Quantum Coding Theory	**(UC Berkeley CS294, Spring 2024)**

Lecture 12: Topological Codes

March 6, 2024

Lecturer: John Wright | *Scribe: Hongxun Wu*

1 Recap: Toric Code

Last lecture, we learnt what is a Toric Code. We first quickly recap the basics. Below, Figure 1 shows a Toric Code. The dots on the edges are the qubits. The left (resp. top) side of the grid is indentified with the right (resp. bottom) side, so that it topologically forms a torus.

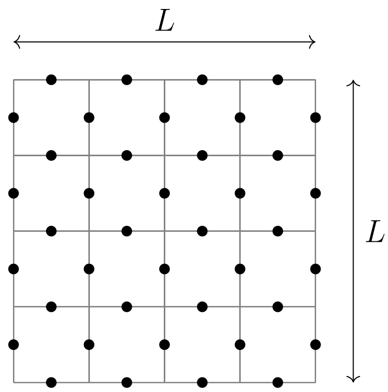


Figure 1: A toric code.

We have the parity checks:

- X-basis parity checks ($C_X^\perp$): For every vertex, there is a parity check with Z-Pauli matrices over the neighbouring qubits.

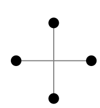


Figure 2: An X parity check.

- Z-basis parity checks ($C_Z^\perp$): For every plaquette, there is a parity check with X-Pauli matrices over the neighbouring qubits.

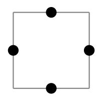


Figure 3: A Z parity check.

They generates the stabilizer group S. Furthermore, we have the following geometric intepretation of C_X and $C_Z^{\perp}$ that helps us in calculating the distance:

- All codewords in C_X has to pass the parity checks in $C_X^{\perp}$ and has an even intersection with every vertex. So they are linear combinations of cycles.

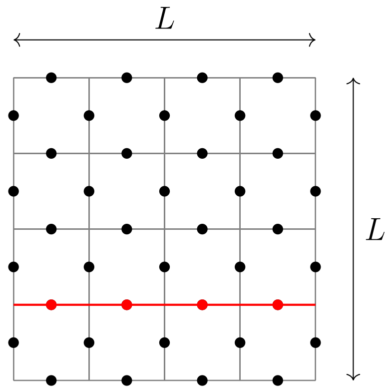


Figure 4: An element in C_X.

- For a codeword in $C_Z^{\perp}$, becuase it is the linear space generated by all the Z-basis parity checks (boundries of plaquettes), it is the linear combination of all boundries.

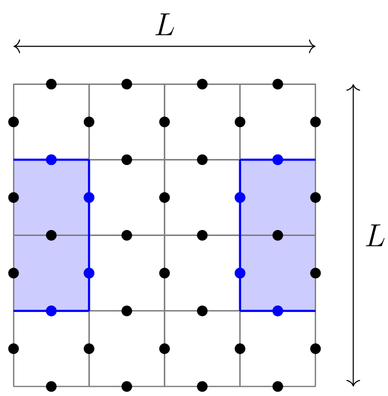


Figure 5: An element in $C_Z^{\perp}$. (It is the boundary of the shaded area.)

As required by CSS codes, we can see that $C_X \subset C_Z^{\perp}$ because any boundary is also a cycle.

Then, recall that for CSS code, we defined

$$d_X^+ = \min_{c \in C_X \setminus C_Z^\perp} |c|.$$

Here $C_X \setminus C_Z^\perp$ are the cycles which are not boundaries. For example, the one in Figure 4 is a cycle but not a boundary. In fact, we can see fairly easily that it is the smallest one. So $d_X^+ = L$.

Then for d_Z^+, we can repeat the same reasoning for the dual lattice, as shown below in Figure 6. Thus $d_Z^+ = L$ as well. The code has distance $\min\{d_X^+, d_Z^+\} = L$.

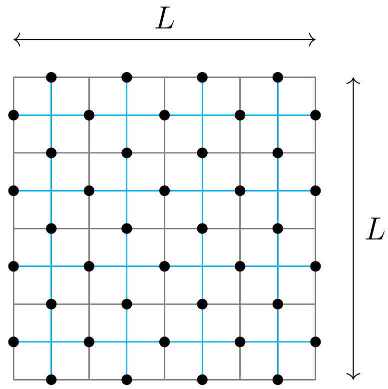


Figure 6: The Dual Lattice.

Finally, let us define the notion of equivalent cycles.

Definition 1.1 (Equivalent Cycles)**.** We define

$$C_X \setminus C_Z^\perp = \{\{c + b \mid b \text{ is boundary}\} \mid \text{c is a cycle}\}.$$

Note that this is a set of cosets (equivalency classes over cycles).

We know that cycles that are not boundaries gives us the logical operators. Two equivalent cycles just correspond to the same logical operator. For example, in the case of Toric Code, the following two cycles are equivalent:

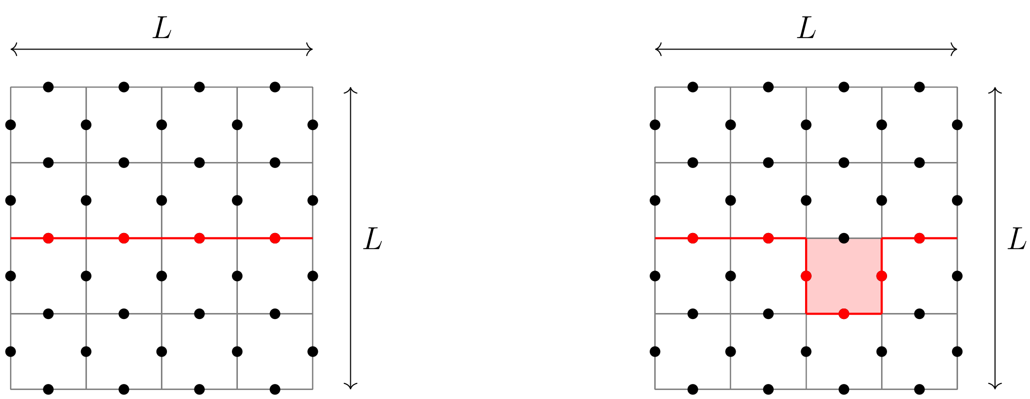


Figure 7: Equivalent Cycles (xoring the boundary of the shaded plaquette).

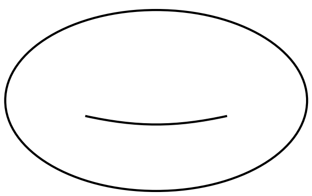
In Figure 7, the two cycles are equivalent, and they are all left-to-right cycles. In fact we can see that all left-to-right cycels are equivalent up to xoring certain plaquettes. There are only four equivalency class for Toric code, left-to-right cycles, up-to-bottom cycles, squiggling cycles (cycles that simutaneously crosses left-to-right and up-to-bottom). These gives four logical Z-operators, $C_X \backslash C_Z^{\perp} = \{I, \overline{Z_1}, \overline{Z_2}, \overline{Z_1 Z_2}\}$. As there are four logical Z-operators, there has to be exactly two qubits.

From this example, we can sort of see that these equivalency classes really only depend on the topology of this surface, but not on the way we divide the grid. It is really the topology which determines the number of qubits! We will further into this property and look at surfaces with different topology in the next section.

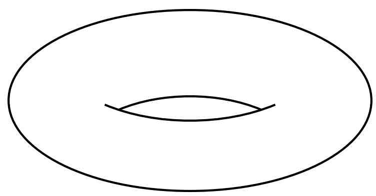
2 Surface Codes

2.1 Basic Topology

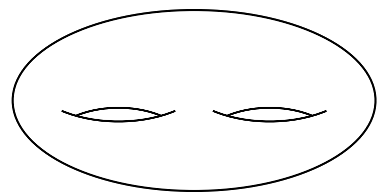
Before we dive into these surface codes, let us first see some examples of topological surfaces:



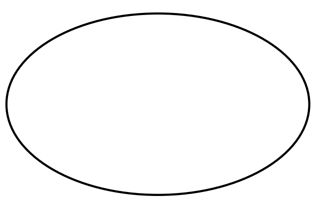
(a) A sphere.



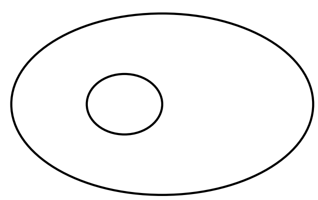
(b) A torus.



(c) A double torus.



(d) A surface with boundry.



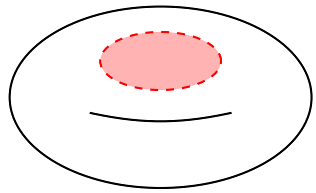
(e) A surface with boundry and a hole.

Above, we see torus with 0, 1, 2 handles. (Never call it a hole! A handle is what you can get in and go around. If you get into a hole, you cannot go anywhere.)

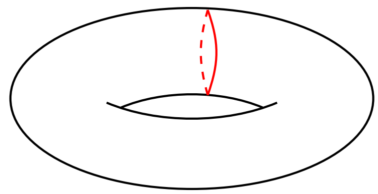
Definition 2.1 (Genus)**.** The genus of a surface is the maximum number of "cut" (non-indersecting closed loop) you can make & keep it connected. In other words, this is the number of handles on a closed face. (Here a closed surface means a face without any boundary).

Now let us see what is the genus of these surfaces:

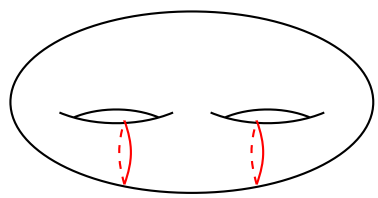
(a) For a sphere, if we draw a closed loop, its interior is immediately disconnected with the outside. So it has genus 0. (See next page for a picture illustration.)



(b) For a torus, we can only draw the following one loop to avoid disconnect it. So it has genus 1.



(c) For a double torus, we can draw two such loops. So it has genus 2.



Definition 2.2 (Cellulation)**.** The <u>cellulation</u> of a surface divdes it into polygons.

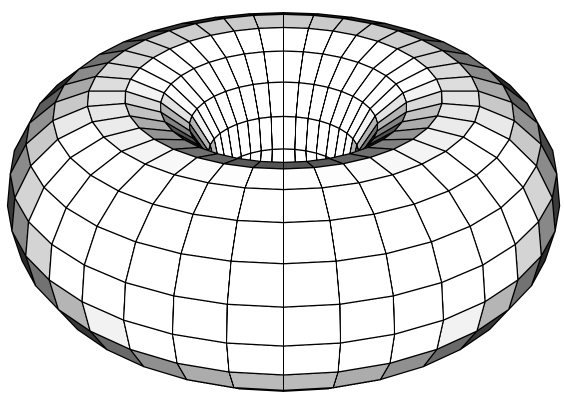


Figure 9: Cellulation of a torus.

Definition 2.3 (Euler characteristic)**.** Given a cellulation, let us denote the nubmer of vertices, edges, and faces by V, E, and F repesctively. The Euler Characteristic is defined as

$$\chi = F - E + V.$$

Now let us see a few examples for Euler characteristic.

Example 2.4. *For any cellulation of a sphere,* $\chi = 2$*.*

While for sphere it might not be obvious, let us calculate it for a cube (which is topologically equivalent to at least one possible cellulation of the sphere).

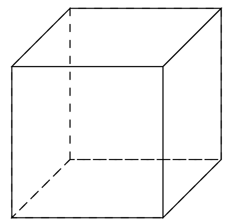


Figure 10: A cube.

For cube, we have $E = 12$, $F = 6$, $V = 8$. Thus $F - E + V = 2$ as we expected.

Example 2.5. *Given a cellulation of a* $M \times N$ *torus,* $\chi = 0$*.*

This we can actually directly calculate. We have $V = M, E = 2MN, F = MN, \chi = NN - 2MN + MN = 0$. Note that this do not depend on M, N and is a fixed number. This motivates the discovery of the following fact.

Fact 2.6. *Euler characteristic depends only on the genus of the surface, not the cellulation.*

- *For closed surfaces,* $\chi = 2 - 2g$*.*
- *For general surfaces with* b *boudaries,* $\chi = 2 - 2g - b$*.*

2.2 Surface code

Definition 2.7. Given a surface, the surface code on it is constructed by the following steps:

1. Celluate.
2. Add qubit to each edge.
3. Add X-basis parity checks to each vertex.
4. Add Z-basis parity check to each face.

Now let us calculate the number of logical qubits in this surface code.

- The number of physical qubits $= E$.
- The number of independent X-basis parity $= V - 1$. (Similar as the torus case, the xor of all parity checks are 0. So we have to remove one to make it independent.)

- The number of independent Z-basis parity $= F - 1$. (Same as above.)

Thus the number of logical qubits is the number of physical qubits minus the number of parity checks, which is $E - (V - 1) - (F - 1) = 2 - \chi = 2g$. This gives the following fact.

Fact 2.8. *There are $2g$ logical qubits in the surface code over a surface with genus g.*

For example, no matter which cellulation we pick, the double torus always gives us four logical qubits. To get more logical qubits, we only have to increase the number of handles on the surface. However, as we have seen in the case of toric code, the distance of the code actually depends on the concrete cellulation.

2.3 Example: 3d toric code.

Before we end this lecture, let us see one last example, which generalize these beyond 2D surfaces.

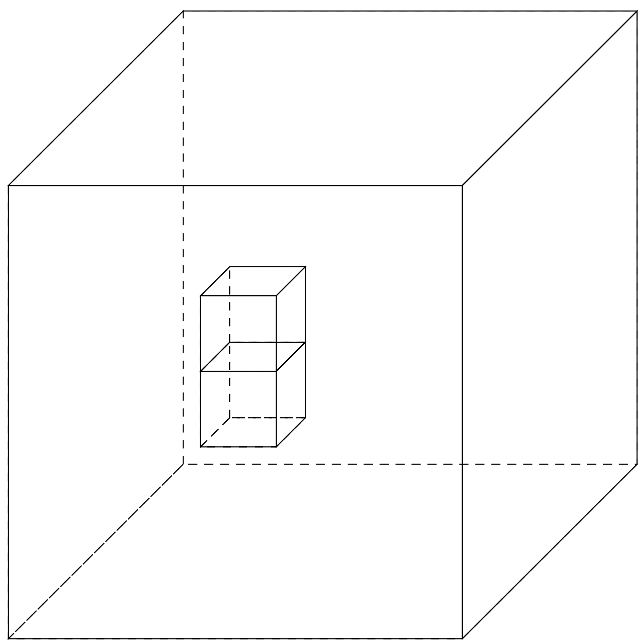


Figure 11: 3D Toric Code. Suppose we divide it to $L \times L \times L$ grid.

Suppose we again put our codewords on edges, and use faces / vertex for Z / X basis parity checks. Then we will have $\approx L^3$ edges ($=$ number of physical qubits n). But the ditance is still $\approx L$ because the loop around it only has length $4L$. Therefore, the distance becomes $\sqrt[3]{n}$. It becomes even worse!