Generalized surface codes and packing of logical qubits

Nicolas Delfosse[1,2] Pavithran Iyer[3] and David Poulin[3]

June 22, 2016

Abstract

We consider a notion of relative homology (and cohomology) for surfaces with two types of boundaries. Using this tool, we study a generalization of Kitaev's code based on surfaces with mixed boundaries. This construction includes both Bravyi and Kitaev's [3] and Freedman and Meyer's [2] extension of Kitaev's toric code. We argue that our generalization offers a denser storage of quantum information. In a planar architecture, we obtain a three-fold overhead reduction over the standard architecture consisting of a punctured square lattice.

1   Surface codes

Kitaev's toric code [1] is one of the most emblematic examples of topological quantum codes. It is defined by local constraints on qubits placed on a torus. The properties of the code depend on the topology of the surface. For instance, the number of encoded qubits is determined by the genus of the surface and the minimum distance is the length of the shortest cycle with non-trivial homology. Such a code can be defined on an arbitrary closed surface.

For practical purposes, a planar layout of the qubits is desirable. Unfortunately, in such a case the prescription of Kitaev to construct quantum codes yields a trivial code that preserves just a single state. The work of Freedman and Meyer [2] and Bravyi and Kitaev [3] extends Kitaev's construction in two directions.

(i) Kitaev construction can be extended to a surface with boundaries, that is a surface punctured with holes [2].

(ii) Two different kinds of boundaries called open and closed boundaries (also called rough and smooth respectively) can be introduced along the outer boundary of a planar lattice [3].

These modifications increase the degeneracy of the ground space of the Hamiltonian, allowing for a non-trivial planar surface code. Punctured planar lattices have been proposed as quantum memory or quantum computing architecture where logical operations are realized by braiding holes [4, 5, 6, 7, 8, 9, 10, 11]. Understanding and optimizing their performance is a central question for the physical implementation of quantum information processing.

[1]Department of Physics and Astronomy, University of California, Riverside, CA, USA
[2]IQIM, California Institute of Technology, Pasadena, CA, USA
[3]Département de Physique and Institut Quantique, Université de Sherbrooke, Québec, Canada
Corresponding author: Nicolas Delfosse - ndelfoss@caltech.edu

Surprisingly, mixed boundaries (partially open and partially closed boundaries) have been examined previously only for the outer boundary of a planar lattice as in (ii), but not along all the punctures. In the present work, we combine the two ideas (i) and (ii), producing better surface codes. First, we introduce a family of generalized surface codes based on punctured surfaces where any holes can have partially open and partially closed boundaries. Justifying the commutation relations between stabilizer generators as well as computing the parameters of these codes is non-trivial and requires an in-depth study of a notion of relative homology. We determine a closed formula for the parameters of these generalized surface codes and we describe graphically their logical operators. Then, we propose a planar architecture based on holes with mixed boundaries which improves over the parameters of standard constructions of two-dimensional surface codes. We obtain a three-fold reduction of the overhead compared to the square lattice punctured with closed holes [9].

Besides providing new constructions of surface codes, the formalism developed here is necessary in order to optimize the performance of different fault-tolerant architecture. If it is clear that surface codes can make a quantum computer fault-tolerant, the details of the architecture of such a fault-tolerant quantum computer are still to be determined. Optimizing the design of surface codes can lead to a much more favourable overhead. Our formalism, which encompasses all the previously considered constructions of surface codes [1, 2, 3, 4, 5, 6, 7, 8, 9, 22, 10, 11], provides an ideal framework to compare and optimize quantum computing architectures based on surface codes.

Generalized surface codes are defined in Section 3. The definition of these codes and the computation of their parameters rely on a particular notion of relative homology of surfaces with boundaries that we study in Section 4. Finally, a planar architecture based on generalized surface codes is proposed in Section 5. This last section, which focuses on the problem of optimizing the packing of logical qubits in a planar lattice, can be read independently.

2   Background on stabilizer codes

Let us recall the definition of *stabilizer codes* [12]. In what follows, I, X, Y and Z denote the usual Pauli matrices. Pauli operators are n-fold tensor products of Pauli matrices $i^a P_1 \otimes P_2 \otimes \cdots \otimes P_n$ where $a \in \mathbb{Z}_4$. Denote by $\mathcal{P}_n$ the set of n-qubit Pauli operators.

A stabilizer code of length n is defined as the common $(+1)$-eigenspace of a family of n-qubit commuting Pauli operators $S_1, S_2, \ldots S_r$. Equivalently, it is the degenerate ground space of the Hamiltonian $H = -\sum_i S_i$. It is a 2^{n-r}-dimensional subspace of $(\mathbb{C}^2)^{\otimes n}$, whenever the r Pauli operators S_i are independent. The quantum code then encodes $k = n - r$ qubits into n qubits and its parameters are denoted as $[[n, k]]$. The group generated by the operators S_i is denoted $\mathcal{S}$ and is called the stabilizer group of the quantum code $C(\mathcal{S})$.

Assume that an encoded state $|\psi\rangle$ is subjected to a Pauli error $E \in \mathcal{P}_n$. The system is then in the state $E|\psi\rangle$. The correction procedure for stabilizer codes is based on the *syndrome measurement*, that is the measurement of the observables S_i for all $i = 1, \ldots, r$. These commuting operators can be measured simultaneously, providing a measurement outcome ± 1 for all i. The outcome of the measurement of a stabilizer S_i is $(-1)^{\sigma_i}$ where $\sigma_i \in \mathbb{F}_2$ is defined by the equation $E S_i = (-1)^{\sigma_i} S_i E$. This defines the syndrome $\sigma(E) = (\sigma_1, \ldots, \sigma_r) \in \mathbb{F}_2^r$. Whenever the system being measured is in a state $|\psi\rangle$ that belongs to the code space the syndrome is trivial. Hence, a non-trivial syndrome indicates the presence of an error.

Among stabilizer codes, *CSS codes* [13, 14] are those defined by r_X operators chosen from $\{I, X\}^{\otimes n}$ and $r_Z = r - r_X$ operators chosen from $\{I, Z\}^{\otimes n}$. Writing Pauli errors as $E = i^a E_Z E_X$ with $E_Z \in \{I, Z\}^{\otimes n}$ and $E_X \in \{I, X\}^{\otimes n}$, we see that the syndrome can be partitioned as a pair of vectors (σ_Z, σ_X), where σ_Z (resp. σ_X) contains the measurement outcomes of the r_X operators of X-type (resp. the r_Z operators of Z-type). As suggested by the notation, the vector σ_Z depends only on E_Z and σ_X depends only on E_X. Since quantum states are defined up to a phase, we can assume that the phase i^a of the error is trivial. Our goal will be to identify E_X from its syndrome σ_X and E_Z from its syndrome σ_Z.

An error which has a trivial syndrome is called a *logical operator* or a *logical error*. These errors preserve the code space. A stabilizer, that is an element of the stabilizer group, is a particular logical operator that has a trivial action on encoded qubits. By a *non-trivial logical operator*, we mean a logical operator that is not a stabilizer. Up to the stabilizer group S, the set of logical operators, denoted $N(S)$ (for normalizer), is a group that has the same structure as the k-qubit Pauli group. More precisely, the quotient $N(S)/S$ is generated by $2k$ operators $\bar{X}_1, \bar{Z}_1 \ldots, \bar{X}_k, \bar{Z}_k$ which satisfy the same relations as the k-qubit Pauli operators $[\bar{P}_i, \bar{Q}_j] = [P_i, Q_j]$. A family which satisfies these relations is called the *symplectic basis* of the logical operators.

The *minimum distance d* of a stabilizer code is defined to be the minimum weight of a non-trivial logical error. It is a proxy indicator for the error-correction capability of the code. When d is known, it is added to the parameters of the code denoted as $[[n, k, d]]$. For CSS codes, the minimum distance is reached either by an error $E_Z \in \{I, Z\}^{\otimes n}$ or by an error $E_X \in \{I, X\}^{\otimes n}$. One can thus obtain the minimum distance as $d = \min\{d_X, d_Z\}$ where d_Z is the minimum weight of a non-trivial logical error $E_Z \in \{I, Z\}^{\otimes n}$ and d_X is the minimum weight of a non-trivial logical error $E_X \in \{I, X\}^{\otimes n}$.

3 Definition and properties of generalized surface codes

Stabilizer codes and CSS codes can be considered as a quantum generalization of classical linear codes. The main obstacle which arises when we try to 'quantize' a classical code construction is the constraint that stabilizers must commute. In Kitaev's construction, qubits are placed on the edges of a cellulation of a surface, X-type stabilizers correspond to vertices and Z-type stabilizers correspond to faces. The commutation relations are then an immediate consequence of basic homological properties (the boundary of a boundary is trivial). The homology of the surface is also a crucial tool to compute the parameters of Kitaev's code.

In this section, we first introduce the surfaces that support our code construction. Then, we define generalized surface codes and we state our main result which provides a full description of the parameters of these codes. The validity of the construction as well as the proof of our main result is delayed to Section 4 through the study of relative homology.

3.1 Surface with open and closed boundaries

The goal of this section is to define an appropriate notion of surface in order to incorporate all extensions of Kitaev's code. Starting from closed surfaces, we bring two modifications. First, we authorize surfaces with boundaries. Then, each edge on the boundary will be declared either open or closed. We choose the term open or closed for its topological connotation. A surface with only closed boundaries is a closed surface. Those boundaries are sometimes

called rough and smooth in quantum information. This notion of surface will allow us to consider a general construction of surface codes which encompasses both Kitaev's original construction [1], its generalizations [3, 2] and extends it to any surface punctured with mixed holes that have open as well as closed edges on their boundaries, as one can see in Figure 1.
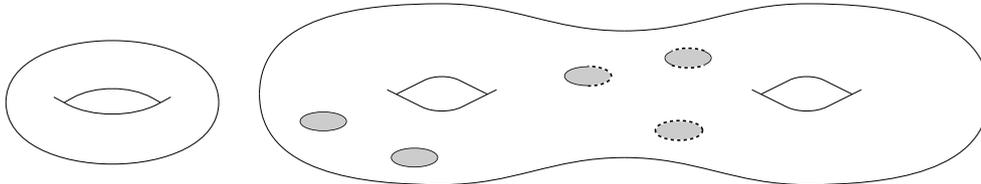

Figure 1: Left: A closed surface of genus 1. Right: A surface of genus 2 with boundaries. The surface is punctured with 5 holes resented by dark disks. The boundary of these holes can be open (dotted lines) or closed (continuous lines). Here 2 holes are closed and 1 hole is open. The 2 remaining holes have both open and closed boundaries.

Since it is only the combinatorial structure of the surface that plays a role in the definition of topological codes, we consider surfaces constructed by gluing a finite set of faces together along their edges in a coherent way. Formally, a *combinatorial surface* or a simply a *surface* is a cellulation of a compact 2-manifold S (possibly with boundaries). In other words, it is a triple (V, E, F) where $G = (V, E)$ is a finite graph embedded on the surface such that each connected component of $S \backslash G$ is a disk which is a face $f \in F$ of the cellulation. We often identify a face f with the set of edges on its boundary, *i.e.* we regard a face f as a subset $f \subset E$. Topologically a face is a disk and its boundary is a set of edges that form a cycle in the graph (V, E). We assume that G is embedded without overlapping edges or multiple edges. We also suppose that no edge belongs to the same face twice and that two faces share at most one edge.

Now that we have a surface $G = (V, E, F)$, let us define its *boundary*. An edge of G is defined to be a boundary if it is incident to a single face of G. The corresponding faces are called boundary faces. The endpoints of boundary edges are the boundary vertices. We denote the sets of boundary vertices, edges and faces, respectively by $\partial V, \partial E$ and ∂F.

Boundary elements will be either *open* or *closed*. First, a subset of edges living on the boundary of the tiling is declared to be open. Once open edges are defined, a vertex of the boundary is said to be open if it is incident to an open edge. Analogously an open face is a face of the boundary containing an open edge. An edge, respectively a vertex or a face of the boundary that is not open is said to be closed. This defines a partition of the boundary $\partial V = \partial_O V \sqcup \partial_C V$, $\partial E = \partial_O E \sqcup \partial_C E$, and $\partial F = \partial_O F \sqcup \partial_C F$, where ∂_O denotes the open subset and ∂_C denotes the closed subset.

The set of non-open vertices, edges and faces are denoted respectively by $\mathring{V} = V \backslash \partial_O V$, $\mathring{E} = E \backslash \partial_O E$ and $\mathring{F} = F \backslash \partial_O F$. By analogy with topology, one may be tempted to define the set $\mathring{X}$ as $X \backslash \partial X$. Our definition makes the statements of our results simpler and emphasizes the similarities with standard results in homology.

3.2 Generalized surface codes

In this section we provide a unified definition for surface codes defined on surfaces with or without boundaries.

In order to construct a quantum error-correcting code from a surface $G = (V, E, F)$, we place a qubit on each non-open edge of G. This leads to a global state living in the Hilbert space $\mathcal{H} = \otimes_{e \in \mathring{E}} \mathbb{C}^2$. Denote by X_e, respectively Z_e, the Pauli operator acting as X, respectively Z, on the qubit indexed by e and which is the identity on the other qubits.

Definition 3.1. *The* surface code *associated with the surface $G = (V, E, F)$ is defined to be the stabilizer code of stabilizer group $S = \langle X_v, Z_f \mid v \in \mathring{V}, f \in F \rangle$, where*

$$X_v = \prod_{\substack{e \in \mathring{E} \\ v \in e}} X_e \quad and \quad Z_f = \prod_{\substack{e \in \mathring{E} \\ e \in f}} Z_e.$$

In other words, this surface code is the ground space of the Hamiltonian

$$H = -\sum_{v \in \mathring{V}} X_v - \sum_{f \in F} Z_f.$$

The commutation between the operators X_v and Z_f, for all $v \in \mathring{V}$ and for all $f \in F$, is ensured by Lemma 4.2.

When the surface G has no boundaries, this definition coincides with Kitaev's original construction [1]. When the surface has only closed boundaries it is Freedman and Meyer's generalization [2]. Finally, we recover Bravyi and Kitaev's construction [3] when the surface is a sphere punctured with a single hole.

Our first objective is to establish a closed formula for the parameters of generalized surface codes. Let us recall the key ingredients in the case of closed connected surfaces.

- The rank of the Z-stabilizer group $S_Z = \{Z_f \mid f \in F\}$ is given by $|F| - 1$.

- A Z-error $E_Z \in \{I, Z\}^{\otimes n}$ has trivial syndrome if and only if its support is a cycle and it is a stabilizer if and only if this cycle is homologically trivial. This proves that the minimum distance d_Z is the minimum length of a cycle with non-trivial homology.

- Replacing the graph G by its dual exchanges the role of X and Z, proving that d_X is the minimum length of a cycle of the dual graph G^* with non-trivial homology and that the X-stabilizer group has rank $|F^*| - 1 = |V| - 1$

Denote by $\chi(G)$ the Euler-Poincaré characteristic of the surface. Based on the first and the third items, $k = 2 - \chi(G)$ which is equal to $2g$ when the surface is orientable and g when it is not orientable. The minimum distance is given by the last two items as the minimum length of a non-trivial cycle of G or its dual G^*.

In the presence of open and closed boundaries, these 3 items are altered. First, the rank of the stabilizer group depends on the boundaries in a non-trivial way (See Theorem 3.2 below). Second, one needs an appropriate notion of homology that reproduces the behaviour of errors and syndrome measurements. Lastly, one also needs to define a dual graph G^* in such a way that graph duality coincides exactly with the duality between X and Z-errors.
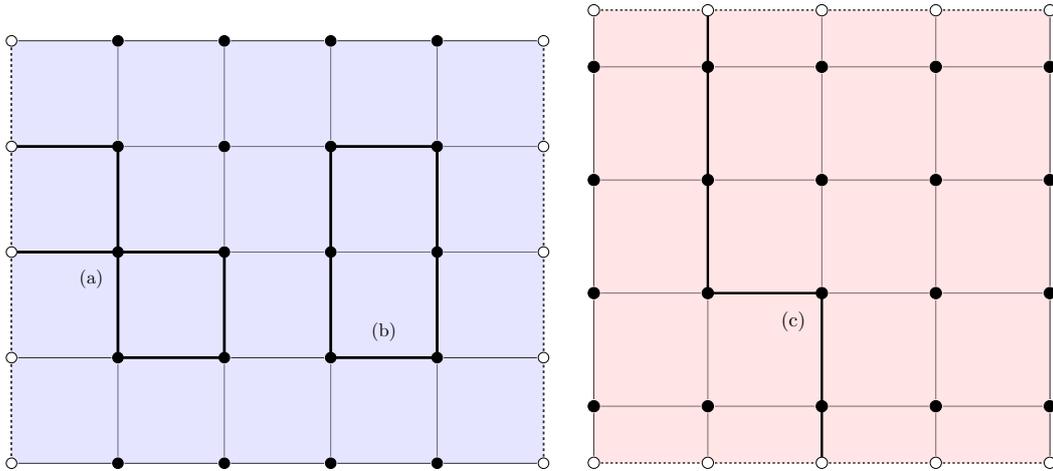
Figure 2: A surface with 2 open boundaries and 2 closed boundaries and its dual. Open edges are represented by dotted lines and open vertices by white nodes. Duality switches open and closed boundaries. Three cycles are depicted. The cycles (a) and (b) are trivial relative cycles whereas the relative cycle (c) is non-trivial.

3.3   Statement of the main result

An important part of the present article is devoted to the study of the homology of surfaces with open and closed boundaries and to the construction of an appropriate notion of dual for these surfaces. In order to state our main result, we first provide a rough definition of these notions. Rigorous definitions of these objects will be given in Section 4.

In order to describe the minimum distance of the code, we need to introduce a generalization of the notion of cycle for surfaces with boundaries. Following Bravyi and Kitaev, we define a *relative cycle* of a surface with boundaries as a set of non-open edges which meet each non-open vertex an even number of times. Three relative cycles are shown in Figure 2. For instance, a path connecting two open vertices, like (a) and (c) in the figure, is a relative cycle though it is not a standard cycle. The support of an error E_Z with syndrome zero is a relative cycle since it can only be detected by measuring the operators X_v associated with the non-open vertices $v \in \mathring{V}$.

A relative cycle is said to be *homologically trivial* or simply *trivial* if it is the boundary of a subset of faces of the surface. Examples of trivial and non-trivial cycles are depicted in Figure 2. By definition of generalized surface codes, a trivial cycle induces an error E_Z which is a stabilizer (again this cycle is the support of the error).

We will also have to extend the definition of the dual to surfaces with open and closed boundaries. Inspired by the standard example of the square lattice with open and closed boundaries depicted in Figure 2 [3], the dual graph will be defined in such a way that open and closed boundaries are switched under duality. This duality will be introduced and motivated in section 4.5.

Our first result is a closed formula for the parameters of generalized surface codes defined on any surface, orientable or not, with boundaries or not, and where boundaries can be either open or closed. We use the notation $\kappa_{\overline{\partial_C E}}(G)$ for the number of connected components of G
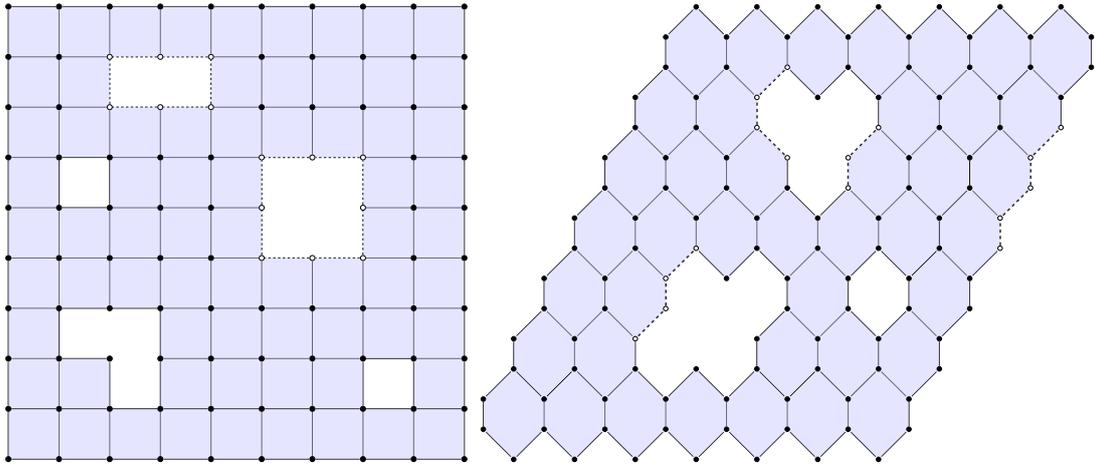
Figure 3: Left: A surface of genus 0 with 6 holes encoding k=4 qubits. Right: A surface with partially open holes encoding 6 qubits. Dashed edges represent open edges.

containing no closed boundary edge $e \in \partial_C E$ and the notation $\kappa_{\overline{\partial_O V}}(G)$ for the number of connected components of G containing no open vertex $v \in \partial_O V$.

Theorem 3.2. *The parameters $[[n, k, d]]$ of the generalized surface code associated with a surface $G = (V, E, F)$ are*

- $n = |\mathring{E}|$,

- $k = -|\mathring{V}| + |\mathring{E}| - |F| + \kappa_{\overline{\partial_O V}}(G) + \kappa_{\overline{\partial_C E}}(G)$,

- $d = \min\{d_Z, d_X\}$ *where d_Z is the minimum length of a non-trivial relative cycle of G and d_X is the minimum length of a non-trivial relative cycle of G^*.*

The first result is obvious. Note that in the last item both the dual graph G^* and the notion of cycle and trivial cycle are generalized to the case of surfaces with open and closed boundaries. This theorem will be proven through the study of the homology group of surface with open and closed boundaries, which is the focus of Section 4. It is the combined conclusion of Corollary 4.8 and Corollary 4.9.

Remark: In order to avoid surface codes with minimum distance $d = 1$, one can assume that any edge whose both endpoints are open is an open edge. See the remark at the end of Section 4.5.

3.4 Applications to surfaces with uniform boundaries

As a first application, we find the parameters of surface codes defined over a surface punctured with open holes and closed holes.

Corollary 3.3. *Let G be a connected surface with b_c closed-boundary holes and b_o open-boundary holes. The number of logical qubits encoded in the corresponding surface code is*

$$\begin{cases} k = 2g + \min\{b_c - 1, 0\} + \min\{b_o - 1, 0\} & \text{if } G \text{ is orientable,} \\ k = g + \min\{b_c - 1, 0\} + \min\{b_o - 1, 0\} & \text{if } G \text{ is not orientable.} \end{cases}$$

As an illustration, a surface of genus 0 with 4 closed holes (the external boundary is a closed hole) and 2 open holes encoding $k = 4$ qubits is represented in Figure 3.

Proof. If G is an orientable surface of genus g with no boundary, then its Euler-Poincaré characteristic $\chi(G) = |V| - |E| + |F|$ equals $2 - 2g$. For a non-orientable surface, we have $\chi(G) = 2 - g$. The case of closed surface is thus a straightforward application of Theorem 3.2. If G is connected and contains b_c closed boundaries, then in the orientable case, $\chi(G) = 2g - b_c$, which in turn yields $k = 2g - b_c - 1$ according to Theorem 3.2. In the non-orientable case, $k = g - b_c - 1$ again according to Theorem 3.2.

The case of a surface with some open boundaries is less standard. Assume that $b_o + b_c > 0$. Denote $\bar{G}$ the surface obtained from G by closing all the boundaries. Any open edge is now declared to be closed. This allows us to apply the previous result to $\bar{G}$. Let us now compare $k(G)$ and $k(\bar{G})$:

$$k(G) - k(\bar{G}) = |V| - |\mathring{V}| + |\mathring{E}| - |E| + \kappa_{\overline{\partial_O V}}(G) + \kappa_{\overline{\partial_C E}}(G) - \kappa_{\overline{\partial_O V}}(\bar{G}) - \kappa_{\overline{\partial_C E}}(\bar{G})$$

Any boundary is either an open cycle or a closed cycle. This implies that the number of open edges $|E| - |\mathring{E}|$ is equal to the number of open vertices $|V| - |\mathring{V}|$.

The terms corresponding to closed boundaries add up to

$$\kappa_{\overline{\partial_C E}}(G) - \kappa_{\overline{\partial_C E}}(\bar{G}) = \begin{cases} 1 & \text{if } b_c = 0 \\ 0 & \text{if } b_c > 0 \end{cases}$$

and the contribution of open boundaries is

$$\kappa_{\overline{\partial_O V}}(G) - \kappa_{\overline{\partial_O V}}(\bar{G}) = \begin{cases} 0 & \text{if } b_o = 0 \\ -1 & \text{if } b_o > 0 \end{cases}$$

Overall, we obtain $k(G) = k(\bar{G}) + \delta_{b_c=0} - \delta_{b_o>0}$. which is $2g + b - 1 + \delta_{b_c=0} - \delta_{b_o>0}$ or $g + b - 1 + \delta_{b_c=0} - \delta_{b_o>0}$ depending on the orientability of the surface. This proves that the result holds. □
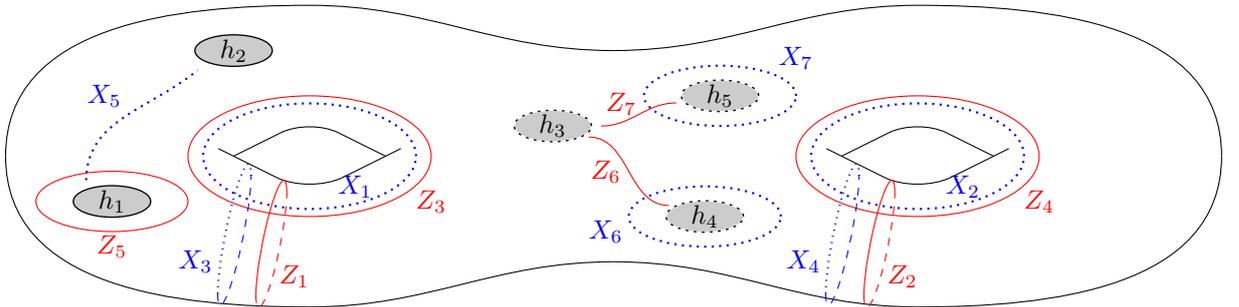

Figure 4: A symplectic basis of the set of logical operators for an orientable surface of genus 2 punctured with 5 holes.

For instance, Figure 4 represents an orientable surface of genus 2 punctured by 2 closed holes h_1 and h_2 and 3 open holes h_3, h_4 and h_5, resulting in a code that encodes $k = 4+1+2 = 7$

logical qubits. A symplectic basis $\bar{X}_1, \bar{Z}_1, \ldots, \bar{X}_k, \bar{Z}_k$ of logical operators is shown in the above figure. Let us detail the construction of this symplectic basis. As usual, for each handle we define 2 pairs of logical $\bar{X}_i, \bar{Z}_i$ of logical operators. When the surface is not orientable, it is a connected sum of g projective planes and we have only one logical of each type per projective plane. It remains to define the logical operators associated with the b_c closed holes and the b_o open holes. Consider the closed holes $h_1, \ldots, h_{b_c}$. The last hole h_{b_c} (the top closed hole h_2 in Figure 4) will play a special role. For the $b_c - 1$ holes $h_1, \ldots, h_{b_c-1}$, define a Z-logical operator $\bar{Z}_i$ whose support is a loop α_i around the cycle h_i for $i = 1, \ldots, b_c - 1$. Then, take a family of paths $\beta_1, \ldots, \beta_{b_c-1}$ in the dual graph G^* such that β_i connects the hole h_{b_c} to the hole h_i and $\beta_i \cap \alpha_j = \delta_{i,j}$ for all i, j. By $\beta_i \cap \alpha_j = \delta_{i,j}$, we mean that the only path α_j which shares an edge with β_i (up to duality) is α_i, and moreover, these paths have exactly one common edge. Each of these $b_c - 1$ paths β_i, define a logical operator $\bar{X}_i$ whose support is β_i. The assumption $\beta_i \cap \alpha_j = \delta_{i,j}$ ensures the commutation relations required for a symplectic basis. Then, we construct $b_o - 1$ pairs of logical operators based on the b_o open holes. This can be done by the same procedure in the dual graph.

3.5 Applications to surfaces with mixed boundaries

We now consider codes based on surface punctured with holes which may have mixed (open and closed) boundaries.

Corollary 3.4. *If G is a connected surface of genus g with b holes that contains $m > 1$ non-cyclic disjoint open paths then the number of logical qubits of the corresponding surface code is*
$$\begin{cases} k = 2g + b + m - 2 & \text{if } G \text{ is orientable,} \\ k = g + b + m - 2 & \text{if } G \text{ is not orientable.} \end{cases}$$

By non-cyclic path, we simply mean a path that is not a cycle. Since these paths live on the boundary of a hole, the only kind of cyclic open path that may appear in this context is a totally open hole.

A similar formula was given for the number of logical qubits of colour codes [15].

Proof. We proceed as in the proof of Corollary 3.3. In order to determine the number of logical qubits $k(G)$ of the code based on the surface G, we introduce the surface $\bar{G}$ which is obtained from G by closing all its boundaries. The open subsets $\mathring{V}$ and $\mathring{E}$ of the surface $\bar{G}$ are then trivial. We already know that $k(\bar{G}) = 2g + b$ or $g + b$, let us relate $k(G)$ and $k(\bar{G})$.

Assume that G is punctured by b holes indexed by $i = 1, \ldots, b$. The perimeter of the i-th hole is a cycle γ_i of length ℓ_i. If some of its boundaries are open, it is partitioned as an alternate sequence of $2m_i$ open and closed path $\gamma_i = p_{i,1} \cup p_{i,2} \cup \cdots \cup p_{i,2m_i}$ where odd paths are open and even paths are closed. When a hole γ_i is totally closed or totally open, its partition is trivial and we set $m_i = 0$. The case of a surface containing only open boundaries was already considered in Corollary 3.3. We focus on surfaces G that contain both types of boundaries. By Theorem 3.2, we have

$$k(G) - k(\bar{G}) = |V| - |\mathring{V}| + |\mathring{E}| - |E| + \kappa_{\overline{\partial_O V}}(G) + \kappa_{\overline{\partial_C E}}(G) - \kappa_{\overline{\partial_O V}}(\bar{G}) - \kappa_{\overline{\partial_C E}}(\bar{G})$$

By definition, $\kappa_{\overline{\partial_O V}}(\bar{G}) = 1$ and $\kappa_{\overline{\partial_C E}}(\bar{G}) = 0$ whereas $\kappa_{\overline{\partial_O V}}(G) = 0$ and $\kappa_{\overline{\partial_C E}}(G) = 0$ which

produces

$$k(G) - k(\bar{G}) = |V| - |\mathring{V}| + |\mathring{E}| - |E| - 1$$
$$= |\partial_O V| - |\partial_O E| - 1$$
$$= \left(\sum_{i=1}^{b} (|\partial_O V(\gamma_i)| - |\partial_O E(\gamma_i)|) \right) - 1$$
$$= \left(\sum_{i=1}^{b} m_i \right) - 1$$

Therein, we simply used the fact that each open path that is not a cycle contains 1 more open vertex than the number of open edges. Altogether, we proved that

$$k(G) = k(\bar{G}) + \sum_{i=1}^{b} m_i - 1$$

where $k(\bar{G}) = 2g + b - 1$ or $g + b - 1$ and $\sum_{i=1}^{b} m_i$ is the number of non-cyclic disjoint open paths on the boundary of G. □

For instance, consider a planar lattice punctured with b holes (that is a sphere punctured by $b + 1$ holes). If the b holes are closed, then the corresponding surface code encodes $k = b$ qubits. But if the boundary of each of these b holes is the union of an open path and a closed path, this generalized surface code encodes $k = 2b - 1$ qubits that is approximately twice more than the standard surface codes based on closed holes. However, one cannot immediately conclude that these codes are always better since this transformation may also reduce the minimum distance.

A second example is shown in Figure 5. It is an orientable surface of genus 2 with 4 mixed boundaries holes. One of these 4 holes is open, another one is closed and each of the other 2 holes have 2 open paths and 2 closed paths on their boundary. Corollary 3.4 tells us that the associated surface code encodes 10 qubits ($g = 2, b = 4$ and $m = 4$). A symplectic basis of the logical operators is represented.
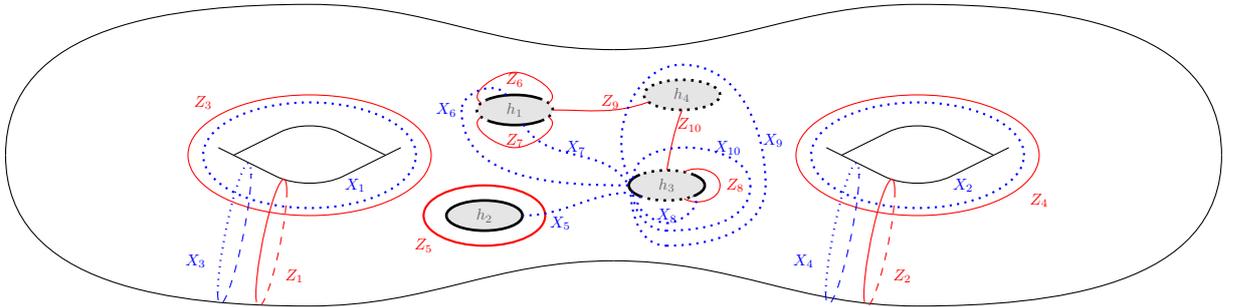

Figure 5: A symplectic basis of the set of logical operators for an orientable surface of genus 2 punctured with holes with mixed boundaries.

To conclude, let us propose a general strategy to construct a symplectic basis of logical operators for generalized surface codes. As before we can attach 2 pairs of logical operators

with any handle (or just one pair for non-orientable surface). The unusual set of logical operators comes from holes and their boundaries. The following strategy produces a symplectic basis to complete the handle logical operators in a symplectic basis.

Assume that the boundary of the surface contains b holes and m open paths. Denote by $h_1, \ldots, h_{b_o}$ the b_o holes of the surface which contains at least one open boundary. Let $\delta_1, \ldots, \delta_{\ell_c}$ be the ℓ_c closed paths (cyclic or not) along the boundary of the surface. In what follows, for any path γ of G or its dual, $P(\gamma)$ denotes the Pauli operator which acts as the Pauli matrix $P = X$ or Z on the qubits supported by γ.

Construction of $Z_1, \ldots, Z_{b+m-2}$:
1. For $i = 1, \ldots, \ell_c - 1$, define $Z_i = Z(\delta_i)$.
2. For $i = 1, \ldots, b_o$, pick an open vertex v_i on the boundary of the hole h_i.
3. For $i = 1, \ldots, b_o - 1$, construct a path γ_i that connects v_i to the last vertex v_{ℓ_o} and define $Z_{\ell_c-1+i} = Z(\gamma_i)$.

Construction of $X_1, \ldots, X_{b+m-2}$:
1. For $i = 1, \ldots, \ell_c - 1$ construct a path δ_i^* of the dual graph which connects δ_i to the last closed path δ_{ℓ_c}. and define $X_i = X(\delta_i^*)$.
2. For $i = 1, \ldots, b_o - 1$, construct a path γ_i^* in the dual graph that connects the path δ_{ℓ_c} to itself after a loop around the hole h_i and define $Z_i = Z(\gamma_i^*)$.

This provides $\ell_c - 1 + b_o - 1$ independent logical operators of each type. One can easily see that $\ell_o - 1 + \ell_c - 1 = b + m - 2$ is exactly the contribution of the holes and the open boundaries to k. These operators satisfy the expected commutation relations of a symplectic basis.

4   Homology of surfaces with mixed boundaries

In this section, we develop a notion of homology appropriate to surfaces with open and closed boundaries. Although homology of surfaces with closed boundaries is well understood [16, 17], we are not aware of any in-depth study of this peculiar notion of homology. Recall that understanding homology of the underlying tiling is crucial for surface codes since it is responsible for the commutation relations of stabilizers and it governs the parameters of these quantum codes.

4.1   Cycle space of open graphs

As explained in Section 3.3, cycles appear naturally in the context of surface codes as the support of logical operators and stabilizers. The number of encoded qubits of a generalized surface code will be obtained by enumerating logical operators, that is cycles. This enumeration relies on a closed formula for the number of cycles in a surface $G = (V, E, F)$, obtained in the present section.

The notion of cycles and cocycles we derive in this section applies not only to a generalized surfaces $G = (V, E, F)$ with open and closed boundaries, but more generally to abstract graphs $H = (V, E)$ with open and closed boundaries. Since open edges have no physical relevance (they do not support any qubit), our goal is to compute the number of relative cycles of the graph $\mathring{H} = (V, \mathring{E})$. First, we recall the definitions of a graph and a cycle and we extend these definitions to graphs with some open edges and vertices. We emphasize that open edges are

mostly irrelevant throughout this section since we are concerned with the cycle structure of $\mathring{H} = (V, \mathring{E})$; their sole purpose is to define the notion of open vertices.

A *graph* is defined to be a pair $H = (V, E)$, where V is a finite set and E is a set of pairs $\{u, v\}$ of elements of V. The elements of V are called vertices and those of E are the edges of the graph H. We assume that all graphs are finite and simple, *i.e.* have no loops nor multiple edges. We refer to [18] or [19] for standard results in graph theory.

We consider a more general class of graphs that we call *open graphs*, where some edges are declared to be open. The set of open edges of H is a subset of E that we denote $\partial_O E$. The set of open vertices $\partial_O V$ is a subset of V formed of the vertices that are adjacent to an open edge. We denote the interiors $\mathring{V} = V \backslash \partial_O V$ and $\mathring{E} = E \backslash \partial_O E$. With these definitions, it is clear that the edges of $\mathring{E}$ have at most one endpoint in $\partial_O V$. A graph, as defined in the previous paragraph, is simply an open graph with trivial open-edge set $\partial_O E = \emptyset$.

In this paper, $\kappa(H)$ denotes the number of connected components of the graph H. We will also have to enumerate the connected components of a graph H satisfying some properties. For this purpose, we introduce the notation $\kappa_X(H)$ (respectively $\kappa_{\bar{X}}(H)$) for the number of connected components of the graph H containing at least one element of the set X (respectively no elements of the set X). The set X will typically be a subset of the vertex set or the edge set of H.

A *cycle* in a graph $H = (V, E)$ is defined to be a subset $\gamma \subset E$ of edges of the graph H that meets every vertex an even number of times. For an open graph, we only require that γ meets the vertices of $V \backslash \partial_O V$ an even number of times. These cycles were called relative cycles in the statement of Theorem 3.2 and will simply be called cycles or cycles of an open graph in what follows. For instance, a path connecting two different open vertices is a cycle. If $\partial_O V$ is empty, these definitions coincide.

The structure of the set of cycles of a graph, which we denote $\mathcal{C}(H)$ is well understood when H has no open vertices. It is a $\mathbb{F}_2$-linear space, where the sum of two cycles is defined as their symmetric difference. This set is called the *cycle space* of H and its dimension is

$$\dim \mathcal{C}(H) = |E| - |V| + \kappa(H). \tag{1}$$

See for instance [18] for an elegant proof of this result.

Here, we prove that the set of cycles of an open graph has a quite similar structure.

Proposition 4.1. *The set of cycles of the interior $\mathring{H}$ of an open graph $H = (V, E)$ with open-vertex set $\partial_O V$ and open-edge set $\partial_O E$ is a $\mathbb{F}_2$-linear code of dimension*

$$\dim \mathcal{C}(\mathring{H}) = |\mathring{E}| - |\mathring{V}| + \kappa_{\overline{\partial_O V}}(\mathring{H}).$$

Recall that $\kappa_{\overline{\partial_O V}}(\mathring{H})$ denotes the number of connected components of $\mathring{H}$ containing no open vertex $v \in \partial_O V$. As expected we recover Eq.(1) for graphs with an empty open-vertex set.

Proof. To prove that the set of cycles of $\mathring{H}$ is a $\mathbb{F}_2$-linear code, it suffices to check that the sum (that is the symmetric difference) of two cycles is also a cycle.

The only non-trivial point is the dimension formula. In order to prove it, we construct a 2-cover H_2 of the graph $\mathring{H}$ which has no open vertex. Eq.(1) provides the dimension of the

cycle code of H_2. Then, we derive the dimension of the cycle space of $\mathring{H}$ from the one of H_2. This idea of introducing a double cover to simplify our problem is quite common. For instance, it is standard to replace a non-orientable surface by its orientable double cover.

This 2-cover is constructed by taking two copies of the graph $\mathring{H}$ and by connecting the corresponding pair of open vertices. Formally, the vertex set of $H_2 = (V_2, E_2)$ is $V_2 = V \times \mathbb{F}_2$. Two vertices (u, x) and (v, y) in V_2 are linked by an edge if and only if $x = y$ and $\{u, v\}$ is an edge of $\mathring{H}$. Moreover, an extra edge is added between all the pairs of vertices $(v, 0)$ and $(v, 1)$ associated with an open vertex $v \in \partial_O V$.

The graph $\mathring{H}$ is embedded in H_2 as the subgraph induced by the vertices of the form $(u, 0)$. This restriction that maps H_2 to $\mathring{H}$ induces a projection of the cycle space of H_2 onto the cycle space of $\mathring{H}$

$$\pi : \mathcal{C}(H_2) \longrightarrow \mathcal{C}(\mathring{H})$$

which sends a cycle $\gamma \subset E_2$ onto its restriction γ_0 to the set of edges of the form $\{(u, 0), (v, 0)\}$. This linear application π is surjective since any cycle γ of $\mathring{H}$ is the image of the cycle of H_2 obtained from two copies $\gamma \times \{0\}$ and $\gamma \times \{1\}$ of the cycle γ, and connecting them through the open vertices incident to γ.

From Eq.(1), the cycle space of H_2 has dimension

$$\dim \mathcal{C}(H_2) = |E_2| - |V_2| + \kappa(H_2) = 2|E| + |\partial_O V| - 2|V| + 2\kappa_{\overline{\partial_O V}}(\mathring{H}) + \kappa_{\partial_O V}(\mathring{H}),$$

where $\kappa_{\partial_O V}(\mathring{H})$ is the number of connected components of $\mathring{H}$ containing at least one open vertex and $\kappa_{\overline{\partial_O V}}(\mathring{H})$ denotes the number of connected components of $\mathring{H}$ containing no open vertices.

Consider the kernel of the projection π. A cycle of H_2 has a trivial image under π if and only if it is a cycle of the subgraph $\bar{H}$ of H_2 induced by the vertices $(u, 1)$ where no vertices are declared to be open. We can therefore apply Eq.(1), which proves that

$$\dim \operatorname{Ker} \pi = \dim \mathcal{C}(\bar{H}) = |E| - |V| + \kappa(\mathring{H}).$$

By the rank nullity theorem, the dimension of the cycle space of $\mathring{H}$ is thus

$$\dim \mathcal{C}(H_2) - \dim \ker \pi = |E| - |V| + |\partial_O V| + 2\kappa_{\overline{\partial_O V}}(\mathring{H}) + \kappa_{\partial_O V}(\mathring{H}) - \kappa(\mathring{H}).$$

To conclude, note that $\kappa(\mathring{H}) = \kappa_{\overline{\partial_O V}}(\mathring{H}) + \kappa_{\partial_O V}(\mathring{H})$. $\qquad\square$

4.2 Definition of Homology

Intuitively, the first homology group of a tiling $G = (V, E, F)$ represents the cycles of the graph up to deformations. To formally define this group, we associate the following *chain complex* with the surface $G = (V, E, F)$.

$$C_2 \xrightarrow{\partial_2} C_1 \xrightarrow{\partial_1} C_0.$$

In this notation, C_2, C_1 and C_0 are the 3 $\mathbb{F}_2$-linear spaces

$$C_0 = \bigoplus_{v \in \mathring{V}} \mathbb{F}_2 v, \quad C_1 = \bigoplus_{e \in \mathring{E}} \mathbb{F}_2 e, \quad C_2 = \bigoplus_{f \in F} \mathbb{F}_2 f.$$

This triple is equipped with two $\mathbb{F}_2$-linear maps $\partial_2 : C_2 \to C_1$ and $\partial_1 : C_1 \to C_0$ defined by

$$\partial_2(f) = \sum_{\substack{e \in \mathring{E} \\ e \in f}} e \quad \text{and} \quad \partial_1(e) = \sum_{\substack{v \in \mathring{V} \\ v \in e}} v.$$

These maps are called *boundary maps*. This definition is motivated by the fact that for the surface codes introduced in Definition 3.1, a Z-error $E_Z \in \{I, Z\}^{\otimes n}$ can be detected through the measurements of the vertex operators X_v corresponding to non-open vertices $v \in \mathring{V}$.

Before going to the definition of homology groups, recall that any subset $S \subset X$ corresponds to a vector δ_S of the space $\oplus_{x \in X} \mathbb{F}_2 x$ by the map $S \mapsto \delta_S = \sum_{x \in S} x$. This allows us to consider subsets of vertices, edges or faces respectively as vectors of C_0, C_1 or C_2 and to put a geometric intuition on these sets. For instance, a subset of E corresponds to a vector of C_1. A subset of edges corresponds to a vector of the kernel of the application ∂_1 if and only if it is called a cycle of the graph $(V, \mathring{E})$ where the vertices of the subset $\partial_O V$ are declared to be open.

The following lemma motivates the choice of the chain spaces. It is also the key ingredient making the definition of surface codes possible on any surfaces with or without boundaries. More precisely, it will be used to justify the commutation relations between the stabilizer generators of a surface code.

Lemma 4.2. *For any tiling G, with or without boundaries, the composition of the two boundary maps is trivial: $\partial_1 \circ \partial_2 = 0$.*

Proof. By linearity, it suffices to check that the image of a face $f \in F$ under $\partial_1 \circ \partial_2$ is trivial. A face of G is an elementary cycle of the form $f = \{\{v_1, v_2\}, \{v_2, v_3\} \dots, \{v_\ell, v_1\}\}$. If it contains only non-open edges (and by consequence only non-open vertices), then clearly $\partial_1 \circ \partial_2$ vanishes. Otherwise, we assume that only the last edge $\{v_\ell, v_1\}$ is open. Its two endpoints are thus open vertices. Then, we get

$$\partial_1 \circ \partial_2(f) = \partial_1\left(\sum_{i=1}^{\ell-1}\{v_i, v_{i+1}\}\right)$$

$$= \partial_1(\{v_1, v_2\}) + \partial_1\left(\sum_{i=2}^{\ell-2}\{v_i, v_{i+1}\}\right) + \partial_1(\{v_{\ell-1}, v_\ell\})$$

$$= v_2 + \sum_{i=2}^{\ell-2}(v_i + v_{i+1}) + v_{\ell-1} = 0.$$

Adapting this argument to prove the general case where more edges of f are open is straightforward. $\square$

It follows directly from the previous lemma that $\operatorname{Im} \partial_2 \subset \operatorname{Ker} \partial_1$. This allows us to define the first homology group of a surface.

Definition 4.3. *The* first homology group *of a tiling G with open and closed boundaries, denoted $H_1^\partial(G)$, is defined to be the quotient space $H_1^\partial(G) = \operatorname{Ker} \partial_1 / \operatorname{Im} \partial_2$.*

In the present work, we consider $\mathbb{F}_2$-homology. As we see in the previous definition, the group $H_1^\partial(G)$ is then a $\mathbb{F}_2$-linear space. The symbol ∂ in $H_1^\partial(G)$ is used to distinguish our generalization of the first homology group to the standard homology group $H_1(G)$ usually considered for surfaces without boundaries.

4.3   Rank of the first homology group

Let us determine the dimension of $H_1^\partial(G)$. Recall that $\kappa_{\overline{\partial_C E}}(G)$ denotes the number of connected components of G containing no closed boundary $e \in \partial_C E$ and $\kappa_{\overline{\partial_O V}}(G)$ is the number of connected components of G containing no open vertex $v \in \partial_O V$.

Proposition 4.4. *The dimension of $H_1^\partial(G)$ is given by*

$$\dim H_1^\partial(G) = -|\mathring{V}| + |\mathring{E}| - |F| + \kappa_{\overline{\partial_O V}}(G) + \kappa_{\overline{\partial_C E}}(G),$$

For instance, for a connected orientable surface $G_{g,b}$ of genus g with b closed boundaries and no open boundaries, we recover the classical formula

$$\dim H_1^\partial(G_{g,0}) = 2g \quad \text{and} \quad \dim H_1^\partial(G_{g,b>0}) = 2g + b - 1.$$

Therein, we used the following property of Euler-Poincaré characteristic: $|V| - |E| + |F| = 2 - 2g - b$.

Proof. In order to compute the dimension of this quotient space, it suffices to determine the dimension of $\operatorname{Ker} \partial_1$ and $\operatorname{Im} \partial_2$.

The space $\operatorname{Ker} \partial_1$ corresponds to the cycle space of the open graph $(V, \mathring{E})$ with open vertex set $\partial_O V$. Its dimension, provided by Proposition 4.1, is

$$\dim \operatorname{Ker} \partial_1 = |\mathring{E}| - |\mathring{V}| + \kappa_{\overline{\partial_O V}}(G).$$

The space $\operatorname{Im} \partial_2$ is generated by the vectors $\partial_2(f)$ for $f \in F$. Consider a connected component C of the tiling G. Clearly, if the induced subtiling $C = (V_C, E_C, F_C)$ contains a closed boundary $e \in \partial_C E$ then the only relation of the form $\sum_{f \in F_C} \lambda_f \partial_2(f) = 0$ is the trivial one. That means that the vectors $\partial_2(f)$ associated with the faces of this component are independent. Assume now that C contains no closed boundary. Then each edge $e \in E \backslash \partial_O E$ belongs to exactly 0 or 2 faces of C. This implies the non-trivial relation $\sum_{f \in F_C} \partial_2(f) = 0$, proving that the vectors $\partial_2(f)$ are not independent. However, by a similar argument any $|F_C| - 1$ of these faces are independent. Considering all the connected components of G together, this proves that the dimension of $\operatorname{Im} \partial_2$ is given by

$$\dim \operatorname{Im} \partial_2 = |F| - \kappa_{\overline{\partial_C E}}(G).$$

where $\kappa_{\overline{\partial_C E}}(G)$ denotes the number of connected components of G having no closed boundary $e \in \partial_C E$.

Altogether, we obtain the dimension of $H_1^\partial(G)$ by $\dim H_1^\partial(G) = \dim \operatorname{Ker} \partial_1 - \operatorname{Im} \partial_2$.   □

4.4   Local structure of the dual

In order to prepare the construction of the dual of a combinatorial surface with boundaries, we provide in this section a precise description of the local structure of a surface around a vertex. We will construct a dual face from the set F_v of faces indicent to a vertex v. Figure 6 illustrates this construction. We do not consider the distinction between an open and a closed boundary yet.

By construction, the faces of any surface (V, E, F) are glued together in such a way that

1. Any two faces meet in at most one edge.

2. Any edge belongs to either one or two faces.

3. Given any vertex $v \in V$, denote by F_v the set of faces incident to v and consider the graph obtained by adding an edge between any two elements of F_v if the corresponding faces of G share an edge. Then, this graph F_v is either a self-avoiding path or a cycle.

We previously defined boundaries through the edges that are on the boundary of a unique face. Alternatively, one may define boundary vertices from the third property above by saying that a vertex v is a boundary if the corresponding set F_v induces a self-avoiding path. Then we would define boundary edges and faces as those incident to a boundary vertex. This leads to an equivalent definition since a vertex v is a boundary if and only if the corresponding set F_v induces a self-avoiding path. The following lemma summarizes these equivalent definitions of boundaries.

Lemma 4.5. *Let $G = (V, E, F)$ be a tiling with boundaries.*

- *$v \in \partial V \Leftrightarrow F_v$ induces a self-avoiding path $\Leftrightarrow v$ is incident to 2 edges of ∂E.*

- *$e \in \partial E \Leftrightarrow e$ is incident to a unique face $\Leftrightarrow$ the two endpoints of e belong to ∂V.*

- *$f \in \partial F \Leftrightarrow f$ is incident to a vertex of $\partial V \Leftrightarrow f$ is incident to an edge of ∂E.*

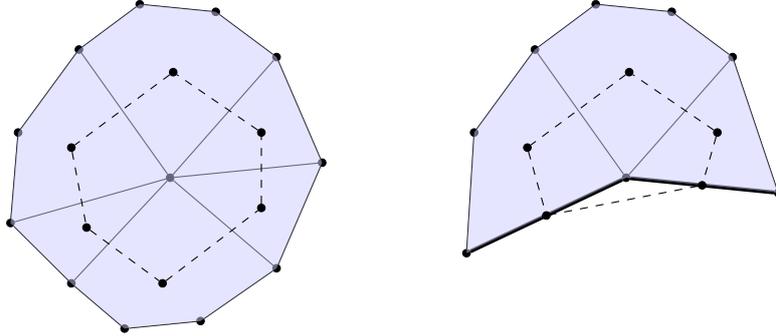This lemma is clear. It is stated only for convenience.

Figure 6: Local structure of the dual graph around a vertex v. The faces containing a vertex v are represented. If v is not a boundary (left), then F_v induces a cycle in the dual graph represented by dashed lines. If v is a boundary (right), then F_v is completed by adding 2 vertices in the middle of the 2 boundary edges and by connecting them by a dual edge. This results in a cycle $\bar{F}_v$ in the dual graph. The two dark edges at the right are the boundaries.

We now show that the set F_v introduced in item 3 above can always be endowed with a structure of cycles as depicted in Figure 6. It will later play the role of a face of the dual surface. Let $v \in V$ be a vertex of a surface G and let F_v be the set of faces of G incident to v. Denote by v_f the elements of F_v where $f \in F$ is a face incident to $v \in V$. Let $\bar{F}_v$ be the set F_v completed with the 2 extra elements v_e and $v_{e'}$ coming from the 2 boundary edges $e, e' \in E$ incident to v. The set $\bar{F}_v$ is regarded as a vertex set and equipped with the edges $\{v_f, v_{f'}\}$ such that f and f' share an edge, the 2 edges $\{v_e, v_f\}$ where f is the unique face containing e, and finally the edge $\{v_e, v_{e'}\}$ connecting the 2 boundaries. When v is not a boundary, $\bar{F}_v$ coincides with F_v.

16

Lemma 4.6. *Let $v \in V$ and let F_v and $\bar{F}_v$ be the corresponding local dual graphs, then the graph F_v is a cycle if and only if $v \in V \backslash \partial V$ and $\bar{F}_v$ is always a cycle.*

This result is an immediate consequence of the previous lemma. The two possible configurations for the local dual graph are represented in Figure 6. The cycle $\bar{F}_v$ is not necessarily included in the original surface but it can always be deformed to be embedded in the surface. This makes it an ideal candidate to define dual faces.

4.5 Dual cellulation

The dual cellulation G^* of a combinatorial surface G without boundaries is simply obtained by replacing each face of G by a vertex and by connecting two such vertices if the corresponding faces of the original graph share an edge. This leads to a correspondence between the edges of the graph and those of its dual. This dual graph naturally defines a cellulation of the same surface whose faces are given by the cycle F_v (see Lemma 4.6). The faces of G^* are therefore in one-to-one correspondence with the vertices of G.

Before stating our definition of the dual of a surface with open and closed boundaries, let us consider two natural notions of dual for surface with boundaries without considering the type (open or closed) of the boundaries.

- **Inner Dual:** Following the standard construction, one could define a dual graph by replacing each face of G by a vertex and connecting two vertices if they correspond to faces sharing an edge. Then by Lemma 4.6, we see that all the vertices $v \in V \backslash \partial V$ induce a cycle F_v in the dual graph. This defines the face set of the dual. No face is associated with boundary vertices. This corresponds to cutting the regions on the boundary of the surface where faces are not trivially defined.

This construction preserves the bijection between the faces of the graph and the vertices of its dual. However an edge of G that belongs to a single face, has no corresponding edge in the dual. The correspondence between the vertices of G and the faces of its dual is also lost. We may also consider the following extension of the inner dual.

- **Outer Dual:** Another dual can be obtained by replacing each vertex of G by a face of the dual. More precisely, from Lemma 4.6, we can replace any vertex $v \in V$, boundary or not, by the face defined by the cycle $\bar{F}_v$. Then these faces are glued together as follows. For any pair of neighbours vertices u and v, the faces $\bar{F}_u$ and $\bar{F}_v$ are stuck together along the edge corresponding to the edge $\{u, v\}$.

This latter dual re-establishes the correspondence between the vertices of G and the faces of G^* but it does not preserve the bijections between E and E^* or between F and V^*.

The definition of a dual surface is thus less straightforward in the presence of boundaries. For our purpose, we also need to consider the type of the boundaries. In order to preserve the surface code structure, we aim for a notion of duality which leads to a correspondence (i) between $\mathring{E}$ and $\mathring{E}^*$ (like in the enlarged dual) since $\mathring{E}$ supports the qubits, (ii) between $\mathring{V}$ and F^*, and (iii) between F and $\mathring{V}^*$. The bijections (ii) and (iii) are required to exchange the roles of X and Z through duality.

We will proceed in two steps. Roughly, we define the dual surface as the inner dual along open boundaries and as the outer dual along closed boundaries. Through steps 1 to 3 below,

we define a dual graph for which the correspondence (i) is restored. Then, we extend this graph by adding some extra edges (the open edges in step 4) to properly define faces. This makes it a combinatorial surface and this brings back the correspondence (ii) and (iii). We can check that the correspondences are satisfied in Table 1.

The dual surface $G^* = (V^*, E^*, F^*)$ is obtained from $G = (V, E, F)$ by the following process.

1. **Non-open vertices**: Define a vertex $v_f \in V^*$ for each face $f \in F$

2. **Open vertices**: Define an open vertex $v_e \in \partial_O V^*$ for each closed edge $e \in \partial_C E$

3. **Non-open edges**: For each edge $e \in E \backslash \partial E$, add an edge $\{v_f, v_{f'}\} \in E^*$, where f and f' are the two distinct faces containing e. For each edge $e \in \partial_C E$, add an edge $\{v_e, v_f\} \in E^*$, where $f \in F$ is the unique face of G containing the boundary edge e.

4. **Open edges**: Add an edge $\{v_e, v_{e'}\} \in \partial_O E^*$ for every pair of distinct edges $e, e' \in \partial_C E$ sharing a vertex $v \in \partial_C V$ in G.

5. **Non-open faces**: Define a face from the cycle F_v for each $v \in V \backslash \partial V$.

6. **Open faces**: Define a face from the cycle $\bar{F}_v \in \partial_O F^*$ for each $v \in \partial_C V$.

Table 1: Correspondences between a surface and its dual.

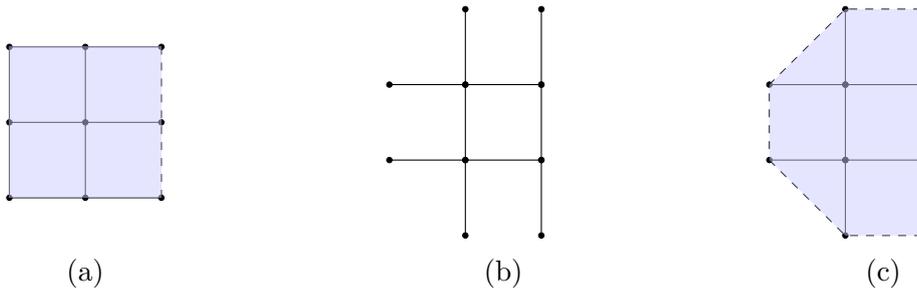| Tiling (V, E, F)	Dual tiling (V^*, E^*, F^*)
F	$\mathring{V}^*$
$\partial_C E$	$\partial_O V^*$
$\mathring{E}$	$\mathring{E}^*$
$\partial_C V$	$\partial_O E^*$
$V \backslash \partial V$	$\mathring{F}^*$
$\partial_C V$	$\partial_O F^*$

(a)   (b)   (c)

Figure 7: (a) A surface with open boundaries represented by dashed lines. (b) Its dual graph is obtained (steps 1 to 3) by replacing each non-open edge by a non-open edge. (c) Faces of the dual surface are obtained by adding open edges to equip the dual graph with a structure of surfaces.

The different steps are chosen to emphasize the different correspondences between the surface and its dual (See Table 1). As previously, the non-open subsets of vertices, edges and faces of G^* are denoted respectively by $\mathring{V}^* = V^* \backslash \partial_O V^*$, $\mathring{E}^* = E^* \backslash \partial_O E^*$ and $\mathring{F}^* = F^* \backslash \partial_O F^*$.
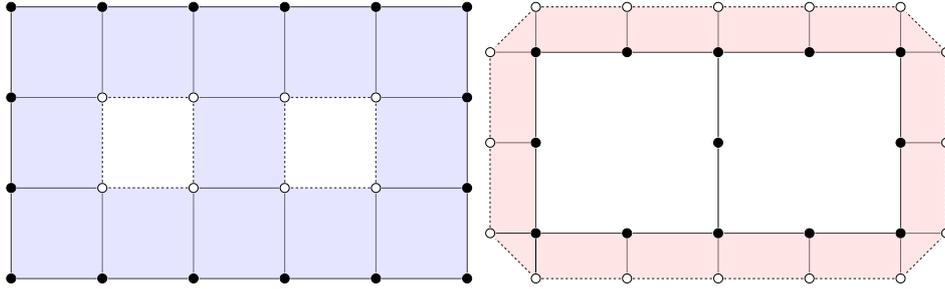
Figure 8: Left: A tiling with open and closed boundaries. Dashed edges represent open edges and white vertices are open vertices. The tiling contains two non-open edges whose endpoints are both open vertices. Right: The two corresponding edges in the dual belong to no face. Then this dual tiling is not tiling.

By construction, G^* defines a cellulation of the same surface as G. We can easily check that open vertices, edges and faces are defined in a coherent way. This is a subgraph of the outer dual which contains the inner dual.

Remark: Property 2 of Section 4.4 may not be satisfied by the dual graph. If the 2 endpoints of a non-open edge are open then the dual of this edge will not belong to any face of the dual as in Figure 8. In the error correction terminology, the surface code would have minimum distance 1. We can simply avoid this configuration by declaring any edge whose both endpoints are open as an open egde.

Remark: To avoid the presence of loop or multiple edges in the dual graph, one can restrict the girth of G to be at least 3.

4.6 Application to generalized surface codes

This notion of homology is chosen to correspond to errors on the surface of Def. 3.1 and their syndromes. Indeed, by definition a Z-error $E_Z \in \{I, Z\}^{\otimes n}$ corresponds to a vector z of C_1 defined by

$$z = \sum_{e \in \mathrm{Supp}(E_Z)} e.$$

Its syndrome $\sigma(E_Z)$ corresponds to the vector $\partial_1(z) \in C_0$ and Z-stabilizers are given by the vectors $z \in \mathrm{Im}\, \partial_2$.

Similarly, an error $E_X \in \{I, X\}^{\otimes n}$ corresponds to the vector $x = \sum_{e \in \mathrm{Supp}(E_X)} e$ of C_1. Two Pauli operators E_X and E_Z commute if and only if the corresponding vectors x and z are orthogonal for the binary inner product in C_1.

Through these isomorphisms, the set of Z-stabilizers can be seen as the image of the map ∂_2 and the X-stabilizers are in bijection with the vectors of $(\ker \partial_1)^\perp$. From Lemma 4.2, we have $\mathrm{Im}\, \partial_2 \subset \ker \partial_1 = ((\ker \partial_1)^\perp)^\perp$. This proves that the two spaces corresponding to S_X and S_Z are orthogonal. In other words, we just proved that the commutation relation between the operators X_v and Z_f is ensured by the relation $\partial_1 \circ \partial_2 = 0$:

Corollary 4.7. *For any surface G, the operators X_v for $v \in \mathring{V}$ and Z_f for $f \in F$ are commuting.*

We know that the number of logical qubits k encoded in the surface G is given by the rank of the quotient of the group of Z-errors of syndrome 0 by the subgroup of Z-stabilizers. As an immediate application, k is the dimension of the first homology group $H_1^\partial(G)$. Proposition 4.4 yields:

Corollary 4.8. *The number of logical qubits k encoded using the surface code associated with G is*

$$k = -|\mathring{V}| + |\mathring{E}| - |F| + \kappa_{\overline{\partial_O V}}(G) + \kappa_{\overline{\partial_C E}}(G).$$

Consider the minimum distance $d = \min(d_X, d_Z)$ of the code. We already noticed that a Z-error has trivial syndrome if and only if its support $z \in C_1$ is a cycle and it is not a stabilizer if and only if this cycle has non-trivial homology. In order to obtain a graphical expression for the minimum distance, we need a similar result for d_X. An error E_X corresponds to a vector $x \in C_1$ but its syndrome and X-stabilizers are less trivial to describe. Consider this same error in the dual graph $G^* = (V^*, E^*, F^*)$. Denote by

$$C_2^* \xrightarrow{\partial_2^*} C_1^* \xrightarrow{\partial_1^*} C_0^*,$$

the homology chain-complex associated with the dual surface G^*. Using the correspondence of Table 1, we see that this duality map $G \to G^*$ transforms the vector $x = \sum_{e \in \mathrm{Supp}(E_X)} e \in C_1$ in the vector $x^* = \sum_{e \in \mathrm{Supp}(E_X)} e^* \in C_1$, where e^* is the dual edge of e. Any stabilizer Z_f is sent onto $Z_{v_f^*}$ where v_f^* is the dual vertex associated with the face f. This relies on the bijection between faces of G and non-open vertices of its dual. The syndrome of E_X can therefore be expressed as $\partial_1^*(x^*)$. The X-stabilizers X_v, which correspond to non-open vertices are mapped onto the vectors of $\mathrm{Im}\, \partial_2^*$. This is also a consequence of the definition of the dual graph and the correspondences of Table 1. This proves that d_X is the shortest length of a non-trivial cycle of G^*. The next corollary follows.

Corollary 4.9. *d_Z is the minimum length of a relative cycle of G with non-trivial homology in $H_1^\partial(G)$ and d_X is the minimum length of a relative cycle of G^* with non-trivial homology in $H_1^\partial(G^*)$.*

An equivalent proof of the graphical expression of d_X might be obtained by considering the cellular cohomology of the surface G. An error E_X of trivial syndrome which is not a stabilizer corresponds to a cocycle with non-trivial cohomology. Then it suffices to remark that the cohomology chain-complex of a surface is isomorphic to the homology complex of its dual.

5 Packing of logical qubits in a planar architecture

In this section we argue that mixed holes with partially open and partially closed boundaries may offer an advantage over usual surface codes for storage of quantum information in a two-dimensional lattice.

Bravyi, Poulin and Terhal [20] proved that the parameters of any two-dimensional local commuting projector code over finite dimensional quantum systems embedded in a square grid are subjected to the bound

$$kd^2 \leq cn$$

for some constant $c > 0$ that depends on the locality of the constraints. This bound can be seen as a tradeoff between the amount of quantum information stored in the lattice (given by the number of logical qubits k) and the error-correction capability of the code (measured by the minimum distance d). It provides a natural figure of merit to compare different quantum computing architectures. Optimizing the constant c can save a large amount of resources while keeping roughly the same performance.

Let us first provide some intuition for the special case of generalized surface codes based on a planar lattice punctured with closed holes. Each hole represents one logical qubit. To preserve a large minimum distance d, these holes must be separated from each other by a distance at least d. In other words, the neighborhoods $B(h, (d-1)/2)$ of each hole h containing all the qubits within distance $(d-1)/2$ from hole h do not overlap. These buffers of physical qubits surrounding each holes each consists $\Omega(d^2)$ qubits due to the two-dimensional geometry of the lattice (here, we assume that our lattice is Eucliean, for instance hyperbolic codes constructed in [21, 22, 23, 11] are locally planar but are not subjected to this tradeoff [24]). This shows that encoding k logical qubits using a uniform planar surface code with minimum distance d requires at least $n = \Omega(kd^2)$ qubits. This argument also emphasizes the resemblance with a sphere packing problem.

In this section, we will use the formalism of generalized surface codes to improve the constant c over previously known constructions.

5.1 Underlying lattice

Let us fix some notations and definitions. We focus on a square lattice of qubits. Starting with a finite region of a planar square lattice with closed boundaries, we will encode qubits as holes in this region. Two kinds of regions of the square lattice are considered in what follows, as depicted in Figure 9.
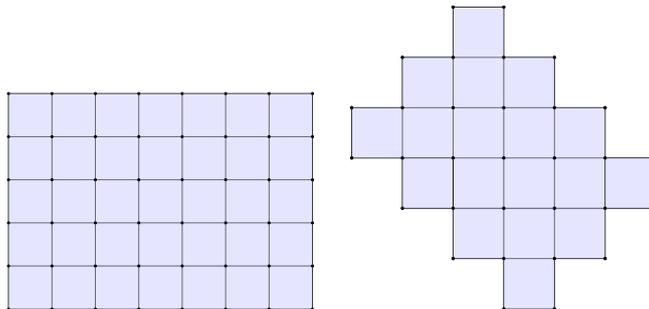

Figure 9: Left: A (7×5) square lattice. Right: A (3×4)-rotated square lattice.

By a $(L \times L')$-*square lattice*, we mean the subgraph of $\mathbb{Z}^2$ induced by the vertices $(x, y) \in [0, L] \times [0, L']$. This lattice contains $LL' + L + L' + 1$ vertices, $2LL' + L + L'$ edges and LL' faces. See Figure 9 for an example.

The second region of the square lattice that we use will be called a $(L \times L')$-*rotated square lattice*. It is a patch of the square lattice $\mathbb{Z}^2$ by a rectangle of length $\sqrt{2}L \times \sqrt{2}L'$ rotated by $45°$. For instance, a (3×4)-rotated square lattice is shown in Figure 9. Such a rotated square lattice is said to have size $L \times L'$. It contains $2LL' + L + L'$ vertices, $4LL'$ edges and $2LL' - L - L' + 1$ faces.

5.2 Square hole architecture

As a first example, we consider a square lattice punctured with square holes. This construction which is one of the most widely studied quantum computing architecture [5, 7, 6, 9], will be referred to as the *square hole architecture* or the *square hole surface code* and will be denoted $\mathrm{Sq}(h, h', t)$ or simply $\mathrm{Sq}(h, t)$ when $h = h'$. The parameters h, h' and t are non-negative integers. Each qubit corresponds to a $(t \times t)$ closed hole and these holes are separated to one another and to the boundary by a distance $4t - 1$. This ensures that the minimum distance of the code is $d = 4t$. This lattice is represented in Figure 10. The number of encoded qubits is given by the number of holes. In order to create $(h \times h')$ holes, this structure requires a $(L_h \times L_{h'})$-square lattice where $L_h = h(5t - 1) + (4t - 1)$.
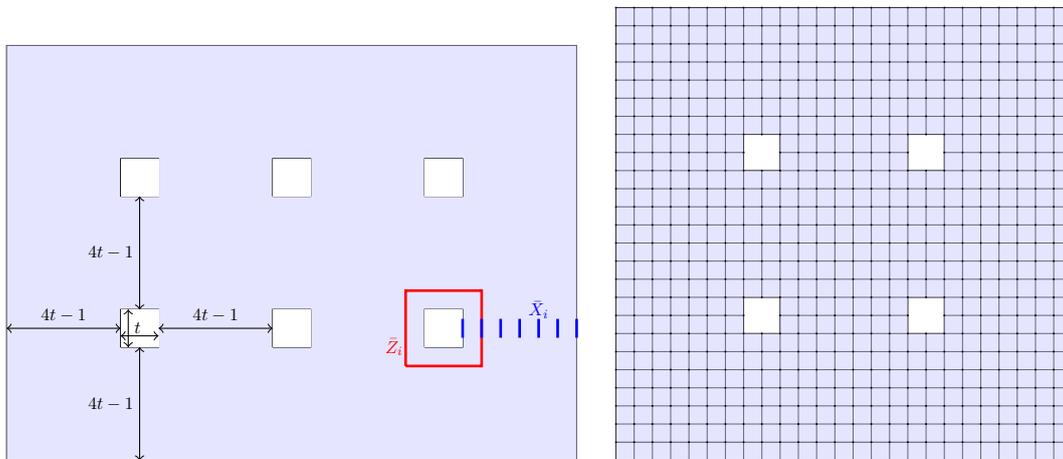

Figure 10: Left: Structure of the square hole architecture with minimum distance $d = 4t$. Right: An example with 4 holes encoding k=4 qubits with minimum distance $d = 8$.

Let us describe logical operations for these codes. Consider for instance a square hole surface code with k holes. As illustrated in Figure 10, one can choose $\bar{Z}_i$ which is a Z-operator whose support is a cycle enclosing the i-th hole and $\bar{X}_i$ which is a path in the dual graph connecting this hole to the boundary of the planar lattice.

Example 5.1. *The square hole architecture* $\mathrm{Sq}(h, t)$ *yields a surface code whose parameters satisfy*

$$n \sim 3kd^2$$

when both k and $d \to +\infty$.

The computation of these parameters relies on the following remark. When puncturing, $2t^2 - 2t$ qubits are removed per hole. There remains $n = 2L^2 + 2L - h^2(2t^2 - 2t)$ physical qubits. Indeed, a $(t \times t)$-square region of the square lattice contains $2t^2 + 2t$ edges, its boundary contains $4t$ edges and its interior contains $2t^2 - 2t$ edges. This proves that when $k, d \to +\infty$.

$$n = 2L^2 + 2L - h^2(2t^2 - 2t) \sim 48kt^2 \sim 3kd^2.$$

Therein, we used $k = h^2$ which is the number of holes and $t \sim d/4$ when d diverges.
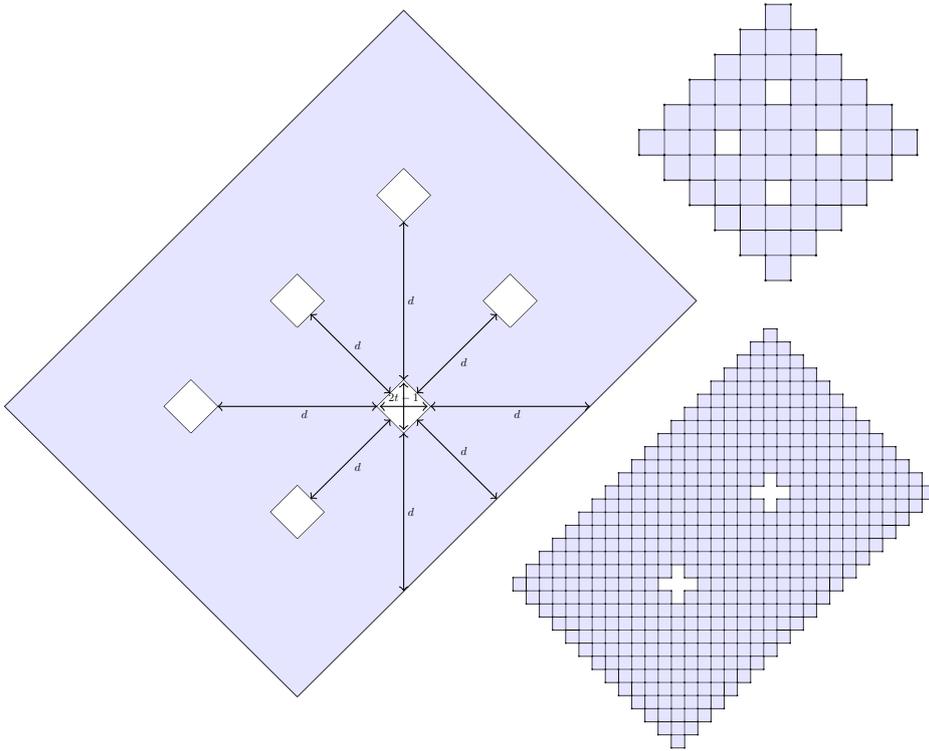
Figure 11: Left: Representation of the diamond hole architecture with minimum distance $d = 4(2t - 1)$ with the X-distance between holes and the boundary. Top right: A diamond-hole surface code with 4 holes encoding $k = 4$ qubits with minimum distance $d = 4$. Bottom right: A diamond-hole surface code with 2 holes encoding $k = 2$ qubits with minimum distance $d = 12$.

5.3 Diamond hole architecture

We can easily reduce the number of physical qubits required in the previous family of codes, for instance by simply truncating the corner. Indeed, the qubits at each of the 4 corners of the lattice are far away from each hole. Cutting these corners will not reduce the minimum distance. More generally, note that a hole in the square hole architecture is surrounded by only 4 closest holes (2 in the horizontal direction and 2 in the vertical direction). We will construct a packing of these holes that requires much less resources.

In order to get some intuition on this packing problem, we introduce the X-*distance* between two holes which is defined as the minimum length of a path in the dual graph connecting these two holes. Such a path is the support of a X-logical operator. Given a punctured lattice, the X-distance between two distinct holes is at least d_X. Two holes separated by a X-distance exactly d_X are called *tangent holes*. They cannot be closer. For instance, in the square hole architecture each hole is tangent to up to 4 holes. Based on the similarity with a sphere packing problem, we choose a punctured lattice such that each hole is tangent to a large number of holes.

Let us now define the *diamond hole architecture* or *diamond-hole surface codes* in which each hole is tangent to up to 8 distinct holes. It will be denoted $D(h, h', t)$ or $D(h, t)$ when $h = h'$, where h, h' and t are non-negative integers. This is also a surface codes based on a

planar square lattice punctured by holes with closed boundaries. It is represented in Figure 11. The parameter t fixes the size of holes. For $t = 1$, each hole is simply a single face. For $t = 2$ we extend this hole by adding the 4 faces incident to it. These 5 faces correspond to a vertex and its neighbours in the dual graph, that is a ball of radius 1 in the dual. More generally, a hole is the dual of a ball of radius t in the dual graph. The minimum distance will be given by the perimeter of holes, that is $d = 4(2t - 1)$. A set of $h \times h'$ holes are punctured in a rotated square lattice as in Figure 11. To keep a minimum distance $d = 4(2t - 1)$, holes are separated to one another and to the boundary of the lattice by a X-distance d. It was already noticed that such a rotation increases the distance for toric codes (without punctures) [26].

Just as with square holes, logical operators are generated by the Z-operators wrapping around holes and X-operators connecting holes to the boundary in the dual graph.

Example 5.2. *The diamond hole architecture* $D(h, t)$ *yields a surface code whose parameters satisfy*

$$n \sim 1.5kd^2$$

when k and $d \to \infty$.

Indeed, the total number of physical qubits in $D(h, t)$ is $n = 4(h(t + d/2) + d/2)^2 - 4h^2t^2$. For this enumeration remark that, before puncturing, the rotated lattice contains $4L_h^2$ edges. Each puncture removes $4t^2$ edges, leaving us with $4L_h^2 - 4ht^2$. When both k and d diverge, we get

$$n = 4(h(5d/8) + d/2)^2 - 4h^2(d/8)^2 \sim (96/64)^2kd^2$$

using $t \sim d/8$ and $k = h^2$.

One may wonder whether we can place these holes in such a way that a hole is tangent to more than 8 holes. This diamond hole lattice is a *perfect lattice* in the following sense. For each qubit of the lattice, either there is a unique hole at X-distance less than $d/2$ or it is at distance less $d/2$ from the outer boundary of the lattice. Stated differently, this proves that the regions $B(h, d/2)$ at X-distance $d/2$ from holes and from the boundary form a partition of the lattice. Therein, the X-distance is extended to measure the distance between 2 edges as follows. The X distance between 2 edges e and f is the minimum length of a path of the dual graph whose first edge is e and last edge is f. These regions are exactly the neighborhood of the holes considered in the introduction of this section to prove the bound $kd^2 = O(n)$. The term perfect lattice is chosen for the resemblance with perfect codes [27]. This argument does not exclude a better tradeoff for holes with a different shape.

5.4 Mixed boundary diamond hole architecture

In order to further improve the parameters of surface codes, we will introduce open boundaries around every hole. Our basic idea is to divide the boundary of a hole in an alternate sequence of open and closed paths. This reduces the minimum distance. One can then shrink the lattice, cutting the required number of physical qubits. Moreover, the number of encoded qubits k increases. Overall, we will prove that this transformation allows us to achieve a better tradeoff.

We start with a lattice punctured with diamond holes like in Section 5.3 and we will open some of the boundaries of the holes as shown in Figure 12.
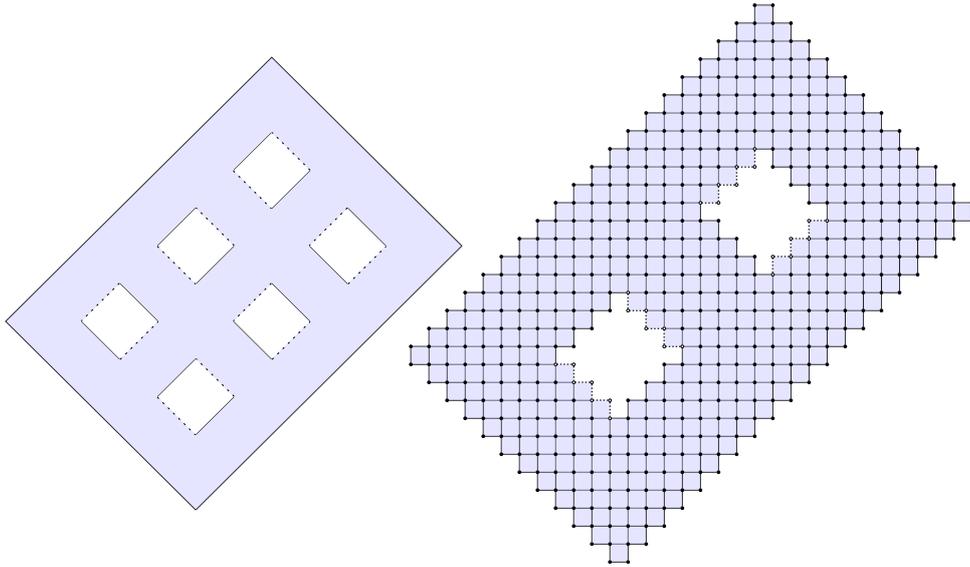
Figure 12: Left: Structure of the mixed diamond hole architecture. Dotted edges represent open boundaries. The distance between two holes coincides with the size of holes. Right: A mixed diamond-hole surface code with 2 holes encoding $k = 5$ qubits with minimum distance $d = 8$.

Let us define the *mixed diamond hole architecture* that we denote $D_4(h, h', t)$ or $D_4(h, t)$ when $h = h'$. The underlying lattice is a rotated square lattice. It is punctured by diamond holes whose size depends on the parameter t just as in the diamond hole architecture. Recall that a diamond hole is the dual of a ball of radius t. The edges on 2 opposite sites of these holes are then declared to be open. The boundary of such a hole is an alternate sequence of 2 closed paths and 2 open paths as we can see in Figure 12. More precisely, the perimeter of such a hole contains $4(2t - 1)$ edges, 2 paths of length $2t - 2$ are declared to be open and 2 paths of length $2t$ are kept closed. The minimum distance of a code based on such punctures is then at most $d = 2t$. We fix the X-distance between holes to be as small as possible and at least $2t$ in order to obtain a minimum distance $d = 2t$ for the code. Open and closed sides of holes are chosen such that adjacent holes face each other with different kinds of boundaries. This considerably reduces the number of low-weight errors.

Example 5.3. *The mixed diamond hole architecture* $D_4(h, t)$ *yields a surface code whose parameters satisfy*

$$n \sim kd^2$$

when k and $d \to \infty$.

Adapting the arguments of the previous section, we see that the architecture $D_4(h, t)$ requires a rotated square lattice of size $L_h \times L_h$ where $L_h = h(d/2 + d/2) + d/2$ and that h^2 rotated square holes of size $d/2 \times d/2$ are removed. We obtain $n \sim 4h^2d^2 - 4h^2(d/2)^2 = 3h^2d^2 = kd^3$ since we have $k = 3h^2 - 1 \sim 3h^2$.

25

6 Concluding remarks

We have obtained a 3-fold reduction in the number of physical qubits compared to the most widely studied surface code architecture [9]. Note that this improvement was achieved without affecting the minimum distance. In general, the performance of a code is dictated not only by the minimum distance but also by the number of distinct errors achieving this minimum distance, and more generally by the entire weight enumerator polynomial. This is why open and closed sides of holes were chosen such that neighbours holes face each other with different kinds of boundaries, considerably reducing the number of low-weight errors. The problem of optimizing a lattice geometry taking into account these combinatorial factors is in general very hard. In an upcoming paper, we will present a linear-time benchmarking algorithm which provides a quick way of numerically comparing different geometries.

The naive bound explained in the introduction of Section 5 proves that $kd^2 \leq n$ for planar square lattice architecture based on closed holes. We believe that the diamond-hole architecture provides the best tradeoff for planar Euclidean surface codes with closed boundaries, so the packing argument presented in the introduction of this section can probably be refined to prove that $kd^2 \leq 1.5n$ for such codes. We have shown that surface codes with mixed boundaries can violate this bound. To encode a single logical qubit, the rotated surface code [26, 10, 28, 29] remains the best alternative.

We exploited our formalism to construct surface codes with better parameters. Optimizing the parameters of surface codes and in particular the minimum distance naturally leads to a better performance for a depolarizing noise. However, a different type noise may require a different strategy. For instance, to fight the effect of an asymmetric Pauli noise with a high Z-error probability and a low X-error probability, one should consider asymmetric surface codes with a much stronger error-correction capability against Z-errors than X-errors [25]. More generally, it is crucial to understand the behaviour of generalized surface codes under more general non-Pauli noise.

References

[1] A. Y. Kitaev. Fault-tolerant quantum computation by anyons. *Annals of Physics*, 303(1):27, 2003.

[2] M. H. Freedman and D. A. Meyer. Projective plane and planar quantum codes. *Foundations of Computational Mathematics*, 1(3):325–332, 2001.

[3] S. B. Bravyi and A. Y. Kitaev. Quantum codes on a lattice with boundary. arXiv:9811052, 1998.

[4] E. Dennis, A. Kitaev, A. Landahl, and J. Preskill. Topological quantum memory. *Journal of Mathematical Physics*, 43:4452, 2002.

[5] R. Raussendorf and J. Harrington. Fault-tolerant quantum computation with high threshold in two dimensions. *Physical Review Letters*, 98(19):190504, 2007.

[6] R. Raussendorf, J. Harrington, and K. Goyal. Topological fault-tolerance in cluster state quantum computation. *New Journal of Physics*, 9:199, 2007.

[7] R. Raussendorf, J. Harrington, and K. Goyal. A fault-tolerant one-way quantum computer. *Annals of Physics*, 321(9):2242 – 2270, 2006.

[8] D. S. Wang, A. G. Fowler, and L. C. L. Hollenberg. Surface code quantum computing with error rates over 1 *Phys. Rev. A*, 83:020302, Feb 2011.

[9] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland. Surface codes: Towards practical large-scale quantum computation. *Physical Review A*, 86(3):032324, 2012.

[10] C. Horsman, A. G. Fowler, S. Devitt, and R. Van Meter. Surface code quantum computing by lattice surgery. *New Journal of Physics*, 14(12):123011, 2012.

[11] N. P. Breuckmann and B. M. Terhal. Constructions and noise threshold of hyperbolic surface codes. *arXiv preprint arXiv:1506.04029*, 2015.

[12] D. Gottesman. *Stabilizer Codes and Quantum Error Correction.* PhD thesis, California Institute of Technology, 1997.

[13] A. R. Calderbank and P. W. Shor. Good quantum error-correcting codes exist. *Physical Review A*, 54(2):1098, 1996.

[14] A. Steane. Multiple-particle interference and quantum error correction. *Proc. of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 452(1954):2551–2577, 1996.

[15] A. Kubica, B. Yoshida, and F. Pastawski. Unfolding the colour code. *New Journal of Physics*, 17(8):083026, 2015.

[16] A. Hatcher. *Algebraic topology.* Cambridge University Press, 2002.

[17] P. Giblin. *Graphs, surfaces and homology.* Cambridge University Press, 2010.

[18] C. Berge. *Graphs and Hypergraphs.* Elsevier, 1973.

[19] B. Bollobas. *Graph theory: an introductory course*, volume 63. Springer Science & Business Media, 2012.

[20] S. Bravyi, D. Poulin, and B.M. Terhal. Tradeoffs for reliable quantum information storage in 2D systems. *Physical Review Letters* 104:050503, 2010.

[21] M. H. Freedman, D. A. Meyer, and F. Luo. Z2-systolic freedom and quantum codes. *Mathematics of Quantum Computation, Chapman & Hall/CRC*, pages 287–320, 2002.

[22] G. Zémor. On Cayley graphs, surface codes, and the limits of homological coding for quantum error correction. In *Proc. of the 2nd International Workshop on Coding and Cryptology, IWCC 2009*, pages 259–273. Springer-Verlag, 2009.

[23] N. Delfosse and G. Zémor. Quantum erasure-correcting codes and percolation on regular tilings of the hyperbolic plane. In *Proc. of IEEE Information Theory Workshop, ITW 2010*, pages 1–5, 2010.

[24] N. Delfosse. A decoding algorithm for CSS codes using the X/Z correlations. In *Proc. of IEEE International Symposium on Information Theory, ISIT 2014*, pages 1071 - 1075, 2014.

[25] N. Delfosse and J.-P. Tillich In *Proc. of IEEE International Symposium on Information Theory, ISIT 2013*, pages 917–921, 2013.

[26] H. Bombin and M. A. Martin-Delgado. Topological quantum error correction with optimal encoding rate. *Physical Review A*, 73(6):062303, 2006.

[27] F. J. MacWilliams and N. J. A. Sloane. *The theory of error correcting codes*. North-Holland mathematical library. North-Holland Pub. Co. New York, Amsterdam, New York, 1977. Includes index.

[28] Y. Tomita and K. M. Svore. Low-distance surface codes under realistic quantum noise. *Phys. Rev. A*, 90:062320, Dec 2014.

[29] J. M. Gambetta, J. M. Chow, and M. Steffen. Building logical qubits in a superconducting quantum computing system. *arXiv preprint arXiv:1510.04375*, 2015.