any error syndrome s.

Let E be a Pauli error that gives syndrome $\sigma(E) = s$. We consider the probability $P(E\bar{L})$ of errors that are stabilizer equivalent to $E\bar{L}$, where $\bar{L}$ is any logical operator. If we have this distribution, then we can construct a maximum-likelihood decoder by undoing the $E\bar{L}$ with the maximal probability of $P(E\bar{L})$ given syndrome s. Similarly, one could apply a Bayesian decoder where $E\bar{L}$ is applied with the probability $p(E\bar{L}|\sigma(E))$ for error correction.

Definition 2. A coset weight enumerator for a stabilizer code is given by $A^s(\mathbf{u}; E_s, \Pi) = A(\mathbf{u}; M_1, M_2)$ where $M_1 = M_2^\dagger = E_s\Pi$ for some (Pauli) operator E_s with syndrome s. Its "dual" enumerator is $B^s(\mathbf{u}; E_s, \Pi) = B(\mathbf{u}; M_1, M_2)$ where $M_1 = \Pi$, $M_2 = E_s\Pi E_s^\dagger$. Their tensorial versions are similarly defined with M_1, M_2 taking on these specific values. The same definition applies for the generalized enumerators $\bar{A}^s, \bar{B}^s$.

Note that M_1, M_2 here are no longer hermitian for A and the operators used for A, B are different. As a result, the "dual" enumerator is very different from its usual form. We do not use or prove a MacWilliams identity in this work, though it may be interesting to see if an analogous relation exists. More generally, the coset enumerators can be defined for E_s that are not Pauli operators so long as $\Pi_s = E_s\Pi E_s^\dagger$ projects onto the error subspace. As the definition of such subspaces and syndromes can be highly code dependent, here we focus on the instance where E_s are Paulis.

Proposition 2. Up to an overall normalization, the coefficients of the coset enumerator $A^s(\mathbf{u}; E_s, \Pi)$ counts the number of coset elements in $E_s\mathcal{S}$ while $B^s(\mathbf{u}; E_s, \Pi)$ enumerates the number of elements $E_s\mathcal{N}(\mathcal{S})$.

Proof. Let

$$\Pi_s = \frac{1}{|\mathcal{S}|}\sum_{D\in E_s\mathcal{S}} D \quad (3.49)$$

then for any $E \in \mathcal{E}^n$

$$\mathrm{Tr}[E\Pi_s]\,\mathrm{Tr}[E^\dagger\Pi_s^\dagger] = |\,\mathrm{Tr}[E\Pi_s]|^2 = q^{2n}/|\mathcal{S}|^2 \quad (3.50)$$

if $E \in E_s\mathcal{S}$ and zero otherwise. Hence, up to a constant normalization factor, the coefficient of the coset enumerator counts the number of coset elements of a particular weight. As we do not track signs in the distribution, no generality is lost by choosing the left vs right coset.

The B-type enumerators have coefficients

$$\mathrm{Tr}[E\Pi E^\dagger\Pi_s] = \mathrm{Tr}[\Pi_E\Pi_s], \quad (3.51)$$

where $\Pi_s = E_s\Pi E_s^\dagger$, $\Pi_E = E\Pi E^\dagger$. As these projectors are orthogonal for different syndromes in stabilizer codes, this coefficient is only nontrivial when $E \in E_s\mathcal{N}(\mathcal{S})$, i.e., when E is any logical error with syndrome s. Therefore, up to normalization, we again obtain an enumerator that captures the weight distribution of $E_s\mathcal{N}(\mathcal{S})$. ■

Practically, the process of preparing this enumerator using tensor network is the same as before except we modify the values of M_1 and M_2. First we identify the physical qubits on which E_s has support. Suppose E_s acts on a particular atomic code nontrivially with E_s^T, then we prepare the A-type tensor coset enumerator of this code with $M_1 = M_2^\dagger = E_s^T\Pi^T$ where Π^T is the projection onto the code subspace of the local QL. Such a tensor enumerator counts elements in the coset $E_s^T\mathcal{S}^T$. We then repeat this for all such

tensors. For ones that E_s does not have support, we compute their tensor enumerator with $M_1 = M_2 = \Pi^T$ as usual. Then we contract these tensor enumerators in the same way as we did for building $A(\mathbf{u}; M_1, M_2)$, e.g., Fig. 10. The resulting enumerator polynomial is the desired $A_s(\mathbf{u}; E_s\Pi)$. Also note that M_1, M_2 take on a special form that satisfy Proposition 3, hence we can compute it more efficiently using a tensor network with reduced bond dimension, much akin to its weight-enumerator counterparts.

With these weight distributions, it is obvious that we can then compute $P(E|s)$. For example, suppose we are given the coset enumerator $A^s(z, w; E_s, \Pi)$ for a code space defined by Π, then under symmetric depolarizing channel with physical error rate p,

$$p_s = B^s(z = p, w = 1 - 3p)/K \tag{3.52}$$

is the probability of returning an error syndrome s with noiseless checks and $A^s(z = p, w = 1 - 3p)/K^2$ is the probability of errors that are stabilizer equivalent to E_s. Indeed, this also extends trivially to double and complete enumerators by evaluating the polynomial at the respective parameters we used for the trivial syndrome examples in Sec. III B.

In fact, such kinds of error probabilities generalize to any error channel. Similar to the nondetectable errors we have analyzed in the previous section, it is possible to compute $p(\bar{L}|s)$ using generalized enumerators as long as we replace M_1, M_2 by the appropriate values used in the coset enumerators.

Theorem 5. Consider a stabilizer code where Π is the projection onto its code subspace of dimension K and let E_s be an error operator with syndrome s. Let the error channel be given by the Kraus form $\mathcal{E}(\cdot) = \sum_{\mathbf{i}} \mathcal{K}_{\mathbf{i}} \cdot \mathcal{K}_{\mathbf{i}}^\dagger$. Then

$$p(E_s\mathcal{S} \cap s) = \frac{1}{K(K+1)}\Big(\sum_{\mathbf{i}} \mathrm{Tr}[\mathcal{K}_{\mathbf{i}}\Pi\mathcal{K}_{\mathbf{i}}^\dagger\Pi_s] + \sum_{\mathbf{i}} \mathrm{Tr}[\mathcal{K}_{\mathbf{i}}\Pi E_s^\dagger]\,\mathrm{Tr}[\mathcal{K}_{\mathbf{i}}^\dagger E_s\Pi]\Big) \tag{3.53}$$

$$p_s = \frac{1}{K}\sum_{\mathbf{i}} \mathrm{Tr}[\Pi\mathcal{K}_{\mathbf{i}}^\dagger\Pi_s\mathcal{K}_{\mathbf{i}}], \tag{3.54}$$

where $\Pi_s = E_s\Pi E_s^\dagger$.

Note that E_s need not be a Pauli operator but can take on a general form $P_s\bar{L}$ where P_s is a Pauli error with syndrome s and $\bar{L}$ is any unitary logical operation. For proof and generalization where the logical error is a general quantum channel, see Appendix D 3. We see that these terms in the expression share a great deal of similarities with Theorem 4, which computes the logical error probability for trivial syndromes. Indeed, by expanding the Kraus operators in the Pauli basis, we see that these expressions can again be written as generalized weight enumerators Eq. (33) that we used to compute the uncorrectable error rates. To obtain these error probabilities, we follow the identical recipe by decomposing the Kraus operators in the Pauli basis $\sum_i K_i \cdot K_i^\dagger = \sum_{P\bar{P}} k_{P\bar{P}} P \cdot \bar{P}$ and evaluate $\mathbf{u}(k_{P\bar{P}})$ at the appropriate values based on that decomposition [c.f. Eq. (28)]. Finally, we set $M_1 = \Pi, M_2 = E_s\Pi E_s^\dagger$ for the B-type enumerator and $M_1 = \Pi E_s^\dagger, M_2 = M_1^\dagger$ for the A type.

Formally,

$$p(E_s\mathcal{S} \cap s) = \frac{1}{K(K+1)}\Big(B(\mathbf{k}; \Pi, \Pi_s) + A(\mathbf{k}; \Pi E_s^\dagger, E_s\Pi)\Big) \tag{3.55}$$

$$p_s = \frac{1}{K}B(\mathbf{k}; \Pi, \Pi_s), \tag{3.56}$$

where $\mathbf{k} = \{k_{P\bar{P}}\}$ are the coefficients from the Pauli expansion. With these coset enumerators in hand, we are now ready to discuss optimal decoders for general noise channels.

G. Decoders from weight enumerators

We see that one can express the probability

$$p(E_S\mathcal{S}|s) = p(E_s\mathcal{S} \cap s)/p_s \tag{3.57}$$

entirely in terms of weight enumerators. Suppose $E_s = P_s\bar{L}$ where P_s is any Pauli error with syndrome s, which can be obtained by solving a set of $n - k$ linear equations, and $\bar{L}$ is a logical operator, then the set of probabilities $\mathcal{P}_s = \{p(P_s\bar{L}\mathcal{S}|s)\}$, as $\bar{L}$ runs over logical operators, is sufficient for us to perform error correction. It is customary to generate the set $\mathcal{P}_s$ for the set of $\bar{L}$ that are logical Pauli operations as they form an operator basis for the code subalgebra and are thus sufficient to generate the conditional probabilities for any unitary logical operators.

1. Maximum-likelihood and Bayesian decoders

It is straightforward to implement a maximum-likelihood decoder where we identify the logical operator $\bar{L}_m$ for which $p(P_s\bar{L}_m\mathcal{S}|s) = \max \mathcal{P}_s$. Then error correction is performed by acting $\bar{L}$ and P_s following the syndrome measurements. In this case, it is sufficient to compute just the A enumerator because the B enumerators are independent of $\bar{L}$ and only add to an overall normalization that not impact our choice of the maximum element. When multiple global maxima exist, then we choose one at random.

One can also correct errors based on the probability distribution of $p(P_s\bar{L}\mathcal{S}|s)$ where we act on the state

using operator $P_s\bar{L}$ with the self-same probability. We call this the Bayesian decoder. As we require that $\sum_{\bar{L}} p(P_s\bar{L}\mathcal{S}|s) = 1$ when summing over all Paulis, it is again sufficient to only compute the type-A enumerator as the constant from B can be obtained by solving the normalization condition.

For each syndrome s, the complexity in implementing these decoders is therefore the complexity $\mathcal{C}(A^s)$ of computing A^s from tensor contractions. For some codes this can be performed efficiently, which we further elaborate in Sec. V. Nevertheless, even if each such contraction is efficient, we would still have to compute q^{2k} number of enumerators as there are q^{2k} number of distinct logical Pauli operators. Therefore, the overall complexity estimate for such a decoder is $O(\mathcal{C}(A^s)q^{2k})$.

2. Marginals

For a code where k is large, building the above maximum-likelihood decoder remains challenging. However, it is possible to compute the "marginals" efficiently. Let us write any logical Pauli operator $\bar{L}$ as

$$\bar{L}(\mathbf{a},\mathbf{b}) \propto \bigotimes_{i=1}^{k} X_i^{a_i} Z_i^{b_i}, \quad a_i, b_i = 0, \ldots, q-1, \tag{3.58}$$

where $\mathbf{a},\mathbf{b}$ are k tuples with $\mathbf{a} = (a_1, a_2, \ldots, a_k)$ and the same for $\mathbf{b}$. Let $\bar{L}_i(a_i, b_i)$ be the logical Pauli acting on the i th logical qubit. Consider the marginal error probability

$$p(E\bar{L}_i) = \sum_{a_j, b_j; \forall j \neq i} p(E\bar{L}(\mathbf{a},\mathbf{b})). \tag{3.59}$$

This can be computed using the mixed enumerator. Recall that one can compute $p(\bar{I}_i|s=0)$ by inserting B-type tensor enumerators for all atomic blocks whose logical leg correspond to qubits $j \neq i$, and A type for blocks whose logical legs correspond to qubit i. This enumerator, which we called $I_i(z)$, records the weight distribution of logical operators $\mathcal{N}_i \subset \mathcal{N}(S)$, which consists of all Pauli logical operators in the code that act as the identity on the i th logical qubit. If we treat the other qubits $\neq i$ as gauge qubits, we can think of it as recording all gauge-equivalent representations of the identity operator. For example, this is what we have done for the Bacon-Shor code and the holographic code. For general cosets of operator E_s, we now compute mixed coset enumerator for operator E_s such that $\sigma(E_s) = s$. Let us construct the mixed enumerators

$$M(\mathbf{u}; E_s, \Pi) = \wedge_{J,J_j}\left[A^J_{(i)}(\mathbf{u}; E_s^A, \Pi) \bigotimes_j B^{J_j}_{\neq i}(\mathbf{u}; E_s^B, \Pi)\right], \tag{3.60}$$

where $E_s = E_s^A \otimes E_s^B$ and $E_s^{A,B}$ are the Pauli substrings that only have support on physical legs of the atomic codes that are mapped to type-A or B tensor enumerators, respectively. We take $\wedge_{J,J_j}$ to be tracing over the appropriate legs required by the tensor network. This now enumerates the weights of $E_s\mathcal{N}_i$. We can repeat this a number of times for different $E_s = \bar{L}_i P_s$ s, and the resulting enumerators would provide the requisite error probabilities $p(P_s\bar{L}_i|\sigma(P_s))$.

For example, under symmetric depolarizing noise with probability p,

$$p(P_s\bar{L}_i|\sigma(P_s)) = M(z=p, w=1-3p; P_s\bar{L}_i, \Pi). \tag{3.61}$$

For other error models, we again select the appropriate parameters for the abstract n-tuple $\mathbf{k}$ and weight function. See Sec. III B and Appendix A.

A decoder can then choose an operator with the highest probability then correct the error by acting $E\bar{L}_i$ on the system. In the case where no other logical qubits are present, this reduces to the maximum-likelihood decoder.

It is easy to generalize this such that $\bar{L}_i$ can include multiple qubits logical qubits in some set κ such that

$$M(\mathbf{u}; E_s, \Pi) = \wedge_{J_i,J_j} \bigotimes_{i\in\kappa} A_i^{J_i}(\mathbf{u}; E_s^A, \Pi) \bigotimes_j B^{J_j}_{\notin\kappa}(\mathbf{u}; E_s^B, \Pi). \tag{3.62}$$

However, in general, if we include $|\kappa|$ qudits in the mixed enumerator such that we only integrate out $j \notin \kappa$, then we need to check $q^{2|\kappa|}$ terms to find the error operator with the highest marginal probability.

H. Logical error rates

1. Exact computations

We have seen previously how one can compute the trivial syndrome enumerators, which yield the uncorrectable error probability. We can interpret the value $p_D = 1 - A/B$ as a logical error rate for a decoder with perfect syndrome measurements such that one discards the state whenever a nontrivial syndrome is measured. For such processes, one can define an error-detection threshold p_{th} such that p_D is suppressed as a function of d for error rates below the threshold. One such example is shown in Fig. 17 for the surface code and 2D color code.

Remark 1. If a class of quantum codes has an error-detection threshold under i.i.d. depolarizing error, then the threshold is $p_{\text{th}} = 1/6$.

Proof. Let $A^*(z), B^*(z)$ be the enumerators with normalization such that $A_0^*, B_0^* = 1$. Then for a quantum code with dimension K, $A(z) = K^2 A^*(z)$ and $B(z) = KB^*(z)$.

Thus

$$\frac{B^*(z)-A^*(z)}{B^*(z)} = 1 - \frac{1}{K}\frac{A(z)}{B(z)}. \tag{3.63}$$

Now, homogenizing, the MacWilliams identity has $B(w,z) = A((w+3z)/2,(w-z)/2)$, and using $z = p = 1/6$ and $w = (1-3p) = 1/2$, we see $A(1/2,1/6) = B(1/2,1/6)$ for every quantum code. Therefore, all curves $p_L(p, 1-3p)$ cross at $(1/6, 1-1/K)$. ■

We may similarly ask whether the current tensor-network method can efficiently compute the exact logical error rate under other decoders. We do not provide such a method in this work, though it may be an interesting direction. A simple application of the current method fails to be efficient in the following examples.

The exact logical error rate with maximum-likelihood decoder can be expressed as

$$p_L = \sum_s [B^s(\mathbf{u}; P_s, \Pi) - \max_{\forall \bar{L}\in\mathcal{L}}\{A^s(\mathbf{u}; P_s\bar{L}, \Pi)\}] \tag{3.64}$$

and the error rate for a Bayesian decoder is

$$p_L = 1 - \sum_s \frac{1}{B^s(\mathbf{u}; P_s, \Pi)} \sum_{\bar{L}} A^s(\mathbf{u}; P_s\bar{L}, \Pi)^2. \tag{3.65}$$

We see that both of them involve nonlinear functions of the weight enumerators, which makes it difficult to compute efficiently through a tensor-network method. It would appear that one has to sum over exponentially many syndromes even if each enumerator can be produced efficiently.

This does not mean that enumerators cannot improve the computation of logical error rates. In practical decoding, it is far more relevant to consider a sample with only polynomially many distinct syndromes after running the decoder for a reasonable amount of time. It is also the case for all sampling-based simulations that are currently used for error and threshold computations.

2. Error-rate estimation

In addition to computing exact error probabilities given syndrome s, one can also use enumerators to provide more accurate estimates for logical error rates in conjunction with sampling-based methods.

Conventional sampling methods generate errors E based on particular noise models. Once the error is generated, its associated syndromes $\sigma(E)$ are determined. Note that for noiseless syndrome measurements, $\sigma(E)$ always outputs a syndrome s deterministically. However, for more realistic models with faulty measurements, the outcome s can depend also on the noisy measurement process. A decoder $\mathcal{D}(\sigma(E))$ then takes the syndrome and suggests a recovery operator R with probability $p_{\mathcal{D}}(R|s)$, $\sum_R p(R|s) = 1$. If $RE \sim \bar{L}$ is equivalent to a nonidentity logical operator, then a logical error has occurred and this adds to the error probability p_L. This process is repeated until a large enough sample size has been established such that the overall p_L estimate is believed to have sufficiently converged.

We can improve up this method, especially those derived from rare events and syndromes with enumerators. Given an error model (e.g., depolarizing noise with fixed error probability p) a set of errors are generated using existing sampling methods. Subsequent syndrome measurements (either noiseless or noisy) lead to a sampled syndrome distribution $\mathcal{P}(s)$ such that $\sum_s \mathcal{P}(s) = 1$ and only has support over polynomially many distinct syndromes. In our case, we assume that we are given the distribution $\mathcal{P}(s)$, the error-correcting code (along with its tensor-network construction), the error model in question, and a decoder $\mathcal{D}$ of the user's choice.

The logical error rate estimates are thus given by

$$\bar{p}_L(\mathbf{k}) = \sum_s \mathcal{P}(s) \sum_R p_{\mathcal{D}}(R|s)\Big(1 - \frac{A_s(\mathbf{k}; R, \Pi)}{B_s(\mathbf{k}; \Pi, \Pi_s)}\Big), \tag{3.66}$$

where $A_s(R)/B_s$ is precisely the expected probability where the decoder's choice of R successfully corrects the error based on the syndrome. For a maximum-likelihood decoder, $p_{\mathcal{D}}(R|s)$ is also trivial except for one R. For a pure sampling-based method, the probability $A_s(R)/B_s$ would usually require a large number of events before the estimate converges to its true value. Therefore, its estimate for rare syndromes can be wildly inaccurate. Here with the enumerator method, we can compute these quantities exactly, thereby improving the accuracy for $\bar{p}_L$. It is also useful sometimes to further sort the logical error rate by operator types. This can be done by excluding certain terms in Eq. (85) from the summation over R. We do not provide its explicit forms here as the extension is somewhat trivial and situation dependent.

In scenarios where the computation cost of enumerators are relatively expensive, one can complement, for instance, the Monte Carlo method, where only error rates associated with rare syndromes are computed using weight enumerators.

Faulty measurements: With logical error probabilities in hand, we can compute error thresholds in the usual way by repeating such calculations or estimations for a class of codes with different distances. Note that the use of enumerators above is compatible with any error model composed of identical single-qubit error channels. The computation also fully accommodates different models of

noisy syndrome measurements, as they only affect the distribution $\mathcal{P}(s)$. Furthermore, the impact of each decoder can be independently evaluated to produce the conditional probability $p(R|s)$. We hasten to point out that the choice of decoder here is completely arbitrary and not limited to the decoders we constructed in Sec. III G based on weight enumerators.

Since the contributions from the error channel, noisy measurements, decoders, and enumerators can be separated into independent modules, one can prepare them separately. For example, one can prepare a syndrome distribution $\mathcal{P}_0(s)$ with noiseless measurements. If the measurements are noisy, they are given by some set of transition probabilities $p(s_f|s_i)$, which depend solely on the noise model associated with the measurement. Composing these probabilities we get

$$\mathcal{P}(s) = \sum_{s_i} \mathcal{P}_0(s_i) p(s|s_i). \tag{3.67}$$

Once the set of relevant syndromes have been established, which we take to be poly$(n-k)$, we create the decoding table from which $p_{\mathcal{D}}(R|s)$ can be obtained. At the same time, the enumerators that depend on s and R may be prepared in parallel, if needed. In many cases, exact contractions may not be needed as we may not require the same level of accuracy for distance verification. In such cases, approximate but efficient contraction algorithms maybe sufficient.

IV. TENSOR NETWORKS FOR CODES

As our primary tool for speedup comes from the QL description of the code, to make use of these methods, one also needs a modular construction for the codes. For codes created from QL, this is automatically true. Here we also provide a general method for decomposing known codes and states into smaller "quantum Lego blocks." We give two approaches for performing this decomposition based on how the code or state is prepared. As the enumerator for graph states [11] is of interest, we also provide an explicit QL decomposition of all graph states.

A. Quantum codes from quantum Tanner graph

For any stabilizer code with Abelian and non-Abelian stabilizer group (such as XP stabilizer codes [37–39]), the codewords can be defined by the simultaneous $+1$ eigenspace of the stabilizer elements.

Any such code can be rewritten with a QL decomposition where the tensor network is isomorphic to the Tanner graph of the code [Fig. 6(a)]. For each check node connected to ℓ qubits and qudits, we place a degree GHZ tensor, which is the encoding tensor of a repetition code. The GHZ tensor is also known as a Z spider in ZX calculus [40]. A similar repetition code with H applied to each leg

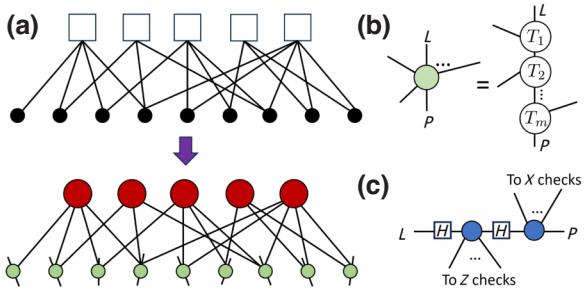

FIG. 6. (a) A quantum code written as a Tanner graph where qubits are nodes and checks are squares. A tensor network isomorphic to this graph can be constructed from Z spiders (red) and local QL codes (green). (b) Each local code has two dangling legs where L marks the logical leg and P marks the physical leg. The remaining legs are contracted with the check node tensors. (c) For the special case of a CSS code, the local code simplifies to the contraction of Hadamard tensors and X spiders (blue).

is known as an X spider. For each physical node checked by m checks with operators $\{g_i, i = 1, \ldots, m\}$ acting on the physical node, we place a tensor (green) with degree $m+2$ shown in Fig. 6(b) where each of the m contracted legs is connected to the corresponding check node tensor the qubit is checked by. The remaining two dangling legs represent the logical and physical degrees of freedom associated with the atomic code at each physical node. The specific T_i tensor of degree 3 is completely determined by the type of operator g_i that is present in the stabilizer check. If the code is a Calderbank-Shor-Steane (CSS) code, then the local tensor takes the simple form of Fig. 6(c), where it is a contraction between X spiders (blue), which are repetition codes in a different basis, and Hadamard tensors. With this construction, one can easily build up tensor networks for existing codes with known stabilizer checks. When applied to topological codes, the quantum Tanner graphs are structurally similar to existing constructions with similar bond dimension, e.g., [41] for the toric code, which is CSS, and for the twisted quantum double model [42,43], which has a non-Abelian stabilizer group. Details of this construction are given in Appendix E. The number of seed codes from the quantum Tanner graph are essentially optimal where we have $2n-k$ atomic Legos for an $[\![n,k]\!]$ code, identical to the usual Tanner graph description. If the code is a low-density parity-check (LDPC) code, then each tensor node also has bounded degree.

B. Circuit-based tensor network

The quantum Tanner graph tensor network covers the vast majority of the existing quantum codes. For codes and states with an encoding circuit, then one can also convert the circuit into a tensor network [44] as it is simply the contraction of tensors of unitary gates and product $|0\rangle$

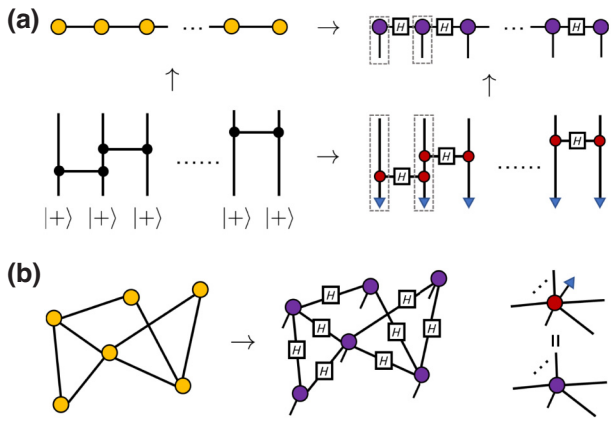

FIG. 7. Any graph state (yellow) can be converted to a tensor network using its encoding circuit constructed from CZ s acting on $|+\rangle$ s. The tensor network consists of the GHZ tensors (red), Hadamard tensors (H), and contraction with $|+\rangle$ (blue triangles). Note that multiple GHZ tensors, which are also Z spiders, can be merged to create a larger Z spider. (a) Example: a 1D cluster-state tensor network from encoding circuit. (b) Any graph state can be converted to such a tensor network using the procedure above. A purple tensor is the GHZ tensor contracted with a $|+\rangle$ tensor.

states. For Clifford gates, the states dual to these tensors are stabilizer states. For instance, concatenated codes can naturally yield a log-depth tree tensor network. In general, the connectivity of a subregion of the network can scale linearly as the number of gates and tensors inside the region. For a one-dimensional (1D) (spatially) local circuit, it is in principle possible to cut through the network in the time direction. The edge cuts are upper bounded by the circuit depth T, and hence each contraction is no costlier than $O(\exp(T))$. For log depth, this clearly yields exponential speedup. For d spatial dimensions, the number of edge cuts is upper bounded by the surface area of the space-time region $\ell^{d-1}T$ where $\ell^d \lesssim n$. Therefore, the upper bound for the cost for each contraction is $O(\exp(n^{1-1/d}T))$. This can still lead to a subexponential speedup as long as $T \lesssim n^{1/d-\epsilon}$ asymptotically.

A simple procedure of such a circuit-to-tensor-network conversion is constructed in Fig. 7 for all graph states. Each CZ gate can be converted into the fusion of three tensors, which can then be individually simplified through local recombinations [Fig. 7(a)]. This produces a tensor network with the same graph connectivity as the graph state with GHZ tensors (Z spiders) on the nodes and Hadamard tensors inserted on the edges [Fig. 7(b)].

It is important to note that given the level of generality of our method, not all tensor networks will be exactly contractible in polynomial time as we see in the next section. This is because finding enumerators is NP hard and exponential time for general tensor-network contraction is unavoidable.

V. COMPUTATIONAL COMPLEXITY

A. General comments

1. Brute-force method

For a generic stabilizer code, the construction of its weight-enumerator polynomial is at least NP hard. We thus expect the same for a generic quantum code. Indeed, as we see that constructing enumerators solves the optimal decoding problem [45], such tasks must be at least $\#P$ complete. A simple brute-force algorithm is exponential in the system size. For stabilizer codes, one can enumerate all of its stabilizer or normalizer elements, which is of $O(q^{n-k})$ and $O(q^{n+k})$, respectively. This extracts the relevant coefficients A_d, B_d. For a general quantum code, each coefficient A_d, B_d is already hard, as it involves $q^n \times q^n$ matrix multiplications. One then has to repeat this $O(q^{2n})$ times for each error basis element. Therefore, the complexity for the brute-force method is $O(q^{O(n)})$ for general quantum codes of local dimension q. A slightly better strategy computes only the coefficients of A_d and then performs a MacWilliams transform, which is polynomial in n. Therefore, for complexity estimates, it is sufficient that we provide the estimate for computing $A(\mathbf{u})$.

2. Tensor-network method

Now we analyze how our method improves this picture assuming the QL constructions are known.

a. Tensor preparation overhead. Let us first revisit the encoding tensor network of an $[\![n,k]\!]$ stabilizer code with local dimension q where each tensor is obtained from a small stabilizer code. We assume that the degree of each tensor (including dangling legs) is bounded by some constant c. This is to ensure that the complexity in constructing the tensor enumerator of each node is upper bounded by a constant overhead [46]. Then consider the graph $G = (V, E)$ produced from the tensor network by removing all dangling legs such that the tensors are vertices and contracted legs are edges. Suppose the tensor-network representation is one such that $|V| \leq C(n+k)$ for some constant C, then preparation of the atomic blocks has worst-case complexity $O((n+k)q^{5c})$. In fact, many tensor networks consist of only a few types of tensors, e.g., recall that any QL structure is constructible from a constant number distinct blocks, making even $O(q^{5c})$ sufficient. Therefore, the overhead for tensor preparation is usually constant while a generous upper bound is at most linear in the system size. Here we assume that the tensor network does not contain an overwhelming number of tensors that have no dangling legs, e.g., a deep quantum circuit. This assumption can always be satisfied (e.g., MPS).

b. Tensor contraction We now contract these tensors to build up the tensor network. Recall that each tensor contraction may be construed as a matrix multiplication. Suppose we have two tensors of $p \le m$ legs, respectively, while we contract $n \le p$ legs. For the most general quantum code, we need to use the full tensor enumerators as building blocks, which have bond dimension $\chi = q^4$ and can be reshaped as a multiplication of two matrices of size $\chi^{(p-n)} \times \chi^n$ and $\chi^n \times \chi^{(m-n)}$. Hence each contraction step with the same parameters above has worst case $O(\chi^{(p+m-n)})$. For codes that only needed reduced enumerators, this can be done with $\chi = q^2$. For stabilizer codes, these matrices are especially sparse and have at most q^p, q^m nonzero elements, thus each such contraction is loosely upper bounded by $O(q^{p+m+\min(p,m)})$. Therefore, the computational complexity scales exponentially with the number of uncontracted legs during tensor contraction.

To incorporate the symbolic functions, additional degrees of freedoms are often needed. The specifics can depend on the implementation. One method is to introduce a separate index with bond dimension $(n+1)^\ell$ to track the power of the polynomial (Appendix C). This adds another factor of n^ℓ to the complexity counting above. The power ℓ depends on the number of independent variables one needs to track. For Shor-Laflamme enumerators $\ell = 1$, but $\ell > 1$ for the refined enumerators. As this cost can vary depending on the treatment of symbolic objects, we do not include their contributions in the following estimates. One can easily restore them when needed.

c. Fully contracted tensor network Aside from minor corrections related to symbolic manipulations and those associated with storing and manipulating for large numbers, the computational complexity would be determined by the contractibility of the tensor network, which is ultimately dominated by the cost of multiplying large matrices. Heuristically, the cost of tensor contraction scales linearly with the bond dimension of the uncontracted indices, or exponentially with the number of minimal edge cuts in the tensor network.

In the ensuing discussion, we will use base e exponential for complexity. For a tensor network with bond dimension χ, we can generally set $e \to \chi$ to obtain the worst-case complexity estimate. As we discussed earlier, the general rule of thumb for bond dimension is $\chi = q^4$ for the full tensor enumerator, $\chi = q^2$ for codes that only requires reduced enumerators. However, using a sparsity argument in stabilizer codes, the effective bond dimension needed in an efficient representation can even be as low as q.

Let us represent a sequence $\mathcal{S}_G$ of tensor contractions by a sequence of induced subgraphs $H_i = (V_i^H, E_i^H)$ where $V_i^H \subset V$, $V_{i+1}^H = V_i^H \cup \{v_{i+1} \in V \setminus V_i^H\}$, and $V_0^H = \{v_0 : v_0 \in V\}$. In other words, we construct a sequence of subgraphs by adding one additional vertex at a time. The sequence terminates at $i = |V| - 1$, when the subgraph contains G. Let $E_c(W, W') = \{e = \{v_a, v_b\} \in E : v_a \in W, v_b \in W'\}$ denote the set of edges connecting any two sets of vertices W, W' and let M_{i+1} be the connected component of H_{i+1} containing v_{i+1}.

Then the complexity for the i th step of contraction is

$$\mathcal{C}_i \lesssim \exp(|E_c(V_{H_i} \cap V_{M_{i+1}}, V \setminus V_{H_i})| + \deg(v_{i+1}) - |E_c(V_{H_i} \cap V_{M_{i+1}}, \{v_{i+1}\})|) \lesssim O(\exp(|C_{\max}|)), \tag{5.1}$$

where $|C_{\max}| = \max_i |E_c(V_{H_i}, V \setminus V_{H_i})|$ is the largest possible cut through the tensor network during contraction. Then we see that the number of computations needed for calculating the final tensor enumerator of the tensor network is given by

$$\mathcal{C} = \sum_{i=0}^{|V|-1} \mathcal{C}_i \lesssim O(|V| \exp(|C_{\max}|)). \tag{5.2}$$

The upper bound is a pretty drastic overcounting especially if H_i contains many disconnected components, as many do not enter the contraction. In other words, as long as each connected component of the induced subgraph has only $\log |V|$ connectivity with its complement throughout the sequence $\mathcal{S}_G$, then the complexity is polynomial in $|V|$.

B. Cost for common codes

a. Tree tensor network. Tree tensor networks can be used to describe concatenated codes over n qubits (leaves). It is also known that these tensor networks can be contracted with polynomial complexity. A contraction algorithm would start from the leaves of the tree and contract into $O(n)$ disconnected components of the graph. Each piece in this first layer of contraction has at most $\mathcal{E} \sim O(c)$ open legs where c is the maximum degree or branching factor in the tree. Then at each iteration, we join the $\le c - 1$ branches with another tensor. The maximum number of open legs on each connected component is always bounded by c, therefore the complexity for each contraction is at most $O(e^{2c})$. For a tree with n leaves, the overall complexity is $O(ne^{2c})$ for tensors of bounded degree, Fig. 8. If the codes on each node are identical, then we only have to perform a separate contraction at each layer, yielding a complexity $O(\log n)$, Table I (general and symmetric). The latter would be doubly exponentially faster than brute-force enumeration.

b. Holographic code. For tensor networks of holographic codes [47–50], the network is taken from a tessellation of the hyperbolic disk. This is slightly more connected than

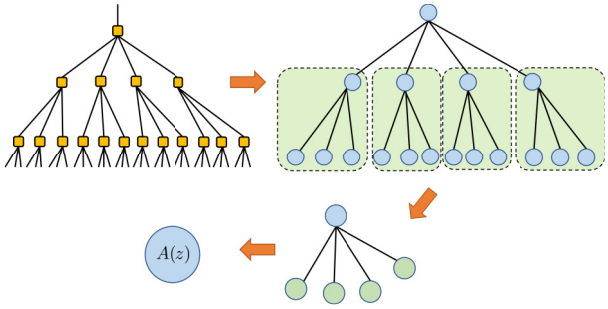

FIG. 8. Tree tensor network for concatenated code. It is efficiently contractible from the bottom up and can be parallelized.

the tree tensor network (TTN) as it contains loops. The contraction strategy is similar to that of the TTN, except now minimum cuts depend on the system size such that each connected component has at most $O(\alpha \log n)$ open legs during the contraction. The parameter α depends on the tessellation. Then

$$\mathcal{C} \sim \sum_{m=1}^{n} \exp(\alpha \log m + 2c) \leq \exp(2c) n^{\alpha+1} \sim O(n^{\alpha+1}). \tag{5.3}$$

A similar counting argument holds for the hyperbolic surface code, where minimal cuts remain logarithmic in the system size.

c. Codes with shallow local circuits. If the encoding circuit of a code is known (e.g., stabilizer code once the check matrices are given), then we can easily convert the circuit into a tensor network. If these circuits are shallow, say, of constant or $\log n$ depth, then one can contract the circuit-induced tensor network in the spacelike direction where the minimal number of edge cuts would be given by the circuit depth. Thus the enumerators of such codes can be prepared in poly(n) time.

d. Codes on a flat geometry. These are codes on an Euclidean geometry of dimension D such as ones where the code words may be described by a PEPS. Some examples include the 2D color code, the surface code, Haah code [51], etc. Constructions like the Bacon-Shor code also fall under this category. Note that the worst-case complexity holds for any such tensor network regardless of the specific tensor construction or its symmetries.

For codes whose discrete geometry are embeddable in the D-dimensional Euclidean space, we simply "foliate" the lattice with co-dimension 1 objects. Each such object can be built up from $O(n^{1-1/D})$ contractions where each contraction retains at most $O(n^{1-1/D})$ open legs. Then $\mathcal{C} \sim O(n \exp(n^{1-1/D}))$. Compared to the brute-force method, this permits a subexponential speedup.

If the geometry of the network allows for fewer open edges during tensor contraction, then it is possible to get further speedups. Note the above counting assumes $n \sim L^D$ for a system that has similar lengths in different directions. If all but one direction have bounded length L then we obtain an exponential speedup. For example, consider a rectangular surface code of size $L \times n/L$ on a long strip where L is bounded, then each contraction along its shorter side is only $O(\exp(L))$.

Note that the hardness of evaluating the weight-enumerator polynomial here is directly tied to the hardness of the tensor-network contraction. It was shown in Ref. [52] that contraction of PEPS is average case $\#P$ complete. Therefore, there is strong reason to believe that an exponential speedup of this process is unlikely for both classical and quantum algorithmic approaches using tensor networks if one disallows postselection and chooses the tensors in a Gaussian random fashion. However, we also note that often the tensors are strictly derived from stabilizer codes. Therefore, it is not impossible that these added structures in the discrete symmetry and contractible 2D tensor networks may permit further speedups.

TABLE I. Tabulates the computational cost for enumerator preparation from tensor-network contractions. There is additional complexity associated with the symbolic manipulation of the polynomial, storage of large numbers, and MacWilliams transforms, which can also contribute an additional cost that can be superlinear.

Network architecture	TN cost	Code examples
Tree	$O(\log n)$	Concatenated (symmetric)
Tree, 1D area law	$O(n)$	Concatenated (general), convolutional
2D hyperbolic	$O(n^{\alpha+1}), \alpha > 0$	Holographic, surface code (hyperbolic)
(Hyper)cubic	$O(n \exp(n^{1-1/D}))$	Topological (Euclidean), Bacon-Shor
(Hyper)cubic (bounded L)	$O(n \exp(L^{D-1}))$	Rectangular surface code
δ-volume law	$O(n \exp(\delta n)), \delta < 1$	Nondegenerate code, random code
Generic encoding circuit	$O(n^2 \exp(n)/\log n)$	Generic stabilizer code

e. Codes with volume-law entanglement. For states that have volume-law entanglement for any subsystem, let us assume that the number of edges connected to vertices in a subregion is proportional to the number of vertices in that region, i.e., $\eta|V|$. For simplicity, let us also assume that the number of tensors and qubits are roughly equal. In general, η need not be less than one. This is because each node may be connected to multiple nodes in the complementary region, while the entanglement captured in each bond is not maximal. However, if a carefully crafted tensor network is efficiently capturing the entanglement of the state, such that each bond is roughly maximally entangled, then we could expect the number of bonds cut to be less than or equal to the total number of qubits in the region for large enough subregions. Then the cost for each contraction is $O(\eta|V|)$. For $\eta < 1$, this provides a polynomial speedup. If the number of bonds cut $\leq d$ for any subsystem and the code distance $d = \delta n, \delta < 1$, which is the case for random codes, then the overall complexity would be

$$\mathcal{C} \sim O(n \exp(\delta n)), \tag{5.4}$$

which again admits a polynomial speedup.

However, if the number of bonds cut for a subsystem is $\geq n$, then we do not get any speedup. This would be the case for all-to-all connected graphs where the edge cuts can be of size $(n/2)^2$, our algorithm at $O(\exp(n^2/4))$ will actually be slower than the brute-force algorithm. For another example, consider the encoding circuit of any stabilizer code has $n^2/\log n$ complexity, which can be thought of as a tensor network. Suppose we simply contract the circuit tensor-network timeslice by time slice, then we expect $|C_{\max}| \sim n$ because each time slice would correspond to a tensor network with $O(n)$ legs and the worst-case complexity scales as approximately $O(n^2 \exp(n)/\log n)$. This is fully expected, as we should not be able to solve a #*P*-complete problem in polynomial time. Therefore, in this regime, even if its tensor network description is optimal and minimizes the number of edge cuts for any subregion, the tensor-network method would still only provide a polynomial speedup at best.

C. Entanglement and cost

In this work, we say that a tensor-network representation is *good* if its graph connectivity reflects the entanglement structure of the underlying state. In other words, the entanglement entropy of any subsystem can be reasonably well approximated by the number of edge cuts when bipartitioning the graph into the subsystem and its complement. This definition does not require the network to be efficiently contractible [52,53]. If we use the tensor-network connectivity interchangeably with subsystem entanglement then we see that the complexity for computing the weight enumerator can be connected with the amount of entanglement present in the codewords. For more highly entangled codewords and states, its tensor network will be more connected, and hence the number of edge cuts for each subsystem will be higher. This provides us a heuristic where the general expectation of its weight-enumerator computation should scale as approximately $\exp(S)$ where S is roughly the maximum amount of entanglement for subsystems we generate during tensor tracing. We see that this is indeed the case for our examples—the complexity is polynomial for codes whose code words are weakly entangled, i.e., $S \lesssim \log n$ and generally subexponential for states that satisfy an area law $S \sim n^{1-1/D}$ for systems with D-dimensional Euclidean geometry.

For *nondegenerate* quantum codes, the $(d-1)$-site subsystem are maximally mixed, hence $d \sim S$. Therefore, up to polynomial-factor corrections, we expect the complexity lower bound for computing the enumerator polynomial to be comparable to that of finding the minimal distance in classical linear codes [17,54], i.e.,

$$\mathcal{C} \sim \exp(O(S)) \sim \exp(O(d)). \tag{5.5}$$

For this high-level analysis, we will neglect other subleading terms and the dependence on rate $R = k/n$. Because stabilizer codes can be identified with classical linear codes over $GF(4)$ [54], it means that the tensor-network method should have comparable complexity scaling with existing algorithms for nondegenerate stabilizer codes.

In *degenerate codes*, however, there exist subsystems where $S \ll d$. For example, a gauge fixed Bacon-Shor code can be constructed from a TTN (Sec. VI D). Although certain subsystems are highly entangled, its much weaker entanglement for some other subsystems allows one to engineer the network such that it is written in an efficiently contractible form, such that each step of the contraction is bounded by a constant. Depending on the gauge, we can get away with an enumerator with as few as $2\sqrt{n}$ such contractions. Although the code has overall distance $d \sim \sqrt{n}$, the cost in preparing its enumerator is only $O(\sqrt{n})$ time, compared to a naive distance scaling of $O(\exp(\sqrt{n}))$ (Fig. 22). Therefore, we expect some degenerate codes to have $\mathcal{C} \ll \exp(O(d))$, which is a substantial speedup compared to known methods.

VI. EXAMPLES

Now we examine a few examples by computing the enumerators for codes that have order a hundred qubits or so. These analyses are to showcase the tensor enumerator method; they are not meant to be exhaustive nor do they represent the largest possible codes one can study with this method.

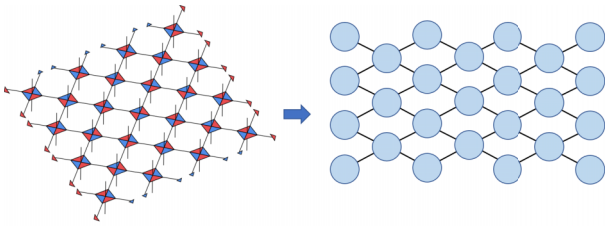

FIG. 9. A surface code and the tensor network of its weight enumerator.

A. Surface code

a. Kitaev's surface code. Recall from [1] that the tensor network for the surface code encoding map, Fig. 9(left), is one where each tensor is a $[\![5,1,2]\!]$ code and the boundaries are contracted with $|0\rangle, |+\rangle$ states (red and blue triangles). The upward pointing dangling legs denote the logical inputs and downward pointing legs denote physical qubits, therefore the encoding map has a nontrivial kernel and a physical qubit sits on each node. For each atomic block, we construct its tensor enumerator and contract them column by column to generate the entire network, Fig. 9 (right). For example, the quantum weight enumerators of a $[\![181,1,10]\!]$ surface code are

$$A(z) = 1 + 36z^3 + 180z^4 + 136z^5 + 1344z^6 + 7084z^7 + 24001z^8 + 60432z^9 + 286748z^{10} + \ldots \quad (6.1)$$

$$B(z) = 1 + 36z^3 + 180z^4 + 136z^5 + 1344z^6 + 7084z^7 + 24001z^8 + 60432z^9 + 286768z^{10} + \ldots, \quad (6.2)$$

where we count only 20 representations of nontrivial logical operators at weight 10.

Using a similar network, we can also find the coset weight distribution. Suppose a Pauli error acts on physical qubits in the form of Fig. 10(left). Note that we do not contract the Pauli errors into the encoding tensor network when defining the encoding map; if we actually contract the Pauli errors onto the physical legs in the tensor-network

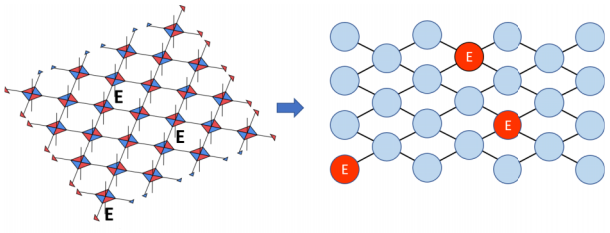

FIG. 10. The coset enumerator of a particular error string that acts trivially on some qubits.

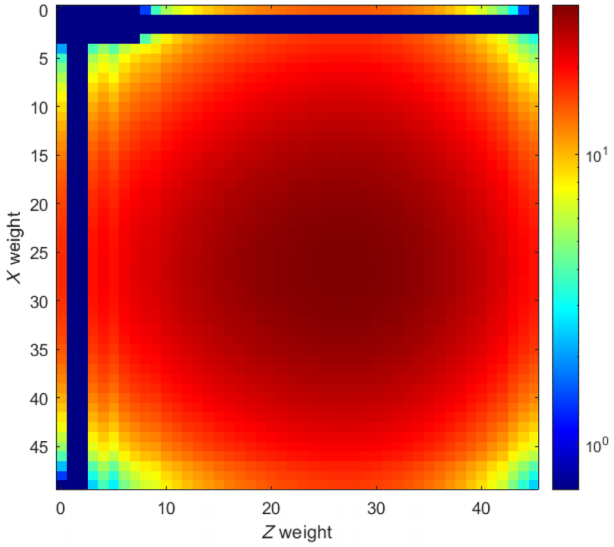

FIG. 11. Double enumerator of a 4 by 8 surface code at $n = 53$. Plotting log of operator weight distribution for nontrivial logical operators. A relatively small code is chosen for clarity in the figure.

construction then obtain enumerators from those networks, it would correspond to finding the stabilizer weight distribution for surface codes that have extra minus signs on certain generators. To build the coset enumerator, we swap out the original tensors in Fig. 9 (right) for the proper coset-tensor weight enumerator of each error node (red). The modified tensor network then computes the weight distribution of coset elements. For example, the coset distribution for a single X error at the bottom left corner for a $[\![113,1,8]\!]$ surface code, is

$$A^{sXbl}(z) = z + z^2 + 2z^3 + 31z^4 + 146z^5 + 284z^6 + 1258z^7 + 5180z^8 + 17627z^9 + \ldots \quad (6.3)$$

These exercises can be easily repeated for the double and complete weight enumerators where the weights are counted differently. For example, see Fig. 3 of Ref. [14] and Fig. 11.

b. Rotated surface code. In practice, it is easier to deal with rotated surface code as the distance scaling is better by a constant factor for a similar value n, Fig. 12. Note that one only has to modify the boundary conditions compared to the original surface code. The rotated surface-code tensor network is also easier to contract exactly. For reference, the enumerator for the $[\![256,1,16]\!]$ rotated surface code at $d = 16$ can be computed on a laptop with a run time of approximately equal to 20 min. The weight enumerators

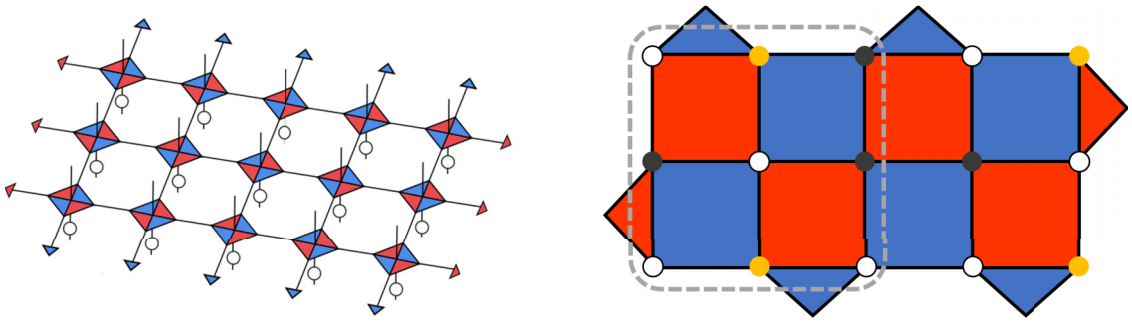

FIG. 12. Tensor network of a rotated surface code where the atomic codes are identical to those of the surface code. Only the boundary conditions are modified. One can also modify each tensor by contracting some other single qubit gate and tensor. The checks are given on the right where qubits (vertices) adjacent to red regions indicate Z checks and blue indicate X checks. For the derandomized local Clifford deformed code [55], white and yellow dots indicate local HSH and H deformations, respectively.

for this code are

$$A(z) = 1 + 30z^2 + 776z^4 + 15538z^6 + 276801z^8 + 4431408z^{10} + 65676619z^{12} + 912021486z^{14} + 12003931907z^{16} + 150911390280z^{18} + \ldots \tag{6.4}$$

$$B(z) = 1 + 30z^2 + 776z^4 + 15538z^6 + 276801z^8 + 4431408z^{10} + 65676619z^{12} + 912021486z^{14} + 12004980483z^{16} + 150970896992z^{18} + \ldots \tag{6.5}$$

Indeed, we see that the two coefficients start deviating at $d = 16$.

One can also obtain an error-detection threshold by assuming a decoder that performs no active error correction, but discards all instances that return a nontrivial syndrome assuming perfect measurements. Recall (Remark 1) that this threshold is at $p = 1/6 \approx 16.67\%$, which is quite similar to the code-capacity thresholds [56] across various decoders under depolarizing noise.

c. Local Clifford deformations. We can perform local modifications [13] on each tensor to perturb the (rotated) surface code. These are represented by the circle tensors that act on each qubit. For the vanilla surface code, these tensors are trivial (identity). However, we may choose them at will. For instance, if they are random single-qubit Clifford operators, then the tensor network reproduces the Clifford-deformed surface codes [25]. Similarly, if choosing every other tensor to be a Hadamard, then one arrives at the $XZZX$ code [57].

Because the Shor-Laflamme enumerator is invariant under local unitary deformations, it is clear that the logical error probabilities of such locally deformed codes would be identical under unbiased noise. However, this local unitary invariance is broken when we consider more general enumerators with other weight functions, which indicate that their performances under biased noise differ. In Fig. 13, we see that the derandomized Clifford deformed code (right) has fewer logical operators that have low Z weight, which is to be contrasted with the rotated surface code (left) and the $XZZX$ code (middle). We use a derandomized Clifford deformed code like the one shown in Fig. 12 (right) where yellow and white dots indicate local HSH and H rotations [55]. More general dimensions of the code follow from repeating the local patterns on the 3×3 blocks (enclosed by dashed lines) periodically.

For example, using the double enumerators, we contrast the performance of the $XZZX$ code and the derandomized Clifford deformed code, Fig. 14, under biased noise with $p = p_X + p_Y + p_Z$ and $p_X = p_Y = p_Z/(2\eta)$. It is clear from the normalized uncorrectable error rate (and hence effective distances) that the Clifford deformed construction vastly outperforms the $XZZX$ at high bias and small p. Note that the weight function for these double enumerators is slightly different from the one used in Appendix A or Ref. [9] because it enumerates the X, Y weight separately from the Z weights.

d. Coherent error. General quantum errors are not limited to random Pauli noise, which are somewhat "classical." Here we compute the coherent error probability of the rotated surface code using techniques introduced earlier.

Efficient methods for computing unitary rotations along X or Z have been introduced by Ref. [23] using a Majorana fermion mapping. Here we instead consider i.i.d. coherent error of the form $U = \exp(itY) = \cos(t)I + i\sin(t)Y$. Note that the normalized logical error rate differs for codes with even or odd X and Z distances because the abundance of Y-only operators differ for these codes, Fig. 15 (left).

When d_x, d_z are odd, the normalized logical error rate under coherent noise with rotation angle t coincides with that under the Y-only Pauli noise with probability

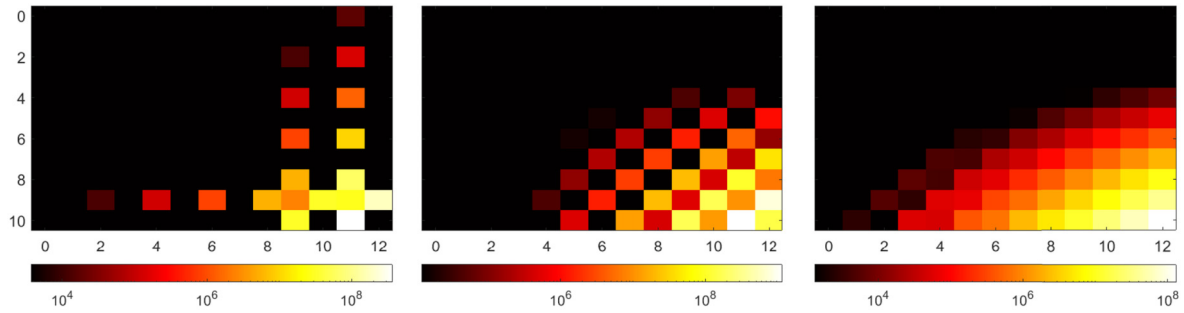

FIG. 13. Truncated X, Z weight distribution of nontrivial logical operators for the 9×9 surface code (left), $XZZX$ code (middle), and the Clifford deformed code (right). Horizontal axis, X weight; vertical axis, Z weight. Note that nonzero weights are invisible in this scale.

$p_Y = \sin^2(t)$. This is because at odd distances, the only Y-type logical operator acts globally on the system. When at least one of d_X or d_Z is even, then the coherent noise yields slightly higher logical error probability, Fig. 15 (right). However this only incurs a small correction with a similar order of magnitude, consistent with earlier results but in different settings [23]. A similar result holds for the $XZZX$ code with coherent noise of Y-only rotations because up to a phase, Y is invariant under Hadamard conjugation.

Although the impact of coherent noise with Z or X only rotations produce very different logical error profiles than those produced by the Z- or X-only Pauli noise in the rotated surface code, there exist $XZZX$ codes where their impact are identical. For instance, for the system sizes tested, the effect of such coherent errors and Pauli errors coincide when we have a square lattice where the width is equal to height. It also holds for some rectangular lattices, though not all. The reason is similar as before, where there is a sole logical operator consisting of only I and X (or Z), but the operator need not act globally. This may be due to special symmetries of the $XZZX$ code, which indicates that local deformations can be tuned to reduce the impact of coherent noise. Though it is also likely that such symmetries are restricted to the $s = 0$ sector. We leave a more systematic characterization of such behaviors to future work.

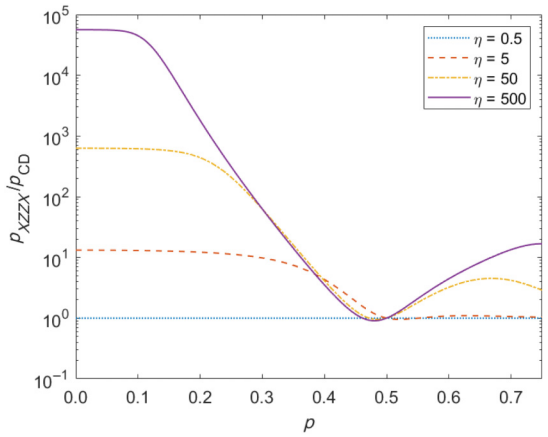

FIG. 14. The ratio between the $XZZX$ code normalized uncorrectable error rate p_{XZZX} and that of the Clifford deformed code p_{CD} as a function of physical error parameter p at different biases η for $d = 7$.

B. 2D color code

We first provide a novel tensor-network construction for the hexagonal 2D color code, which is a self-dual CSS code constructed entirely from Steane codes, Fig. 16. The class of such tensor networks constructs a family of $[\![3\ell(\ell+1)+1, 1, 2\ell]\!]$ codes. Similar color codes with hexagonal plaquettes can also be constructed by following the same contraction pattern in the bulk and imposing different boundary conditions. Just like the surface-code construction, this tensor network represents an encoding map with a nontrivial kernel [58]. One can similarly construct a codeword of this code, e.g., $|\bar{0}\rangle$ by contracting all the dangling logical legs with $|0\rangle$. Recall that each Steane code can be built from contracting two $[\![4, 2, 2]\!]$ atomic codes, which was used to construct the surface code. As such, this tensor network can indeed be construed as a double copy [59] of the surface code in some sense.

Each tensor in the left figure is a Steane code where the logical leg is suppressed. For the remaining seven physical legs, six are drawn in-plane while the remaining one is represented as a dot that corresponds to a physical qubit in the color code. Each stabilizer generator that acts on the plaquette of the $[\![7, 1, 3]\!]$ code is mapped to a stabilizer that acts on the four physical legs adjacent to a colored quadrilateral in the tensor description. Given this QL construction, its enumerator can be computed using the same method. For example, the enumerators for a $[\![91, 1, 11]\!]$ code are

$$A(z) = 1 + 54z^4 + 297z^6 + 2889z^8 + 24258z^{10} + 197493z^{12} + 1629738z^{14} + 13287999z^{16} + 108647952z^{18} + \ldots \quad (6.6)$$

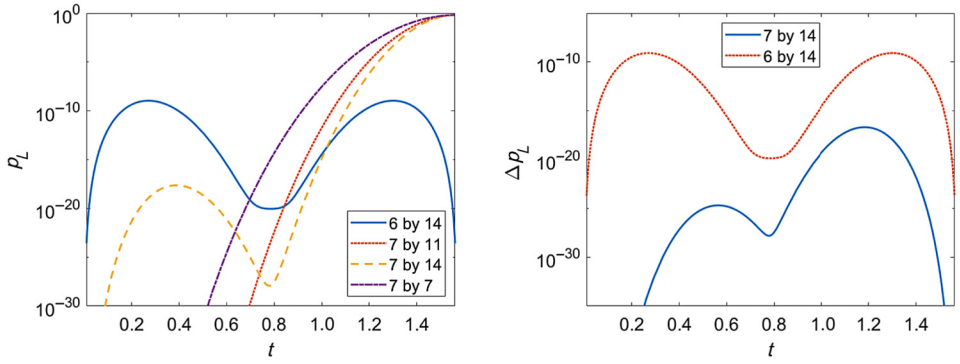

FIG. 15. Left: normalized logical error rate as a function of the rotation angle t for codes with size $n = d_x \times d_z$. Right: differences in normalized logical error rates $\Delta p_L = p_L^{\text{coherent}} - p_L^{Y\text{ only}}$. The plots shown are for trivial syndromes.

$$B(z) = 1 + 54z^4 + 297z^6 + 2889z^8 + 24258z^{10} + 4176z^{11} + 197493z^{12} + 67242z^{13} + 1629738z^{14} + 1066740z^{15} + 13287999z^{16} + 14401674z^{17} + 108647952z^{18} + \ldots \tag{6.7}$$

We see that the two coefficients start deviating at $d = 11$, thus verifying its adversarial distance. The computation time is only tens of seconds, but a better encoding is needed to avoid unnecessary allocation of memory space for 0s in the sparse matrix. Also note that the cancellation at even weights between A and B.

As we discussed in Remark 1, these codes admit a common error detection threshold at $p = 1/6$ (Fig. 17) thanks to the MacWilliams identity, and is close to the known code capacity threshold.

C. Holographic code

To demonstrate the usefulness of mixed enumerators, we now look at a class of finite rate holographic code [47] also known as the HaPPY (pentagon) code, originally conceived as a toy model of the AdS/CFT correspondence. Different versions of this code have been proposed in various contexts [27,49] where preliminary studies have examined some of its behaviors under erasure errors and symmetric depolarizing noise. However, the application of such codes in quantum error correction is far less understood compared to the surface code. Here we analyze the HaPPY code as a useful benchmark using our mixed weight-enumerator technology and present some novel results.

This code can be constructed from purely $[\![5, 1, 3]\!]$ atomic codes. It is known that, as a stabilizer code, it has an adversarial distance 3 regardless of n because of the bulk qubits that are close to the boundary. However, from AdS/CFT, we expect the logical qubits deeper in the bulk to be better protected and hence having different "distances." We can analyze the distances of these bulk qubits in different ways.

First as a stabilizer code, we define the *stabilizer distance* d_S of each bulk qubit as the minimal weight of all stabilizer equivalent nonidentity logical operator that acts on a bulk leg and qubit [27]. To enumerate such operators, we can build a mixed enumerator by contracting a B-type tensor enumerator associated with the bulk

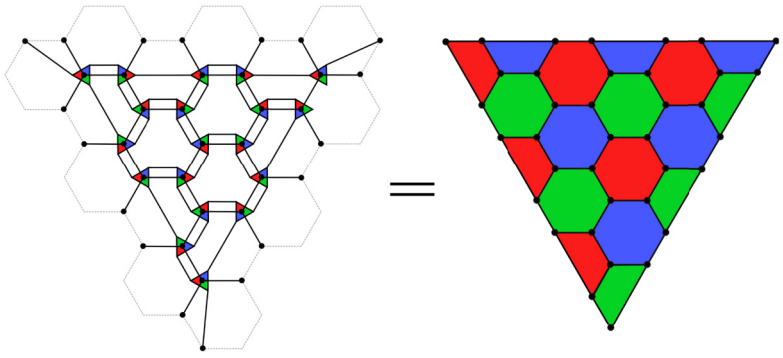

FIG. 16. A $[\![37, 1, 7]\!]$ 2D color code (left) tensor-network construction where (right) its stabilizer generators are all X or all Z operators acting on the vertices of each colored plaquette.

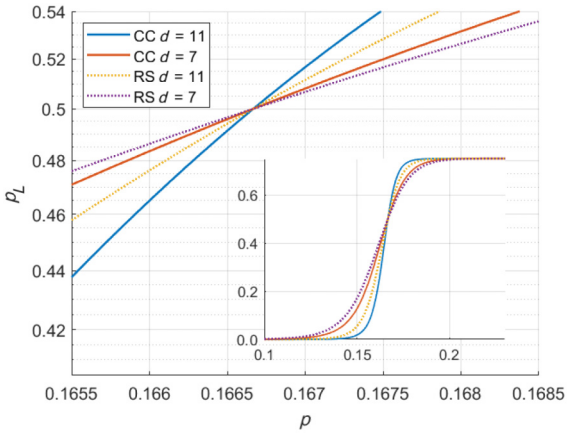

FIG. 17. Error-detection thresholds coincide for the 2D color code (CC) and the surface code (RS). Zoomed out plot on the corner shows the error probability in a greater range. Only two distinct distances are shown in the plot, since other distances cross at the same value.

tile that contains the logical qubit for which we compute the distance, with A-type tensor enumerators on the other tiles. Subtracting the enumerator polynomial $A(\mathbf{u})$ of the stabilizers, we then obtain a distribution for all the nonidentity logical operators acting on that bulk qubit Fig. 5 (top right).

One can also define the word distance of this code, as in Ref. [27], where it is simply the distance of the resulting subsystem code if we isolate one bulk qubit as the logical qubit and the rest as gauge qubits. To compute the word distance, we construct an $\tilde{A}(\mathbf{u})$ enumerator by contracting A-type tensor enumerator on the central tile with B-type tensor enumerator on the rest of the network. This enumerates the logical identities in the gauge code. Then subtracting it from the scalar $B(\mathbf{u})$ enumerator of the whole code yields the distribution of all gauge-equivalent nontrivial logical operators, Fig. 5 (bottom right).

For each code of a fixed size n, we then repeat this for bulk qubits at different radii from the center of the graph measured in graph distance. An explicit labeling of the qubits we study is shown in Fig. 5 (left) [60]. We give a summary for $n = 25$ and $n = 85$ in Table II, where $\mathcal{N}_S, \mathcal{N}_W$ denote the number of minimal weight stabilizer or gauge-equivalent representations of the nonidentity logical operators.

Although the stabilizer distance decreases as a function of radius, the word distance is more or less constant with respect to the radius. This is a particular consequence of the tiling and the atomic codes, such that erasure of four certain boundary qubits can lead to the erasure of the innermost bulk qubit [47]. Under depolarizing noise with probability p, the normalized uncorrectable error probability p_L is shown in Fig. 18(left). We see that the central bulk qubit in fact suffers from more errors because it has a greater number of minimal weight-equivalent representations despite having the same word distance as most other bulk qubits. We see a crossing because the outermost bulk qubit has a slightly lower distance compared to the rest.

Despite the constant word distance as a function of system size for logical qubits that are deep in the bulk, and presumably the lack of erasure threshold for the central bulk qubit [61], a larger n does hint at a greater degree of error suppression. Let $\Delta p_L = p_L(n = 85) - p_L(n = 25)$, we see that the error rate difference for the inner most bulk qubit has a slight suppression at small p while the outermost bulk qubit is the opposite. Intuitively, this is expected for general holographic codes as its construction is a slight generalization of code concatenation. As such, a crossing is expected, where adding more layers of code would generally lead to noisier bulk qubits in the deep IR when the physical error rates are sufficiently large while the opposite happens for the logical qubits in the UV. A more in-depth analysis of other holographic codes with varying word distances can be interesting as future work.

Let us also briefly examine its properties under biased noise using the double enumerator. The asymmetric distances d^X/d^Z are recorded in Table II. The XZ weight distribution is not symmetric, but the normalized logical error probability is fairly symmetric with respect to p_X, p_Z. Here we compare the logical error probability $\Delta p_L = p_L^{r=0} - p_L^{r=3}$ for the $n = 85$ code, Fig. 19. Like the

TABLE II. Tabulated stabilizer distances d_S and word distances d_W for two HaPPY pentagon codes at different sizes. $\mathcal{N}_S, \mathcal{N}_W$ denote the number of minimal weight stabilizer equivalent and gauge-equivalent representations of nontrivial logical operators, respectively. We also provide the corresponding asymmetric stabilizer and word distances sorted by X and Z weights. Radial distance r is the graph distance of the bulk qubit from the central tile for a code of fixed n. The qubits we studied are labeled according to Fig. 5.

	$[\![25,11,3]\!]$						$[\![85,41,3]\!]$					
r	d_S	$\mathcal{N}_S$	d_W	$\mathcal{N}_W$	d_S^X/d_S^Z	d_W^X/d_W^Z	d_S	$\mathcal{N}_S$	d_W	$\mathcal{N}_W$	d_S^X/d_S^Z	d_W^X/d_W^Z
0	9	30	4	60	5/5	2/2	23	240	4	60	13/13	2/2
1	5	6	4	54	3/3	2/2	13	48	4	36	7/7	2/2
2	3	3	3	3	1/2	1/2	9	12	4	24	5/5	2/2
3	n/a	n/a	n/a	n/a	n/a	n/a	3	12	3	12	1/2	1/2

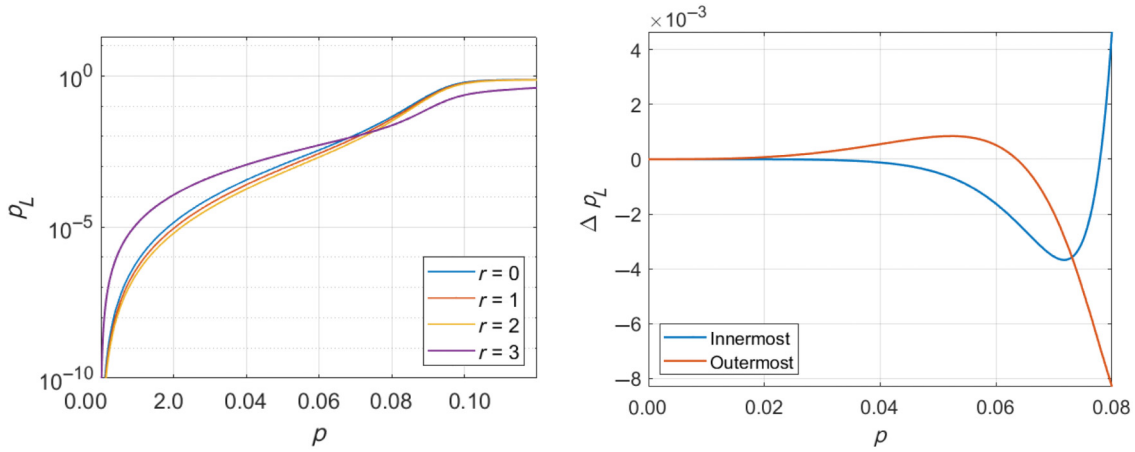

FIG. 18. Left: logical error probability of bulk qubits at different radii for a $[\![85, 41, 3]\!]$ HaPPY code. Right: the difference between logical error rates for two HaPPY codes of radii 3 and 2. At higher n, the innermost bulk qubit has lower logical error rate while that for the outermost is higher for sufficiently low physical error rate p. The opposite is true at higher p.

symmetric depolarizing noise, the bulk qubit deeper in the bulk provides slightly better protection for the encoded information, but becomes noisier at higher physical error rates. However, the bulk qubit at $r = 0$ does not provide better protection compared to the bulk qubits close to the boundary for any noise parameter in the heavily biased regime.

D. 2D Bacon-Shor code

For another example of the subsystem code, we study the 2D Bacon-Shor code. The tensor network for this code is identical to that of the surface code except we designate the physical legs every other row as gauge legs; see Appendix G.4 of Ref. [13]. It is conceptually convenient to think of these blocks as $[\![4, 2, 2]\!]$ stabilizer codes or $[\![4, 1, 2]\!]$ Bacon-Shor codes. As a subsystem code, it is most relevant to obtain its word distance. To that end, we construct its mixed enumerator $I(z)$ for the logical identity. The enumerator for the nontrivial logical operators (non-identity logical operators multiplying any element of the gauge group) is $C(z) = B(z) - I(z)$. It is most convenient to express these enumerators graphically, Fig. 20.

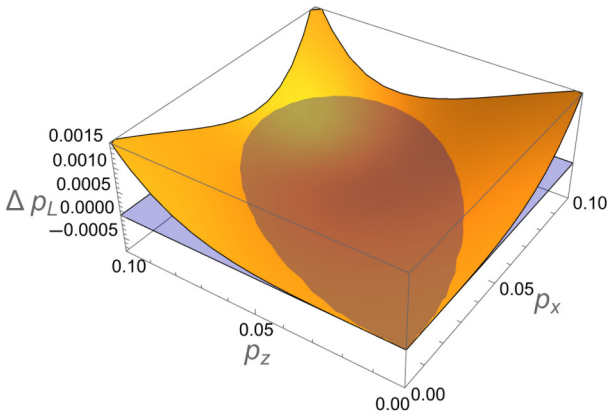

FIG. 19. $\Delta p_L = p_L^{r=0} - p_L^{r=3}$ as a function of p_x, p_z the bit-flip and phase-error probabilities. The blue translucent plane marks $\Delta p_L = 0$, below which the bulk qubit at $r = 0$ provides better protection.

Computing $B(z)$ is relatively straightforward, as we build it by contracting all $\mathbf{B}(z)$ of the $[\![5, 1, 2]\!]$ and $[\![4, 2, 2]\!]$ codes in the tensor network and then renormalize B_0 to 1. Practically, we compute $A(z)$ by contracting all the $\mathbf{A}(z)$ of these tensors then perform a MacWilliams transform. However $I(z)$ requires extra care as we need to place $\mathbf{B}(z)$ on the odd number rows for the regular $[\![5, 1, 2]\!]$ codes and $\mathbf{A}'(z)$ for the $[\![4, 1, 2]\!]$ Bacon-Shor codes on even rows. Although these tensors in the encoding map are identical, the downward pointing legs in the $[\![5, 1, 2]\!]$ code now maps to a gauge leg in the $[\![4, 1, 2]\!]$ code. Therefore, we must account for its weight distributions appropriately. It can be checked that the logical legs on the odd rows and columns are correlated with the logical legs on the even rows and columns. Therefore, they only contribute to an overall normalization.

Above computations can also be easily generalized to double and complete enumerators for the Bacon-Shor code. For example, the X, Z weight distributions of all nontrivial logical Pauli operator representations in this subsystem code is shown in Fig. 21 for the 2D Bacon-Shor code of different sizes.

Note that it has a very different structure from the surface-code operator weight distribution, a likely consequence of the even weight gauge generators.

1. 2D compass code

Now we examine different instances of gauge fixed Bacon-Shor codes. For an $\ell \times \ell'$ Bacon-Shor code, let

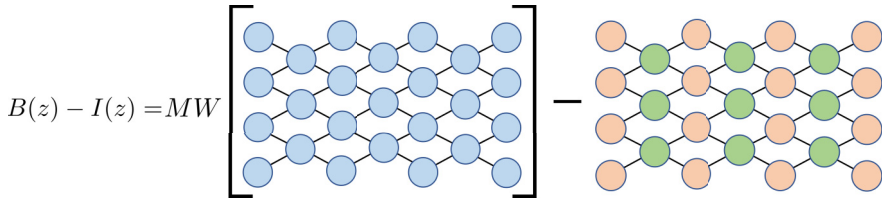

FIG. 20. Distribution of nonidentity logical operators in the 2D Bacon-Shor code, where blue tensors indicate $\mathbf{A}(z)$ of the $[\![5,1,2]\!]$ code (odd columns) and $[\![4,2,2]\!]$ codes (even columns). Green tensors are $\mathbf{A}'(z)$ of the $[\![4,1,2]\!]$ subsystem code while orange tensors are $\mathbf{B}(z)$ of the $[\![5,1,2]\!]$ codes.

us fix the XX gauge by promoting $(\ell-1)(\ell'-1)$ weight-2 X-type gauge operators to stabilizer generators. This yields a stabilizer group with $(\ell-1)+(\ell'-1)+(\ell-1)(\ell'-1)=\ell+\ell'-2+\ell\ell'-\ell-\ell'+1=\ell\ell'-1$ generators, which is a $[\![\ell\ell',1,\min(\ell,\ell')]\!]$ stabilizer code.

The tensor network for this gauge can be built from the tensor of two different repetition codes

$$W_R = |00\rangle\langle 0| + |11\rangle\langle 1|$$
$$W_B = (|00\rangle + |11\rangle)\langle 0|/\sqrt{2} + (|10\rangle + |01\rangle)\langle 1|/\sqrt{2}. \tag{6.8}$$

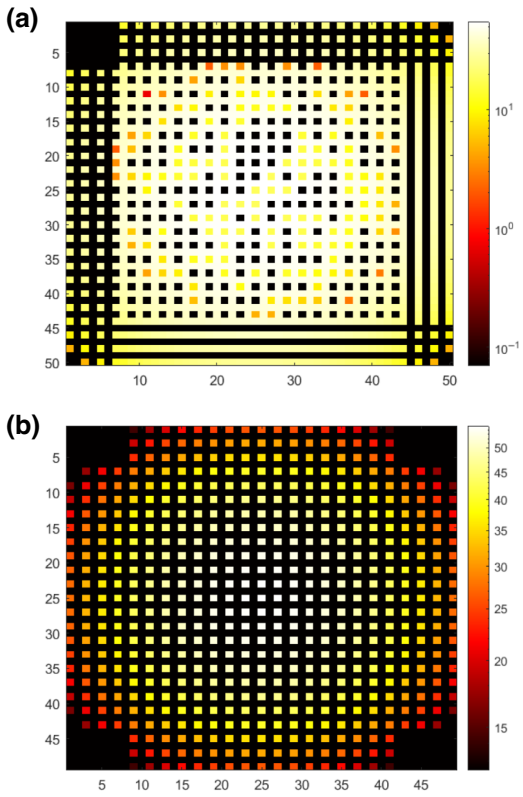

FIG. 21. Plotting $\log(C_{w_x,w_z})$ in log scale, where X and Z weights are labeled by the vertical and horizontal axes, respectively. (a) 7 by 7 Bacon-Shor code. (b) 6 by 8 Bacon-Shor code.

The code defined by W_R has stabilizer ZZ, and $\bar{X}=XX, \bar{Z}=IZ$ and the one with W_B has $X\leftrightarrow Z$ with stabilizer XX, $\bar{X}=IX, \bar{Z}=ZZ$. Their tensors are colored red and blue, respectively. The output legs of the tensors are connected into a ring while leaving the inputs dangling. This constructs tensors in the tree tensor network, Fig. 22. Each of the bigger red nodes corresponds to a stabilizer state with stabilizer group $\langle$all X, even weight $ZZ\rangle$. The same holds for the big blue nodes but with $X\leftrightarrow Z$.

Although the code has $d\sim\sqrt{n}$, the entanglement for some subsystems of size approximately d can be much weaker. This allows us to write down a more efficiently contractible tensor network by taking advantage of these low entanglement cuts [62]. The total time complexity for obtaining the enumerator is thus $O(\ell+\ell')\sim O(d)$ if $\ell\approx\ell'$.

By fixing the gauges in other ways, one produces a class of codes known as the 2D compass codes [63], which includes a gauge that reproduces the surface code and the XX (or ZZ) gauge we examined. Coincidentally, these are also two gauges of the Bacon-Shor code with the highest ($O(n\exp(\sqrt{n}))$) and lowest ($O(\sqrt{n})$) computational cost, respectively. The entanglement structure of the underlying quantum state generally depends on the gauge choice. While this speedup is not surprising, as the example can be built from code concatenation, we can estimate how cost would scale for other patterns of gauge fixings that are everywhere-in-between provided we have tensor networks whose connectivity captures the entanglement feature. Intuitively, we can roughly understand the speedup as a statement about entangled clusters. When the code is fixed in the pure XX gauge, for instance, there is little entanglement across the columns or rows of the code. If we now introduce gauge fixing such that ZZ stabilizers can occur with some nonvanishing fraction, this introduces more entanglement across these clusters and the resulting tensor-network minimal cut now has to cut through these additional bridges of entanglement. Generally, we then expect the complexity to scale exponentially as the width of these bridges, or the minimal cuts that separates these clusters. In the extreme case of the surface code, the bridges are of $\sqrt{n}$, and in the pure XX or ZZ gauge, the bridge is of $O(1)$. By slowly deforming from the XX or

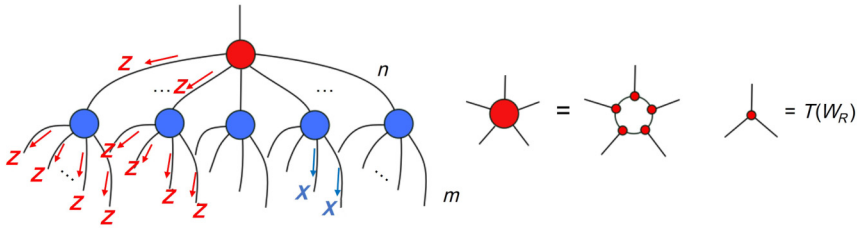

FIG. 22. Tree tensor network for a $m \times n$ Bacon-Shor code in the XX gauge. Some stabilizers are shown via operator pushing. The tensors are obtained from repetition code encoding maps.

ZZ gauge, one may also explore the intermediate regime of complexities $O(\sqrt{n}) \to \text{poly}(n) \to O(\exp(\sqrt{n}))$ [64]. A more comprehensive study of this complexity transition and gauge fixing can be an interesting subject for future exploration.

VII. DISCUSSION

In this work, we generalize the existing weight-enumerator formalism to study cosets, subsystem codes, and all single-qubit error channels. In conjunction with tensor networks, we extend their applications in quantum error correction. We show that weight enumerators can be computed more efficiently using tensor-network methods once a QL construction of the code is known. The complexity can vary depending on the tensor-network connectivity, and is dominated by the cost of tensor contractions. For a QL construction that faithfully reflects the entanglement structure of the code words, the cost for finding their enumerator is approximately $O(\exp(d))$ for nondegenerate codes and up to exponentially faster for degenerate codes. As a novel distance-finding protocol, our proposal constitutes the only and the best current algorithm for finding the distance beyond stabilizer codes. In the case of Pauli stabilizer codes, this provides a comparable performance for nondegenerate codes, and up to exponential speedup for degenerate codes.

Using the generalized coset enumerators, we also construct (optimal) decoders for all codes using weight enumerators for all i.i.d. single-qubit error channels. As a corollary, it improves the simulation accuracy when estimating fault-tolerant thresholds if used in conjunction with existing methods. Since QL includes all quantum codes, and thus stabilizer codes, the enumerator method can also be understood as a generalization of tensor-network decoders. Finally we applied our method numerically to codes with sizes of order 100 to 200 qubits, showing that it is practical to study codes of relevant sizes in near-to-intermediate term devices. We also provide novel analysis of the surface code, color code, holographic code, and the Bacon-Shor code using exact analytical expressions. These include their full operator weight distributions and certain code performance under coherent or biased noise. For the holographic code, we also present new results on asymmetric distances and the varied behavior of different bulk qubits under (biased) Pauli error.

This advance also has a wide range of applications in the context of measurement-based quantum computation quantum many-body physics. We have shown that higher genus weight enumerators computes the stabilizer Renyi entropy, or magic, of a quantum state. It is also known that Shor-Laflamme enumerator, or sector length, is a powerful tool to study the entanglement structure of cluster states. With these novel connections between coding-theoretic objects and quantum resource-theoretic quantities, our method provides a far more efficient method to characterize entanglement and magic in quantum many-body systems compared to brute-force evaluation. For the case of graph states, existing methods to compute sector lengths have been limited to order 30 qubits. Our numerics from the 2D tensor network suggests that this may be pushed to about 10 times higher with modest hardware requirements on 2D cluster states. Numerical computation of quantum many-body magic is also widely recognized as a challenging problem. Here we provide an alternative method that is readily implementable for a variety of tensor-network architectures.

We also provided a systematic method for building QL decomposition of existing quantum codes, filling the void left by our previous work [13]. In particular, our novel tensor-network construction applied to quantum LDPC codes provides a simple algorithmic method to analyze such codes from the perspective of quantum many-body systems using bounded-degree tensor networks.

A. Connection with stat mech mapping

We comment on the connection between optimal decoding and distance from the point of view of the statistical mechanical mapping and weight enumerators. Recall that the coset weight enumerator polynomial $A(\bar{E}, \mathbf{u})$ of E captures the weight distribution of all operators that are stabilizer equivalent to E. By plugging in the corresponding coefficients $\mathbf{k}$ from decomposing the error channels, one obtains the probability of incurring any errors that are equivalent to E. This is nothing but the partition function Z_E by solving the stat mech mapping [65] associated with

a noise model that satisfies the Nishimori condition for all parameters βJ_i where β is the inverse temperature and $\{J_i\}$ are coupling strengths of the model.

Conversely, if the error probability from the stat mech model can be obtained exactly, then it must agree with $A(\bar{E}, \mathbf{u})$ in some domain that is a connected region near the origin. Since if two polynomials f, g agree in an infinite number of points, $f-g$ must have an infinite number of roots. This cannot happen for any nontrivial $f-g$ because the degree $\deg(f-g) \leq \max(\deg(f), \deg(g))$ is bounded. This implies that the solution $Z_E = A(\bar{E}, \mathbf{u})$ must be unique. Therefore, by solving the stat mech model and obtaining its partition function for different values of βJ_i, we must also have sufficient information to uniquely fix the enumerator polynomial. For example, for symmetric Pauli noise, one can in principle fix the coefficients of the polynomial by computing the values of $Z_E(\beta)$ at different temperatures. As there are only finitely many coefficients for A^s, one can solve an overconstrained system of equations with integer solutions.

In practice, however, the expression for $Z_E = \Pr(\bar{E})$ in the stat mech model is often obtained numerically. Therefore, unless $P = NP$ (or $NP = RP$) the reverse process going from the stat mech output to the enumerator can only be trusted to produce the correct results only when the values of $\Pr(\bar{E})$ hold to exponential accuracy generally. This is expected, because otherwise one can solve the minimal distance problem approximately [66] using the stat mech model in polynomial time with approximate tensor contraction. In instances where the partition functions can be (or have been) obtained with relatively high accuracy such that the cost is less expensive compared to the current enumerator method, one can also acquire polynomially many values of the partition functions at different coupling strengths. One can then fit the coefficients of the enumerator polynomial to these data points. This allows us to derive (an approximation of) the enumerator and thus also extrapolat the error probabilities to other regimes instead of evaluating those points individually using the stat mech model.

B. Future directions

Recently, it was shown that asymptotically good quantum LDPC codes like Ref. [67] have a circuit depth lower bound that is $\log n$. Since these codes are highly degenerate and some may sustain linear distances even with a much lower entanglement along some cuts, it is possible that a good tensor-network description may lead to a more efficient distance-verification protocol for codes whose code words saturate the entanglement lower bound. However, we also note that small-sized examples, their tensor-network descriptions, and a tight entanglement lower bound are still open problems as of the time of writing, the advantage our method provides only remains a theoretical possibility [68]. Therefore, a general QL recipe for building quantum LDPC codes would be useful.

As weight enumerators are applicable for non-(Pauli)-stabilizer codes, they can be used to study or search for such codes while providing crucial information on their distances. This would extend the examples in this work beyond stabilizer codes and would also have relevant applications in optimization-based methods that need not produce stabilizer codes [24]. For example, XS or XP codes [37,38] do not have Abelian stabilizer groups and currently lack a protocol for computing their code distances. However, for general codes, reduced enumerators are likely insufficient, and a higher bond dimension will be needed.

Note that beyond QECC literature, Shor-Laflamme enumerators, also known as sector lengths in graph states [11,69], have been used to study the structure as well as the robustness of entanglement in entangled resource states. Sector lengths of graph states are difficult to compute using the brute-force method. Given our tensor-network decomposition of all graph states, we expect our method to carry immediate impact in the analysis of 2D or planar cluster states with >100 qubits using sector lengths as well as more general applications in the context of fault-tolerant resource state preparation for measurement and fusion-based quantum computations and quantum networks.

In the context of quantum many-body magic, nonstabilizerness has been difficult to compute numerically. As stabilizer Renyi entropy and other measures have been related to quantum chaos, entanglement spectrum and the emergence of gravity in AdS/CFT correspondence, it is interesting to explore whether the tensor-network methods based on enumerators are advantageous for more efficient magic computations. It is also intriguing to understand whether the quantum MacWilliams identity can provide important constraints for quantum many-body entanglement and magic.

Tensor-enumerator methods are also useful when used in conjunction with machine-learning (especially reinforcement-learning)-based methods for QECC search [26,70]. As one would typically need to evaluate certain code properties, such as distance, that are resource intensive, the tensor enumerator method can be used to drastically decrease the time needed to evaluate the cost function. It is also of interest to study the effect of approximate tensor contractions and how they impact the accuracy of the weight distribution and related distance information.

While we have treated all i.i.d. single-qubit errors, the current formalism does not tackle general location-based or correlated error efficiently. For the former, a straightforward extension exists where one can either introduce an additional variable for each location that has independent error pattern. This remains efficient as long as the types of distinct error channels is small, but can quickly become intractible if it scales with the system size. Alternatively,

one can precontract the tensor with a fixed error parameter $\{p_i\}$ instead of describing them as variables. The latter reduces to a more general tensor-network decoder [21,71–74]. In particular, [74] shows that correlated errors can be efficiently studied using PEPO with approximate tensor contraction. Both correlated noise and approximate contraction schemes can be interesting future avenues of research in the context of tensor weight enumerators. In the same vein, further extension is needed to describe fault-tolerant processes, which are fundamentally dynamical. Therefore, an enumerator framework compatible with space-time quantum error correction that incorporates gadgets that includes measurement errors, midcircuit noise and POVMs will be needed.

Finally, while enumerators were first defined in classical coding theory, one yet needs an efficient method to compute them for classical codes. Therefore, it is natural to extend the current QL-based approach to classical codes and compute their weight-enumerator polynomials. Such tasks may be accomplished by directly applying the current formalism for classical codes and rephrasing them as quantum stabilizer codes with trivial generators, or devising a more efficient method that performs the analog of the trace or conjoining operation for classical codes.

ACKNOWLEDGMENTS

We thank Y.D. Li, D. Miller, G. Sommers, and Y.J. Zou for helpful discussions and comments on the manuscript. C.C. acknowledges the support by the U.S. Department of Defense and NIST through the Hartree Postdoctoral Fellowship at QuICS, the Air Force Office of Scientific Research (Grant No. FA9550-19-1-0360), and the National Science Foundation (Grant No. PHY-1733907). M.J.G. acknowledges support from the National Science Foundation (QLCI Grant No. OMA-2120757). The Institute for Quantum Information and Matter is an NSF Physics Frontiers Center. Certain commercial equipment, instruments, or materials are identified in this paper in order to specify the procedure adequately and do not reflect any endorsement by NIST.

APPENDIX A: COMMON SCALAR ENUMERATORS

For completeness, we review a few examples below that we we have used in this work.

1. Shor-Laflamme weight enumerator

The original weight enumerators [75,76] are important objects in classical coding theory. Their quantum counterparts were introduced by Shor and Laflamme [4], which capture some key properties of an error-correcting code. They feature a duo of polynomials that take the forms of

$$A(z,w)=\sum_{d=0}^{n}A_d(M_1,M_2)z^d w^{n-d}, \tag{A1}$$

$$B(z,w)=\sum_{d=0}^{n}B_d(M_1,M_2)z^d w^{n-d}, \tag{A2}$$

where

$$A_d(M_1,M_2)=\sum_{E\in\mathcal{E}[d]}\mathrm{Tr}(EM_1)\,\mathrm{Tr}(EM_2), \quad \text{and} \tag{A3}$$

$$B_d(M_1,M_2)=\sum_{E\in\mathcal{E}[d]}\mathrm{Tr}(EM_1EM_2) \tag{A4}$$

for some Hermitian M_1, M_2, and $\mathcal{E}[d]$, which denotes unitary errors of weight d. Here without loss of generality we can simply choose the Pauli basis. Note that they are a special case of the abstract enumerator [14], and we may recover them by setting $\mathbf{u}=(w,z)$ and

$$\mathrm{wt}(E)=\begin{cases}(1,0) & \text{if } E=I\\ (0,1) & \text{otherwise.}\end{cases} \tag{A5}$$

So that $\mathbf{u}^{\mathrm{wt}(E)}=w^{n-\mathrm{wt}(E)}z^{\mathrm{wt}(E)}$, where $\mathrm{wt}(E)$ is simply the operator weight of the Pauli string E.

These polynomials are related by the MacWilliams identity

$$B(w,z)=A\left(\frac{w+(q^2-1)z}{q},\frac{w-z}{q}\right). \tag{A6}$$

Therefore, it is sufficient to obtain one of them, and perform MacWilliams transform to get the other. In practice, for a brute-force algorithm, it is often easier to recover $A(z,w)$.

Note that these polynomials are sometimes expressed in the inhomogeneous form where $A(z)=A(w=1,z), B(z)=B(w=1,z)$. As it is simple to recover the homogenized form by setting $A(z)\to w^nA(z/w)$ and similarly for B, we refer to both of them weight enumerators as they contain the same information as encoded by the coefficients.

2. Refined enumerators

We can also consider a generalization of the Shor-Laflamme polynomial (A1) where we separate the weights by type [9]. One such example is the double weight

enumerator. Using variables $\mathbf{u} = (w, x, y, z)$

$$\mathrm{wt}(E) = \begin{cases} (0,1,0,1) & \text{if } E = I \\ (0,0,1,1) & \text{if } E = X \\ (1,0,1,0) & \text{if } E = Y \\ (1,1,0,0) & \text{if } E = Z \end{cases}. \tag{A7}$$

This is useful when, for instance, we consider a biased error model where bit flip (X) and phase (Z) errors occur independently with different probabilities. Depending on the form of the biased Pauli noise, other weight functions may be used for the weight function. Such double enumerators may be used as long as the biased Pauli noise only admits two independent physical error parameters. The polynomials are

$$D(x,y,z,w;M_1.M_2) = \sum_{w_x,w_z}^{n} D_{w_x,w_z} y^{w_x} w^{w_z} x^{n-w_x} z^{n-w_z}, \tag{A8}$$

$$D^{\perp}(x,y,z,w;M_1,M_2) = \sum_{w_x,w_z}^{n} D^{\perp}_{w_x,w_z} y^{w_x} w^{w_z} x^{n-w_x} z^{n-w_z}, \tag{A9}$$

where

$$D_{w_x,w_z} = \sum_{E\in\mathcal{E}[w_x,w_z]} \mathrm{Tr}[EM_1]\,\mathrm{Tr}[E^{\dagger}M_2], \tag{A10}$$

$$D^{\perp}_{w_x,w_z} = \sum_{E\in\mathcal{E}[w_x,w_z]} \mathrm{Tr}[EM_1E^{\dagger}M_2], \tag{A11}$$

and $\mathcal{E}[w_x, w_z]$ is the set of Paulis that have X and Z weights w_x, w_z, respectively.

The MacWilliams identity was derived in Ref. [9] for local dimension 2 where $M_1 = M_2$ are projection operators onto the code subspace. In Ref. [14], it was extended arbitrary local dimension q and M_1, M_2. We reproduced the relation here for convenience

$$D^{\perp}(x,y,z,w) = D\left(\frac{x+(q-1)y}{\sqrt{q}}, \frac{z-w}{\sqrt{q}}, \frac{z+(q-1)w}{\sqrt{q}}, \frac{x-y}{\sqrt{q}}\right). \tag{A12}$$

The inhomogeneous forms are

$$D(y,w) = \sum_{w_x,w_z}^{n} D_{w_x,w_z} y^{w_x} w^{w_z}, \tag{A13}$$

$$D^{\perp}(y,w) = \sum_{w_x,w_z}^{n} D^{\perp}_{w_x,w_z} y^{w_x} w^{w_z}. \tag{A14}$$

One can easily restore the x, z dependence as their powers are fixed by n, w_x, w_z.

Theorem 6. If t_x, t_z are the two largest integers such that $D_{w_x,w_z} = D^{\perp}_{w_x,w_z}$ for $w_x < t_x, w_z < t_z$, then $d_x = t_x, d_z = t_z$.

Proof. See Theorem 8 of Ref. [9]. ■

An even more refined weight function distinguish all the Pauli operators by their types

$$\mathrm{wt}(E) = \begin{cases} (1,0,0,0) & \text{if } E = I \\ (0,1,0,0) & \text{if } E = X \\ (0,0,1,0) & \text{if } E = Y \\ (0,0,0,1) & \text{if } E = Z \end{cases}. \tag{A15}$$

This is known as the complete weight enumerator [9]. Again, let $\mathbf{u} = (w, x, y, z)$

$$E(x,y,z,w;M_1,M_2) = \sum_{w_x,w_y,w_z} E_{w_x,w_y,w_z} x^{w_x} y^{w_y} z^{w_z} w^{n-w_x-w_y-w_z} \tag{A16}$$

$$F(x,y,z,w;M_1,M_2) = \sum_{w_x,w_y,w_z} F_{w_x,w_y,w_z} x^{w_x} y^{w_y} z^{w_z} w^{n-w_x-w_y-w_z}, \tag{A17}$$

where

$$E_{w_x,w_y,w_z} = \sum_{Q\in\mathcal{E}[w_x,w_y,w_z]} \mathrm{Tr}[QM_1]\,\mathrm{Tr}[Q^{\dagger}M_2]\mathbf{u}^{\mathrm{wt}(Q)}, \tag{A18}$$

$$F_{w_x,w_y,w_z} = \sum_{Q\in\mathcal{E}[w_x,w_y,w_z]} \mathrm{Tr}[QM_1Q^{\dagger}M_2]\mathbf{u}^{\mathrm{wt}(Q)}, \tag{A19}$$

and $\mathcal{E}[w_x, w_y, w_z]$ are the Pauli operators with those X, Y, and Z weights, respectively. See Ref. [14] for general MacWilliams identities at any q.

3. Applications to stabilizer codes

Before we move on to tensor enumerators, let us build up some intuition as to what these polynomials are enumerating. Let us examine a special case where we set $M_1 = M_2 = \Pi$ to be the projection onto the code subspace of a quantum code. Furthermore, let us suppose that this is a $[\![n, k]\!]$ stabilizer code, meaning that

$$\Pi = \frac{1}{2^{n-k}} \sum_{S\in\mathcal{S}} S. \tag{A20}$$

It is clear that $\mathrm{Tr}[E\Pi] \neq 0$ if and only if $E \in \mathcal{S}$ is a stabilizer element and $\mathrm{Tr}[E\Pi E^{\dagger}\Pi] \neq 0$ if and only if $E \in \mathcal{N}(\mathcal{S})$ is a normalizer element. Therefore, we see that, up to a constant normalization factor, the coefficients A_d of $A(z;\Pi,\Pi)$ is simply enumerating the number of stabilizer elements with weight d and B_d enumerating the number of logical operators with weight d. Consequently, $\sum_d A_d = 2^{n-k}$ and $\sum_d B_d = 2^{n+k}$ for a $[\![n, k]\!]$ code.

For the refined enumerators, the coefficients of the double enumerator $D, D^{\perp}$ are simply recording the number of stabilizer and normalizer elements that have X, Z weights (w_x, w_z). Similarly, the complete enumerator coefficients $E_{w_x,w_y,w_z}, F_{w_x,w_y,w_z}$ count the elements with those corresponding X, Y, and Z weights.

One can also set M_1, M_2 to different operators to extract different information about the code. For example, in coset enumerators, A_d^s counts the number of coset elements with a particular weight.

APPENDIX B: INSTANCES OF TENSOR ENUMERATORS

We have seen previously particular instances of tensor enumerators with $\mathbf{u} = z$. One can extend examples in the main text to other enumerators, which we have used to study other error models.

1. Refined tensor enumerators

Similar to the scalar forms, we apply $\mathbf{u} = (w, x, y, z)$ and the weight function Eqs. (A7) to (8). The tensor coefficients are

$$\begin{aligned} D^{(J)}_{w_x,w_z}(E,\bar{E};M_1,M_2) &= \sum_{F\in\mathcal{E}^{n-m}[w_x,w_z]} \mathrm{Tr}((E\otimes_J F)M_1)\,\mathrm{Tr}((\bar{E}\otimes_J F)^{\dagger}M_2), \\ D^{\perp(J)}_{w_x,w_z}(E,\bar{E};M_1,M_2) &= \sum_{F\in\mathcal{E}^{n-m}[w_x,w_z]} \mathrm{Tr}((E\otimes_J F)M_1(\bar{E}^{\dagger}\otimes_J F^{\dagger})M_2), \end{aligned} \tag{B1}$$

where $J \subseteq \{1, 2, \ldots n\}$ are the locations of open legs in the tensor enumerator. As in the main text, $\otimes_J$ denotes the operation where we insert E s at corresponding positions of J indices to form a n-qubit Pauli string with F which has length $n - m$ for m open indices.

For complete tensor enumerators, we replace $\mathcal{E}[w_x, w_z] \to \mathcal{E}[d_x, d_y, d_z]$ and

$$\begin{aligned} D^{(J)}_{w_x,w_z}(E,\bar{E},M_1,M_2) &\to E^{(J)}_{d_x,d_y,d_z}(E,\bar{E},M_1,M_2), \\ D^{\perp(J)}_{w_x,w_z}(E,\bar{E},M_1,M_2) &\to F^{(J)}_{d_x,d_y,d_z}(E,\bar{E},M_1,M_2) \end{aligned} \tag{B2}$$

in Eq. (B1). $\mathcal{E}[d_x, d_y, d_z]$ is the set of Pauli operators with X, Y, Z weights given by d_x, d_y, d_z, respectively.

2. Generalized tensor enumerators

For the most general noise model, it is also useful to define generalized abstract enumerator

$$\tilde{\mathbf{A}}^{(J)}(\mathbf{u};M_1,M_2) = \sum_{E,\bar{E}\in\mathcal{E}^m}\sum_{F,\bar{F}\in\mathcal{E}^{n-m}} \mathrm{Tr}((E\otimes_J F)M_1)\,\mathrm{Tr}((\bar{E}^{\dagger}\otimes_J \bar{F}^{\dagger})M_2)\mathbf{u}^{\mathrm{wt}(F,\bar{F})}e_{E,\bar{E}}, \tag{B3}$$

$$\tilde{\mathbf{B}}^{(J)}(\mathbf{u};M_1,M_2) = \sum_{E,\bar{E}\in\mathcal{E}^m}\sum_{F\in\mathcal{E}^{n-m}} \mathrm{Tr}((E\otimes_J F)M_1(\bar{E}^{\dagger}\otimes_J \bar{F}^{\dagger})M_2)\mathbf{u}^{\mathrm{wt}(F,\bar{F})}e_{E,\bar{E}}, \tag{B4}$$

where the forms are similar to the conventional tensor enumerator but the sum and weight function now depend on two independent variables $F, \bar{F}$. These are useful for computing generalized scalar weight enumerators (Sec. II B), which finds applications in noise models such as coherent noise or amplitude damping channel (Sec. III B).

Theorem 7. Suppose $j, k \in J \subset \{1, \ldots, n\}$. Then

$$\wedge_{j,k}\tilde{\mathbf{A}}^{(J)}(\mathbf{u};M_1,M_2) = \tilde{\mathbf{A}}^{(J\setminus\{j,k\})}(\mathbf{u};\wedge_{j,k}M_1,\wedge_{j,k}M_2) \tag{B5}$$

and similarly for $\tilde{\mathbf{B}}$.

Proof.

$$\begin{aligned}
\wedge_{jk}\tilde{\mathbf{A}}^{(J)}(\mathbf{u};M_1,M_2) &= \sum_{E,\bar{E},F,\bar{F}} \left[\mathrm{Tr}((E\otimes_J F)M_1)\,\mathrm{Tr}((\bar{E}\otimes_J \bar{F})^\dagger M_2\right]\mathbf{u}^{\mathrm{wt}(F,\bar{F})}\left[\wedge_{j,k} e_{E,\bar{E}}\right] \\
&= \sum_{F,\bar{F}} \sum_{\substack{E\setminus\{E_j,E_k\},\\ \bar{E}\setminus\{\bar{E}_j,\bar{E}_k\}}} \sum_{G} \Big\{ \mathrm{Tr}([((G\otimes G^*)\otimes_{j,k} E\setminus\{E_j,E_k\})\otimes_J F]M_1) \\
&\quad \times \sum_{\bar{G}} \mathrm{Tr}([((\bar{G}\otimes \bar{G}^*)\otimes_{j,k}\bar{E}\setminus\{\bar{E}_j,\bar{E}_k\})\otimes_J \bar{F}]^\dagger M_2)\Big\}\mathbf{u}^{\mathrm{wt}(F,\bar{F})} e_{E\setminus\{E_j,E_k\},\bar{E}\setminus\{\bar{E}_j,\bar{E}_k\}} \\
&= \sum_{E',\bar{E}',F} \mathrm{Tr}((E'\otimes_{J\setminus\{j,k\}} F)(|\beta\rangle\langle\beta|_{j,k}M_1))]\,\mathrm{Tr}((\bar{E}'\otimes_{J\setminus\{j,k\}}\bar{F})^\dagger(|\beta\rangle\langle\beta|_{j,k}M_2))\mathbf{u}^{\mathrm{wt}(F,\bar{F})}e_{E',\bar{E}'} \\
&= \sum_{E',\bar{E}',F} \mathrm{Tr}((E'\otimes_{J\setminus\{j,k\}} F)(\wedge_{j,k}M_1))\,\mathrm{Tr}((\bar{E}'\otimes_{J\setminus\{j,k\}}\bar{F})^\dagger(\wedge_{j,k}M_2))\mathbf{u}^{\mathrm{wt}(F,\bar{F})}e_{E',\bar{E}'} \\
&= \tilde{\mathbf{A}}^{(J\setminus\{j,k\})}(\mathbf{u};\wedge_{jk}M_1,\wedge_{jk}M_2),
\end{aligned} \tag{B6}$$

where the wedge acts on the vector basis in the usual way.

We used the fact that

$$|\beta\rangle\langle\beta| = \frac{1}{q}\sum_{P\in\mathcal{P}} P\otimes P^*. \tag{B7}$$

Similarly, we can repeat this argument for B-type generalized enumerators. We do not use MacWilliams identity for this proof as we have not been able to identify any. ■

Note that it is often possible to cut down the computational cost when the weight function satisfies the form

$$\mathbf{u}^{\mathrm{wt}(E,F)} = \mathbf{u}_1^{\mathrm{wt}(E)}\mathbf{u}_2^{\mathrm{wt}(F)}. \tag{B8}$$

Then we can write the generalized enumerator as

$$\bar{A}(\mathbf{u};M_1,M_2) = \sum_{E\in\mathcal{E}^n}\mathrm{Tr}[EM_1]\mathbf{u}_1^{\mathrm{wt}(E)}\sum_{F\in\mathcal{E}^n}\mathrm{Tr}[F^\dagger M_2]\mathbf{u}_2^{\mathrm{wt}(F)} \tag{B9}$$

that factorize into two separate sums such that each piece can be computed separately. For each M_i, we can rewrite as a tensor network. This allows us to compute either sum using a tensor network of $\chi = q^2$ by tracing reduced tensor enumerators. We see that for stabilizer codes, the coefficients for each term are identical to the usual A-type scalar weight enumerator up to a constant factor normalization.

3. Stabilizer codes and reduced enumerators

Again, let us come back to stabilizer codes for intuition behind these constructions. Consider the reduced tensor enumerator polynomial with open indices $J = \{j_1,\ldots,j_m\}$ where we set $M_1 = M_2 = \Pi$ to be the projection onto stabilizer code subspace. We see that each coefficient $A_d^{j_1,\ldots,j_m}$ simply enumerates, up to a constant normalization, the number of stabilizer elements that has Pauli string $\sigma^{(j_1)}\otimes\ldots\otimes\sigma^{(j_m)}$ on the first through m th qubit and qudit and has weight d on the remaining qubit and qudits. Similarly for the reduced double, complete, and the generalized enumerators, the same intuition applies, except the weights are separated and recorded according to the types of the Pauli operators.

The tracing of reduced enumerators for stabilizer codes can be understood as a simple consequence of operator matching. Recall that stabilizers and logical operators in the QL construction come from matching such operators on the smaller tensors. Since the tensor enumerator is counting the number of stabilizers with weight d and a particular Pauli type on the open legs, tracing it with another tensor enumerator retains precisely the weight distribution of Pauli elements that are matching on the legs being glued. This in turn produces the desired weight distribution of the larger tensor network. Although Theorem 2 provides a construction that is sufficient for building weight enumerator of any quantum code, the above intuition suggests that the reduced enumerators are sufficient for stabilizer codes, which allows us to reduce the bond dimension from q^4 to q^2.

Definition 3. A diagonal trace is defined by

$$\wedge^{DT}_{j,k} e_{E,\bar{E}} = \begin{cases} e_{E\setminus\{E_j,E_k\},\bar{E}\setminus\{\bar{E}_j,\bar{E}_k\}} & \text{if } E_j = E_k^* = \bar{E}_j = \bar{E}_k^* \\ 0 & \text{otherwise.} \end{cases} \tag{B10}$$

Proposition 3. Suppose

$$M_1 = \frac{1}{|S|}\sum_{S\in P\mathcal{S}} S, \tag{B11}$$

$$M_2 = \frac{1}{|S|}\sum_{S\in P\mathcal{S}} \omega_S S \tag{B12}$$

for any coset $P\mathcal{S}$ of Pauli operator P and $\omega_S \in \mathbb{C}$. Let $\wedge_{\text{all}}$ be the set of self-contractions that reduce an even rank tensor enumerator to a scalar, then

$$\wedge^{DT}_{\text{all}} \mathbf{A}^J(\mathbf{u}; M_1, M_2) \propto A(\mathbf{u}; \wedge_{\text{all}} M_1, \wedge_{\text{all}} M_2) \tag{B13}$$

and similarly for $\mathbf{B}$. The same holds if the forms of M_1, M_2 are switched.

Proof. The proof is similar to Theorem 7.1 of Ref. [14]. Let us begin with the case where there are only two open legs in the tensor enumerator. It is clear that

$$\text{Tr}[(G\otimes G^*\otimes F)M_i] \neq 0 \tag{B14}$$

if and only if $G\otimes G^*\otimes F$ is a coset element.

Suppose $\mathcal{G}$ is the set of all G s for which the trace Eq. (B14) is nonzero, then

$$\sum_{G,\bar{G}\in\mathcal{E}} \text{Tr}[(G\otimes G^*\otimes F)M_1]\,\text{Tr}[(\bar{G}\otimes\bar{G}^*\otimes F)^\dagger M_2] = |\mathcal{G}|\sum_G \text{Tr}[(G\otimes G^*\otimes F)M_1]\,\text{Tr}[(G\otimes G^*\otimes F)^\dagger M_2], \tag{B15}$$

which is proportional to the diagonal trace $\wedge^{DT}$. We can see this by the following. For each $G\in\mathcal{G}$, we sum over $\bar{G}$, which leads to some constant $\propto \sum_S \omega_S$. Repeating for each G, we simply get back the same constant $|\mathcal{G}|$ times. If we only sum over the diagonal terms with $G=\bar{G}$, then we obtain the constant $\propto \sum_S \omega_S$ once. This only works because one of M_1, M_2 is an equal superposition of Pauli operators.

Furthermore, note that for the F in Eq. (B15), each $\bar{G}\in\mathcal{G}, \bar{G}=PG, P\neq I$, it is clear that $P\otimes P^*$ is a stabilizer of the code. Therefore, for any other F such that $G\otimes G^*\otimes F\in P\mathcal{S}$, it must follow that $(P\otimes P^*\otimes I)(G\otimes G^*\otimes F) = \bar{G}\otimes\bar{G}^*\otimes F\in P\mathcal{S}$ for each $\bar{G}\in\mathcal{G}$. Therefore, the overcounting is identical for all F s by a factor of $|\mathcal{G}|$. Therefore, the diagonal elements contain sufficient information to reproduce the scalar enumerator.

For any tensor enumerator with four open legs that needs two self-traces on two pairs a_0 and a_1. From the above arguments we know that a full trace on a_1 followed by diagonal trace on a_0 produces the correct scalar enumerator. Therefore, it is sufficient to show that a diagonal trace on a_1 produce the correct diagonal elements for the pair a_0. Let E denote the Pauli for open legs associated with pair a_0. Under a full trace on a_1, the diagonal elements of the remaining tensor then come from coefficients of the form

$$\sum_{G,\bar{G}\in\mathcal{E}} \text{Tr}[(G\otimes G^*\otimes E\otimes F)M_1] \times \text{Tr}[(\bar{G}\otimes\bar{G}^*\otimes E\otimes F)^\dagger M_2], \tag{B16}$$

where the sum comes from tracing over the legs of a_1. We notice that the same argument above applies by setting $E\otimes F\to F$ since F is arbitrary. Hence we conclude that the full trace on a_1 produce the same diagonal elements on a_0 as a diagonal trace up to a constant multiple. Proceed inductively with $2k$ open legs, it is clear that the diagonal components are sufficient for generating the scalar weight enumerators.

To show that the B type enumerator is also correctly produced via diagonal trace, recall that diagonal trace is linear and commutes with the generalized Wigner transform as shown in the proof of Prop. VI.1, it can also be generated with only diagonal trace operations. ■

Therefore, for practical analysis of Pauli stabilizer codes, we only need to consider the reduced tensor enumerators, that is, restricting to the diagonal elements $E=\bar{E}$ of each tensor enumerator in Definition 4.2 of Ref. [14].

The same proof does not apply for tracing generalized enumerators $\tilde{\mathbf{A}}, \tilde{\mathbf{B}}$ because two separate sums are required separately for F and $\bar{F}$ whereas the argument is valid only when $F=\bar{F}$. Therefore, we have to perform a full tensor trace even for stabilizer codes. Such is needed to analyze more general error channels like coherent noise.

APPENDIX C: TENSOR-ONLY IMPLEMENTATION

1. Multilinear formulation

Although the enumerator polynomials can be implemented symbolically, it is also possible to rephrase them purely as tensors with complex coefficients.

For each tensor enumerator, one can take the coefficients in the polynomial, as a tensorial object by itself. For example, for Shor-Laflamme enumerators, coefficient A_d in the scalar enumerator can be treated as a rank-1 tensor with bond dimension $\leq n+1$. Similarly, A^j_d in vector enumerator has two indices where j marks a bond dimension q^4 index (or q^2 in reduced enumerators) and d has bond dimension $\leq n+1$. Generally, $a(n)$ (abstract) tensor enumerators can be represented by the tensor components

$$A^{j_1,\ldots,j_i}_{d_{\mathbf{u}}} \quad \text{and} \quad B^{j_1,\ldots,j_i}_{d_{\mathbf{u}}},$$

where $d_{\mathbf{u}}$ can be l tuple of indices that tracks the powers of the monomials. For instance, for complete enumerator, $d_{\mathbf{u}} \to (d_x, d_y, d_z)$. For now, let us focus on reduced enumerators over $q=2$ where the upper and lower indices carry no additional physical meaning. To avoid clutter, let us also drop the subscript of $d_{\mathbf{u}}$. For concreteness one can take d to be the usual operator weight, but it is straightforward to restore it to the most general form.

In a tensor network, instead of tracing the tensor enumerator polynomials, we now trace together these tensors. However, we need an additional operation on the two legs d_1, d_2 that add the powers of the monomials during polynomial multiplication.

$$A^{j_{l+1},\ldots,j_i,r_{l+1},r_k}_{d} = \sum_{d_1,d_2}^{n_1,n_2} M^{d_1 d_2}_{d} \sum_{j_1,j_2,\ldots,j_l} A^{j_1,j_2,\ldots,j_l,\ldots,j_i}_{d_1} A^{j_1,j_2,\ldots,j_l,r_{l+1}\ldots r_k}_{d_2}, \tag{C1}$$

where $M^{d_1 d_2}_d$ is a tensor such that

$$M^{d_1 d_2}_d = \begin{cases} 1 & \text{if } d = d_1 + d_2 \\ 0 & \text{else.} \end{cases} \tag{C2}$$

On the formalism level, this trace with M tensor can be completed at any time. In practice, however, we perform such an operation every time a tensor trace like Eq. (C1) is completed.

The modified trace operation $\tilde{\mathrm{Tr}}$ that reduces the tensor rank can also be performed by contracting another rank-3 tensor

$$T_{d'dj} = \begin{cases} \delta_{d'd} & \text{if } j=0 \\ \delta_{d'd-1} & \text{else.} \end{cases} \tag{C3}$$

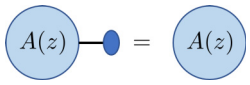

FIG. 23. Modified trace operation.

For example, to recover the scalar enumerator from a vector enumerator (Fig. 23), we use $A_{d'} = T_{d'dj} A^j_d$, where repeated indices are summed over.

The method for tracing other tensor enumerators, such as the double and complete tensor weight enumerators, is largely the same.

For example, the contraction of two reduced double enumerator is

$$D^{j_{l+1},\ldots,j_i,r_{l+1},r_k}_{d^x,d^z} = M^{d^x_1 d^x_2}_{d^x} M^{d^z_1 d^z_2}_{d^z} D^{j_1,j_2,\ldots,j_l,\ldots j_i}_{d^x_1,d^z_1} D^{j_1,j_2,\ldots,j_l,r_{l+1}\ldots r_k}_{d^x_2,d^z_2}, \tag{C4}$$

where repeated indices are summed. The modified trace is

$$D_{d'_x,d'_z} = \sum_{d_x,d_z,j} T_{d'_x,d'_z,d_x,d_z,j} D^j_{d_x,d_z}, \tag{C5}$$

with

$$T_{d'_x d'_z d_x d_z j} = \begin{cases} \delta_{d'_x d_x}\delta_{d'_z d_z} & \text{if } j=0 \\ \delta_{d'_x-1 d_x}\delta_{d'_z d_z} & \text{if } j=1 \\ \delta_{d'_x-1 d_x}\delta_{d'_z-1 d_z} & \text{if } j=2 \\ \delta_{d'_x d_x}\delta_{d'_z-1 d_z} & \text{if } j=3 \end{cases}. \tag{C6}$$

If $\mathbf{u}$ carries more variables, then an additional M contraction is needed for each separate variable index [77].

For the full tensor enumerator polynomial, one has to take extra care of potential sign changes where we have $I \leftrightarrow I, X \leftrightarrow X, Z \leftrightarrow Z$ but $-Y \leftrightarrow Y$ matchings. Suppressing the d index for now, we can think of each tensor enumerator index in a representation $A^j \to A^{\alpha,\bar{\alpha}}$ with $\alpha = 1,\ldots,q^2=4$. Furthermore, we prepare the Minkowski metric $\eta_{\alpha,\beta} = \mathrm{diag}(1,1,-1,1)$ so that tensor contractions are only performed between upper and lower indices. Indices are raised and lowered in the usual way with $A_{\beta\bar{\beta}} = \eta_{\alpha\beta}\eta_{\bar{\alpha}\bar{\beta}} A^{\alpha\bar{\alpha}}$ and contracting the two vector enumerators is by contracting the covariant vector with the contravariant one, i.e., $A^{\alpha\bar{\alpha}} A'_{\alpha\bar{\alpha}}$. We see the raised or lowered index does not matter for reduced enumerators because the diagonal elements for $\eta_{\alpha\beta}\eta_{\bar{\alpha}\bar{\beta}}$ at $\alpha=\bar{\alpha}, \beta=\bar{\beta}$ only carry positive signs.

2. MacWilliams identity as a linear transformation

We derive the matrix representation of MacWilliams identities in the polynomial basis $\{z^d w^{n-d} : 0 \leq d \leq n\}$ to facilitate MATLAB numerics.

By Corollary 5 of Ref. [5],

$$\begin{aligned} A'(w,z) &= A((w+z)/q, z/q), \\ B'(w,z) &= B((w+z)/q, z/q). \end{aligned} \tag{C7}$$

By Theorem 3 of Ref. [5], $A'(w,z) = B'(z,w)$ and $A'_d = B'_{n-d}$, which is equivalent to the quantum MacWilliams identity (Theorem 7 of Ref. [5]):

$$B(w,z) = A\left(\frac{w+(q^2-1)z}{q}, \frac{w-z}{q}\right). \tag{C8}$$

We chose to work with A' and B' because the relation $A'_d = B'_{n-d}$ can be easily expressed by an antidiagonal matrix with every element equal to 1 in the polynomial basis $\{z^d w^{n-d} : 0 \le d \le n\}$.

To express B_d in terms of A_d, we only need to express A_d in terms of A'_d, and B'_d in terms of B_d. By Corollary 5 of [5],

$$\begin{aligned} A'(w,z) &= \sum_{d=0}^{n} A_d \left(\frac{z}{q}\right)^d \left(w+\frac{z}{q}\right)^{n-d} \\ &= \sum_{d=0}^{n} A_d \left(\frac{z}{q}\right)^d \left(\sum_{e=0}^{n-d} \binom{n-d}{e} \left(\frac{z}{q}\right)^{n-d-e} w^e\right) \\ &= \sum_{d=0}^{n} \sum_{e=0}^{n-d} A_d \binom{n-d}{e} \left(\frac{z}{q}\right)^d \left(\frac{z}{q}\right)^{n-d-e} w^e \\ &= \sum_{d=0}^{n} \sum_{e=0}^{n-d} A_d \binom{n-d}{e} \left(\frac{z}{q}\right)^{n-e} w^e \\ &= \sum_{e=0}^{n} \sum_{d=0}^{n-e} A_d \binom{n-d}{e} \left(\frac{z}{q}\right)^{n-e} w^e \\ &= \sum_{e=0}^{n} \left(\sum_{d=0}^{n-e} A_d \binom{n-d}{e}\right) \left(\frac{z}{q}\right)^{n-e} w^e \\ &= \sum_{e'=0}^{n} \left(\sum_{d=0}^{e'} A_d \binom{n-d}{n-e'}\right) \left(\frac{z}{q}\right)^{e'} w^{n-e'} \\ &= \sum_{d=0}^{n} \left(\frac{1}{q^d} \sum_{m=0}^{d} A_m \binom{n-m}{n-d}\right) z^d w^{n-d}. \end{aligned} \tag{C9}$$

Since

$$A'(w,z) = \sum_{d=0}^{n} A'_d z^d w^{n-d}, \tag{C10}$$

we have

$$A'_d = \frac{1}{q^d} \sum_{m=0}^{d} A_m \binom{n-m}{n-d}, \quad 0 \le d \le n. \tag{C11}$$

In other words,

$$A'_d = \sum_{m=0}^{d} T_{dm} A_m, \tag{C12}$$

where

$$T_{dm} = \frac{1}{q^d} \binom{n-m}{n-d}, \quad 0 \le m \le d,\ 0 \le d \le n. \tag{C13}$$

Similarly, $B'(w,z) = B(w+z/q, z/q)$ implies that

$$B'_d = \sum_{m=0}^{d} T_{dm} B_m. \tag{C14}$$

Hence

$$A_d = \sum_{d'=0}^{n} (T^{-1}JT)_{dd'} B_{d'}, \quad 0 \le d,\ d' \le n, \tag{C15}$$

where

$$J = \begin{pmatrix} 0 & \cdots & 0 & 1 \\ 0 & \cdots & 1 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 1 & \cdots & 0 & 0 \end{pmatrix}_{(n+1)\times(n+1)}$$

$$T = \begin{pmatrix} \binom{n}{n} & 0 & \cdots & 0 \\ \frac{1}{q}\binom{n}{n-1} & \frac{1}{q}\binom{n-1}{n-1} & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots \\ \frac{1}{q^n}\binom{n}{0} & \frac{1}{q^n}\binom{n-1}{0} & \cdots & \frac{1}{q^n}\binom{0}{0} \end{pmatrix}_{(n+1)\times(n+1)}.$$

Similarly,

$$B_d = \sum_{d'=0}^{n} (T^{-1}JT)_{dd'} A_{d'}, \quad 0 \le d,\ d' \le n, \tag{C16}$$

because

$$A = T^{-1}JTB \Longleftrightarrow B = T^{-1}JTA. \tag{C17}$$

3. Connection with Farrelly, Tuckett, and Stace

Reference [20] proposed a method to compute distance in local tensor-network codes, which are qubit stabilizer codes obtained from contracting other smaller stabilizer codes in a manner similar to Ref. [13]. In particular, we see that when applied to an $[\![n,k]\!]$ stabilizer code, the tensor $C_w^{l_1,\ldots,l_k}$ in Ref. [20] is exactly a reduced tensor enumerator in the multilinear form where w is precisely the degree of the monomial and $l_i = 0,1,2,3$ are the open indices that track the Pauli type I, X, Y, Z. This corresponds to computing the tensor weight enumerator $B(z)^{l_1,\ldots,l_k}$ if we keep all

the logical legs open. Similarly, $C_w^{0,\ldots,0}$ is the coefficient of an A-type tensor enumerator. Indeed, D_w which is obtainable from $C_w^{l_1,\ldots,l_k}$ is precisely the tensor coefficients of the scalar enumerators $B_d - A_d$. Both enumerators are in the usual Shor-Laflamme form.

Although both approaches rely on the tensor-network method to produce weight distributions, the detailed construction differs somewhat in how the tensors in the network is implemented—we construct the enumerator from encoding maps while Ref. [20] directly enumerates the logical operators of an encoding tensor and then computes their weights by contracting with another weight tensor that tabulates 4^n Pauli weights. Reference [20] also produces a tensor network [see Fig. 4(d)] with a double bond on each contraction where each edge has bond dimension 4. Naively this appears to lead to bond dimension 16 objects [Figs. 4(b) and 4(c)]. While such a description is sufficient, we know from the tensor enumerator formalism that it is possible to obtain the enumerator for Pauli stabilizer codes with a reduced bond dimension 4 (Thm 6), hence enabling more efficient tensor contractions.

It is unclear how the complexity estimates for these methods compare, as none was performed for Ref. [20] except for 1D codes that are prepared by log depth circuits. However, given the similarities in their structure and their overall efficiency for tree tensor networks and holographic codes, they should be polynomially equivalent in that regime. In practice, however, we note that even a constant factor difference can be quite substantial. Therefore, a more in-depth comparison in their performance can be an interesting problem for future work. In particular, it is important to understand whether these methods are optimal with respect to different networks.

Although Ref. [20] does not discuss weight distributions for other error models, it is possible to adapt their formalism to produce double and complete weight enumerators by modifying their weight tensor. For example, to obtain the double enumerator, one can replace $W_w^{g_1,\ldots,g_m}$ with $W_{w_x,w_z}^{g_1,\ldots,g_m}$ such that the tensor coefficient is unity when a Pauli string $\sigma^{g_1} \otimes \cdots \otimes \sigma^{g_m}$ has X and Z weights w_x, w_z, respectively. A similar extension should be possible for abstract enumerators. It is currently unclear whether a similar extension is possible for generalized abstract enumerators.

Another key difference lies in the use of MacWilliams transform in our work, which is polynomial [78] in n. Generally, the MacWilliams identity can help reduce computational cost. When the tensor network is efficiently contractible, and when we overlook the cost in manipulating large integers, the difference of keeping B- vs A-type tensor enumerators should be relatively insignificant. However, when the minimal cuts are large, e.g., when the tensor network represents a volume law or even some area law states, the B-type tensor can become far more populated than the A type by as much as $O(e^k)$. For instance, in the limiting case where the cost of tensor-network contraction approaches that of the brute-force method [79], e.g., in random stabilizer codes, we see that B is 2^{2k} more expensive to compute compared to A. Hence in some instances, the MacWilliams identities can help reduce computational cost that is exponential in k.

The decoder [20] uses is formally similar to the usual tensor network decoder where error probability is computed for some fixed p using a tensor contraction whereas the enumerator method produces an analytical expression for the error probability as a function of $\mathbf{u}$. The former is computationally advantageous when the error probabilities are heavily inhomogeneous and carry a strong locational dependence. The latter is more powerful for obtaining a continuous range of error probabilities when the physical errors are relatively uniform across the entire system.

Overall, the formalism based on tensor weight enumerator is more general as it applies to all quantum codes with uniform local dimensions. When specialized to the case of Pauli stabilizer codes over qubits, both methods can compute scalar and tensor enumerators associated with the code using tensor-network methods. In this case, our method improves upon Ref. [20] with a reduced bond dimension and with the use of the MacWilliams identities. The former provides a polynomial speedup while the latter can provide an $O(e^k)$ speedup in some regimes. With the extension to biased error and general noise models, we also extend the maximum-likelihood decoders for such stabilizer codes to general error channels. However, the enumerator method is less efficient in tackling highly inhomogeneous errors.

APPENDIX D: ERROR DETECTION FOR GENERAL NOISE CHANNELS

1. Nondetectable error

Proof. Let us compute here the probability of incurring a noncorrectable (logical) error. Suppose the error channel is $\mathcal{E}(\rho)$, which can be written as the Kraus form in Theorem 4 being the tensor product of single site errors. Let the initial state be $\rho = |\tilde{\psi}\rangle\langle\tilde{\psi}| \in L(\mathcal{C})$, and $\dim \mathcal{C} = K$. Then the probability of a nondetectable error is

$$\begin{aligned} p_{nd}(\rho) &= \mathrm{Tr}[(\Pi - \rho)\Pi\mathcal{E}(\rho)\Pi] \\ &= \sum_{\mathbf{i}} ||(I - |\tilde{\psi}\rangle\langle\tilde{\psi}|)\Pi\mathcal{K}_{\mathbf{i}}|\tilde{\psi}\rangle||^2, \end{aligned} \tag{D1}$$

where Π is the projection onto the code subspace. It is simply the overlap between the error state and the part of the code subspace that is orthogonal to the original codeword. ■

Now averaging over all initial codewords $|\tilde{\psi}\rangle$ with respect to the normalized uniform measure $\mu(|\tilde{\psi}\rangle)$, we have

$$\begin{aligned} p_{nd} &= \sum_{\mathbf{i}} \int_{|\tilde{\psi}\rangle\in\mathcal{C}} ||(I-|\tilde{\psi}\rangle\langle\tilde{\psi}|)\Pi\mathcal{K}_{\mathbf{i}}|\tilde{\psi}\rangle||^2 d\mu(|\tilde{\psi}\rangle) \\ &= \sum_{\mathbf{i}} \int_{|\tilde{\psi}\rangle\in\mathcal{C}} \langle\tilde{\psi}|\mathcal{K}_{\mathbf{i}}^\dagger\Pi(I-|\tilde{\psi}\rangle\langle\tilde{\psi}|)(I-|\tilde{\psi}\rangle\langle\tilde{\psi}|)\Pi\mathcal{K}_{\mathbf{i}}|\tilde{\psi}\rangle d\mu(\tilde{\psi}) \\ &= \sum_{\mathbf{i}} \int_{|\tilde{\psi}\rangle\in\mathcal{C}} \langle\tilde{\psi}|\mathcal{K}_{\mathbf{i}}^\dagger\Pi\mathcal{K}_{\mathbf{i}}|\tilde{\psi}\rangle d\mu(|\tilde{\psi}\rangle) - \sum_{\mathbf{i}} \int_{|\tilde{\psi}\rangle\in\mathcal{C}} \langle\tilde{\psi}|\mathcal{K}_{\mathbf{i}}^\dagger\Pi|\tilde{\psi}\rangle\langle\tilde{\psi}|\Pi\mathcal{K}_{\mathbf{i}}|\tilde{\psi}\rangle d\mu(|\tilde{\psi}\rangle). \end{aligned} \tag{D2}$$

Similar to Ref. [80], the integral in Eq. (D2) can be evaluated. The first term is

$$\begin{aligned} \sum_{\mathbf{i}} \int_{|\tilde{\psi}\rangle\in\mathcal{C}} \langle\tilde{\psi}|\mathcal{K}_{\mathbf{i}}^\dagger\Pi\mathcal{K}_{\mathbf{i}}|\tilde{\psi}\rangle d\mu(|\tilde{\psi}\rangle) &= \sum_{\mathbf{i}} \mathrm{Tr}[\mathcal{K}_{\mathbf{i}}^\dagger\Pi\mathcal{K}_{\mathbf{i}} \int_{|\tilde{\psi}\rangle\in\mathcal{C}} |\tilde{\psi}\rangle\langle\tilde{\psi}| dv(|\tilde{\psi}\rangle)] \\ &= \frac{1}{K}\sum_{\mathbf{i}} \mathrm{Tr}[\mathcal{K}_{\mathbf{i}}^\dagger\Pi\mathcal{K}_{\mathbf{i}}\Pi], \end{aligned} \tag{D3}$$

where we use Lemma 7 in Ref. [80] for the last step, which evaluates the integral.

The second term is

$$\sum_{\mathbf{i}} \int_{|\tilde{\psi}\rangle\in\mathcal{C}} \langle\tilde{\psi}|\mathcal{K}_{\mathbf{i}}^\dagger\Pi|\tilde{\psi}\rangle\langle\tilde{\psi}|\Pi\mathcal{K}_{\mathbf{i}}|\tilde{\psi}\rangle d\mu(|\tilde{\psi}\rangle) = \frac{1}{K(K+1)}\left(\sum_{\mathbf{i}} \mathrm{Tr}[\mathcal{K}_{\mathbf{i}}^\dagger\Pi\mathcal{K}_{\mathbf{i}}\Pi] + \sum_{\mathbf{i}} \mathrm{Tr}[\mathcal{K}_{\mathbf{i}}^\dagger\Pi]\,\mathrm{Tr}[\mathcal{K}_{\mathbf{i}}\Pi]\right), \tag{D4}$$

where we integrate over the same measure and use Lemma 8 in Ref. [80]. This completes our proof for Theorem 4.

2. Errors with nontrivial syndromes

Proof. When $\mathcal{C}$ is a stabilizer code, we can talk about syndrome measurements and decoding in the usual sense. While Π denotes the projection onto the code subspace, i.e., measuring trivial syndromes, we can similarly ask what the probability is for measuring some other syndromes s where the state is taken to a subspace $\Pi_s = E_s\Pi E_s^\dagger$, where E_s is an error with syndrome s. ■

Again, this is given by the overlap between the state suffering from the error and some final state in the error subspace.

$$\begin{aligned} \bar{p}_s &= \sum_{\mathbf{i}} \int_{|\tilde{\psi}\rangle\in\mathcal{C}} d\mu(|\tilde{\psi}\rangle)\,\mathrm{Tr}[\Pi_s\mathcal{K}_{\mathbf{i}}\rho\mathcal{K}_{\mathbf{i}}^\dagger\Pi_s] \\ &= \sum_{\mathbf{i}} \int_{|\tilde{\psi}\rangle\in\mathcal{C}} d\mu(|\tilde{\psi}\rangle)\,\mathrm{Tr}[E_s\Pi E_s^\dagger\mathcal{K}_{\mathbf{i}}\rho\mathcal{K}_{\mathbf{i}}^\dagger E_s\Pi E_s^\dagger] \\ &= \sum_{\mathbf{i}} \mathrm{Tr}[\int_{|\tilde{\psi}\rangle\in\mathcal{C}} d\mu(|\tilde{\psi}\rangle)|\tilde{\psi}\rangle\langle\tilde{\psi}|\mathcal{K}_{\mathbf{i}}^\dagger\Pi_s\mathcal{K}_{\mathbf{i}}] \\ &= \frac{1}{K}\sum_{\mathbf{i}} \mathrm{Tr}[\Pi\mathcal{K}_{\mathbf{i}}^\dagger\Pi_s\mathcal{K}_{\mathbf{i}}]. \end{aligned} \tag{D5}$$

Therefore, this quantity can be easily obtained from the B-type generalized complete enumerator when we replace one of Π by Π_s. Note that this recovers the syndrome probability with Pauli errors using the coset enumerator.

It is also useful for decoding purposes to consider the probability $p(\bar{L}|s)$ so as to correct the most likely logical error. The probability that $\tilde{E}_s = E_s\tilde{L}$ occurs is

$$\begin{aligned}\bar{p}(\tilde{L}\cap s) &= \sum_{\mathbf{i}}\int_{|\tilde{\psi}\rangle\in\mathcal{C}} d\mu(|\tilde{\psi}\rangle)\,\mathrm{Tr}[\tilde{E}_s|\tilde{\psi}\rangle\langle\tilde{\psi}|\tilde{E}_s^\dagger\Pi_s\mathcal{K}_{\mathbf{i}}|\tilde{\psi}\rangle\langle\tilde{\psi}|\mathcal{K}_{\mathbf{i}}^\dagger\Pi_s]\\ &= \sum_{\mathbf{i}}\int_{|\tilde{\psi}\rangle\in\mathcal{C}} d\mu(|\tilde{\psi}\rangle)(\langle\tilde{\psi}|\tilde{E}_s^\dagger\Pi_s\mathcal{K}_{\mathbf{i}}|\tilde{\psi}\rangle)(\langle\tilde{\psi}|\mathcal{K}_{\mathbf{i}}^\dagger\Pi_s\tilde{E}_s|\tilde{\psi}\rangle).\end{aligned} \tag{D6}$$

Because $\Pi_s\tilde{E}_s|\tilde{\psi}\rangle = \tilde{E}_s|\tilde{\psi}\rangle$.

$$\begin{aligned}\bar{p}(\tilde{L}\cap s) &= \sum_{\mathbf{i}}\int_{|\tilde{\psi}\rangle\in\mathcal{C}} d\mu(|\tilde{\psi}\rangle)(\langle\tilde{\psi}|\tilde{E}_s^\dagger\mathcal{K}_{\mathbf{i}}|\tilde{\psi}\rangle)(\langle\tilde{\psi}|\mathcal{K}_{\mathbf{i}}^\dagger\tilde{E}_s|\tilde{\psi}\rangle)\\ &= \frac{1}{K(K+1)}\Big(\sum_{\mathbf{i}}\mathrm{Tr}[\tilde{E}_s^\dagger\mathcal{K}_{\mathbf{i}}\Pi\mathcal{K}_{\mathbf{i}}^\dagger\tilde{E}_s\Pi] + \sum_{\mathbf{i}}\mathrm{Tr}[\tilde{E}_s^\dagger\mathcal{K}_{\mathbf{i}}\Pi]\,\mathrm{Tr}[\mathcal{K}_{\mathbf{i}}^\dagger\tilde{E}_s\Pi]\Big)\\ &= \frac{1}{K(K+1)}\Big(\sum_{\mathbf{i}}\mathrm{Tr}[\mathcal{K}_{\mathbf{i}}\Pi\mathcal{K}_{\mathbf{i}}^\dagger\Pi_s] + \sum_{\mathbf{i}}\mathrm{Tr}[\mathcal{K}_{\mathbf{i}}\Pi\tilde{E}_s^\dagger]\,\mathrm{Tr}[\mathcal{K}_{\mathbf{i}}^\dagger\tilde{E}_s\Pi]\Big).\end{aligned} \tag{D7}$$

Hence

$$\bar{p}(\tilde{L}|s) = \bar{p}(\tilde{L}\cap s)/\bar{p}_s = \frac{1}{K+1}\left(1 + \frac{\sum_{\mathbf{i}}\mathrm{Tr}[\mathcal{K}_{\mathbf{i}}\Pi\tilde{E}_s^\dagger]\,\mathrm{Tr}[\mathcal{K}_{\mathbf{i}}^\dagger\tilde{E}_s\Pi]}{\sum_{\mathbf{i}}\mathrm{Tr}[\mathcal{K}_{\mathbf{i}}\Pi\mathcal{K}_{\mathbf{i}}^\dagger\Pi_s]}\right). \tag{D8}$$

Each term in Eq. (D8) can be computed by setting M_1, M_2 to the appropriate values in the weight enumerator $M_1 = \Pi\tilde{E}_s^\dagger, M_2 = \tilde{E}_s\Pi$ for the $\bar{A}$-type enumerator and $M_1 = \Pi, M_2 = \Pi_s$ for the $\bar{B}$-type enumerator.

For the purpose of building a decoder, we do not care about the overall normalization, hence computing the $\bar{A}$-type enumerator will be sufficient. The first term in $\bar{p}(\tilde{L}\cap s)$ is independent on the logical operation $\tilde{L}$, and thus does not modify our decision based on the maximum likelihood.

3. General logical error channel

Nonunitary logical error: Under this more general channel, it is also natural to consider a more general logical error where for some ρ_s in the error subspace with syndrome s beyond the kind of coherent logical error $\tilde{L}$. For instance, we can discuss the error probability that the logical information suffers from a logical error channel in that subspace

$$\tilde{\mathcal{N}}(\tilde{\rho}) = \tilde{\rho} \to \sum_j \tilde{\eta}_j\tilde{\rho}\tilde{\eta}_j^\dagger, \tag{D9}$$

after obtaining syndrome s by measuring the checks. More precisely, we find that

$$\begin{aligned}\bar{p}(\tilde{\mathcal{N}}(\cdot)\cap s) &= \sum_{\mathbf{i}}\int_{|\tilde{\psi}\rangle\in\mathcal{C}} d\mu(|\tilde{\psi}\rangle)\,\mathrm{Tr}[E_s\tilde{\mathcal{N}}(\tilde{\rho})E_s^\dagger\Pi_s\mathcal{K}_{\mathbf{i}}|\tilde{\psi}\rangle\langle\tilde{\psi}|\mathcal{K}_{\mathbf{i}}^\dagger\Pi_s]\\ &= \sum_{\mathbf{i},j}\int_{|\tilde{\psi}\rangle\in\mathcal{C}} d\mu(|\tilde{\psi}\rangle)\,\mathrm{Tr}[E_s\tilde{\eta}_j|\tilde{\psi}\rangle\langle\tilde{\psi}|\tilde{\eta}_j^\dagger E_s^\dagger\mathcal{K}_{\mathbf{i}}|\tilde{\psi}\rangle\langle\tilde{\psi}|\mathcal{K}_{\mathbf{i}}^\dagger]\\ &= \sum_{\mathbf{i},j}\int_{|\tilde{\psi}\rangle\in\mathcal{C}} d\mu(|\tilde{\psi}\rangle)\,\mathrm{Tr}[O_{j\mathbf{i}}^s|\tilde{\psi}\rangle\langle\tilde{\psi}|O_{j\mathbf{i}}^{s\dagger}|\tilde{\psi}\rangle\langle\tilde{\psi}|]\\ &= \frac{1}{K(K+1)}\sum_{\mathbf{i},j}\Big(\mathrm{Tr}[O_{j,\mathbf{i}}^{s\dagger}\Pi O_{j,\mathbf{i}}^s\Pi] + \mathrm{Tr}[O_{j,\mathbf{i}}^{s\dagger}\Pi]\,\mathrm{Tr}[O_{j,\mathbf{i}}^s\Pi]\Big),\end{aligned} \tag{D10}$$

where we defined $O^s_{j,\mathbf{i}} = \mathcal{K}^\dagger_{\mathbf{i}} E_s \tilde{\eta}_j$. Note that just like for calculating error probability of syndrome s under depolarizing noise with B-type enumerators, the first term can be expressed as a B type and one can perform the sum over j in defining

$$\Pi^s_\eta = \sum_j E_s \tilde{\eta}_j \Pi \tilde{\eta}_j^\dagger E_s^\dagger \tag{D11}$$

and then substitute and compute the first term as

$$\sum_{\mathbf{i}} \mathrm{Tr}[\mathcal{K}_{\mathbf{i}} \Pi \mathcal{K}^\dagger_{\mathbf{i}} \Pi^s_\eta], \tag{D12}$$

which is basically identical to our computation of the nondetectable error probability except we set $M_2 = \Pi^s_\eta$ and the remaining procedures for decomposing $\mathcal{K}_{\mathbf{i}}$ carries over identically.

For the second term, however, we have to repeat the enumerator computations for each j by summing over $\mathbf{i}$. If we set $\tilde{E}^j_s = E_s \tilde{\eta}_j$, then it has the identical form as the coset enumerator we analyzed earlier except for the j dependence,

$$\mathrm{Tr}[\mathcal{K}_{\mathbf{i}} \Pi \tilde{E}^{j\,\dagger}_s] \,\mathrm{Tr}[\mathcal{K}^\dagger_{\mathbf{i}} \tilde{E}^j_s \Pi]. \tag{D13}$$

For generic errors, $j = 1, \ldots, 4^k$, so it is more relevant for k small.

For a fixed error channel, the enumerator fully captures the likelihood of all error channels by decomposing $\tilde{\eta}_j$ into Paulis and varying their coefficients. It could be interesting to analyze the extrema of error probabilities with respect to these variables to find the most likely error channel. We can imagine building a decoder that seeks to undo the effect of the most likely error channel, though it is unclear when such recovery procedures exist in general. To correct such errors, we apply first a coset element of E_s. Then depending on the availability of the recovery map given the logical error, we (partially) reverse the effect of the logical error channel based on the relevant information of the code and syndrome measurement outcomes.

APPENDIX E: QUANTUM TANNER GRAPH FROM QUANTUM LEGO

Given any $[\![n, k]\!]$ stabilizer code whose codewords can be obtained from measuring the check operators and postselecting on the trivial syndrome outcome, one can express the encoding or state-preparation process as a tensor network.

Consider a measurement-based state-preparation process where we entangle all physical qubits with a reference using Bell pairs $|00\rangle + |11\rangle$. Then to apply the checks by measuring them. The measurement process is straightforward—the physical qubits on which the check has support

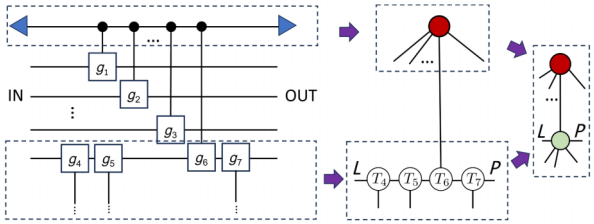

FIG. 24. An ancillary is prepared and projected onto $|+\rangle$ (blue triangle). This condenses the ancillary into a check node tensor that is simply the encoding tensor of a repetition code (Z spider). The data qubit condenses into another tensor node (green) by combining the degree 3 tensors T_i. Here IN or L labels the input and logical degrees of freedom while OUT or P labels the output and physical degrees of freedom in the final atomic code it produces. The bottom row indicates that the qubit is checked by other checks, e.g., $g_i, i = 4 \sim 7$.

is entangled with a ancillary qubit using the usual circuit. Then one measures and postselects on the trivial syndrome. Although the actual preparation of such a state in a quantum computer requires either adaptive measurements, decoding, and/or postselection, thus rendering the actual process much more complicated, there is no such obstacle in the classical tensor-network description where postselection simply corresponds to contraction of a particular type of tensor.

Suppose each check acts with a unitary g_i on the physical qubit (bottom wire of Fig. 24), then the action of this gate on the wire can be converted into a tensor [Fig. 25(a)]. The resulting tensor for common gates used in the preparation process for XP stabilizer codes are also given in Fig. 25(b). Generally, this conversion can be performed for any two-qubit controlled gate by choosing the appropriate tensor T in purple.

Here φ is a tensor with elements

$$\varphi_{ij} \propto \begin{pmatrix} 1 & 1 \\ 1 & e^{i\varphi} \end{pmatrix}, \tag{E1}$$

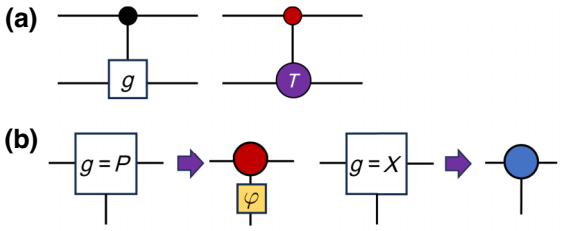

FIG. 25. (a) Depending on the nature of the controlled-g gate, the target action can be condensed into a tensor. (b) The action of a controlled-phase gate and that of a controlled X gate can be simplified into the corresponding tensors with φ defined below. Red and blue tensors are Z and X spiders, respectively.

where an overall normalization is added as needed. For $\varphi = \pi$ it is the Hadamard gate and tensor.

Note that a code prepared this way has a nontrivial kernel in the encoding map. This is the same for the tensor network we built for the surface code or color code in the main text.

For CSS codes, without loss of generality, one can perform first the Z checks then the X checks. By suitably substituting g, one can simplify the data nodes (green) into the form of Fig. 6(c).

[1] A. Yu. Kitaev, Fault-tolerant quantum computation by anyons, Ann. Phys. (N. Y.) **303**, 2 (2003).

[2] Matthew B. Hastings, Jeongwan Haah, and Ryan O'Donnell, in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing* (Association for Computing Machinery, New York, NY, USA, 2021), pp. 1276–1288.

[3] Pavel Panteleev and Gleb Kalachev, in *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing* (Association for Computing Machinery, New York, NY, USA, 2022), pp. 375–388.

[4] Peter Shor and Raymond Laflamme, Quantum analog of the MacWilliams identities for classical coding theory, Phys. Rev. Lett. **78**, 1600 (1997).

[5] E. M. Rains, Quantum weight enumerators, IEEE Trans. Inf. Theory **44**, 1388 (1998).

[6] E. M. Rains, Quantum shadow enumerators, IEEE Trans. Inf. Theory **45**, 2361 (1999).

[7] E. M. Rains, Shadow bounds for self-dual codes, IEEE Trans. Inf. Theory **44**, 134 (1998).

[8] Alexei Ashikhmin, Alexander Barg, Emanuel Knill, and Simon Litsyn, Quantum error detection I: Statement of the problem, IEEE Trans. Inf. Theory **46**, 778 (2000).

[9] Chuangqiang Hu, Shudi Yang, and Stephen S.-T. Yau, Weight enumerators for nonbinary asymmetric quantum codes and their applications, Adv. Appl. Math. **121**, 102085 (2020).

[10] Patrick Rall, Signed quantum weight enumerators characterize qubit magic state distillation, ArXiv:1702.06990.

[11] Daniel Miller, Daniel Loss, Ivano Tavernelli, Hermann Kampermann, Dagmar Bruß, and Nikolai Wyderka, Shor-Laflamme distributions of graph states and noise robustness of entanglement, J. Phys. A- Math. Theor. **56**, 335303 (2023).

[12] Thomas Schuster and Norman Y. Yao, Operator growth in open quantum systems, Phys. Rev. Lett. **131**, 160402 (2023).

[13] ChunJun Cao and Brad Lackey, Quantum Lego: Building quantum error correction codes from tensor networks, PRX Quantum **3**, 020332 (2022).

[14] ChunJun Cao and Brad Lackey, Quantum weight enumerators and tensor networks, IEEE Trans. Inf. Theory **70**, 3512 (2024).

[15] Alexander Vardy, in *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, STOC '97* (Association for Computing Machinery, New York, NY, USA, 1997), p. 92.

[16] Ilya Dumer, Alexey A. Kovalev, and Leonid P. Pryadko, in *2014 IEEE International Symposium on Information Theory* (2014), pp. 1086–1090.

[17] Alexey A. Kovalev, Ilya Dumer, and Leonid P. Pryadko, in *2013 Information Theory and Applications Workshop (ITA)* (2013), pp. 1–6.

[18] Ilya Dumer, Alexey A. Kovalev, and Leonid P. Pryadko, Distance verification for classical and quantum LDPC codes, IEEE Trans. Inf. Theory **63**, 4675 (2017).

[19] M. N. Vyalyi, Hardness of approximating the weight enumerator of a binary linear code, ArXiv:cs/0304044.

[20] Terry Farrelly, David K. Tuckett, and Thomas M. Stace, Local tensor-network codes, New J. Phys. **24**, 043015 (2022).

[21] Andrew S. Darmawan and David Poulin, Tensor-network simulations of the surface code under realistic noise, Phys. Rev. Lett. **119**, 040502 (2017).

[22] Yasunari Suzuki, Keisuke Fujii, and Masato Koashi, Efficient simulation of quantum error correction under coherent error based on the nonunitary free-fermionic formalism, Phys. Rev. Lett. **119**, 190503 (2017).

[23] Sergey Bravyi, Matthias Englbrecht, Robert König, and Nolan Peard, Correcting coherent errors with surface codes, npj Quantum Inf. **4**, 55 (2018).

[24] Chenfeng Cao, Chao Zhang, Zipeng Wu, Markus Grassl, and Bei Zeng, Quantum variational learning for quantum error-correcting codes, Quantum **6**, 828 (2022).

[25] Arpit Dua, Aleksander Kubica, Liang Jiang, Steven T. Flammia, and Michael J. Gullans, Clifford-deformed surface codes, PRX Quantum **5**, 010347 (2024).

[26] Vincent Paul Su, ChunJun Cao, Hong-Ye Hu, Yariv Yanay, Charles Tahan, and Brian Swingle, Discovery of optimal quantum error correcting codes via reinforcement learning, ArXiv:2305.06378.

[27] Robert J. Harris, Elliot Coupe, Nathan A. McMahon, Gavin K. Brennen, and Thomas M. Stace, Decoding holographic codes with an integer optimisation decoder, Phys. Rev. A **102**, 062417 (2020).

[28] Fernando Pastawski and John Preskill, Code properties from holographic geometries, Phys. Rev. X **7**, 021022 (2017).

[29] In practice, it may be more efficient to compute the tensor enumerator $\mathbf{A}(z)$ of the entire code, then perform a MacWilliams transform on the tensor enumerator.

[30] William Duke, On codes and Siegel modular forms, Int. Math. Res. Not. **1993**, 125 (1993).

[31] Bernhard Runge, Codes and Siegel modular forms, Discrete Math. **148**, 175 (1996).

[32] Sergey Bravyi, Subsystem codes with spatially local generators, Phys. Rev. A **83**, 012320 (2011).

[33] In fact this is true of every subsystem code: Using the usual symplectic formalism of stabilizer groups, the gauge group becomes a subspace, and a Darboux basis for this subspace provides the two isotropic subspaces that characterize $\mathcal{G}_1$ and $\mathcal{G}_2$.

[34] Lorenzo Leone, Salvatore F. E. Oliviero, and Alioscia Hamma, Stabilizer Rényi entropy, Phys. Rev. Lett. **128**, 00 (2022).

[35] Tobias Haug, Soovin Lee, and M. S. Kim, Efficient quantum algorithms for stabilizer entropies, Phys. Rev. Lett. **132**, 240602 (2024).

[36] Poetri Sonya Tarabunga, Emanuele Tirrito, Mari Carmen Bañuls, and Marcello Dalmonte, Nonstabilizerness via matrix product states in the Pauli basis, ArXiv:2401.16498.

[37] Xiaotong Ni, Oliver Buerschaper, and Maarten Van den Nest, A non-commuting stabilizer formalism, J. Math. Phys. **56**, 052201 (2015).

[38] Mark A. Webster, Benjamin J. Brown, and Stephen D. Bartlett, The XP stabiliser formalism: a generalisation of the Pauli stabiliser formalism with arbitrary phases, Quantum **6**, 815 (2022).

[39] Ruohan Shen, Yixu Wang, and ChunJun Cao, Quantum Lego and XP stabilizer codes, ArXiv:2310.19538.

[40] Bob Coecke and Ross Duncan, Interacting quantum observables: Categorical algebra and diagrammatics, New J. Phys. **13**, 043016 (2011).

[41] Zheng-Cheng Gu, Michael Levin, and Xiao-Gang Wen, Tensor-entanglement renormalization group approach as a unified method for symmetry breaking and topological phase transitions, Phys. Rev. B **78**, 205116 (2008).

[42] Michael A. Levin and Xiao-Gang Wen, String-net condensation: A physical mechanism for topological phases, Phys.Rev. B **71**, 045110 (2005).

[43] Oliver Buerschaper, Miguel Aguado, and Guifré Vidal, Explicit tensor network representation for the ground states of string-net models, Phys. Rev. B **79**, 085119 (2009).

[44] In the context of stabilizer codes, its Clifford encoding circuit is also easily obtainable [81].

[45] Pavithran Iyer and David Poulin, IEEE Trans. Inf. Theory **61**, 5209 (2015).

[46] For stabilizer codes, if there are k_v logical legs on a tensor on a node v, then building $\mathbf{A}_v(z)$ is upper bounded by complexity $O(q^{c-2k_v})$ and is less expensive compared to that of $\mathbf{B}_v(z)$. For general quantum codes where one uses the full tensor enumerator, preparing the coefficients of $\mathbf{A}_v(z)$ requires a worst case of $O(q^{(5c-4k_v)})$ operations.

[47] Fernando Pastawski, Beni Yoshida, Daniel Harlow, and John Preskill, Holographic quantum error-correcting codes: Toy models for the bulk/boundary correspondence, J. High Energy Phys. **2015**, 149 (2015).

[48] Robert J. Harris, Nathan A. McMahon, Gavin K. Brennen, and Thomas M. Stace, Calderbank-Shor-Steane holographic quantum error-correcting codes, Phys. Rev. A **98**, 052301 (2018).

[49] ChunJun Cao and Brad Lackey, Approximate Bacon-Shor code and holography, J. High Energy Phys. **2021**, 127 (2021).

[50] M. Steinberg, S. Feld, and A. Jahn, Holographic codes from hyperinvariant tensor networks, Nat. Commun. **14**, 7314 (2023).

[51] Jeongwan Haah, Local stabilizer codes in three dimensions without string logical operators, Phys. Rev. A **83**, 042330 (2011).

[52] Jonas Haferkamp, Dominik Hangleiter, Jens Eisert, and Marek Gluza, Contracting projected entangled pair states is average-case hard, Phys. Rev. Res. **2**, 013010 (2020).

[53] Yimin Ge and Jens Eisert, Area laws and efficient descriptions of quantum many-body states, NJP **18**, 083026 (2016).

[54] A. R. Calderbank, E. M. Rains, P. M. Shor, and N. J. A. Sloane, Quantum error correction via codes over GF(4), IEEE Trans. Inf. Theory **44**, 1369 (1998).

[55] Arpit Dua, Private communication, 2023.

[56] Kitaev surface code, in *The Error Correction Zoo*, edited by Victor V. Albert and Philippe Faist (2023).

[57] J. Pablo Bonilla Ataides, David K. Tuckett, Stephen D. Bartlett, Steven T. Flammia, and Benjamin J. Brown, The XZZX surface code, Nat. Commun. **12**, 2172 (2021).

[58] A previous tensor network construction of the $[\![19, 1, 5]\!]$ color code can be found Ref. [20], which requires both the $[\![7, 1, 3]\!]$ codes and $[\![9, 0, 3]\!]$ stabilizer states as building blocks. However, the protocol does not generalize to $d > 5$ due to concavity of the polygonal region.

[59] Aleksander Kubica, Beni Yoshida, and Fernando Pastawski, Unfolding the color code, New J. Phys. **17**, 083026 (2015).

[60] Note that this radius is different from that in [27] where on the central bulk qubit is singled out and its distances are computed with respect to codes of different n s.

[61] This result assumes a particular decoder applied to small sized systems using Monte Carlo methods. It is possible that a different asymptotic behaviour can emerge with larger codes and greater accuracy.

[62] Note that this speedup would not be possible for nondegenerate codes as all subsystems of size d has the same entanglement approximately d.

[63] Muyuan Li, Daniel Miller, Michael Newman, Yukai Wu, and Kenneth R. Brown, 2D compass codes, Phys. Rev. X **9**, 021041 (2019).

[64] In this way, the approximately $\exp(d)$ cost of computing the Bacon-Shor weight enumerator is not surprising as the unfixed tensor network encompasses all 2D compass code configurations.

[65] Christopher T. Chubb and Steven T. Flammia, Statistical mechanical models for quantum codes with correlated noise, Ann. Inst. Henri Poincare D **8**, 269 (2021).

[66] I. Dumer, D. Micciancio, and M. Sudan, Hardness of approximating the minimum distance of a linear code, IEEE Trans. Inf. Theory **49**, 22 (2003).

[67] Anurag Anshu, Nikolas P. Breuckmann, and Chinmay Nirkhe, in *Proceedings of the 55th Annual ACM Symposium on Theory of Computing* (Association for Computing Machinery, New York, NY, USA, 2023), pp. 1090–1096.

[68] The expansion property of these codes may naively indicate the edge cut to scale with volume. However, as they are not the corresponding tensor networks, and that edge cuts are only upper bounds of entanglement, it may be possible that a sparser tensor network can be found that permits fewer cuts.

[69] Christopher Eltschka and Jens Siewert, Maximum N-body correlations do not in general imply genuine multipartite entanglement, Quantum **4**, 229 (2020).

[70] Caroline Mauron, Terry Farrelly, and Thomas M. Stace, Optimization of tensor network codes with reinforcement learning, ArXiv:2305.11470.

[71] Andrew J. Ferris and David Poulin, Tensor networks and quantum error correction, Phys. Rev. Lett. **113**, 030501 (2014).

[72] Christopher T. Chubb, General tensor network decoding of 2D Pauli codes, ArXiv:2101.04125.

[73] Sergey Bravyi, Martin Suchara, and Alexander Vargo, Efficient algorithms for maximum likelihood decoding in the surface code, Phys. Rev. A **90**, 032326 (2014).

[74] Andrew S. Darmawan and David Poulin, Linear-time general decoding algorithm for the surface code, Phys. Rev. E **97**, 051302 (2018).

[75] Jessie MacWilliams, A theorem on the distribution of weights in a systematic code, Bell Syst. Tech. J. **42**, 79 (1963).

[76] Florence Jessie MacWilliams and Neil James Alexander Sloane, *The Theory of Error Correcting Codes* (North-holland Publishing Company, Amsterdam, 1977), Vol. 16.

[77] In the MATLAB code implementation, $j = 3$ and $j = 2$ are swapped in the indexing convention such that Y is mapped to the last index and Z is mapped to the second last.

[78] The complexity roughly scales as $O(n^3)$ from matrix multiplication. However, this is not counting the cost needed to manipulate large integers.

[79] Intuitively, computing B enumerators of an $[\![n, k]\!]$ stabilizer code involves enumerating 2^{n+k} elements instead of 2^{n-k}.

[80] A. E. Ashikhmin, A. M. Barg, E. Knill, and S. N. Litsyn, Quantum error detection I: Statement of the problem, IEEE Trans. Inf. Theory **46**, 778 (2000).

[81] Scott Aaronson and Daniel Gottesman, Improved simulation of stabilizer circuits, Phys. Rev. A **70**, 052328 (2004).