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

Lecture 16: Hypergraph Product Codes

March 15, 2024

Lecturer: John Wright *Scribe: Francisca Vasconcelos*

Last class we saw hypergraph product codes. They are a natural family of codes that generalize the toric code and are similar to classical product codes. The hope is that they will allow us to get around some of these limitations that we saw when designing codes from topology.

1 Recall: The Hypergraph Product Code

We will begin with a reminder of how hypergraph product code construction of [TZ13] works. This code is constructed by combining two chain complexes. Namely, let (A_0, A_1) have a boundary map ∂_A and let (B_0, B_1) have a boundary map ∂_B. Thus, a *hypergraph product code* is a 2-dimensional chain complex, consisting of 2-dimensional, 1-dimensional, and 0-dimensional objects with relations depicted in Figure 1. Recall that we put Z checks on faces, qubits on edges, and X checks on vertices.

$$\begin{tikzcd}
\textbf{2-dim :} & & (e_A, e_B) \arrow[dl, dash, "\partial_A \otimes I"'] \arrow[dr, dash, "I \otimes \partial_B"] & & \text{Z checks} \arrow[ll] \\
\textbf{1-dim :} & (v_A, e_B) \arrow[dr, dash, "I \otimes \partial_B"'] & & (e_A, v_B) \arrow[dl, dash, "\partial_A \otimes I"] & \text{qubits} \arrow[l] \\
\textbf{0-dim :} & & (v_A, v_B) & & \text{X checks} \arrow[ll]
\end{tikzcd}$$

Figure 1: The chain complex of the hypergraph product code.

We saw last class that this construction results in a valid quantum CSS code. This class, we will discuss the parameters of this code.

We saw that the toric code was an example of a hypergraph product code, where you take the product of two repetition codes. The key idea was that, if you take the parameters of the individual codes translate into the parameters of the hypergraph product code. Not only do the parameters translate, but the errors of the base code also translate to the errors of the hypergraph product code. For example, an error in the repetition code is all 1s, which translates into a cycle in the toric code. We are going to see that the same thing applies more generally.

2 Properties of Codes and Cell Complexes

2.1 Properties of the 1-Dimensional Base Code

First, it will be useful to understand these 1-dimensional codes and cell complexes. Specifically, we will look at how these 1-dimensional cell complexes relate to the properties of the code.

Recall that the 1-dimensional cell complex consists of 1-dimensional objects (edges) and 0-dimensional objects (vertices). There is also a boundary map which identifies some of the vertices with the boundaries of some of the edges, as depicted in Figure 2. This looks a lot like a Tanner graph, as we saw last time. As such, we can associate a classical code to this 1D chain complex. Since the Tanner graph is symmetric in this case, we can decide whether we assign variables to the edges and parity checks to the vertices or vice versa.

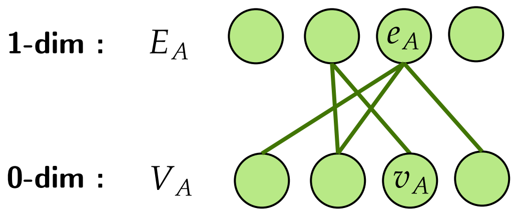

Figure 2: A 1-dimensional code cell complex.

Suppose we design a code C_A where we assign n_A variables to the edges and ℓ_A checks to the vertices. Then, the codewords are going to be the elmenents of the code, $e \in \mathbb{Z}_2^{E_A}$, with parity check 0, i.e. satisfying $\partial_A(e) = 0$. Let's now look at this code and how its properties relate to the properties of the chain complex.

The important properties of the 1-dimensional objects of this chain complex are its 1-cycles and 1-boundaries, which we compute to be:

- **1-cycles**: $Z_1 = \{e \in \mathbb{Z}_2^{E_A} : \partial_A(e) = 0\}$= codewords $= C_A$
- **1-boundaries**: $B_1 = \{0\}$

This implies that the first homology group is $H_1 = Z_1/B_1 = \{\{e\}$: e is a codeword $\}$. Note that $\dim(H_1) = \dim(C)$. Therefore, for the 1-dimensional code, its properties correspond nicely to the properties of the chain complex. In fact, the first homology group *is* the code.

Now let's do all these computations again, but this time for the 0-cycles and 0-boundaries:

- **0-cycles**: $Z_0 = \{v \in \mathbb{Z}_2^{V_A} : \partial_0(v) = 0\}$= all 0-chains
- **0-boundaries**: $B_0 = \{\partial_A(e_A) : e_A \in \mathbb{Z}_2^{E_A}\} = \{$ distinct syndromes $\}$

To elaborate, these distinct syndromes of the 0-boundaries are governed by the parity check matrix:

$$\partial_A = \begin{bmatrix} | & | & & | \\ c_1 & c_2 & \cdots & c_{n_A} \\ | & | & & | \end{bmatrix} \tag{1}$$

Concretely, the number of distinct syndromes is given by the number of distinct linear combinations of the columns, which is determined by the rank of ∂_A, i.e. $\dim(B_0) = \text{rank}(\partial_A)$. Furthermore, the dimension of the zeroth homology group, $H_0 = Z_0/B_0$ is:

$$\dim(H_0) = \dim(Z_0) - \dim(B_0) = \ell_A - \text{rank}(\partial_A) = \dim(C^T). \tag{2}$$

Now that we understand these basic properties of our base code, let's see how these can be used in a hypergraph product code.

2.2 Properties of the Hypergraph Product Code

Now we want to understand the hypergraph product code described in Section 1. For this code, the objects we care about are the 2-chains, 1-chains, and 0-chains. We depict the matrices corresponding to each of these objects in Figure 3.

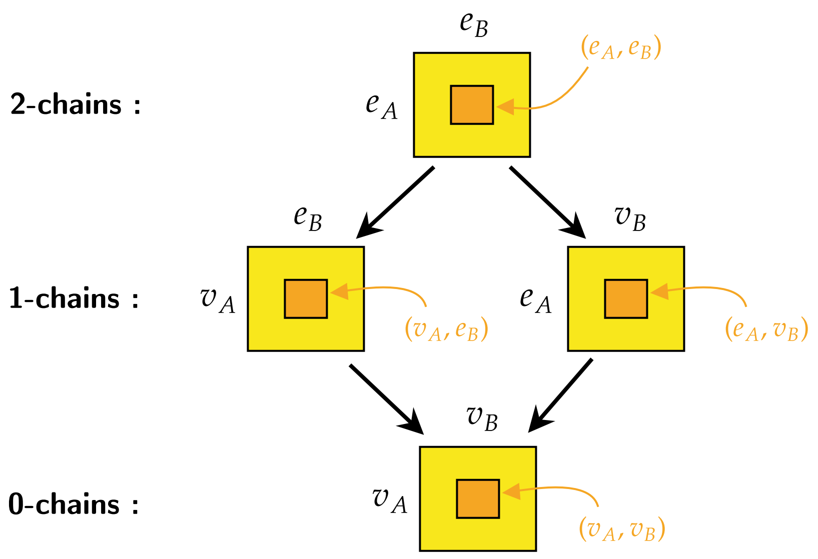

Figure 3: Depiction of matrices encoding the chain complex of the hypergraph product code.

We now want to figure out the logic operators of this code, corresponding to the first homology group of the code (which consists of all the 1-cycles with the 1-boundaries removed). This would tell us the rate (number of encoded qubits) and distance of the code. We will not go through formal proofs, but instead get intuition for what these logical operators look like.

2.2.1 Intuitive 1-Cycles

We will begin by looking at the 1-cycles, which are 1-chains where applying the boundary map results in zero for the 0-chain. For our high-level intuitive discussion, we will consider the case in which one of the two blocks in the 1-chain (depicted in Figure 3) is set to zero, i.e. only one of the blocks of the 1-chain is used. Specifically, we consider the case in which the second block is set to zero.

Furthermore, we are going to imagine that our 1-chain is expressible as a tensor product $X_A \otimes X_B$, where $X_A \in \mathbb{Z}_2^{V_A}$ and $X_B \in \mathbb{Z}_2^{E_B}$. Thus, all the non-zero columns are going to be equal to X_A and all the non-zero rows are going to be equal to X_B. As depicted in Figure 4, applying the boundary map $I \otimes \partial_B$ to this gives the 0-chain $X_A \otimes \partial_B(X_B)$. This is equal to zero when either X_A is zero (not interesting) or if the boundary of X_B is zero, which implies that $X_B \in Z_1(B)$. Intuitively, this says that the 1-cycles of the product code consist of any chain from the A code and 1-cycles from the B code. Mathematically,

$$Z_1 \approx A_0 \otimes Z_1(B) = A_0 \otimes H_1(B) = Z_0(A) \otimes H_1(B) \tag{3}$$

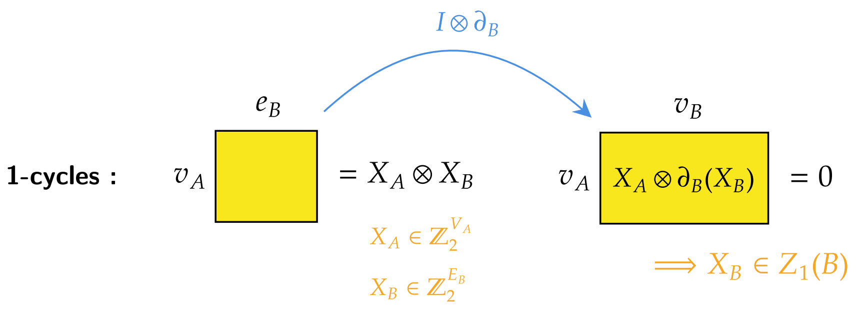
$I \otimes \partial_B$
e_B
v_B
1-cycles :
v_A
$= X_A \otimes X_B$
v_A
$X_A \otimes \partial_B(X_B)$
$= 0$
$X_A \in \mathbb{Z}_2^{V_A}$
$X_B \in \mathbb{Z}_2^{E_B}$
$\implies X_B \in Z_1(B)$

Figure 4: Intution for the 1-cycles of the hypergraph product code.

2.2.2 Intuitive 1-Boundaries

We will now consider the 1-boundaries. In this case, we will only consider 2-chains which are tensor products $X_A \otimes X_B$, where $X_A \in \mathbb{Z}_2^{E_A}$ and $X_B \in \mathbb{Z}_2^{E_B}$. Applying boundary maps $\partial_A \otimes I$ and $I \otimes \partial_B$ to this, we obtain a 1-chain that will split into two parts, as depicted in Figure 5.

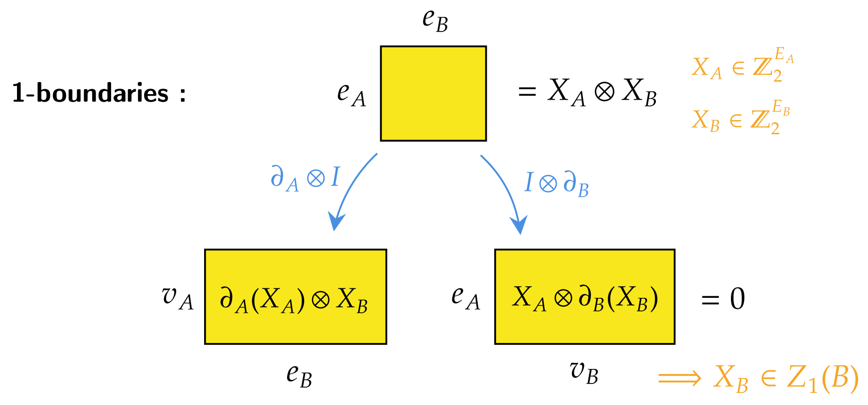
e_B
1-boundaries :
e_A
$= X_A \otimes X_B$
$X_A \in \mathbb{Z}_2^{E_A}$
$X_B \in \mathbb{Z}_2^{E_B}$
$\partial_A \otimes I$
$I \otimes \partial_B$
v_A
$\partial_A(X_A) \otimes X_B$
e_A
$X_A \otimes \partial_B(X_B)$
$= 0$
e_B
v_B
$\implies X_B \in Z_1(B)$

Figure 5: Intution for the 1-boundaries of the hypergraph product code.

Since we are considering the setting in which the second block is set to zero, this implies that $X_A \otimes \partial_B(X_B) = 0$ and, thus, $X_B \in Z_1(B)$. Mathematically, this means that

$$Z_1 \approx A_0 \otimes Z_1(B) = A_0 \otimes H_1(B) = Z_0(A) \otimes H_1(B). \tag{4}$$

2.2.3 Intuitive Logical Operators

Via this "sketchy" intuition, we have computed the 1-cycles [Equation (3)] and 1-boundaries [Equation (4)] of the hypergraph product code. Now, to find the logical operators of our code, we need to look at

$$H_1 = Z_1/B_1 \approx (Z_0(A)/B_0(A)) \otimes H_1(B) = H_0(A) \otimes H_1(B). \tag{5}$$

All of this high-level analysis was assuming that our 1-chains were non-zero on the first block and zero on the second block. However, if we instead do the reverse, where we consider that the first block is zero and the second block is non-zero, it turns out that the same analysis would result in

$$H_1 = H_1(A) \otimes H_0(B). \tag{6}$$

So, overall, allowing both blocks to be non-zero results in the first homology group

$$H_1 = (H_0(A) \otimes H_1(B)) \oplus (H_1(A) \otimes H_0(B)) \tag{7}$$

Despite the "sketchy intuition," it turns out that the equality of Equation (7) is a true fact!

3 Parameters of the Hypergraph Product Code

As we saw in the last section, the overall logical operators of a hypergraph product code are a combination of the logical operators belonging to each of the individual base codes. Let's now use this fact to prove the parameters of the hypergraph product code.

We are going to derive our cell complexes from two codes. Namely, we define the codes and transpose codes:

- $C_A = [n_A, k_A, d_A]$ with locality ℓ
- $C_A^T = [n_A^T, k_A^T, d_A^T]$ with locality ℓ
- $C_B = [n_B, k_B, d_B]$ with locality ℓ
- $C_B^T = [n_B^T, k_B^T, d_B^T]$ with locality ℓ.

Given codes with these parameters, the *number of physical qubits* for the hypergraph product code is

$$n_A \cdot n_B^T + n_A^T \cdot n_B \tag{8}$$

and the *number of logical qubits* is

$$k_A \cdot k_B^T + k_A^T \cdot k_B. \tag{9}$$

Note that the number of logical qubits is computed by looking at the logical operators, which come from the first homology group. To elaborate, the number of independent logical operators is the dimension of first homology group, which is governed by Equation (7). The "$\oplus$" indicates that we sum the dimensions of each of the components. Furthermore, recall that $\dim(H_0(A)) = \dim(C_A^T)$, $\dim(H_1(B)) = \dim(C_B)$, and so forth, resulting in Equation (9).

Finally, the *distance* of the hypergraph product code is the weight of the smallest non-trivial logical operator. The smallest logical operator will come from one of the two sets in the first homology group, meaning the distance is lower-bounded by

$$\min\{d_A, d_B, d_A^T, d_B^T\}. \tag{10}$$

3.1 Picking Parameters

Now let's see how we can pick parameters for our base codes C_A and C_B to construct a good hypergraph product code.

It turns out that we will only need to pick one overall code, which we will refer to as C. Let $C = [n, k, d]$ be a good[1] LDPC code, meaning $k, d = \Omega(n)$. We will also pick our parity checks to be linearly independent, i.e. $P_1, ..., P_\ell$ where $\ell = n - k$. Picking these checks imposes that the transpose code has parameters $C^T = [\ell, 0, \infty]$.

We are going to use this code to define the classical codes $C_A = C$ and $C_B = C^T$. Using these codes in the hypergraph product code construction obtains a quantum code with parameters

$$[[n^2 + \ell^2, k^2, d]] = \left[\left[\geq n^2, \Omega(n^2), \Omega(n)\right]\right] = \left[\left[\geq n', \Omega(n'), \Omega(\sqrt{n'})\right]\right]. \tag{11}$$

Recall that the parameters of the toric code were $[[n, O(1), \sqrt{n}]]$. More logical qubits could be added to the toric code by inserting more handles into the torus, but this would simultaneously decrease the distance. However, with this hypergraph product construction, we are able to maintain a distance of order $\sqrt{n}$, while increasing the number of logical qubits to almost optimal. Therefore, the hypergraph product code beats the toric code!

3.2 Generalizing to Higher-Dimensional Chain Complexes

It turns out that the hypergraph product code is not constrained to 1-dimensional chain complexes, but can be generalized to higher dimensional chain complexes. Concretely, let's consider the d-dimensional chain complexes $(A_0, ..., A_d)$ and $(B_0, ..., B_d)$, with hypergraph product depicted in Figure 6.

[1]Here, "good" means that all the checks are local.

$$\begin{tikzcd}
\textbf{2d-dim :} & & & & A_d \otimes B_d \arrow[dll, bend right, "\partial_A \otimes I"'] \arrow[drr, bend left, "I \otimes \partial_B"] & & & \\
\textbf{(2d-1)-dim :} & & A_{d-1} \otimes B_d \arrow[dl, bend right, "\partial_A \otimes I"'] \arrow[dr, bend left, "I \otimes \partial_B"] & & & & A_d \otimes B_{d-1} \arrow[dl, bend right, "\partial_A \otimes I"'] \arrow[dr, bend left, "I \otimes \partial_B"] & \\
\textbf{(2d-2)-dim :} & A_{d-2} \otimes B_d & & A_{d-1} \otimes B_{d-1} & & A_{d-1} \otimes B_{d-1} & & A_d \otimes B_{d-2}
\end{tikzcd}$$

Figure 6: The hypergraph product of d-dimensional chain complexes.

Any three contiguous layers of the resulting structure can be used to obtain a CSS code. To understand the logical operators of this code, we use the standard *Kunneth Theorem*, which states that:

$$H_i \simeq (H_i(A) \otimes H_0(B)) \oplus (H_{i-1}(A) \otimes H_1(B)) \oplus ... \oplus (H_0(A) \otimes H_0(B)) \tag{12}$$

3.3 Adding Twists (Improving Distance)

We will now discuss a generic way of improving the distance of these hypergraph product codes, namely by adding "twists." We will discuss this specifically in the context of the toric code. Given an $L = 5$ toric code, we can "twist" one of the layers by changing the connections between some of the faces ever so slightly, as depicted in Figure 7.

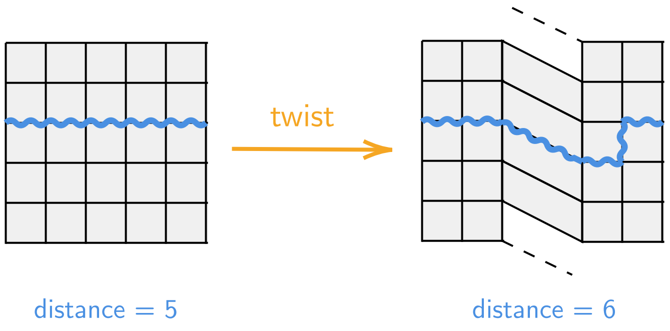

Figure 7: Adding a twist to the toric code improves its distance.

When we twist our code, how does this affect the logical operators? In the toric code, our logical operators correspond to non-contractible loops. When we twist the toric code, the length of this loop increases by 1, thereby increasing the distance by 1. In fact, adding more twists can be used to increase the code distance even more. In recent years, people have been trying to make hypergraph product codes with increasingly clever twists, which ultimately led to the recent breakthrough in good QLDPC codes. We will see these codes next week.

4 Papers

Let's conclude by discussing the line of papers that got us to where we are:

1. [FML02] : The first paper that really improved upon the toric code. It achieved a code of dimension 2 and distance $\sqrt{n}\log^{\frac{1}{4}} n$. This was done before the hypergraph product work, which was why their dimension stayed the same as that of the toric code. However, they introduced some really crazy twists.

2. [TZ13]: This was the hypergraph product code we discussed in lecture today. They achieved a dimension of n and distance of $\sqrt{n}$.

3. [EKZ22] : This work developed a hypergraph product code with a slightly worse dimension of $n/\log n$, but an improved distance of $\sqrt{n}\log n$

4. [KT21] : They were able to show that for any constant k you could sacrifice a factor in the dimension, i.e. $n/\log^k n$, to gain a factor in distance, i.e. $\sqrt{n}\log^k n$.

5. [HHO21] : They managed to finally break the $\sqrt{n}$ distance barrier, achieving a code of dimension $n^{3/5}/\text{poly}\log n$ and distance $n^{3/5}/\text{poly}\log n$. This was a breakthrough paper, achieving a notably better distance than the toric code. However, their methods were still a combination of product codes and twists.

6. [PK21] : They achieved a code with dimension $\log n$ and distance $n/\log n$, again leveraging hypergraph product codes with twists. Furthermore, for any $\alpha \in [0,1]$, they gave a framework for producing codes of dimension $n^{1-\alpha}/\log n$ and distance $n^{1-\alpha/2}/\log n$.

7. [PK22] : They were able to achieve a code of dimension n and distance n, via a generalization of twisted product codes known as homological product codes.

References

[EKZ22] Shai Evra, Tali Kaufman, and Gilles Zémor. Decodable Quantum LDPC Codes beyond the $\sqrt{n}$ Distance Barrier Using High-Dimensional Expanders. *SIAM Journal on Computing*, (0):FOCS20–276, 2022. 3

[FML02] Michael H Freedman, David A Meyer, and Feng Luo. Z2-systolic freedom and quantum codes. In *Mathematics of quantum computation*, pages 303–338. Chapman and Hall/CRC, 2002. 1

[HHO21] Matthew B Hastings, Jeongwan Haah, and Ryan O'Donnell. Fiber bundle codes: breaking the $n^{1/2}\log(n)$ barrier for quantum LDPC codes. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1276–1288, 2021. 5

[KT21] Tali Kaufman and Ran J Tessler. New cosystolic expanders from tensors imply explicit Quantum LDPC codes with Ω (n log kn) distance. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1317–1329, 2021. 4

[PK21] Pavel Panteleev and Gleb Kalachev. Quantum LDPC codes with almost linear minimum distance. *IEEE Transactions on Information Theory*, 68(1):213–229, 2021. 6

[PK22] Pavel Panteleev and Gleb Kalachev. Asymptotically good quantum and locally testable classical ldpc codes. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 375–388, 2022. 7

[TZ13] Jean-Pierre Tillich and Gilles Zémor. Quantum LDPC codes with positive rate and minimum distance proportional to the square root of the blocklength. *IEEE Transactions on Information Theory*, 60(2):1193–1202, 2013. 1, 2