4 Topological codes from homology

Armed with these homology and cohomology tools, we can revisit the toric code. We will see that the code can be defined almost entirely in homological terms and that homology concepts will describe both logical operators and the relationship between errors and syndromes.

This will be more than just a refresh of terminology. Since homological concepts are properties of the surface which hold for any cellulation, the homological approach is much more general. The definitions we write down and properties we define homologically immediately apply to generalisations of the toric code based on any cellulation of any genus $g$ closed surface.

We will later see that the family of topological codes defined in this manner can be broadened further, by extending these definitions to codes defined on surfaces with boundary.

4.1 Topological codes on surfaces with no boundary - a homological definition

Recall the definition of the toric code introduced in part 2. We defined the toric code as a stabilizer code on a square lattice with periodic boundary conditions. Qubits were associated with edges of the lattice, and stabilizer generators, and logical Pauli operators defined in terms of properties of the lattice and dual lattice (see part 2 for details).

Let us now recast these definitions in homological terms. The square lattice represents a cellulation of the surface, with edges representing 1-cells, vertices 0-cells and plaquettes 2-cells. The dual lattice represents co-cells (0-cocells are dual lattice plaquettes, 1-cocells dual edges and 2-cocells dual vertices). Boundary maps are defined exactly as before.
4.1.1 Qubits, Z-chain operators and X-cochain operators

We now define our code as follows. Let us first set the scene:

• For the chosen cellulation, we associate a qubit with every 1-cell and with the 1-cocell that intersects it.

• We now identify tensor products of Pauli $Z$ operators with 1-chains and tensor products of $X$ operators with 1-cochains.

More specifically, let us associate with every 1-chain $c$, the Z-chain operator $Z(c)$ which we can write

$$Z(c) = \bigotimes_{x \in c} Z^x$$  \hspace{1cm} (4.1)

where $x$ is the group element assigned to 1-cell in $c$, and $Z^0 = 1$. In other words, for every 1-cell assigned 1, the operator acts as $Z$ on the associated qubit, and for every 1-cell assigned 0, the identity on that qubit.

Similarly, we can define a X-cochain operator $X(\tilde{p})$ as

$$X(\tilde{p}) = \bigotimes_{\tilde{x} \in \tilde{p}} X^{\tilde{x}}$$  \hspace{1cm} (4.2)

where $\tilde{x}$ is the group element assigned to 1-cocell in $\tilde{p}$, and $X^0 = 1$. Here, for every 1-cocell assigned 1, the operator acts as $X$ on the associated qubit, and for every 1-cocell assigned 0, the identity on that qubit.

Note that the group composition for 1-chains and 1-cochains precisely captures the multiplication algebra for these subgroups of Pauli operators.

To summarise, 1-chains are identified with tensor products of $Z$ and 1-cochains with tensor products of $X$. The commutation relation between these operators can be characterised homologically. An X-cochain operator will commute with a Z-chain operator if and only if the support of the two operators coincide on an even number of qubits. This is captured precisely by the inner product between cochains, and thus

$$X(\tilde{p})Z(c) = (-1)^{\langle \tilde{p}, c \rangle} Z(c)X(\tilde{p})$$  \hspace{1cm} (4.3)

The two operators commute if and only if the inner product between cochain and chain is zero. Note also that every $n$-qubit Pauli operator can be written $P = \alpha X(\tilde{p})Z(c)$ where $\alpha = \pm 1$ or $\alpha = \pm i$ and $\tilde{p}$ and $c$ are 1-(co)chains.
4.1.2 Stabilizer generators

The stabilizer generators will be associated with the boundaries and co-boundaries of 2-cells, and 2-cocells.

Let us consider first the plaquette generators. These were defined, for each plaquette in the lattice, as $Z$ operators acting on the edges of the plaquette. The homological definition is thus very natural

- A **plaquette generator** is the operator associated with a 2-cell $c$ and defined as the $Z$-chain operator $Z(\partial c)$ corresponding to the boundary of $c$.

We identify vertices on our lattice with 0-cells and corresponding 0 − cocells, vertex generators are then defined as $X$-cochains:

- A **vertex generator** is the operator associated with a 0-cocell $\tilde{p}$ and defined as the $X$-cochain operator $X(\tilde{\delta}\tilde{p})$ corresponding to the boundary of $c$.

We can now prove that this set of operators commute for any cellulation. From equation (4.3), the operators commute if the inner product of chains and cochains are zero, i.e. if for all 2-cells $c$ and for all 0-cocell $\tilde{p}$,

$$\langle \tilde{\delta}\tilde{p}, \delta c \rangle = 0$$  \hspace{1cm} (4.4)

But via the definition of co-boundary,

$$\langle \tilde{\delta}\tilde{p}, \delta c \rangle = \langle \tilde{p}, \delta \delta c \rangle$$  \hspace{1cm} (4.5)

but further, the fundamental lemma of homology states that $\delta \delta = 0$, hence

$$\langle \tilde{\delta}\tilde{p}, \delta c \rangle = \langle \tilde{p}, 0 \rangle = 0$$  \hspace{1cm} (4.6)

where we use the fact that the inner product with a null chain is zero.

Thus, defined in these terms, the commutation of the stabilizer generators follows from two foundational definitions in homology and cohomology theory. Since the above definitions are valid for any closed compact surface and any cellulation, the proof holds for any topological code defined according to any cellulation of a closed compact surface in the above manner.

Note that the plaquette/vertex generators are (by definition) the generators of the 1-boundary / 1-boundary groups $B_1$ and $B_0$. 

---

3
4.1.3 Logical operators

Having established that the code stabilizer can be defined (and its commutation verified) solely in homological (and cohomological) terms, we now wish to determine the number of encoded qubits in the code and complete their definition by specifying the logical Pauli operators.

Previously we determined the number of logical operators by counting independent stabilizer generators and qubits. However, these numbers are cellulation dependent. We need a more general homological approach.

This approach is given to us by the first homology and first cohomology groups. Recall that the logical Pauli operators are operators within the centraliser of the code - i.e. they commute with every stabilizer - and that they are divided into equivalence classes, where equivalence is up to multiplication by an element of the stabilizer.

It suffices to define encoded $Z$ and $X$ operators for every encoded qubit. Let us consider the encoded $Z$ operators first. We wish to find the set of $Z$-chains $Z(\tilde{c})$ which commute with every stabilizer generator. We are thus looking for the set of 1-chains $\tilde{c}$ which satisfy:

$$\langle \tilde{\partial} \tilde{p}, \tilde{c} \rangle = 0$$

for every 0-chain $\tilde{p}$. Now let us apply the definition of co-boundary:

$$\langle \tilde{\partial} \tilde{p}, c \rangle = \langle \tilde{p}, \partial c \rangle = 0$$

for every 0-chain $\tilde{p}$. The only $n$-chain whose inner product is zero with all $n$-cochains is the null $n$-chain. Hence, equation 4.8 can only hold for every 0-chain $\tilde{p}$ if $\partial c = 0$.

This is the set of 1-chains with zero boundary. Thus the set of $Z$-chains $Z(c)$ which commute with every stabilizer generator is the set where $c$ is a 1-cycle, $c \in C_1$. Let us call such an operator a $Z$-cycle.

Analogously, one can show that the set of $X$-chains $X(\tilde{p})$ which commute with every stabilizer generator is the set of $X$ co-cycles, where $\tilde{p}$ is a 1-co-cycle, $\tilde{p} \in C^1$.

Since every Pauli operator is a product of a $Z$-chain and an $X$-cochain up to a phase, we have identified the full centralizer of the code, consisting of products of a $Z$-cycle and an $X$-co-cycle.

Thus 1-cycles and 1-co-cycles on the lattice represent logical operators, but to complete the analysis we must group them into equivalence under
4.1. Topological codes on surfaces with no boundary - a homological definition

multiplication by a stabilizer element. Above we identified that the stabilizer operators consist of Z-chains defined by 1-boundaries, and X-cochains defined by 1-coboundaries. Equivalence up to stabilizer multiplication is therefore equivalence under addition with a (co)boundary - this is nothing other than the definition of (co)homological equivalence we encountered previously.

Therefore we can describe the equivalence classes of logical operators as the equivalence classes under homology. This is nothing other than the elements of the 1st homology group. Hence:

- The logical $Z$ operators are identified the elements of the 1st homology group.
- The logical $X$ operators are identified with the elements the 1st cohomology group.

One can verify that these sets of operators commute and anticommute as is necessary to represent these operators (proving this is left as an exercise - one can choose a set of chain generators for $H_1$ and a set of cochain generators for $H^1$ such that each chain generator intersects with just one cochain generator).

The logical Pauli operators are thus fully characterised by the 1st homology groups of the surface. The number of encoded qubits is then determined by the rank of this group - it follows that the 1st Betti number $\beta_1$, the rank of $H_1$ determines the number of qubits encoded in the code. (Note that isomorphism between $H_1$ and $H^1$ ensures that the two groups have equal rank, as is necessary (since we need equal numbers of independent encoded $X$ and $Z$ operators).

Note that the distance of the code - the minimal weight of the set of non-identity logical Pauli operators - is not determined by the homology of the code and is therefore not a topological property. It depends upon the detailed implementation of the code (e.g. for the toric code on a square lattice, the size of that lattice).

4.1.4 Error detection

The error detection and correction properties of these codes can also be understood in homological terms. From the analysis in part 2, the conclusion is almost immediate.
Let us consider first $Z$-type errors only. Here the error operator is now represented by a $Z$-chain $Z(c)$. Errors are detected by measuring generators of the stabilizer, and the syndrome for this error will be represented by the set of vertex generators which anticommute with $Z(c)$, i.e. the set of 0-cocells $\tilde{v}$ which satisfy,

$$\langle \tilde{\partial} \tilde{v}, c \rangle = 1$$  \hspace{1cm} (4.9)

but from the definition of coboundary:

$$\langle \tilde{\partial} \tilde{v}, c \rangle = \langle \tilde{v}, \partial c \rangle = 1$$  \hspace{1cm} (4.10)

This is only satisfied when the primal lattice vertices corresponding to $\tilde{v}$ lie in the boundary of $c$. Hence the set of vertices whose outcomes are flipped by this error are the 0-cocells which intersect the boundary of $c$. On the primal lattice, we’d thus identify the error syndrome corresponding to $c$ with the set of vertices at its boundary $\partial c$.

Similarly, the syndrome associated with an $X$-cochain is the set of plaquettes at the coboundary of this cochain.

### 4.1.5 Error correction

Recall that to correct an error, it suffices to apply a correction operator which is equal to it up to multiplication with an element of the stabilizer. This has a direct homological interpretation. A correction chain $Z(c')$ will correct error chain $Z(c)$ if $c$ and $c'$ are homologically equivalent.

If a decoder returns a correction chain $d$ which shares the boundary of $c$ but is in a different homological equivalence class, then applying this correction will lead to an encoded error and represents a correction failure.

This provides a compact description of the optimal decoder - it is the decoder which returns the most likely homological equivalence class for the error consistent with the observed syndrome.

The minimal weight perfect matching algorithm, on the other hand, returns the shortest weight correction chain (not a homological concept) which is usually, though not always in the most likely homological equivalence class. This is the reason why this decoder performs well - but is not optimal.
4.1.6 Some examples and some thresholds

The above homological formulation is valid for any cellulation of any closed (i.e. boundaryless) compact surface. We therefore can immediately generalise the toric code.

For example, while keeping the torus as the defining surface, we can modify the cellulation to other lattices. For example, we can replace a square lattice with a hexagonal lattice, a kagome lattice, or indeed an irregular lattice.

The square lattice has the special feature that it is self dual - e.g. if the dual of a square lattice is a square lattice. However, most lattices are not self-dual. For example, the dual of the hexagonal lattice is a triangular lattice, while the dual of the kagome lattice is the rhombic star.

The effect of this asymmetry is that codes defined by such cellulations protect against $Z$ and $X$ errors to different degrees and have different code thresholds. The following thresholds were estimated recently by Fujii and Tokunaga, Physical Review A, 86, 020303(R) (2012), [http://arxiv.org/abs/1202.2743](http://arxiv.org/abs/1202.2743). Their estimates of thresholds for the minimum weight perfect matching decoder (so lower bounds to optimal thresholds) results are summarised in the table below:

<table>
<thead>
<tr>
<th>Primal lattice</th>
<th>Z-error threshold</th>
<th>Dual Lattice</th>
<th>X-error threshold</th>
</tr>
</thead>
<tbody>
<tr>
<td>Square</td>
<td>0.103</td>
<td>Square</td>
<td>0.103</td>
</tr>
<tr>
<td>Kagome</td>
<td>0.116</td>
<td>Rhombic star</td>
<td>0.095</td>
</tr>
<tr>
<td>Hexagonal</td>
<td>0.159</td>
<td>Triangular</td>
<td>0.065</td>
</tr>
</tbody>
</table>

This emphasises that the threshold of the toric code is not a topological property.

4.1.7 High genus codes

As already remarked, high genus $g$ closed surfaces can be used to define codes. The 1st Betti number for such surfaces is $\beta_1 = 2g$ which tells us there are $2g$ encoded qubits is $2g$, twice the number of handles in the surface.

4.1.8 Higher dimensional codes

In this lecture course, we will focus on 2-d surfaces, due to time constraints. However, the above homological definitions generalise immediately to higher
Figure 4.1: Examples of some regular lattices. The kagome lattice is dual to the rhombic star lattice. The hexagonal lattice is dual to the triangular lattice. Images from Wikipedia.
4.2 Surfaces with boundary – Planar codes

In higher dimensional spaces, where we identified qubits and qubit operators with 1-chains and 1-cochains, we can define codes where qubits are identified with \( n \)-chains and \( n \)-cochains. All of the above homological definitions are derivations go through unchanged.

For example, in the 4-dimensional toric codes, qubits are identified with 2–chains and 2–cochains, the 2 dimensional faces of the three-dimensional cubes which form the “faces” of the hypercube 4-cells. The stabilizer generators correspond to the boundary of 3-cells and 1-cocells (cubes on the primal and dual lattices) and thus are weight 6 (the cube has 6 faces). Logical operators here are identified with the 2nd homology and cohomology groups, which have rank 6. The 4-dimensional toric code thus encodes 6 logical qubits.

This change in dimensionality gives the 4-dimensional toric code some properties not shared by its 2-d equivalent. Most importantly, it can be used to define a self-correcting quantum memory. For more details, see Dennis et al, Physical Review A, J. Math. Phys. 43, 4452-4505 (2002) [arxiv.org/abs/quant-ph/0110143].

4.2 Surfaces with boundary – Planar codes

So far, we have focussed almost entirely on closed surfaces, surfaces without a boundary, both in terms of homology and in terms of the codes we consider. A disadvantage of this from the code point of view, is that the practical advantage of using local stabilizers, stabilizers where the measurement is on a small subset of close-lying qubits, is lost by the need to construct a toric surface. It would be desirable to construct a code on a flat planar surface. However, finite planar surfaces can never be closed – they always have a boundary.

In this section we will consider homology on surfaces with boundary and show how they can be used to define topological codes. We will see in the following section that the presence of boundaries provides extra structures that give us new ways to define qubits.

4.2.1 Homology of a simple planar surface

Consider the simple cellulation illustrated in figure 4.2. This surface has a single boundary surround it, and is homeomorphic to a disk. In common
with the surfaces we have seen before, we can identify 0-cells (vertices), 1-cells (edges) and 2-cells (plaquettes), and thus can define 0-chains, 1-chains and 2-chains entirely analogously to before.

Just as before we can identify cycles, boundaries and thus homology groups. The 0th homology group is very similar to the cases we saw earlier, we divide the 0-chains into subsets of even numbers of vertices (in $B_1$) and odd numbers of vertices (not in $B_1$). $H_0$ is thus isomorphic to $\mathbb{Z}_2$, and thus $\beta_0 = 1$.

The 1st homology group for this surface is trivial, every closed loop of edges is the boundary of a 2-chain. Thus $H_1$ is also isomorphic to the trivial group $\mathbb{Z}_1$, and thus $\beta_1 = 0$.

Both of these examples so far have been very similar to the homology groups of a sphere. The second homology group $H_2$ exposes the first difference with respect to closed surfaces. The set of 2-cycles for the sphere consisted of two inequivalent 2-chains, the null 2-chain and the “all ones” 2-chain. The all-ones 2-chain was a cycles since the surface had no boundary. If we now return to our example of a surface with boundary we see that the

Figure 4.2: A cellulation of a disk, its single boundary surrounds the entire surface.
4.2. Surfaces with boundary – Planar codes

“all-ones” 2-chain is no longer a cycle – it has a boundary – the boundary of the surface. Thus the second homology group $H_2$ contains just one element and is isomorphic to the trivial group. Thus $\beta_2 = 0$.

We saw previously (and you can verify by counting the figure) that $\chi = 1$ for a surface with a single boundary. We can verify that the Betti numbers and Euler characteristic satisfy the relation $\chi = \beta_0 - \beta_1 + \beta_2$.

Since the number of encoded qubits in a surface is equal to $\beta_1$, there would be no encoded qubits on a topological quantum code defined by a disk. Before we consider cohomology of planar surfaces, therefore let us consider a more complicated surface - a surface with 2 boundaries.

To summarise the homology groups of a disk:

- $H_0 = \mathbb{Z}_2$, $\beta_0 = 1$
- $H_1 = \mathbb{Z}_1$, $\beta_0 = 0$
- $H_2 = \mathbb{Z}_1$, $\beta_0 = 0$

4.2.2 Homology of a disk with a hole

Now consider the cellulation illustrated in figure 4.3. This surface is homoeomorphic to a disk with a single hole in it. The hole provides a second boundary, (thus this surface has two boundaries) and means that the “plaquettes” inside the whole is not part of the surface.

Let us first compute the Euler characteristic of the surface. After a count we compute that $\chi = 32 - 80 + 48 = 0$. The zeroth homology group $H_0$ and the second homology group $H_2$ are unchanged by this new hole. The reasoning used to derive them for the disk applies here as well. It is unaffected by the hole.

The first homology group, however, is changed by the hole in the surface. Consider the 1-chain indicated in figure 4.4. This is clearly a 1-cycle, however it is the boundary of no 2-chain. This surface therefore has a non-trivial first homology group.

You can convince yourself that all 1-cycles which encircle the whole an odd number of times are inequivalent to the 1-cycles which encircle the hole an even number of times. The 1-st homology group therefore consists of two equivalence classes and is isomorphic to $\mathbb{Z}_2$. The first Betti number is $\beta_1 = 1$.

For surfaces with no boundary, the first Betti number was equal to the number of qubits encoded on any topological code defined via a cellulation
on that surface, using the definitions in part 4.1. We might therefore hope that the disk with a whole would encode a logical qubit in a similar manner.

To generalise our definition of topological codes, however, we first need to explore the cohomology of surfaces with a boundary.

Before we do so, let's summarise the homology groups for a disk with a hole (a surface with 2 boundaries)

- $H_0 = \mathbb{Z}_2, \beta_0 = 1$
- $H_1 = \mathbb{Z}_1, \beta_0 = 1$
- $H_2 = \mathbb{Z}_1, \beta_0 = 0$

4.2.3 **Cohomology of surfaces with a boundary.**

We define cochains for surfaces with a boundary in the same way as for closed surfaces. Formally the cochains represent linear functionals (group
4.2. Surfaces with boundary – Planar codes

Figure 4.4: A non-trivial 1-cycle. This 1-cycle is not the boundary of any 2-chain. Note that this non-trivial cycle “wraps around” the hole, similar to how the non-trivial cycles on the torus wrapped around the handle. In contrast to the torus, however, there is only one way to encircle the hole. All cycles which encircle the hole are homologically equivalent. The first homology group is thus isomorphic to \( \mathbb{Z}_2 \) and the first Betti number \( \beta_1 = 1 \).

homomorphisms to \( \mathbb{Z}_2 \) on each chain group. More concretely, we can make the same identification as before – identifying cocells with vertices, edges and plaquettes on the dual lattice.

Thus the 0-cocells are associated with plaquettes on the dual lattice dual to the vertices on the primal lattice, the 1-cocells are associated with dual lattice edges, and 2-cocells with dual lattice vertices. You can verify that the \( n \)-cocells defined in this way do generate the full \( n \)-cochain groups via the inner product definitions that were defined in part 3 – to recap, the 0 and 2-cochain / chain inner products count the number mod 2 of vertices of the (co)chain lying inside plaquettes from the (co)chain. The 1-chain / cochain inner product is the number mod 2 of edges of the 1-chain which intersect the edges of the cochain.
So far, the presence of a boundary on this surface has not had a large effect on our definitions. However, the presence of boundaries has important implications for the coboundary map. Consider, a dual lattice edge (1-cocell) adjacent to the boundary. Normally the coboundary of a dual lattice edge would be the two dual lattice vertices adjacent to the edge. For a dual lattice edge adjacent to the boundary, however, there is only one adjacent dual vertex. The other “end” of the edge is beyond the boundary of the lattice where there is no dual vertex. We therefore must define the coboundary map for this dual lattice edge as the single dual lattice vertex adjacent to it, as depicted in figure 4.7. The reader can verify that this is a valid coboundary map satisfying the defining equation (3.21).

The 2-boundary map for 0-cocells (dual lattice plaquettes) adjacent to the edge is similarly affected. Such plaquettes may have one or two of its expected coboundary dual lattice edges outside of the surface. We thus redefine the boundary map for these 0-cocells accordingly (see figure 4.8). 2-coboundary maps are unaffected by the presence of a surface boundary. The 2-coboundary map sends every 2-cochain to 0.
4.2. Surfaces with boundary – Planar codes

Figure 4.6: The dual lattice cellulation of a disk with a hole depicted on its own. Note the effect that a boundary on the primal lattice has on the “form” of the dual lattice near that boundary.

Figure 4.7: When a surface has a boundary, the 1-coboundary map for the 1-cocells adjacent to the boundary must be modified as shown.
Figure 4.8: An example of the 0-coboundary map for a 0-cochain consisting of two dual plaquettes adjacent to the surface boundary. The coboundary map is modified as shown to remove dual lattice-edges which would lie outside the surface.

The redefined coboundary maps have a profound affect on the cohomology groups. Let us first consider the group of 1-cocycles. Recall that the 1-cocycles are the group of 1-cochains with no coboundary (i.e. their coboundary is the null cochain). As before, closed “loops” of edges represent 1-cocycles. However, there is now a new type of possible 1-cocycle. A “string” of edges which passes starts and ends at a boundary also has null coboundary (due to the modified coboundary map). It is thus also a 1-cocycle. See figure 4.9 for examples of 1-cocycles.

Of these 1-cycles, which “begin” and “end” at the primal lattice boundaries can be split into two homological equivalence classes. Those whose source and termination is the same boundary are homologically equivalent to the null 1-cocycle, while those that connect two different boundaries form a distinct homology class, inequivalent to the null 1-cocycle. There are, in fact, just two homological equivalence classes of 1-cocycles. Those homologically equivalent to the null cocycle, and those homologically equivalent to a string between the two different boundaries. The group $H_1$ is therefore isomorphic to $\mathbb{Z}_2$ and its rank, the second Betti number $\beta_2$ is 1.

For completeness, we should briefly consider the group $H^0$. We find that this group is isomorphic to $\mathbb{Z}_2$, since, just like the closed surface, the “all-ones” 0-cochain has null boundary (this is due to the “missing” dual edges which would lie outside the surface), and, hence, like a closed surface, $H^0$ has two inequivalent elements.

We see that the non-trivial $H_0$ group – reflecting the fact that in the primal lattice, every edge (1-cell) has a boundary chain containing 2 vertices –
4.2. Surfaces with boundary – Planar codes

Figure 4.9: 1-cocycles on a surface with a pair of boundaries. As well as forming loops, 1-cocycles can “begin” and “end” at the primal lattice boundary. The upper two 1-cocycles are representatives of the trivial homological equivalence class, the lower two of the (only) nontrivial class.

goes hand in hand with the non-trivial $H^0$ group – which reflects the lack of a “boundary” on the dual lattice. At the same time, the trivial $H_2$ group – reflecting the fact that every 2-chain except the null chain now has a boundary, so $Z_2$ has only one element – is matched by a trivial $H^2$ group, where the fact that some dual lattice edges have only one dual vertex in their coboundary means that every 2-cochain is a 2-coboundary. This symmetry is just a hint of the elegance of cohomology.

To summarise:

- $H_0 = Z_2, \beta_0 = 1$
- $H_1 = Z_1, \beta_0 = 1$
- $H_2 = Z_1, \beta_0 = 0$
Figure 4.10: Examples of plaquette and vertex stabilizer generators for the “disk with hole” planar code. Note that the presence of the boundary lowers the weight of the adjacent vertex operators, but does not do so for adjacent plaquette operators (reflecting the fact that the presence of the boundary modified the coboundary maps not the boundary maps).

For us, these symmetries have a fortunate consequence, since we need $H_1$ to be isomorphic to $H^1$ to define a quantum code. The Betti numbers $\beta_1 = 1$ implies that this code encodes 1 qubit, and we shall see an explicit construction in the next section.

4.2.4 A planar surface code on a disk with a hole

Using the definitions of boundary and coboundary defined at the beginning of this chapter, we can immediately define a quantum code. This is the planar surface code for the disk with a hole.

Plaquette stabilizer generators are defined as the $Z$-chains $Z(\partial(c))$ for every 2-cell $c$. Vertex operators are also defined via the coboundary map, and their form is modified at the boundary reflecting to modified coboundary maps here (see figure 4.10).

A single qubit is encoded in the code (since the 1st Betti number is 1) and logical operators are defined by the generators of the first homology and first cohomology groups defined above.

Notice that we did not need to consider the independence or otherwise of the plaquette and vertex operators to determine the number of encoded
4.2. Surfaces with boundary – Planar codes

Figure 4.11: Examples of logical Z and logical X operators with minimum weight. Note that this minimum weight is determined by the “radius” of the hole and how far it lies from the boundary of the surface.

qubits. The homology and cohomology – specifically the 1st Betti numbers – take care of this for us. (As an exercise, work out the relationship between the independence of the plaquette and vertex generators here, and the 0th and 2nd Betti numbers).

Looking at figure 4.11 we see that the weight of the lowest weight $Z$ operator is given by the circumference of the hole and the weight of the lowest weight $X$ operator is given by the minimum (spatial) distance (plus one) between the boundary of the surface and the boundary of the hole. We call this type of qubit a primal hole qubit (below we will meet “dual hole qubits”).

Error detection and correction for such a qubit is analogous to the toric code, and similar decoder strategies (e.g. minimum weight perfect matching) can be used. Note, however, that there are now only 2 homology classes for errors on this code. The code threshold for such qubits was found (by Dennis, Landahl, Kitaev and Preskill) to be similar to the (square lattice) toric code at approximately 11%.

4.2.5 Surfaces with many holes

What if add further “holes” to the surface? This is straightforward to analyse using the same methods as above. The number of qubits is equal to the first Betti number of the surface. We find that, in general, such a code with $n$ holes (and thus with $(n + 1)$ boundaries has Betti number $\beta_1 = n$ and will
Figure 4.12: A surface with (primal) boundary and \( n \) (primal) holes encodes \( n \) qubits. Minimal weight logical operators are indicated, the thick black lines, 1-cycle Z operators which enclose the holes, and the green lines 1-cocycle X operators stretching from the hole boundary to the surface boundary. It is clear that these operators respect the commutation relations for X and Z since each X-cochain intersects only with its respective Z cochain.

thus encode \( n \) logical qubits. Logical Z and X operators for such a code are illustrated in figure 4.12.

4.2.6 A different kind of boundary - “dual boundary” and “dual holes”

There was a striking assymmetry between the effect of adding a boundary to the surface between the boundary and coboundary maps and the first homology and first cohomology groups.

The boundary operators and first homology group for this surface was very similar to what we saw for closed surfaces. The coboundary operators, however, needed to be modified to take the boundary into account, and the weights of the boundary chains for 1-cells and 2-cells adjacent to the boundary was reduced. This had an effect on the structure of the cohomology groups. For example, the 1st cohomology group now consisted of stringlike 1-chains stretching from one boundary to the other, rather than “loop-like” chains.

This assymmetry can be very clearly seen in the “shape” of the boundary itself. See figure 4.13 for an example. On the primal lattice, we see a “smooth” boundary, but on the dual lattice, the boundary is “spiky” or
4.2. Surfaces with boundary – Planar codes

Figure 4.13: The primal lattice (left) and the dual lattice (right) for a surface enclosed by a (primal) boundary with a single (primal) hole.

Figure 4.14: The primal lattice (left) and the dual lattice (right) for a surface enclosed by a dual boundary with a single dual hole.

“rough” with adjacent dual lattice edges connecting with no dual vertex, and adjacent dual lattice plaquettes missing one or more sides – both effects forcing us to redefine the boundary map. Similarly, around the hole in the lattice, we see a smooth boundary on the primal and a spiky boundary on the dual lattice.

We can, however, reverse these properties and define a dual boundary. A dual boundary creates a smooth boundary on the dual lattice and a spiky
boundary on the primal lattice – see figure 4.14. We call a hole which has a dual boundary a dual hole. To distinguish these two types of boundary, we call the type of boundary introduced previously a primal boundary and a hole with such a boundary a primal hole.

When a surface has a dual boundary or a dual hole, the boundary maps for 1-cells (edges) and 2-cells (plaquettes) along that boundary need to be redefined - removing the vertices and edges which would lie beyond this dual boundary and which are not part of the surface.

We can construct the homology and cohomology groups for this surface – a cellulation of a disk with dual boundary and a dual hole – and construct a corresponding quantum code. This code is depicted in figure 4.15. This surface encodes a single bit, as expected, due to the similarity with a primal hole qubit, and we see that the logical operators share the structure of the primal hole qubit, just with the logical $X$ and logical $Z$ exchanging roles. The 1st cohomology group generator (and hence logical $X$) is now a loop around the dual hole, while the 1st homology group generator (and hence logical $Z$) is a stringlike chain connecting the two dual boundaries. We call a qubit defined in this way a dual hole qubit.

### 4.2.7 Surfaces with mixed boundary

Having defined primal and dual boundaries, we can now consider surfaces with both types of boundary. Consider the surface in figure 4.16. The boundary of this surface is divided into 4 regions, alternating primal - dual - primal - dual. The alternating boundary allows for the existence of a non-trivial first homology and cohomology groups – (as an exercise, convince yourself that a surface whose boundary is divided into a single primal boundary and a single dual boundary has only trivial 1st (co)homology groups with Betti number $\beta_1 = 0$ and thus encodes no qubits).

We find that the first Betti number for this surface is 1, and that 1st homology and cohomology groups are isomorphic to $\mathbb{Z}_2$ - a single qubit can therefore be encoded. The generators of the (co)homology groups and hence the encoded logical operators are depicted in figure 4.17. The distance of this code is the determined by the width or height (whichever is smallest) of the lattice.

By alternating the boundary type further one can encode more qubits. Determine, as an exercise, how many qubits are encoded on a disk where the boundary alternates $n$ times (where $n$ is an even number).
4.2. Surfaces with boundary – Planar codes

4.2.8 Twin-hole qubits

We have seen that the presence of primal and dual boundaries provides new ways to create qubits on a planar surface code, and that this can be achieved by inserting primal or dual holes into the surface, and by alternating the boundary type along a single boundary.

The planar surface code is the basis for a fully fault tolerant universal model of quantum computation - developed by Raussendorf, Harrington and Goyal within an equivalent (but different in some details) measurement-based quantum computation framework – which we will survey in the final part of this course.

In fault tolerant quantum computation, it is useful to be able to change the distance of an encoded qubit, initialise fresh ancilla qubits and remove unneeded measured qubits. In the next part, we will show explicitly how this can be achieved via code deformation, changing the shape of holes and boundaries, to change a codes properties. On a toric code or the mixed boundary planar code illustrated above, to change the code distance you

Figure 4.15: A dual hole qubit, defined on a surface enclosed by a dual boundary with a single dual hole.
Figure 4.16: A cellulation of a disk with 4 alternating boundary types - primal, dual, primal, dual. The surface defines an encoding of a single qubit as a topological code. Some example stabilizer generators (reflecting the redefined boundary maps due to the boundaries) are depicted.

Figure 4.17: The encoded logical $X$ and $Z$ operators for the cellulation of a disk with 4 alternating boundary types - primal, dual, primal, dual.
4.2. Surfaces with boundary – Planar codes

need to change the lattice dimension or change the boundaries. While possible in principle, this would be cumbersome in practise.

A more elegant solution is provided by twin-hole qubits. In the next section we will see how such qubits can be robustly created, initialised and how their distance can be changed. Here, we shall define the qubits.

The basic idea in a twin-hole qubit is to use two holes of the same type – a primal with a primal or a dual with a dual – to represent a single qubit. This is an example of a very simple repetition encoding.

Consider the 2-qubit repetition code, with code words

\[ |0\rangle_L = |0\rangle|0\rangle \quad |1\rangle_L = |1\rangle|1\rangle \]  (4.11)

The stabilizer for this code is \( Z_1Z_2 \) and the logical operators are \( \overline{Z} = Z_1 \) and \( \overline{X} = X_1X_2 \).

We can create such a repetition code using two primal hole qubits on a planar surface code. Consider the two primal holes depicted in figure 4.4. We can define a redundantly encoded qubit from these two holes by defining encoded \( \overline{Z} = Z_1 \) and \( \overline{X} = X_1X_2 \). In this way we have a single qubit represented by the pair of holes.

There are a number of advantages in this representation. Notice that logical operators longer contains strings to the surface boundary and thus the encoded qubit is more “self-contained” with the pair of holes. This becomes useful when one wants to encode many qubits in a surface and implement encoded quantum gates.

Notice also that the weight of the code depends only on properties of the pair of qubits. The minimal weight of the logical \( Z \) is given by the radius of the smallest hole, while the minimal weight of the logical \( X \) is determined by the distance between the holes.

The two-qubit repetition code has a single stabilizer generator \( Z_1Z_2 \). To fully protect this encoding, we would need to measure this operator - and add it to our set of stabilizer generators. But this operator is of very high weight - consisting of Pauli Z’s around the boundaries of both holes. We would not want to have to measure such an high-weight operator to check errors. Fortunately, we don’t need to. The surface code is protecting the individual hole qubits, and the sort of error which the \( Z_1Z_2 \) would detect would be a logical \( X \) on one of the qubits. Thus if the individual hole qubits are protected against \( X \) errors (which they are, in the surface code) the twin hole qubit is protected, and the encoded \( Z_1Z_2 \) operator does not need to be measured.
Figure 4.18: An example two hole qubits, and the corresponding twin (primal) hole qubit, created from them by using a 2-qubit repetition code, where $\bar{Z} = Z_1$ and $\bar{X} = X_1X_2$. The shaded green area represents the 2-cocell whose boundary operator transforms the product of operators $X_1$ and $X_2$ to the homologically equivalent, but lower weight, form depicted here.
4.2. Surfaces with boundary – Planar codes

This surface encodes two qubits via a twin primal hole qubit and a twin dual hole qubit. The grey shaded holes represent primal holes and the green shaded holes represent dual holes.

This does not mean that we can ignore the $Z_1Z_2$ altogether. As we see in the next section, we need to measure it to initialise a twin qubit in a certain state.

Via an analogous construction, we can define a twin dual hole qubit, from a pair of dual hole qubits. We use a repetition code in the $|\pm\rangle$ basis, i.e.

$$|+\rangle_L = |+\rangle|+\rangle \quad |-\rangle_L = |-\rangle|-\rangle$$  \hspace{1cm} (4.12)

which has stabilizer generator $X_1X_2$ and logical operators are $\tilde{Z} = Z_1Z_2$ and $\tilde{X} = X_1$. In the fault-tolerant quantum computing scheme we will consider in the next part, we shall use twin primal qubits and twin dual qubits to encode our quantum bits. An example of a surface with both sorts of qubits is depicted in figure 4.19.

4.2.9 A remark on terminology

Before we complete this section, a brief note on terminology. Primal and dual boundaries are not concepts widely used in homology theory – mathematicians usually consider only a single type of boundary (our primal boundary) on the surfaces they study – therefore the quantum researchers studying planar surface codes have invented their own terminology. Unfortunately there is no consistency in the literature and different authors use different terms.
For example, in the first paper on planar surface codes, Bravyi and call primal boundaries Z-boundaries and dual boundaries X-boundaries (since the logical Z/X operators can “start” and “finish” at these boundaries). Other authors call primal boundaries “smooth” and dual boundaries “rough” due to their appearance.

In the final section of this course, we will outline how planar surface codes with twin primal hole and twin dual hole qubits may be used to implement fault tolerant quantum computing.