Systematic approach to hyperbolic quantum error correction codes

Ahmed Adel Mahmoud [1,2,*] Kamal Mohamed Ali [1] and Steven Rayan [1,2,†]
[1]*Centre for Quantum Topology and Its Applications (quanTA), University of Saskatchewan, Saskatoon, SK, Canada*
[2]*Department of Mathematics and Statistics, University of Saskatchewan, Saskatoon, SK, Canada*

(Received 12 November 2025; accepted 13 March 2026; published 10 April 2026)

Quantum error correction codes defined on hyperbolic lattices leverage the unique geometric properties of the hyperbolic space to enhance the performance of quantum error correction. By embedding qubits in hyperbolic lattices, these codes achieve higher encoding rates and lower qubit overhead compared to those defined on conventional Euclidean lattices. Building on recent advances in hyperbolic crystallography, we introduce a unified framework for the systematic construction and scalable benchmarking of CSS quantum error correction codes on hyperbolic lattices. A central component of this framework is the Hyperbolic Cycle Basis algorithm, which employs graph-theoretic methods to efficiently identify all plaquette cycles (parity-check supports) and nontrivial cycles (logical operators). This enables scalable and automated benchmarking of a broad class of CSS codes defined on hyperbolic geometries. We apply this framework to construct and simulate two representative hyperbolic quantum error correction codes (HQECCs), evaluating key performance metrics such as encoding rate, error threshold, and code distance for different sublattices. While HQECCs serve as concrete examples, the framework can be adapted to a wide range of CSS codes, including those with more intricate stabilizer structures such as Floquet codes. This work establishes a foundation for systematic exploration and benchmarking of CSS codes on hyperbolic lattices, paving the way toward practical, high-performance quantum error correction.

DOI: 10.1103/95mp-w7kr

I. INTRODUCTION

Quantum error correction is a prerequisite for the scalable and practical implementation of quantum computing, ensuring computational reliability in the presence of noise and decoherence [1]. Among the various families of quantum error correction codes, quantum low-density parity-check (LDPC) codes have gained significant attention due to their favorable properties. In particular, their parity-check operators act on a constant number of qubits, and each qubit participates in a constant number of parity-check measurements, making them highly efficient for fault-tolerant quantum computation [2,3]. Over the past few years, several LDPC codes have been developed, with the toric code, introduced by Kitaev [4], being one of the most well-known examples. The toric code offers high error tolerance but protects only two logical qubits regardless of lattice size [5]. This limitation was overcome by extending the framework to hyperbolic lattices, leading to the discovery of HQECCs [6–9]. Like the toric code, HQECCs offer high error tolerance but with the crucial advantage that the number of logical qubits grows linearly with physical qubits. This advantage of hyperbolic lattices in topological quantum error correction is further confirmed by state-of-the-art Floquet codes, which achieve better performance and lower qubit overhead on hyperbolic lattices than on their original Euclidean honeycomb design [10–12]. These advances established a clear paradigm: hyperbolic lattices are a fundamentally better foundation for topological quantum error correction, yet they expose a critical gap between theoretical advances in topological quantum error correction and practical quantum hardware. The design space of hyperbolic lattices is infinitely vast compared to Euclidean alternatives [13]. Yet, out of this immense family of lattices, it is not known which hyperbolic lattice is optimal for a given topological quantum error correction code and noise profile. Answering this question is essential not only for benchmarking current quantum hardware but also for guiding the design of next-generation quantum processors optimized for topological quantum error correction. To answer this question, a unified framework for constructing and benchmarking topological codes on hyperbolic lattices is needed. In this work, we present a systematic framework for constructing and benchmarking CSS codes on hyperbolic lattices, building on recent advances in hyperbolic crystallography [14]. This framework is applicable to any hyperbolic $\{p, q\}$ tessellation of the Poincaré disk with an underlying $\{p_B, q_B\}$ Bravais lattice. A central component of this framework is the Hyperbolic Cycle Basis algorithm, which, to the best of our knowledge, is the first systematic and scalable method for identifying all plaquette cycles (representing parity-check supports) in a hyperbolic tessellation, as well as all nontrivial cycles (representing logical operators) that generate the first homology group $H_1(M)$ of the underlying Riemann surface M.

To demonstrate the versatility of this framework, we apply it to simulate two HQECCs based on two hyperbolic tessellations of the Poincaré disk, namely $\{8, 3\}$ and $\{10, 3\}$. These tessellations correspond to two different Bravais lattice structures, each associated with a unique Fuchsian group. The unit cells of these lattices are embedded in genus-2 Riemann surfaces, obtained by compactifying the corresponding Bravais lattices $\{8, 8\}$ and $\{10, 5\}$, respectively.

*Contact author: ahmed.mahmoud@usask.ca
†Contact author: rayan@math.usask.ca

Finally, even though this work focuses mainly on quantum error correction codes, the Hyperbolic Cycle Basis algorithm also has useful applications in condensed matter physics. In the spectral analysis of hyperbolic materials derived from kagome-like lattices [15], it has been shown that the ground state exihibts a highly degenerate flat band whose eigenstates are highly localized and therefore dubbed compact localized states [16,17]. Each of these states is supported exclusively on an independent cycle—either trivial or nontrivial—of the underlying lattice. As a result, the full set of compact localized states naturally forms a cycle basis of the underlying hyperbolic graph, a structure that can be efficiently identified using the HCB algorithm.

II. MATHEMATICAL PRELIMINARIES

A. Hyperbolic geometry

Two models of hyperbolic geometry are equivalent. The first model is the upper-half plane $\mathbb{H} = (z \in \mathbb{C} : \mathrm{Im}(z) > 0)$ with boundary $\partial\mathbb{H} = \mathbb{R} \cup \{\infty\}$. The second model is the Poincaré disk model $\mathbb{D} = \{z' \in \mathbb{C} : |z'| < 1\}$ with boundary $\partial\mathbb{D} = \{z \in \mathbb{C} : |z| = 1\}$. The hyperbolic space $\mathbb{H}$ is equipped with the hyperbolic metric

$$ds^2 = \frac{dx^2 + dy^2}{y^2}. \tag{1}$$

An isometry between the two spaces $h : \mathbb{H} \to \mathbb{D}$ is given by

$$h(z) = \frac{zi + 1}{z + i}, \tag{2}$$

where $z = x + iy \in \mathbb{C}$. The metric induced on $\mathbb{D}$ by h is given by

$$ds^2 = \frac{4|dz|^2}{(1 - |z|^2)^2}. \tag{3}$$

Since the two models are equivalent, one can work in either model. However, since the Poincaré disk model is a bounded subset of the Euclidean plane, it is more convenient for visualization. Therefore we will utilize the Poincaré disk model throughout this paper.

The hyperbolic distance between any two points $z_1, z_2 \in \mathbb{D}$ is given by

$$d(z_1, z_2) = \mathrm{arcosh}\left(1 + \frac{2|z_1 - z_2|^2}{(1 - |z_1|^2)(1 - |z_2|^2)}\right). \tag{4}$$

Geodesics in $\mathbb{D}$ are circles that, when extended, are orthogonal to the boundary of the Poincaré disk and its diameter. The hyperbolic angle between two geodesics that intersect at a point is the usual Euclidean angle between the tangent vectors to these two geodesics. A hyperbolic polygon with p edges, called a p-gon, is a convex closed set consisting of p hyperbolic geodesic edges. The point at which two segments intersect is a vertex. A polygon is called regular if all its internal angles are equal. A regular tessellation of $\mathbb{D}$ is achieved by covering the Poincaré disk by regular p-gons that either do not overlap or overlap only at their boundaries. For convenience, we refer to regular tessellations as *patterns*. Formally, a pattern is a finite hyperbolic graph embedded into a closed Riemann surface that determines the shape and size of the unit cell of a hyperbolic lattice. The Schläfli symbol of a pattern is $\{p, q\}$ if each face is a p-gon, and each vertex is surrounded by q faces. To construct a $\{p, q\}$ pattern, one starts by constructing one polygon representing the unit cell of the pattern. Let r be the radius of the polygon given by

$$r = \sqrt{\frac{\cos\left(\frac{\pi}{p} + \frac{\pi}{q}\right)}{\cos\left(\frac{\pi}{p} - \frac{\pi}{q}\right)}}. \tag{5}$$

Then, the positions of the unit cell vertices in $\mathbb{D}$ are given by

$$z_k = re^{2\pi ik/p + \delta}, \tag{6}$$

where $k = 1, \ldots, p$ and δ is an arbitrary phase.

Upon imposing periodic boundary conditions (PBCs), a hyperbolic $\{p, q\}$ pattern can be embedded in a closed Riemann surface of genus $g \geqslant 2$ [18,19]. In this setting, the following combinatorial relations hold. Since each face contains p edges and every edge is shared by two faces, counting face–edge incidences gives $pF = 2E$. Likewise, since q edges meet at every vertex and each edge connects two vertices, counting edge-vertex incidences gives $2E = qV$. By combining these relations, we obtain

$$pF = 2E = qV, \tag{7}$$

where F, E, and V are the number of faces, edges, and vertices of the pattern, respectively. An important topological invariant that we shall use later is the Euler characteristic. Given a closed Riemann surface M tessellated by F faces, E edges, and V vertices, the Euler characteristic is given by

$$\chi(M) = F - E + V. \tag{8}$$

If χ is even, then the tessellation can be embedded in an orientable surface M of genus g; in this case [20],

$$\chi(M) = 2 - 2g. \tag{9}$$

It follows that the number of faces F and the genus g of the closed Riemann surface are not independent. More concretely, consider the Gauss-Bonnet theorem that relates the curvature of a surface to its topology. It states that for a closed, orientable surface M, the genus g of M is proportional to its Gaussian curvature as follows [21]:

$$\int_M K\, dA = 4\pi(1 - g).$$

In the case of hyperbolic geonmetry $K = -1$; therefore, the area of the underlying Riemann surface M is given by

$$A(M) = 4\pi(g - 1). \tag{10}$$

In hyperbolic geometry, the sum of the angles of a hyperbolic triangle is less than π. A special case of the Gauss-Bonnet theorem states that the area of a hyperbolic triangle Δ with internal angles α, β, and ζ is given by

$$A(\Delta) = \pi - (\alpha + \beta + \zeta). \tag{11}$$

Generally, the area of a regular hyperbolic p-gon Λ is given by its angular defect

$$A(\Lambda) = (p - 2)\pi - \sum_{j=1}^{p} \alpha_j, \tag{12}$$

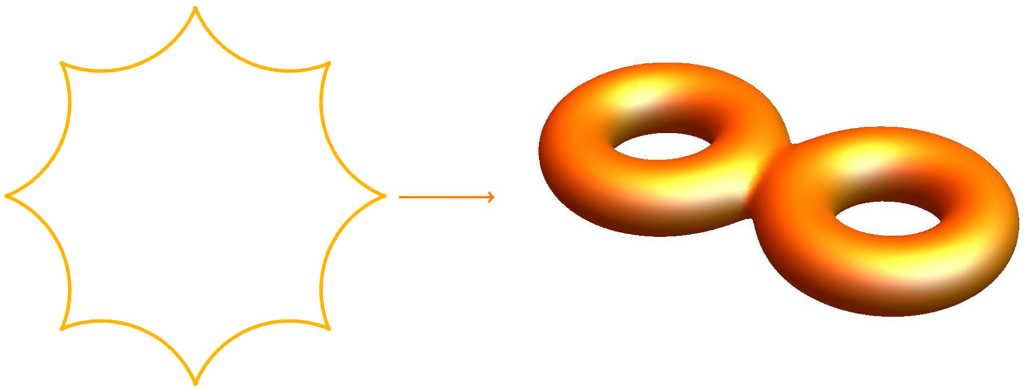

FIG. 1. The left half of the figure shows the unit cell of the {8, 8} Bravais lattice, which, when compactified, can be embedded in the genus-2 Riemann surface shown on the right.

where α_j are the internal angles of Λ. In a regular $\{p, q\}$ pattern, the internal angles of each p-gon are given by $2\pi/q$. Therefore the area of a p-gon is given by

$$A(\Lambda) = \left(p - 2 - \frac{2p}{q}\right)\pi. \tag{13}$$

Now, assume that a closed Riemann surface M is tessellated by a regular $\{p, q\}$ pattern; then, $A(M) = FA(\Lambda)$. That is,

$$4\pi(g-1) = F\left(p - 2 - \frac{2p}{q}\right)\pi, \tag{14}$$

where F is the number of faces of the underlying tessellation of M. Solving for F, we get

$$F = \frac{4q(g-1).}{pq - 2q - 2p}. \tag{15}$$

This is in contrast to the Euclidean case in which the number of faces of the tessellation is independent of the genus of the embedding surface.

Let (F, E, V) be a solution to (7), then (nF, nE, nV) is also a solution to (7) for some $n \in \mathbb{Z}^+$. In this case, however, $\chi \to n\chi$. Therefore increasing the number of faces is equivalent to increasing the genus of the underlying Riemann surface. Thus, for every pattern $\{p, q\}$, there is a minimal solution (F_m, E_m, V_m) to (7), with a minimal number of faces F_m. An interesting class of patterns is one that satisfies $F_m = 1$. In this case, PBCs can be consistently defined for the associated polygon. That is, the edge pairing transformations are sufficient to embed the polygon into a closed Riemann surface of genus $g \geqslant 2$. In this case,

$$(F, E, V) = (1, p/2, p/q);$$

therefore, p is even and $p \geqslant q$. Four prominent infinite families of lattices satisfying this condition were presented in Ref. [14]. Throughout this paper, we will only be focusing on the two families:

$$\begin{aligned} \{4g, 4g\} &: (F_m, E_m, V_m) = (1, 2g, 1), \\ \{2(2g+1), 2g+1\} &: (F_m, E_m, V_m) = (1, 2g+1, 2), \end{aligned} \tag{16}$$

where g denotes the genus of the underlying Riemann surface. In particular, we focus on two patterns: {8, 3} and {10, 3}. The Bravais lattice of the {8, 3} pattern, namely {8, 8}, belongs to the $\{4g, 4g\}$ family for $g = 2$. Meanwhile, the Bravais lattices of the, {10, 3} pattern, namely {10, 5} belongs to the $\{2(2g+1), 2g+1\}$ family for $g = 2$.

The closed Riemann surface obtained by imposing PBCs on the fundamental domain of the {8, 8} Bravais lattice was shown to possess the largest systole among all compact Riemann surfaces of genus $g = 2$ [22]. This property is of particular importance in the context of quantum error correction, as the systole—defined as the length of the shortest noncontractible closed geodesic—provides a lower bound on the length of nontrivial logical operators. Consequently, a larger systole implies a greater code distance, enhancing the code's robustness against logical errors. This property of the {8, 8} Bravais lattice makes the {8, 3} lattice an attractive candidate for constructing high-performance HQECCs. The compactified unit cell of this lattice, which realizes the corresponding genus-2 Riemann surface, is illustrated in Fig. 1.

B. Fuchsian groups

Isometries of the Poincaré disk $\mathbb{D}$ are maps that preserve the hyperbolic metric and, in particular, the hyperbolic distance. We will be concerned with orientation-preserving isometries: these maps are given by elements of the group $PSU(1,1) = SU(1,1)/\{\pm\mathbb{I}\}$. Elements of $PSU(1,1)$ are given by linear transformations of the form

$$z \to gz = \frac{az + b}{b^* z + a^*}, \tag{17}$$

where $g \in PSU(1,1)$, $z \in \mathbb{C}$ and $|a|^2 - |b|^2 = 1$. The full isometry group of $\mathbb{D}$ is given by the semi-direct product $G \ltimes \mathbb{Z}_2$, where elements of $\mathbb{Z}_2$ are the identity and the orientation reversing map $z \to z^*$. In other words, any isometry of $\mathbb{D}$ can be represented by an orientation-preserving map that is either combined or not combined with the orientation-reversing map $z \to z^*$. Had we used the upper-half plane model of the hyperbolic space, the isometry group would have been $PSL(2, \mathbb{R})$. However, because the two models are equivalent, the two groups $PSL(2, \mathbb{R})$ and $PSU(1,1)$ are isomorphic, as expected.

Every closed Riemann surface M can be expressed as a quotient S^2/Γ, $\mathbb{R}^2/\Gamma$, or $\mathbb{H}/\Gamma$, where S^2, $\mathbb{R}^2$, and $\mathbb{H}$ denote the sphere, the Euclidean plane, and the hyperbolic plane,

respectively, and Γ is a discrete subgroup of isometries acting properly discontinuously on each space. If the Euler characteristic of $\chi(M) = 0$, then $g = 1$ and M can be described as $\mathbb{R}^2/\Gamma$. If $\chi(M) > 0$, then $g = 0$ and M can be described as S^2/Γ. Finally, if $\chi(M) < 0$, then $g \geqslant 2$ and M can be described as $\mathbb{H}/\Gamma$. We are concerned with hyperbolic surfaces for which $\chi(M) < 0$ and $g \geqslant 2$. In this case, Γ is called a Fuchsian group. A Fuchsian group Γ is a discrete subgroup of $PSU(1, 1)$; the elements of a Fuchsian group are the transformations that preserve the hyperbolic distance and hence leave the lattice invariant.

The space group of a hyperbolic pattern $\{p, q\}$ is given by

$$SG_{\{p,q\}} = \langle a, b, c \,|\, a^2 = b^2 = c^2 = (ab)^2 = (bc)^p = (ca)^q = \mathbb{I} \rangle.$$

Elements of this group are the set of words consisting of $\{a, b, c, a^{-1}, b^{-1}, c^{-1}\}$, whereas group multiplication is simply a concatenation of words. This group contains orientation-reversing elements and is therefore not a subgroup of Γ. If one considers the quotient by the orientation-reversing elements, one gets a Fuchsian group, that is, a subgroup of $PSU(1, 1)$ given by

$$\Gamma = SG_{\{p,q\}}/\mathbb{Z}_2.$$

The Fuchsian group Γ has the presentation

$$\Gamma = \langle A, B \,|A^p = B^q = (AB)^2 = \mathbb{I} \rangle, \tag{18}$$

where $A = bc$ and $B = ca$. Geometrically, A is a rotation through the center of a face by an angle $\alpha = 2\pi/p$ while B is a rotation through a vertex by an angle $\beta = 2\pi/q$.

Elements of a Fuchsian group are classified as elliptic, parabolic, or hyperbolic if their trace, in the two-dimensional matrix representation, is less than, equal to, or greater than 2, respectively. Elliptic elements have one fixed point; therefore, these elements represent rotations and we denote them by $R(\theta)$, where θ is the angle of rotation. On the other hand, hyperbolic elements have no fixed points; therefore, they are considered translations (or boost transformations). We denote them by $T(\eta)$ where η is the translation parameter.

A Fuchsian translation group is a torsion-free Fuchsian group, that is, no element is of the form $\gamma^n = \mathbb{I}$. It is then obvious that Γ given by (18) is not a Fuchsian translation group. The Fuchsian translation groups associated with Bravais lattices of the form $\{4g, 4g\}$ and $\{2(2g + 1), 2g + 1)\}$ in a hyperbolic space have been given an explicit representation in [14]. It has been shown that, for the two families, only $2g$ generators are independent, and the group is given by the following representation:

$$\Gamma = \langle \gamma_1, \ldots, \gamma_{p_B/2} \,|X_{\{p_B,q_B\}} = \mathbb{I} \rangle, \tag{19}$$

where γ_1 is a boost transformation. The other generators γ_m, where $m = 1, \ldots, 2g$, are obtained by conjugating γ_1 with a rotation by a multiple of $\alpha = 2\pi/p$

$$\gamma_m = R((m-1)\alpha)\gamma_1 \, R(-(m-1)\alpha). \tag{20}$$

Furthermore, the constraint $X_{\{p_B,q_B\}}$ is the same for both families and only depends on the $2g$ independent generators

$$X_{\{p_B,q_B\}} = \gamma_1\gamma_2^{-1} \cdots \gamma_{2g-1}\gamma_{2g}^{-1}\gamma_1^{-1}\gamma_2 \cdots \gamma_{2g-1}^{-1}\gamma_{2g}. \tag{21}$$

From our previous discussion, we can conclude that a hyperbolic polygon has a twofold characteristic, it is the fundamental domain of a symmetry group, and it can be used to construct closed surfaces via edge-pairing identification. We close this section by providing the necessary and sufficient conditions for a p-gon to be embedded in a closed Riemann surface [20]. A hyperbolic p-gon Λ can be embedded in a closed Riemann surface M if the former is the fundamental domain of an orientation-preserving isometry group of $\mathbb{D}$ and if it satisfies the side and angle conditions given as follows:

(1) for each edge e in Λ there is a unique edge e' such that $e' = \gamma(e)$ for $\gamma \in \Gamma$, where γ are the side-pairing transformations,

(2) For each set of vertices identified as a result of the edge-pairing transformations, the sum of the angles has to be equal to 2π.

Theorem 1. (Poincaré) A compact polygon P satisfying the side and angle conditions is the fundamental domain of the group Γ generated by the side-pairing transformations of P.

III. FINITE HYPERBOLIC LATTICES

In this section, we outline a method for constructing finite hyperbolic $\{p, q\}$ lattices given the Fuchsian group of the underlying Bravais lattice. Let $\{p, q\}$ be a hyperbolic lattice whose unit cell U lies within the fundamental domain of the Bravais lattice $\{p_B, q_B\}$. Denote by Γ the Fuchsian group associated with the Bravais lattice $\{p_B, q_B\}$, defined by the group presentation given by (19).

A finite $\{p, q\}$ lattice $\mathcal{L}$ can be constructed by replicating the unit cell U N times, where each copy is generated by applying an element $g \in \Gamma$ to U. To ensure that the resulting lattice remains connected, we adopt a hierarchical approach to constructing these copies. We begin by applying elements of Γ that consist of a single generator—specifically, the generators $\gamma_1, \ldots, \gamma_{2g}$ and their inverses. Next, we apply elements of the form $\gamma_j\gamma_k$ (where $\gamma_j^{-1} \neq \gamma_k$), then proceed to elements with longer word representations, iterating this process until the desired lattice size is achieved.

This finite $\{p, q\}$ lattice can be embedded in a genus $g \geqslant 2$ Riemann surface by imposing appropriate PBCs. Mathematically, imposing the PBCs corresponds to selecting a normal subgroup Γ_{PBC} of index N within the Fuchsian group Γ. The quotient group $\Gamma/\Gamma_{\text{PBC}}$ then represents a finitely generated residual translation group acting on the finite lattice.

To identify index-N normal subgroups of a finitely presented group, computational group theory provides several algorithms [23,24]. In this work, we employ the freely available computational algebra system GAP [25], utilizing the LINS package [26], which implements the low-index normal subgroup algorithm described in [27]. However, this implementation is computationally expensive, as it relies on the Todd-Coxeter coset enumeration procedure [28]. Two distinct sources of inefficiency arise. First, the enumeration becomes impractical for large subgroup index N (e.g., $N > 25$) due to the large number of subgroups enumerated. Second, Fuchsian groups associated with Bravais lattices whose unit cells are embedded in higher-genus Riemann surfaces ($g \geqslant 3$) have presentations with many generators, which significantly

increases the complexity of the enumeration even for moderate values of N. We emphasize that this limitation affects only the subgroup search stage; once a suitable subgroup is identified, the subsequent construction and simulation procedures of the framework apply without modification to an arbitrary genus. In practice, Bravais lattices with unit cells embedded in higher-genus surfaces ($g \geqslant 3$) typically correspond to tessellations with larger values of p and q, leading to higher-weight parity-check operators and consequently less favorable error-correction performance.

To overcome these limitations, a more efficient algorithm was proposed in [29], which enables the computation of normal subgroups of large indices N. This method exploits the fact that if H and K are normal subgroups of a group G, then their intersection $L = H \cap K$ is also a normal subgroup of G. Moreover, if h, k, and l denote the indices of H, K, and L in G, respectively, then

$$\mathrm{lcm}(k, h) \leqslant l \leqslant kh,$$

where equality holds when k and h are coprime.

In practice, we first compute low-index normal subgroups using the LINS package and then take intersections of different subgroups to generate normal subgroups of higher indices. This approach significantly expands the set of accessible normal subgroups beyond what the LINS package alone can achieve, making it a powerful tool for constructing finite hyperbolic lattices embedded in Riemann surfaces.

Let Γ_{PBC} be a subgroup of index N in Γ. Then, Γ has the following coset decomposition:

$$\Gamma = \Gamma_{\mathrm{PBC}} T_1 \cup \Gamma_{\mathrm{PBC}} T_2 \cup \cdots \cup \Gamma_{\mathrm{PBC}} T_N, \tag{22}$$

where the union is disjoint, and

$$T = \{T_1, T_2, \ldots, T_N\} \subset \Gamma \tag{23}$$

is a set of coset representatives, with T_1 being the identity element in Γ. The set T is known as the *right transversal* of Γ_{PBC} in Γ. The finite $\{p, q\}$ lattice $\mathcal{L}$ is then given by

$$\mathcal{L} = \bigcup_{j=1}^{N} T_j U. \tag{24}$$

The choice of a transversal is not unique since an element $T_j \in T$ can be multiplied by any element of Γ_{PBC} while still satisfying (22). However, a physically meaningful choice of a transversal is one that ensures $\mathcal{L}$ forms a connected graph when nearest-neighbor vertices are linked [30].

Topologically, imposing PBCs can be understood in the framework of covering theory. If Γ_{PBC} is a normal subgroup of Γ of finite index N, then the quotient group $\Gamma_N = \Gamma/\Gamma_{\mathrm{PBC}}$ is a finitely presented group of order N. Each quotient group Γ_N serves as the symmetry group of a finite $\{p, q\}$ lattice with N faces of the Bravais lattice.

The minimal representation of the infinite $\{p_B, q_B\}$ lattice is the compactified unit cell obtained as the quotient space $M = \mathbb{D}/\Gamma$, which defines a Riemann surface of genus $g \geqslant 2$. The Fuchsian group Γ is isomorphic to the fundamental group of the quotient space, i.e.,

$$\Gamma \cong \pi_1(M).$$

Similarly, imposing PBCs on a finite lattice $\mathcal{L}$ with N faces by selecting a symmetry group Γ_{PBC} results in a Riemann surface M_N of genus h, where M_N is an N-sheeted cover of M. In this case,

$$\Gamma_{\mathrm{PBC}} \cong \pi_1(M_N),$$

and the Euler characteristic of M_N is given by

$$\begin{aligned} \chi(M_N) &= N\chi(M), \\ 2 - 2h &= N(2 - 2g). \end{aligned} \tag{25}$$

Thus we recover a well-known result in algebraic geometry, the Riemann-Hurwitz formula:

$$h = N(g - 1) + 1. \tag{26}$$

We conclude this section by presenting a systematic framework for constructing a periodic $\{p, q\}$ lattice with an underlying $\{p_B, q_B\}$ Bravais lattice, given a Fuchsian group Γ of $\{p_B, q_B\}$ and a normal subgroup Γ_{PBC} of index N. We outline this framework in Algorithm 1. This algorithm incorporates the procedure outlined in [29,31] as a subroutine to construct the adjacency matrix $A_{\{p,q\}}$ of the $\{p, q\}$ lattice. An illustrative example of applying this procedure to impose PBCs on a hyperbolic lattice is shown in Fig. 2.

IV. CSS QUANTUM CODES ON HYPERBOLIC LATTICES

Topological quantum error correction codes introduced by Kitaev are a special class of stabilizer codes [4,32,33]. These codes are obtained by exploiting the topologies of closed surfaces and are constructed from tessellations of these surfaces, which determine the placement of qubits and stabilizer operators. In the special case where the underlying geometry is Euclidean, one can tessellate a genus one torus by one of three patterns, {4, 4}, {6, 3}, or {3, 6}. Embedding the qubits in a periodic {4, 4} pattern, one obtains the toric code, first introduced by Kitaev [4]. This construction generalizes naturally to hyperbolic geometry once the underlying closed Riemann surface has genus $g \geqslant 2$. In this section, we review the construction of a special class of topological quantum codes, the hyperbolic quantum error correction codes introduced in [6]. We start with a brief review of $\mathbb{Z}_2$ homology that is essential for the construction of HQECCs.

Let M be a closed surface tessellated with a $\{p, q\}$ pattern. The faces, edges, and vertices of the tessellation define the 2-, 1-, and 0-cells of a cellular decomposition of M, respectively. For each dimension $n \in \{0, 1, 2\}$, the n-chains, defined as $\mathbb{Z}_2$ linear combinations of the corresponding geometric elements, form a vector space over $\mathbb{Z}_2$ [34]. We denote this vector space by $C_n(M)$, or simply C_n when the surface M is clear from context. The boundary operators are defined by

$$\partial_n : C_n \to C_{n-1}.$$

That is, the boundary of an element $c_n \in C_n$ is the sum of all the $n - 1$ cells incident on c_n. For example, let v_1, v_2, v_3 be the vertices of a triangle. The boundary of the edge e_{12} connecting v_1 and v_2 is $\partial_1(e_{12}) = v_2 + v_1$. Moreover, the boundary of the triangle f_{123} is given by $\partial_2(f_{123}) = e_3 + e_2 + e_1$, where e_1, e_2, and e_3 are the three edges incident on that triangle. The

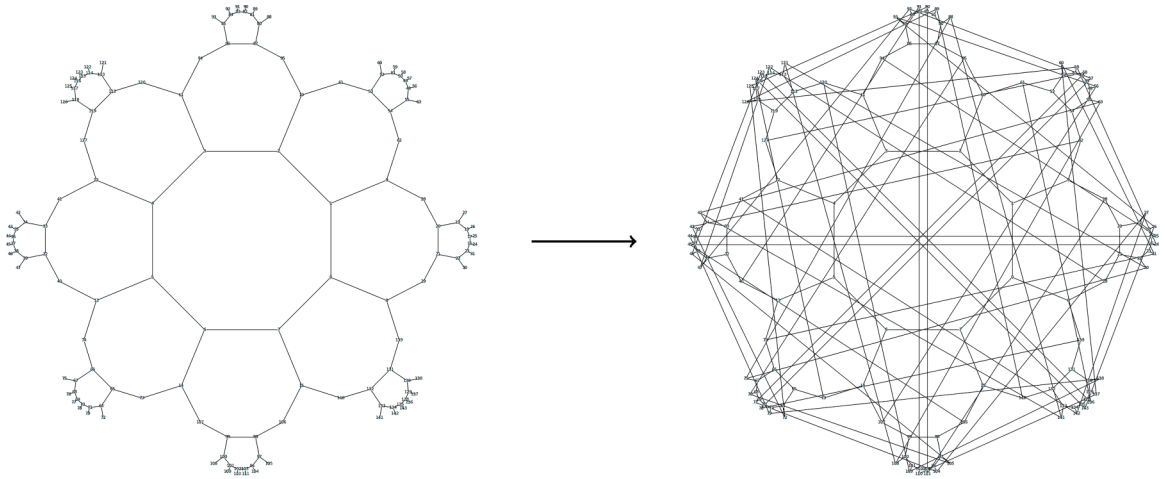

FIG. 2. The left half of the figure shows the finite $\{8, 3\}$ lattice generated by $N = 9$ faces of the Bravais lattice. The graph is generated by replicating the unit cell 8 times using the generators $\gamma_1, \ldots, \gamma_4$ and their inverses. The right half of the figure shows the same lattice after imposing PBCs.

boundary operators satisfy the fundamental property

$$\partial_{n-1} \circ \partial_n = 0, \tag{27}$$

that is, the boundary of a boundary is zero. The boundary operator ∂_n defines two spaces, the cycle space $Z_1 = \ker(\partial_2)$ and the boundary space $B_1 = \mathrm{im}(\partial_2)$. An obvious consequence of (27) is that $B_1 \subset Z_1$. The first homology group is defined as $H_1 = Z_1/B_1$. For an orientable surface, dimension of the first homology group is twice the genus g of the surface, that is $\dim(H_1) = 2g$.

The dual map, the coboundary operator, is defined as follows

$$\delta_n : C_n \to C_{n+1}. \tag{28}$$

Working over $\mathbb{Z}_2$, we identify C_n with its dual via the natural inner product $\langle c, c' \rangle = |c \cap c'| \pmod 2$. Under this identification, the coboundary operator is the transpose of the boundary operator. Equivalently, for an n-cell, δ_n assigns the sum (modulo 2) of all $(n+1)$-cells incident on it, and is extended linearly to arbitrary chains.

The coboundary operator defines the cocycle space

$$Z^1 = \ker(\delta_1),$$

and the coboundary space

$$B^1 = \mathrm{Im}(\delta_0),$$

where $B^1 \subset Z^1$. The first cohomology group is defined as

$$H^1 = Z^1/B^1.$$

With respect to the inner product above, one obtains the orthogonality relations

$$Z^1 = B_1^{\perp}, \quad B^1 = Z_1^{\perp}.$$

Moreover, the groups $H_1(M)$ and $H^1(M)$ are related by the fact that i-cells of a tiled surface M correspond to $(2-i)$-cells of the dual tiling M^*. In the present case $d = 2$, and this correspondence induces an isomorphism

$$* : C_i(M) \longrightarrow C_{2-i}(M^*). \tag{29}$$

Under this identification, the coboundary operator on M corresponds to the boundary operator on the dual tiling,

$$\delta_i = *^{-1} \circ \partial_{2-i} \circ *. \tag{30}$$

Now, we show how to utilize $\mathbb{Z}_2$ homology groups of a tiled surface to define topological quantum codes. A stabilizer code encoding n physical qubits into k logical qubits is a 2^k subspace $\mathcal{C} \subset \mathcal{H}$ of the Hilbert space of n qubits $\mathcal{H} = (\mathbb{C}^2)^{\otimes n}$. The subspace $\mathcal{C} \subset \mathcal{H}$ is defined as the $+1$ eigenspace of an Abelian subgroup S of the Pauli group. The transition from a tessellated surface M to stabilizer codes is done by identifying all edges with qubits. The boundary of each face is identified with a Z parity check operator while the coboundary of each vertex is identified with an X parity check operator. This canonical set of operators are the generators of the stabilizer code. The weight of an operator is the number of qubits on which it acts nontrivially. Thus, for a $\{p, q\}$ tessellation of a closed surface M, the Z parity check operators have weight p, while the X parity check operators have weight q.

The number of physical qubits n equals the number of edges, that is $n = \dim(C_1)$. On the other hand, the number of logical qubits is given by n minus the constraints imposed by the stabilizer operators, that is

$$\begin{aligned} k &= \dim(C_1) - \dim(B_1) - \dim(B^1), \\ &= \dim(Z_1) - \dim(B_1), \\ &= \dim(H_1), \end{aligned} \tag{31}$$

where we used the fact that

$$\dim(B^1) = \dim\left(Z_1^{\perp}\right) = \dim(C_1) - \dim(Z_1).$$

The distance of a stabilizer code is the minimum weight of a Pauli operator that commutes with all stabilizer generators but is not contained in the stabilizer group. In topological CSS surface codes with qubits placed on edges, the Z- and X-distances correspond to the lengths of the shortest homologically nontrivial cycles on the primal and dual tessellations, respectively; the code distance is $d = \min(d_X, d_Z)$. The distance of a quantum CSS surface code can be computed by an

ALGORITHM 1. Construction of a Periodic Regular $\{p, q\}$ Lattice.

Input:

- The values p, q, p_B and q_B defining the $\{p, q\}$ lattice and the $\{p_B, q_B\}$ Bravais lattice.
- A normal subgroup Γ_{PBC} of index N of the Fuchsian group Γ associated with $\{p_B, q_B\}$.

Output: The graph G_{PBC} representing the $\{p, q\}$ lattice after imposing the PBCs.

Step 1: Generate the Unit Cell

- Compute the positions of the unit cell vertices of the $\{p, q\}$ lattice given by (6).

Step 2: Generate the Fuchsian Group Generators

- Construct the generators of the Fuchsian group defined by (19).

Step 3: Generate the Planar Graph G

- Apply a set of elements $\{g_1, g_2, \ldots, g_{N-1}\} \subset \Gamma$ to the unit cell vertices to create additional $N-1$ copies of the unit cell.
- Connect nearest neighbors to construct the planar graph G representing the $\{p, q\}$ lattice prior to imposing the PBCs.
- The resulting graph G consists of bulk vertices, each having degree q, and edge vertices, each having a degree smaller than q.

Step 4: Construct the Adjacency Matrix $A_{\{p,q\}}$

- Let V be the adjacency matrix of the unit cell of the $\{p, q\}$ lattice and initialize
$$A_{\{p,q\}} = \mathbb{I}_N \otimes V,$$
where $\mathbb{I}_N$ is the identity matrix of size $N \times N$.
- For each $j = 1, \ldots, p_B$, define the intercell matrix I_j to represent the edges connecting each unit cell to its neighbor in the direction of γ_j, where γ_j belongs to the set of generators of Γ and their inverses. The matrix I_j specifies the intercell connectivity pattern induced by γ_j.
- Let $A_{\{p_B,q_B\}}$ be the adjacency matrix of the Bravais lattice.
- For each pair of neighboring faces $(A_{\{p_B,q_B\}})_{n,m} = 1$ connected by γ_j, update $A_{\{p,q\}}$:
$$A_{\{p,q\}} \to A_{\{p,q\}} + U \otimes I_j$$
where U is an $N \times N$ matrix with elements all zeros except $U_{nm} = 1$.

Step 5: Imposing the PBCs

- Augment the graph G by adding the edges in $A_{\{p,q\}}$ that are not already present in G. These additional edges arise from imposing the PBCs.
- We denote the resulting graph G_{PBC}; this graph is embedded in a Riemann surface whose genus is given by equation (26).

algorithm due to Bravyi, as described in [7]. For completeness, we outline this procedure in Algorithm 2.

ALGORITHM 2. Computation of the Distance d_Z in a Topological CSS Code.

Input:

- Graph $G_{PBC} = (V, E)$ representing the finite hyperbolic lattice with PBCs.
- The set of logical operators $\{\bar{X}_1, \bar{X}_2, \ldots, \bar{X}_k\}$.

Output: The distance d_Z of the topological CSS code.

for $j = 1, \ldots, k$ **do**

Select a logical operator $\bar{X}_j$ and define its qubit support as $E(\bar{X}_j) \subset E$. Then, construct an auxiliary graph $\tilde{G}$ as follows:

- Create two copies of each vertex $v \in V$, labeled as v^+ and v^-.
- Initialize the edge set of $\tilde{G}$:
  - For each edge $e = (u, v) \in E$:
    - If $e \in E(\bar{X}_j)$, add edges (u^+, v^-) and (u^-, v^+) to $\tilde{G}$.
    - If $e \notin E(\bar{X}_j)$, add edges (u^+, v^+) and (u^-, v^-) to $\tilde{G}$.

Compute the shortest path distance $d(v^+, v^-)$ in $\tilde{G}$ for each vertex $v \in E(\bar{X}_j)$.

Compute the code distance: Set $d_Z = \min d(v^+, v^-)$ over all $v \in E(\bar{X}_j)$ and all logical operators $\bar{X}_j$.

Computation of d_X: The same procedure can be applied to the set of logical operators $\{\bar{Z}_1, \bar{Z}_2, \ldots, \bar{Z}_k\}$ to determine the distance d_X.

V. HYPERBOLIC CYCLE BASIS

In this section, we present an algorithm for computing all plaquettes and logical operators in an HQECC. A plaquette is defined as a trivial cycle of length p, corresponding to a face of the finite $\{p, q\}$ hyperbolic lattice. In contrast, logical operators are associated with nontrivial cycles on the underlying Riemann surface M of genus $g \geqslant 2$. While our method is inspired by the fundamental cycle basis algorithm of Paton [35] and the minimum cycle basis algorithm of Kavitha *et al.* [36], it is fundamentally distinct. Rather than relying purely on graph-theoretic properties, our approach leverages the intrinsic geometry of the surface to construct a specific cycle basis that is essential for the implementation of HQECCs.

Let G be a connected, finite, undirected graph with E edges and V vertices. A cycle C in G is any subgraph in which every vertex has an even degree. Each cycle C can be represented by an incidence vector $I_C \in \mathbb{Z}_2^{\otimes E}$, where $I_C^e = 1$ if and only if edge e is part of the cycle C (i.e., $e \in C$). The vector space generated by these incidence vectors is known as the cycle space of G. A cycle basis of G is a set of cycles that spans the cycle space. That is, given a cycle basis $\{C_1, C_2, \ldots, C_n\}$ of

the graph G, any cycle C in G can then be expressed as:

$$C = \sum_{j=1}^{n} \alpha_j C_j, \tag{32}$$

where $\alpha_j \in \mathbb{Z}_2$.

A *fundamental cycle basis* (FCB) is a specific type of cycle basis for a graph G. To construct an FCB, one first selects a spanning tree $T(G)$ of the graph. For each edge $e \in G \setminus T(G)$ that is not part of the spanning tree, a unique cycle is formed by combining e with the unique path in $T(G)$ that connects the endpoints of e. Each such cycle is called a fundamental cycle, and the collection of all such cycles constitutes the fundamental cycle basis.

Since a spanning tree on a graph with V vertices contains exactly $V-1$ edges, the number of nontree edges is $E-V+1$. Therefore the dimension of the cycle space spanned by a fundamental cycle basis is:

$$\dim(\mathrm{FCB}(G)) = E - V + 1.$$

However, not every cycle basis is necessarily a fundamental cycle basis. Any set of linearly independent $E-V+1$ cycles that span the cycle space is a valid cycle basis of the graph. Since we are looking for a specific set of cycles, our choice is highly constrained by the hyperbolic geometry of the underlying lattice. To accommodate these constraints, the algorithm constructs a cycle basis that is not a fundamental cycle basis, but instead consists of cycles more naturally aligned with the geometry of the hyperbolic lattice.

Before presenting our algorithm for computing all plaquettes and logical operators of a given HQECC, we first prove that the set of plaquettes, excluding one plaquette, along with the set of logical operators, forms a valid cycle basis for the underlying graph.

Theorem 2. Let G_{PBC} be a graph representing a finite hyperbolic $\{p, q\}$ lattice embedded in a closed Riemann surface M of genus $g \geqslant 2$. Suppose that G_{PBC} has V vertices, E edges, and F faces. Then a valid cycle basis for G_{PBC} is given by the union of:

(1) $F-1$ contractible cycles of length p, each corresponding to a plaquette of the hyperbolic lattice, and

(2) $2g$ noncontractible cycles each of length $l > p$ that generate the first homology group $H_1(M)$.

We denote this cycle basis as the *Hyperbolic Cycle Basis (HCB)*.

Proof. For the first part of the proof, we proceed by contradiction. Suppose that HCB is not a valid cycle basis for G_{PBC}, then there is a cycle $C \in Z_1$ that does not belong to the vector space spanned by the elements of HCB. Based on the discussion in Sec. IV, there are two types of cycles in the graph G_{PBC}, trivial cycles that are elements of B_1 and nontrivial cycles that are elements of $Z_1 \setminus B_1$. The trivial cycles are either plaquettes or product of plaquettes. Since each edge is contained in exactly two plaquettes, the sum of all plaquette cycles (mod 2) is the identity. Hence, only a set of $F-1$ plaquette cycles are linearly independent, and the sum of these plaquette cycles equals the remaining plaquette cycle in the graph. Therefore any set $\{Pl_1, Pl_2, \ldots, Pl_{F-1}\}$ of $F-1$ plaquette cycles spans the space B_1. On the other hand, the vector space of nontrivial cycles is spanned by a set of $2g$ linearly independent cycles $\{h_1, h_2, \ldots, h_{2g}\}$ that generate the first homology group of the underlying Riemann surface $H_1(M)$ and can not be expressed as a sum of plaquettes. To ensure linear independence, we require that these cycles are pairwise orthogonal. That is, $|h_j \cap h_k| \equiv 0 \pmod 2$, for all $j, k \in \{1, \ldots, 2g\}$.

Now, consider any cycle C in the graph G_{PBC}. If $C \in B_1$, then C can be expressed as

$$C = \sum_{j=1}^{F-1} \alpha_j Pl_j, \tag{33}$$

where $\alpha_j \in \mathbb{Z}_2$. On the other hand, if $C \in Z_1 \setminus B_1$, then C can be expressed as

$$C = \sum_{j=1}^{2g} \beta_j h_j, \tag{34}$$

where $\beta \in \mathbb{Z}_2$. Thus C belongs to the vector space spanned by the elements of the HCB. In other words,

$$\mathrm{HCB} = \{Pl_1, Pl_2, \ldots, Pl_{F-1}, h_1, h_2, \ldots, h_{2g}\}$$

is a valid cycle basis for G_{PBC}.

To complete the proof, we need to show that the set HCB has the correct dimension of a cycle basis, that is

$$\dim(\mathrm{HCB}) = E - V + 1. \tag{35}$$

Combining (8) and (9),

$$\begin{aligned} 2g &= 2 - \chi(M), \\ &= 2 - F + E - V. \end{aligned} \tag{36}$$

Therefore the dimension of the HCB is

$$\begin{aligned} \dim(\mathrm{HCB}) &= 2g + (F-1), \\ &= 2 - F + E - V + (F-1), \\ &= E - V + 1. \end{aligned}$$

■

Having established that the HCB forms a valid cycle basis for the graph G_{PBC}, we now provide additional context before presenting Algorithm 3, which computes the HCB. The computational complexity of Algorithm 3 is analyzed separately in Appendix A.

The graph G, constructed by replicating the unit cell of the underlying Bravais lattice, contains only trivial cycles—those belonging to the boundary subgroup B_1. Thus any cycle of length p in G corresponds to a valid plaquette. Upon imposing PBCs, the resulting graph G_{PBC} contains $2E/p$ plaquettes, along with additional nontrivial cycles some of which might be of length p. Therefore the plaquettes introduced as a result of imposing PBCs can be identified as those cycles of length p that intersect all other plaquettes in at most one edge. Moreover, there are $2g$ independent, mutually commuting Pauli-Z logical operators and $2g$ mutually commuting Pauli-X logical operators, each associated with a distinct logical qubit. Importantly, these logical operators come in anticommuting pairs, satisfying $X_j Z_j = -Z_j X_j$ for all $j = 1, \ldots, 2g$. In this construction, Pauli-Z logical operators are represented by nontrivial cycles in the primal graph G_{PBC}, whereas Pauli-X

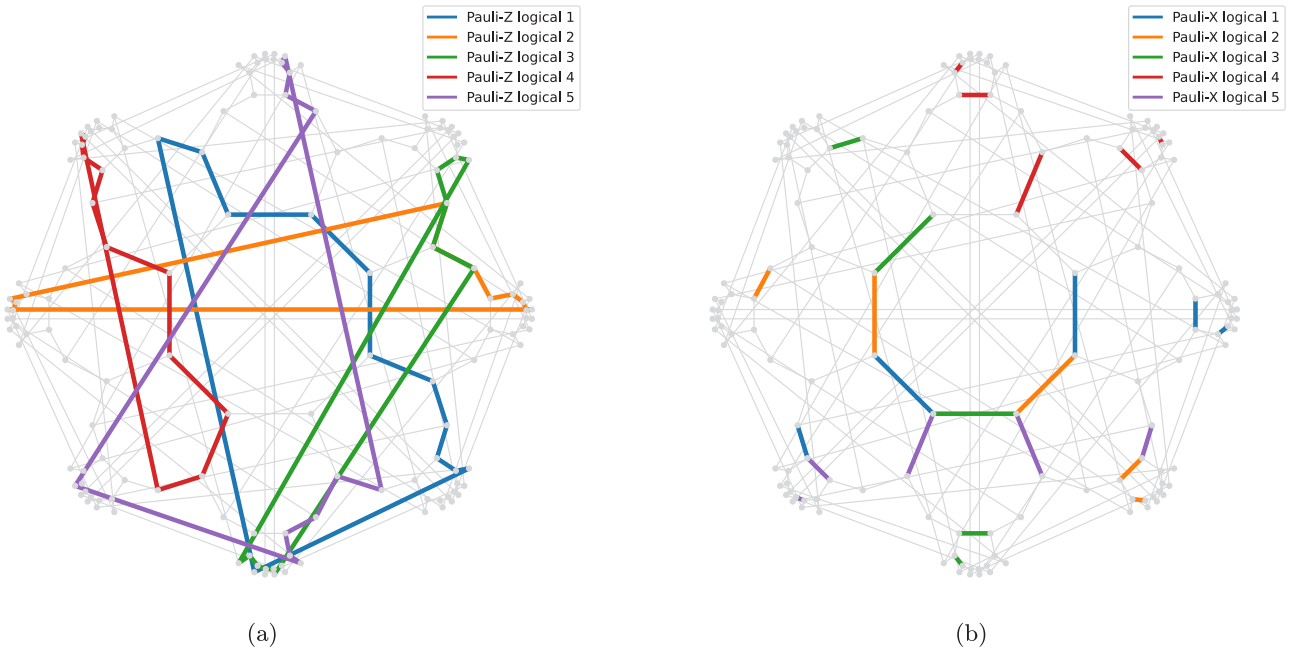

FIG. 3. Logical operators are defined by canonical nontrivial cycles: Z-type operators correspond to nontrivial cycles on the primal graph, while X-type operators correspond to nontrivial cycles on the dual graph. (a) Z-type logical operators. (b) X-type logical operators.

logical operators correspond to nontrivial cycles in the dual graph G^*_{PBC}, which translate to nontrivial cocycles in G_{PBC}. Identifying these nontrivial cycles requires more than selecting cycles of length greater than p, as many such cycles are trivial, belong to B_1, and can be formed as products of plaquettes. Therefore it is necessary to search specifically for cycles of length greater than p that can not be obtained as products of plaquettes.

Identifying a higher-order trivial cycle can be done iteratively as follows. For every two plaquettes F_1 and F_2 that are neighbours, the higher-order trivial cycle is given by their symmetric difference. We call these cycles twofold trivial cycles. The number of twofold trivial cycles in a lattice equals the number of edges in this lattice, and they all have length $2p-2$. Now consider a twofold trivial cycle C_p. This cycle intersects some plaquettes in the lattice in at most two edges. For every plaquette intersecting this cycle, their product is a threefold trivial cycle that has length $3p-2(i+1)$ where $i \in \{1,2\}$ is the length of the intersection between C_p and the given plaquette. This process can be used iteratively to construct n-fold trivial cycles. An n-fold trivial cycle is given by the symmetric difference between an $(n-1)$-fold trivial cycle and an intersecting plaquette. Its length belongs to $np-2(n-1)-2k$, where $k \in \{0,\ldots,n-2\}$. In practice, it is typically sufficient to compute trivial cycles involving up to threefold intersections, as these cycles already exhibit relatively large lengths, specifically $3p-4$ and $3p-6$. Such large lengths are generally adequate to identify all nontrivial cycles present in the lattice.

Fig. 3 illustrates examples of logical operators obtained using the HCB algorithm for the $\{8,3\}$ lattice with $N=9$. It is important to note that, whereas every cycle necessarily includes at least one edge introduced by the imposition of PBCs, the same requirement does not apply to cocycles.

Finally, we summarize our systematic framework for constructing HQECCs and computing various code parameters.

(1) Select a hyperbolic tessellation $\{p,q\}$ of the Poincaré disk with an associated Bravais lattice $\{p_B,q_B\}$.

(2) Apply Algorithm 1 to construct a periodic $\{p,q\}$ lattice. Let G denote the graph representation of the $\{p,q\}$ lattice before imposing the PBCs, and let G_{PBC} denote the corresponding graph after imposing the PBCs.

(3) Input G and G_{PBC} into Algorithm 3 to compute all plaquette cycles and determine the logical operators of the HQECC.

(4) Use G_{PBC} obtained from Algorithm 1 and the set of logical operators obtained from Algorithm 3 as inputs into Algorithm 2 to compute the code distance d.

(5) As discussed before, the number of physical qubits is the number of edges E in G_{PBC}, and the number of logical qubits is given by $2h$, where h is the genus of the underlying Riemann surface defined in (26). These are the parameters of the HQECC.

(6) Using this framework, one can also simulate a HQECC and make an estimate of the code's error threshold corresponding to a specified error model.

VI. NUMERICAL SIMULATIONS

To demonstrate the utility and versatility of this framework, we apply it to simulate two representative HQECCs derived from two hyperbolic tessellations $\{8,3\}$, $\{10,3\}$ of the Poincaré disk, each associated with a distinct underlying Bravais lattice. The $\{8,3\}$ lattice admits an underlying $\{8,8\}$ Bravais lattice belonging to the $\{4g,4g\}$ family for $g=2$. On the other hand, the $\{10,3\}$ lattice has an underlying $\{10,5\}$ Bravais lattice belonging to the $\{2(2g+1),2g+1\}$ family for $g=2$.

ALGORITHM 3. Hyperbolic Cycle Basis Algorithm.

Input:

- Graph $G = (V, E')$ of the finite hyperbolic lattice before imposing PBCs.
- Graph $G_{PBC} = (V, E)$ of the lattice after imposing PBCs.
- Number of plaquettes $N_F = 2E/p$ in G_{PBC}.
- Genus of the embedding Riemann surface g.

Output: `HCB`, `all_plaquettes`, `Z_logicals`, `X_logicals`.

Step 1: Initial Plaquette Detection
Find all cycles of length p in G and store them in two sets: `HCB` and `all_plaquettes`.
Let $N_i =$ `len(HCB)`.

Step 2: Add Remaining Plaquettes from G_{PBC}
Find all cycles of length p in G_{PBC} that are not already in `HCB`; call this set `RP`.
foreach $C_k \in$ *RP* **do**
  if $|C_k \cap C_p| \in \{0, 1\}$ *for all* $C_p \in$ *HCB* **then**
    if len(HCB) $< N_F - 1$ **then**
      Append C_k to `HCB`.
      Append C_k to `all_plaquettes`.
    else if len(HCB) $== N_F - 1$ **then**
      Append C_k to `all_plaquettes`.
      break

Step 3: Generate Higher-Order Trivial Cycles
Generate n-fold trivial cycles iteratively to obtain all trivial cycles up to a given maximum length l.
Store the output in the set `trivial_cycles`.

Step 4: Find Non-Trivial Cycles (Z-type Logical Operators)
Find all cycles C_k of length $p < x \leq l$ in G_{PBC} that are not in `trivial_cycles`; denote this set as `potential_logicals`.
while len(Z_logicals) $< 2g$ **do**
  foreach $Z_l \in$ *potential_logicals* **do**
    if $|Z_l \cap Z_n| \equiv 0 \pmod 2$ *for all* $Z_n \in$ *Z_logicals* **then**
      Append Z_l to `HCB`.
      Append Z_l to `Z_logicals`.
      break

Step 5: Find Non-Trivial Cocycles (X-type Logical Operators)

- Construct the dual graph G^*_{PBC}, corresponding to a $\{q, p\}$ lattice, by placing a vertex at the center of each plaquette in G_{PBC} and connecting two vertices whenever their corresponding plaquettes share a common edge.
- Apply steps 1-4 to the graph G^*_{PBC} to find the set of non-trivial cycles in G^*_{PBC}.
- Convert each non-trivial cycle C in G^*_{PBC} into its corresponding cocycle in G_{PBC} and denote the resulting set of cocycles by `dual_potentials`.

while len(X_logicals) $< 2g$ **do**
  $k \leftarrow len($`X_logicals`$)$
  foreach $X_l \in$ *dual_potentials* **do**
    if
    $(|X_l \cap X_n| \equiv 0 \pmod 2$ *for all* $X_n \in$ *X_logicals*$)$
    and $(|X_l \cap Z_{k+1}| \equiv 1 \pmod 2)$ **then**
      Append X_l to `X_logicals`.
      break

For each simulation, we employ a phenomenological noise model in which each qubit experiences a Pauli-Z error independently with probability p, and we assume that the syndrome measurements are error-free. After that, we use a minimum-weight perfect matching (MWPM) decoder in order to infer the occurrence of a logical error. A logical error occurs if the product of the real and inferred errors anticommutes with any of the X-type logical operators $\{X_1, \ldots, X_{2g}\}$. The results of the simulations are depicted in Fig. 4. An archived version of the simulation code used in this work is available at [37], while an actively maintained repository is hosted on GitHub at [38].

For each HQECC, we compute the logical-error probability p_L as a function of the physical error probability p for several code sizes. To estimate the threshold, we examine the family of curves and identify the parameter regime where the behavior with respect to system size inverts: for p below the threshold, p_L decreases with increasing code size, whereas for p above the threshold, p_L increases with increasing code size. The quoted threshold values $\approx 3\%$ and $\approx 2\%$ for the $\{8, 3\}$ and the $\{10, 3\}$ HQECCs, respectively, are obtained from this criterion and should be understood as approximate estimates. These estimates are also in agreement with prior results showing that HQECCs thresholds range from $1 - 3\%$ when decoded with MWPM [6]. While faster decoders, such as union–find and belief propagation, have been proposed for surface codes, their decoding performance remains suboptimal compared with MWPM [39–41].

The presented HQECCs possess a Z/X asymmetry; therefore, the Z-error threshold differs from the X-error threshold. This contrasts with the toric code, which possesses Z/X symmetry and exhibits an error threshold of approximately 10% for the same noise model [5].

The reduced threshold for the HQECCs compared with the toric code arises for two main reasons. First, the code distance of an HQECC grows only logarithmically with the number of physical qubits, $d = \Theta(\log n)$, in contrast to the toric code whose distance scales with the linear system size $d = \Theta(\sqrt{n})$ [6,42]. This behavior reflects the tradeoff between distance and encoding rate: HQECCs achieve a constant encoding rate, whereas the toric code has vanishing rate in the thermodynamic limit. Consequently, the modest increase of the distance observed in Fig. 4 is not a peculiarity of the specific tessellations considered, but a generic feature of HQECCs, and this logarithmic distance scaling increases the logical failure probability compared with the toric code.

Second, hyperbolic tessellations correspond to closed surfaces of higher genus. Such higher-genus surfaces contain a large number of homologically nontrivial cycles, which increases the number of possible logical error paths and makes HQECCs more susceptible to logical failures. Together, these effects lead to lower thresholds for HQECCs compared with the toric code.

While our simulation present Z-error thresholds, the same framework can be applied to compute the X-error thresholds for both HQECCs. Generally, the framework naturally extends to circuit-level noise models with faulty syndrome measurements. In this case, repeated rounds of syndrome extraction stack the periodic hyperbolic lattice of Algorithm 1 into a space–time decoding graph, where the plaquette cycles

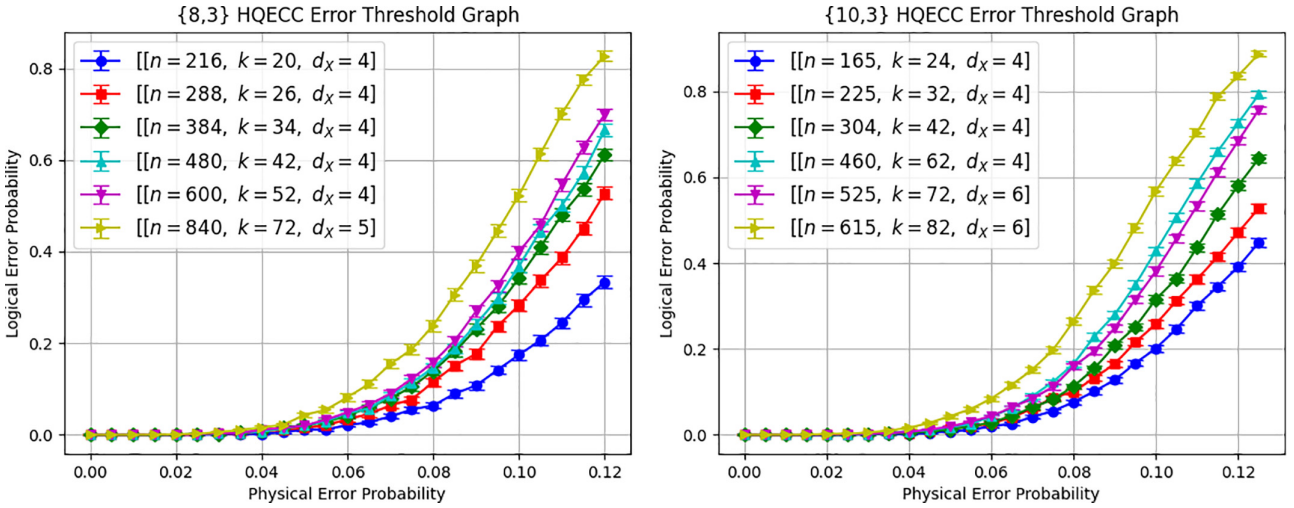

FIG. 4. Estimated Z-error thresholds for two HQECCs with different tilings: {8, 3} with a threshold of approximately 3% (left) and {10, 3} with a threshold of approximately 2% (right). Each data point was obtained via 10^4-trial Monte Carlo simulations under a phenomenological noise model, where each qubit experiences an independent Pauli-Z error with probability p per trial.

and representatives of the logical operators derived from lgorithm 3 are defined on each temporal layer, and measurement faults introduce additional timelike edges between successive layers.

VII. RESULTS

In this work, we presented a systematic framework for constructing and benchmarking CSS quantum error correction codes on hyperbolic lattices. Central to this framework is the HCB algorithm, which efficiently computes all plaquette cycles and logical operators, represented by nontrivial cycles and cocycles on the hyperbolic lattice. The HCB algorithm enables a scalable and automated approach to defining the support of the code's stabilizers and logical operators—an otherwise intractable task for large lattices if done manually. This capability is essential for benchmarking large-scale HQECCs.

To demonstrate the capabilities of the framework, we applied it to compute the Z-error thresholds of two HQECCs constructed from the {8, 3} and {10, 3} hyperbolic tessellations. The simulations were performed using a phenomenological noise model in which each qubit independently experiences a Pauli-Z error with probability p. Logical failures were identified when the error and its correction anticommute with an X-type logical operator, and thresholds were estimated using Monte Carlo sampling over 10^4 trials per data point.

Although we focused on HQECCs as representative examples, the framework can be adapted, with appropriate modifications, to benchmark other CSS codes defined on hyperbolic lattices. In particular, it extends naturally to codes with time-dependent stabilizer measurements, such as CSS Floquet codes [43,44]. Floquet codes are defined by a periodic measurement schedule in which the instantaneous stabilizer group, and hence the chosen representatives of the logical operators, evolve in time. These codes can be implemented on 3-colorable hyperbolic lattices. Algorithm 1 can be used to construct regular hyperbolic lattices of the form $\{p, 3\}$ with PBCs. For suitable choices of p, these lattices are 3-colorable, and therefore compatible with such Floquet measurement schedules [45]. Although the measured stabilizers and explicit logical representatives evolve during the Floquet cycle, they are associated with canonical plaquette cycles and nontrivial homology classes of the underlying lattice, respectively, which can be obtained from Algorithm 3 [12,43]. Over one period, errors correspond to paths in the space–time decoding graph, and a logical failure occurs when an error path forms a nontrivial cycle that anticommutes with a logical operator. Algorithm 2, which finds the shortest nontrivial spatial cycle with this property, thus provides a geometric baseline for estimating the code distance in the Floquet setting, with the precise distance depending on the full space–time decoding graph and the underlying noise model.

This work serves as a first step toward a systematic study of how lattice geometry influences the performance of CSS quantum error correction codes on hyperbolic lattices, a key question for optimizing future fault-tolerant quantum architectures. A companion study applying this framework to benchmark CSS Floquet codes on hyperbolic lattices will be presented elsewhere.

ACKNOWLEDGMENTS

We thank the anonymous referees for their careful reading of the manuscript and for their constructive and valuable feedback, which significantly improved the clarity and quality of this work. We thank Joseph Maciejko for fruitful discussions around the use of GAP. We would also like to thank Srinivas Vamsi Parasa for helping in optimizing and parallelizing the Python code used to simulate the HQECCs. We acknowledge the use of Qiskit's AerSimulator for executing the HQECC circuits. S.R. has been partially supported by a Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant and a Canadian Tri-Agency New Frontiers in Research Fund (NFRF) Exploration Grant. A.A.M. was supported during this work by the Discovery Grant of SR.

DATA AVAILABILITY

The data that support the findings of this article are openly available [37,38].

APPENDIX A: HCB ALGORITHM COMPLEXITY ANALYSIS

In this Appendix, we analyze the computational complexity of the HCB algorithm. Throughout, we consider the graph of a regular $\{p, q\}$ hyperbolic lattice. For such lattices, the number of vertices V, edges E, and faces F scale linearly with one another according to (7), i.e.,

$$F = \Theta(E) = \Theta(V). \tag{A1}$$

All complexity estimates can, therefore, be expressed in terms of E. The HCB algorithm restricts attention to cycles of length at most l, where l is an algorithmic cutoff parameter.

1. Step 1. Initial plaquette detection

In Step 1, the algorithm identifies all cycles of length p in the graph G prior to imposing PBCs. Since G admits a planar embedding without nontrivial cycles, every cycle of length p corresponds to the boundary of a plaquette. Therefore the number of length p cycles in G is $\mathcal{O}(F)$.

Enumerating all simple cycles in a graph is a classical problem, with Johnson's algorithm being its classical solution [46]. For undirected graphs, an asymptotically optimal algorithm exists [47]. Here we use an implementation of Gupta's and Suzumura's algorithm to enumerate cycles of length up to l; this algorithm runs in $\mathcal{O}((\eta + n)(l-1)D^l)$, where η is the number of cycles enumerated, n is the number of vertices, and D is the average degree of the graph [48]. For a $\{p, q\}$ regular hyperbolic lattice, $D = q$ and $n = V$. Moreover, in this step, $l = p$; therefore, Step 1 runs in

$$T_1 = \mathcal{O}(F + V) = \mathcal{O}(E) \tag{A2}$$

2. Preprocessing. Detecting length-l cycles

For Steps 2 and 4, we need to detect all cycles up to a given length l. Cycles of length p will constitute the remaining plaquettes needed for Step 2, while higher-order cycles will be utilized to detect nontrivial cycles that represent Z-type logical operators. In practice, l is initialized to a modest constant $l > p$ and increased iteratively until $2g$ independent nontrivial cycles are obtained in Step 4. The cost of finding all cycles up to a given length l is

$$T_{l-\text{cycles}} = \mathcal{O}((\eta + V)(l-1)q^l), \tag{A3}$$

where η is the total number of cycles found.

3. Step 2. Adding remaining plaquettes from G_{PBC}

In Step 2, the algorithm identifies any remaining plaquette cycles in G_{PBC} and augments the HCB set while enforcing the required intersection constraints. The number of candidate plaquettes is $\mathcal{O}(F)$, and the size of the HCB set grows to $\mathcal{O}(F)$.

The dominant cost in this step arises from checking pairwise intersections between candidate plaquettes and elements of the HCB set. Since each plaquette contains p edges, each intersection test takes $\mathcal{O}(1)$ time. In the worst case, the nested loop performs $\mathcal{O}(F^2)$ such checks, yielding a total runtime

$$T_2 = \mathcal{O}(F^2) = \mathcal{O}(E^2). \tag{A4}$$

4. Step 3. Generating higher-order trivial cycles

In Step 3, the algorithm constructs higher-order trivial cycles by taking symmetric differences of plaquette boundaries. An n-fold trivial cycle corresponds to the boundary of a connected cluster of n adjacent plaquettes.

The construction of twofold trivial cycles examines all face pairs. Since each face has constant size p, each intersection test costs $\mathcal{O}(1)$. Therefore the twofold stage runs in $\mathcal{O}(F^2) = \mathcal{O}(E^2)$.

For $n \geqslant 3$, the algorithm iterates over all previously generated $(n-1)$-fold cycles and checks each cycle with all F plaquettes. If the $(n-1)$-fold cycle and the plaquette are adjacent, the symmetric difference between them is taken to generate an n-fold cycle. For each candidate pair, the symmetric difference between a cycle of size at most l and a plaquette of constant size p costs $\mathcal{O}(l)$. Consequently, the runtime for fixed n is

$$T_n = \mathcal{O}(FN_{n-1}l), \tag{A5}$$

where N_{n-1} is the number of $(n-1)$-fold trivial cycles produced in the previous step. Summing over $n = 3, \ldots, n_{\max}$, we get

$$T_{n-\text{cycles}} = \mathcal{O}\left(\sum_{n=3}^{n_{\max}} FN_{n-1}l\right). \tag{A6}$$

Let N_{tot} be the total number of trivial cycles generated, then

$$T_3 = \mathcal{O}(E^2 + EN_{\text{tot}}l). \tag{A7}$$

5. Step 4. Identifying Z-type logical operators

In Step 4, the algorithm identifies nontrivial cycles representing Z-type logical operators by testing candidate cycles against previously selected logical operators using a parity-of-intersection criterion.

Let γ denote the number of candidate cycles considered in this step. Each candidate cycle has length $\mathcal{O}(l)$, and the number of logical operators selected is $2g$. By (14), $g = cF + 1$, where $c > 0$ is a constant depending on $\{p, q\}$. Therefore $g = \Theta(F) = \Theta(E)$. For each candidate, the algorithm checks its intersection parity with all previously selected logical operators. Each such check takes $\mathcal{O}(l)$ time; therefore, the total runtime of Step 4 is

$$T_4 = \mathcal{O}(\gamma gl) = \mathcal{O}(\gamma El). \tag{A8}$$

6. Step 5: Identifying X-type logical operators

Step 5 constructs X-type logical operators by working on the dual lattice. This step consists of two parts.

First, the algorithm constructs the dual graph G^*_{PBC} corresponding to the $\{q, p\}$ hyperbolic tiling. By (7), the number of edges in the primal and the dual graph is the same, while the number of faces and vertices are interchanged $F \leftrightarrow V$ in the two graphs. We start by placing vertices in the center of each

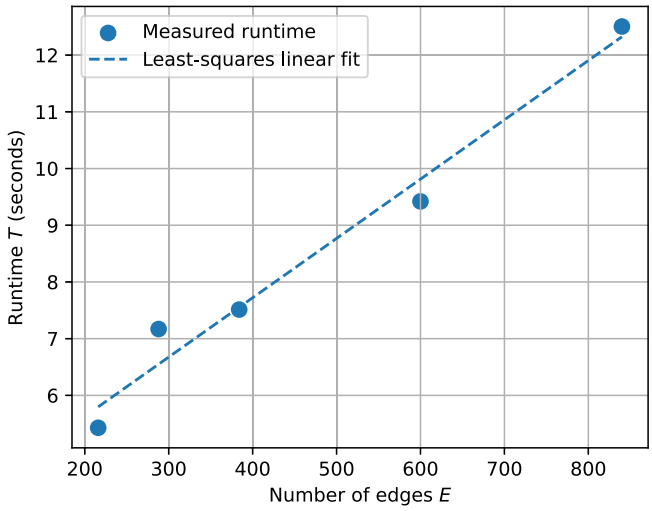

FIG. 5. Measured runtime required to execute the HCB algorithm for five {8, 3} tilings considered in this work as a function of the number of edges E. The cycle–enumeration cutoff was set to $l = 25$ for the primal graph and $l = 10$ for the dual graph. The dashed line shows a least-squares linear fit to the data.

plaquette and then connect vertices corresponding to adjacent faces. The double loop to connect the vertices takes $\mathcal{O}(F^2)$ time.

Second, Steps 1–4 of the HCB algorithm are applied to the dual graph G^*_{PBC} in order to identify nontrivial cycles in the dual lattice. Because G^*_{PBC} is again a bounded-degree hyperbolic tiling with constant face size, and the number of edges in the primal and the dual graphs is the same, the same complexity bounds derived above apply.

7. Summary

The runtime complexity of the HCB algorithm is dominated by three components: the enumeration of cycles up to a prescribed cutoff length l (A3) and the computation of intersections used to construct higher-order trivial cycles (A7) and identify logical operators (A8). These three components are executed separately for the primal and dual graphs.

To complement the asymptotic analysis, we measured the runtime required to generate the HCB graph for the {8, 3} tiling as a function of E. The results are shown in Fig. 5. For the system sizes accessible in this work ($E < 1000$), the construction completes within a few seconds on a single CPU core. The figure serves to document the observed computational runtime for the instances considered here and is not intended to represent the asymptotic scaling behavior of the algorithm.

[1] D. Gottesman, An introduction to quantum error correction and fault-tolerant quantum computation, arXiv:0904.2557.

[2] B. M. Terhal, Quantum error correction for quantum memories, Rev. Mod. Phys. **87**, 307 (2015).

[3] N. P. Breuckmann and J. N. Eberhardt, Quantum low-density parity-check codes, PRX Quantum **2**, 040101 (2021).

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

[5] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill, Topological quantum memory, J. Math. Phys. **43**, 4452 (2002).

[6] N. P. Breuckmann and B. M. Terhal, Constructions and noise threshold of hyperbolic surface codes, IEEE Trans. Inf. Theory **62**, 3731 (2016).

[7] N. P. Breuckmann, C. Vuillot, E. Campbell, A. Krishna, and B. M. Terhal, Hyperbolic and semi-hyperbolic surface codes for quantum storage, Quantum Sci. Technol. **2**, 035007 (2017).

[8] C. D. Albuquerque, R. Palazzo, Jr., and E. Silva, Topological quantum codes on compact surfaces with genus $g \geqslant 2$, J. Math. Phys. **50**, 023513 (2009).

[9] I. H. Kim, Quantum codes on Hurwitz surfaces, Ph.D. thesis, Massachusetts Institute of Technology, 2007.

[10] M. B. Hastings and J. Haah, Dynamically generated logical qubits, Quantum **5**, 564 (2021).

[11] C. Gidney, M. Newman, A. Fowler, and M. Broughton, A fault-tolerant honeycomb memory, Quantum **5**, 605 (2021).

[12] O. Higgott and N. P. Breuckmann, Constructions and performance of hyperbolic and semi-hyperbolic floquet codes, PRX Quantum **5**, 040327 (2024).

[13] J. G. Ratcliffe, *Foundations of Hyperbolic Manifolds* (Springer, New York, NY, 2006).

[14] I. Boettcher, A. V. Gorshkov, A. J. Kollár, J. Maciejko, S. Rayan, and R. Thomale, Crystallography of hyperbolic lattices, Phys. Rev. B **105**, 125118 (2022).

[15] A. J. Kollár, M. Fitzpatrick, and A. A. Houck, Hyperbolic lattices in circuit quantum electrodynamics, Nature (London) **571**, 45 (2019).

[16] T. Bzdušek and J. Maciejko, Flat bands and band-touching from real-space topology in hyperbolic lattices, Phys. Rev. B **106**, 155146 (2022).

[17] A. J. Kollár, M. Fitzpatrick, P. Sarnak, and A. A. Houck, Line-graph lattices: Euclidean and non-euclidean flat bands, and implementations in circuit quantum electrodynamics, Commun. Math. Phys. **376**, 1909 (2020).

[18] A. F. Beardon, *The Geometry of Discrete Groups* (Springer, New York, NY, 2012), Vol. 91.

[19] S. Katok, *Fuchsian Groups* (University of Chicago Press, Chicago, IL, 1992).

[20] J. Stillwell, *Geometry of Surfaces* (Springer, New York, NY, 1995).

[21] M. P. Do Carmo, *Differential Geometry of Curves and Surfaces: Revised and Updated Second Edition* (Courier Dover Publications, Mineola, NY, 2016).

[22] P. Schmutz, Riemann surfaces with shortest geodesic of maximal length, Geom. Funct. Anal. (GAFA) **3**, 564 (1993).

[23] M. Conder and P. Dobcsányi, Applications and adaptations of the low index subgroups procedure, Math. Comput. **74**, 485 (2005).

[24] D. Firth, An algorithm to find normal subgroups of a finitely presented group, up to a given finite index, Ph.D. thesis, University of Warwick, 2005.

[25] The GAP Group, GAP—Groups, Algorithms, and Programming, version 4.11.1 (2021), https://www.gap-system.org/.
[26] F. Rober, LINS, provides an algorithm for computing the normal subgroups of a finitely presented group up to some given index bound, version 0.9 (2024), https://gap-packages.github.io/LINS/.
[27] A. Dietze and M. Schaps, Determining subgroups of a given finite index in a finitely presented group, Can. J. Math. **26**, 769 (1974).
[28] J. A. Todd and H. S. M. Coxeter, A practical method for enumerating cosets of a finite abstract group, Proc. Edinburgh Math. Soc. **5**, 26 (1936).
[29] A. Chen, J. Maciejko, and I. Boettcher, Anderson localization transition in disordered hyperbolic lattices, Phys. Rev. Lett. **133**, 066101 (2024).
[30] J. Maciejko and S. Rayan, Automorphic bloch theorems for hyperbolic lattices, Proc. Natl. Acad. Sci. **119**, e2116869119 (2022).
[31] T. Tummuru, A. Chen, P. M. Lenggenhager, T. Neupert, J. Maciejko, and T. Bzdušek, Hyperbolic non-abelian semimetal, Phys. Rev. Lett. **132**, 206601 (2024).
[32] M. Freedman, A. Kitaev, M. Larsen, and Z. Wang, Topological quantum computation, Bull. Am. Math. Soc. **40**, 31 (2003).
[33] D. Gottesman, Stabilizer codes and quantum error correction, Ph.D. thesis, California Institute of Technology, 1997.
[34] A. Hatcher, *Algebraic Topology* (Cambridge University Press, Cambridge, 2002).
[35] K. Paton, An algorithm for finding a fundamental set of cycles of a graph, Commun. ACM **12**, 514 (1969).
[36] T. Kavitha, K. Mehlhorn, D. Michail, and K. E. Paluch, An algorithm for minimum cycle basis of graphs, Algorithmica **52**, 333 (2008).
[37] A. A. Mahmoud and K. M. Ali, HQECC-Threshold Simulation Code, Zenodo (2026), https://doi.org/10.5281/zenodo.18784824.
[38] A. A. Mahmoud and K. M. Ali, HQECC-Threshold GitHub Repository (2026), https://github.com/AhmeedAdelMahmoud/HQECC-Threshold.
[39] N. Delfosse and N. H. Nickerson, Almost-linear time decoding algorithm for topological codes, Quantum **5**, 595 (2021).
[40] S. Huang, M. Newman, and K. R. Brown, Fault-tolerant weighted union-find decoding on the toric code, Phys. Rev. A **102**, 012419 (2020).
[41] J. Old and M. Rispler, Generalized belief propagation algorithms for decoding of surface codes, Quantum **7**, 1037 (2023).
[42] N. Delfosse, Tradeoffs for reliable quantum information storage in surface codes and color codes, in *Proceedings of the 2013 IEEE International Symposium on Information Theory* (IEEE, Istanbul, Turkey, 2013), pp. 917–921.
[43] M. Davydova, N. Tantivasadakarn, and S. Balasubramanian, Floquet codes without parent subsystem codes, PRX Quantum **4**, 020341 (2023).
[44] M. S. Kesselring, J. C. Magdalena de la Fuente, F. Thomsen, J. Eisert, S. D. Bartlett, and B. J. Brown, Anyon condensation and the color code, PRX Quantum **5**, 010342 (2024).
[45] W. S. Soares Jr. and E. B. da Silva, Hyperbolic quantum color codes, Quantum Inf. Comput. **18**, 306 (2018).
[46] D. B. Johnson, Finding all the elementary circuits of a directed graph, SIAM J. Comput. **4**, 77 (1975).
[47] E. Birmelé, R. Ferreira, R. Grossi, A. Marino, N. Pisanti, R. Rizzi, and G. Sacomoto, Optimal listing of cycles and st-paths in undirected graphs, in *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms* (SIAM, New Orleans, LA, 2013), pp. 1884–1896.
[48] A. Gupta and T. Suzumura, Finding all bounded-length simple cycles in a directed graph, arXiv:2105.10094.