國立臺灣大學理學院數學研究所

碩士論文

Graduate Institute of Mathematics

College of Science

National Taiwan University

Master Thesis

拓樸量子修正碼

Topological Quantum Code

徐滘玢

Chin-Bin Hsu

指導教授: 謝銘倫 教授

Advisor: Ming-Lun Hsieh, Professor

中華民國 111 年 07 月

July 2022

doi:10.6342/NTU202201694

Topological Quantum Code

Chin-Bin Hsu

Advisor: Ming-Lun Hsieh

Graduate Institute of Mathematics

National Taiwan University

Taipei, Taiwan

July 2022

doi:10.6342/NTU202201694

國立臺灣大學碩士學位論文
口試委員會審定書

拓樸量子修正碼

Topological Quantum Code

本論文係徐滑玢君（R09221002）在國立臺灣大學數學系完成之碩士學位論文，於民國 111 年 4 月 15 日承下列考試委員審查通過及口試及格，特此證明

口試委員： [illegible] （簽名）

（指導教授）

夏俊雄

賴青瑞

系主任、所長 ＿＿＿＿＿＿＿＿ （簽名）

（是否須簽章依各院系所規定）

Abstract

Quantum computer is known of being more efficient in computation than classical computer, and hence plays a crucial role in computer science. However, due to the instability of physical particles, error correction becomes an important issue when running quantum algorithms. In contrast to classical error correction, quantum error correction has its difficulty because of the freedom of quantum bits. In 1996, Gottesman invented the stabilizer formalism , which established a purely mathematical construction for quantum codes. In 1997, Kitaev first used topological ideas to construct the toric code, which then generalized to the theory of topological quantum codes. In this article we will introduce various constructions of quantum error correction codes and emphasize mainly on the mathematical idea.

Keywords: Quantum Mechanics, Error correction, Topology, Graph theory, Homological Algebra

doi:10.6342/NTU202201694

Contents

doi:10.6342/NTU202201694

doi:10.6342/NTU202201694

List of Figures

doi:10.6342/NTU202201694

Chapter 1

Introduction

Throughout this article, we will denote the Pauli matrices by X, Y, Z , and use a subscript to indicate which qubit it is acting on. For example, X_7 means X acting on the 7th qubit, and leaving other qubits unaffected. For tensor products of Pauli matrices, we will often omit the symbol $\otimes$ such as $X_1Z_2 := X_1 \otimes Z_2$. Firstly, let us briefly recall the basic idea of classical and quantum error correction codes and set up some notations.

1.1 Classical linear Code

A classical linear $[n, k]$ code C is a k-dimensional subspace of $\mathbb{Z}_2^n$. The parameters n, k mean that we use n bits to protect a k-bit message. A vector in C is called *codeword*, which presents an encoded message. The relation between a codeword $w \in C$ and an input message $s \in \mathbb{Z}_2^k$ is given by

$$w = G(C) \cdot s$$

where $G(C)$ is a $n \times k$ matrix with entries in $\mathbb{Z}_2$, called the *generating matrix* of C. Put another way, the columns of $G(C)$ form a basis for C.

Another frequently used notion is the *parity-check matrix*, denoted by $H(C)$. It is a $(n-k) \times n$ matrix with $\ker(H(C)) = C$. For simplicity, we

doi:10.6342/NTU202201694

will write G, H instead of $G(C), H(C)$ when there is no ambiguity.

Example. (Classical repetition code) A simple example for classical linear code is the *repetition code.* The code encodes a single bit $x \in \{0,1\}$ by

$$x \mapsto xxx \in \mathbb{Z}_2^3$$

This code has $n = 3$ and $k = 1$. A generating matrix is $G = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ and a parity-check matrix is $H = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$. The two rows of H each indicates a *parity-check*: the first row says that the 1st and 2nd entries of the code has to be equal, and the second row says that the 2nd and 3rd entries of the code has to be equal.

Note that for a given code C, its generating matrix and parity-check matrix are not unique. For example, adding a row of H to another one still yields a parity-check matrix. Moreover, sometimes we also append some row vectors to H which preserves its null space. For example, the matrix

$$\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

can also be a parity-check matrix for the repetition code. Thus in the following, we adapt this generalized definition for parity-check matrix:

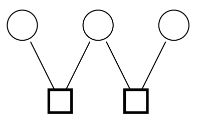
Definition. Let C be an $[n,k]$ code. A *parity-check matrix* for C, $H = H(C)$, is an $r \times n$ matrix with $\ker(H) = C$, where $r \geq n - k$. If $r = n - k$, we say it is a *full-rank parity-check matrix.*

Another way to express a classical linear code is by using the *Tanner graph.*

doi:10.6342/NTU202201694

Definition. Let C be an $[n,k]$ linear code with a parity-check matrix $H = H(C)$ being chosen. The *Tanner graph* of C, denoted by $T(C) = (V,P)$, is a bipartite graph $V \times P$, where $|V| = n$, $|P| = r$, and for $v_i \in V$, $p_j \in P$, there is an edge connecting $v_i p_j$ if and only if the (j,i)-entry of H is 1.

That is to say, each vertex in V represents a bit, and each vertex in P represents a parity-check. There is an edge between a check and a bit when that bit participates in the parity-check. We often use round nodes to represent V and square nodes to represent P. For example, for the repetition code above with $H = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$, its Tanner graph is



Conversely, given a bipartite graph $G = (V,P)$, where V, P are two partitions of G, we can regard it as a Tanner graph of some parity-check matrix.

Lastly, we introduce the concept of coding distance.

Definition. For a codeword $x = (x_1, x_2, ..., x_n) \in \mathbb{Z}_2^n$, its *(Hamming-)weight* $\mathrm{wt}(x)$ is defined to be the number of nonzero bits in x. That is,

$$\mathrm{wt}(x) := |\{j | x_j \neq 0\}|.$$

For a linear code C, the *distance* of C is given by

$$d := \min\{\mathrm{wt}(x) | x \in C, x \neq 0\}.$$

Under this setting, we often say C is a $[n,k,d]$ code. For the repetition code, the only non-trivial codeword is 111, so its distance $d = 3$ and it is a [3,1,3] code.

One checks easily that wt is a metric on $\mathbb{Z}_2^n$. Hence by triangle inequality,

doi:10.6342/NTU202201694

we see that a linear code with distance $d \geq 2t+1$ can correct errors up to t bits.

1.2 Quantum code

The main difference between classical and quantum error correction is that a classical bit can only be 0 or 1, while a qubit is any unit vector in $\mathbb{C}^2$. As a result, classical error correction only needs to deal with bit-flip error, but quantum error correction needs to handle various types of errors. Fortunately, we know that evolution of qubits are unitary, and the unitary group $U(2)$ is spanned by the Pauli matrices, I, X, Y, Z. Since $Y = iXZ$, it turns out that we just need to focus on X-errors (bit-flip) and Z-errors (phase-flip).

Example. (Quantum repetition code) Let S_n denote the set of all n-qubit states. If we mimic the classical reputation code, using

$$\begin{aligned} S_1 &\to S_3 \subseteq \mathbb{C}^8 \\ |0\rangle &\mapsto |0_L\rangle := |000\rangle \\ |1\rangle &\mapsto |1_L\rangle := |111\rangle \end{aligned}$$

to encode a single qubit, it does protects the qubits from X-error. Instead of parity-check, we use *Z-check* to detect the error, that is, we measure our qubits by the operators (called syndrome) Z_1Z_2, Z_2Z_3, which does not collapse the states and also does the job of parity-check due to the properties of their eigenvalues.

Just as classical codes, if we use n qubits to encode k qubits, we call it an $[[n,k]]$ quantum code. More precisely, an $[[n,k]]$ quantum code is a 2^k-dimensional subspace of $\mathbb{C}^{2^n}$. The encoded qubits are called *logical qubits*, just as $|0_L\rangle$ and $|1_L\rangle$ in the preceding example.

doi:10.6342/NTU202201694

The repetition code can only correct X-error. Perhaps the most well-known code on 1 qubit that can correct both Z and X-error is the Shor code:

Example. (Shor code) The following $[[9,1]]$ code is called the *Shor code*:

$$|0\rangle \mapsto |0_L\rangle := \frac{1}{\sqrt{8}}(|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle) \otimes (|000\rangle + |111\rangle)$$
$$|1\rangle \mapsto |1_L\rangle := \frac{1}{\sqrt{8}}(|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle) \otimes (|000\rangle - |111\rangle)$$

The error syndromes for this code are

$$Z_1Z_2, Z_2Z_3, Z_4Z_5, Z_5Z_6, Z_7Z_8, Z_8Z_9$$
$$X_1X_2X_3X_4X_5X_6, X_4X_5X_6X_7X_8X_9$$

Operators in the first line can detect single qubit X-error, and those in the second line can detect single qubit Z-error. Thus this is a code that can protect a single qubit from any possible errors. However, it uses 9 qubits to protect a single qubit, which is somewhat wasting. We will see in short that there is a $[[5,1]]$ code does this job and turns out to be optimal.

The coding distance of a quantum code will be defined in the next chapter, after we have the stabilizer formalism.

doi:10.6342/NTU202201694

Chapter 2

Stabilizer Formalism

In this chapter we introduce the stabilizer formalism of a quantum codes. Most results are from chapter 10.5 of [1].

2.1 Introduction

Again we use the repetition code to explain the idea. The logical qubits for this code are

$$|0\rangle \mapsto |0_L\rangle := |000\rangle\,;$$

$$|1\rangle \mapsto |1_L\rangle := |111\rangle\,.$$

The error detection uses the operators Z_1Z_2, Z_2Z_3. These operators have the property that they fix the logical qubits, since we don't want to collapse the code when correcting error. This motivates us to use group actions to describe the relation between syndromes and the logical qubits. The group we use here is the Pauli group, which contains all the Pauli matrices on n-qubits.

Definition. The *Pauli group on 1 qubit* P_1 is the subgroup of $U(2)$ generated

doi:10.6342/NTU202201694

by X, Y, Z:

$$P_1 := \{\pm I, \pm iI, \pm X, \pm iX, \pm Y, \pm iY, \pm Z, \pm iZ\}.$$

For a positive integer n, the *Pauli group on n qubits* P_n is the subgroup of $U(2^n)$ whose elements are the tensor products of n elements in P_1.

2.2 Action of Pauli group

We first recall some terminologies from group theory.

For a subgroup $H \subseteq P_n$, the *normalizer* of H is the set $N(H) := \{g \in P_n \big| g^{-1}Hg = H\}$, and the *centralizer* of H is the set $Z(H) := \{g \in P_n \big| g^{-1}hg = h \text{ for all } h \in H\}$. It is straightforward to see that each of these set is a subgroup of P_n. Since P_n is finitely generated, so are the subgroups of P_n. A set of elements $\{g_1, g_2, ..., g_k\}$ of P_n is called *independent* the group they generate $\langle g_1, g_2, ..., g_k\rangle$ cannot be generated by any proper subset of $\{g_1, g_2, ..., g_k\}$.

Now the group P_n acts on the n-qubit state space S_n simply by matrix multiplication. For a set A of n-qubit states, the *stabilizer* of A is the set $\text{Stab}_A := \{g \in P_n \big| P\left|\psi\right\rangle = \left|\psi\right\rangle \text{ for all } \left|\psi\right\rangle \in A\}$.

Thus the error syndromes of a code must be in the stabilizer of the subspace spanned by all states in the code space. In the quantum repetition code, we see that given the logical qubits, we can associate the code to its stabilizer, namely the error syndrome. Now we shall ask the converse construction: given a subset of P_n, when does it stabilizes any n-qubit states and how can it define quantum codes? This is solved in the next chapter.

doi:10.6342/NTU202201694

2.3 Stabilizer generators

In the quantum repetition code, the code space is $\operatorname{span}\{|000\rangle, |111\rangle\}$. One can check directly that the stabilizer of this subspace is generated by Z_1Z_2 and Z_2Z_3. That is to say, these two operators are the *stabilizer generators* of the repetition code. The goal of this chapter is to give an intrinsic property characterizing the stabilizer generators, and from where we can define codes in terms of the stabilizers.

First, for a single state $|\psi\rangle$, the stabilizer of $|\psi\rangle$ must be a commuting set, for if g_1, g_2 fixes $|\psi\rangle$, then so does g_1g_2 and g_2g_1. We know that elements in P_n either commute or anti-commute. If g_1, g_2 anti-commute, then $|\psi\rangle = g_2g_1|\psi\rangle = -g_1g_2|\psi\rangle = -|\psi\rangle$, which is a contradiction.

Also, the element $-I$ fixes no states. Thus any stabilizer group of a set of states must not contain $-I$.
It turns out that these two properties characterize the stabilizers.

Theorem 1 (Characterization of stabilizer generators)**.** *A subgroup $S \subseteq P_n$ is the stabilizer of a nontrivial subspace $C(S) \subseteq \mathbb{C}^{2^n}$ if and only if all elements in S commute with each other and $-I \notin S$. In this case, the space $C(S)$ is 2^k-dimensional if S is generated by $n-k$ independent elements $g_1, g_2, ..., g_{n-k}$.*

Remark. In general groups, the number of independent elements generating a subgroup is not unique. For example, in the group $\mathbb{Z}$, both $\{1\}$ and $\{2, 3\}$ are independent set generating $\mathbb{Z}$. However, Theorem 1 implies that any set of independent generators of a subgroup in P_n satisfying the requirement necessarily have same numbers of elements.

To prove this result, we introduce the notion of *check matrix*. Each element in P_n takes the form $c \cdot M_1 \otimes M_2 \otimes ... \otimes M_n$, where $c = \pm 1, \pm i$, and each M_r is one of X, Y or Z. By writing $Y = iXZ$ and modifying the scalar c, we may rewrite the expression as $c \cdot (X^{i_1}Z^{j_1}) \otimes (X^{i_2}Z^{j_2}) \otimes ... \otimes (X^{i_n}Z^{j_n})$, where $i_1, i_2, ..., i_n, j_1, j_2, ..., j_n \in F_2$.

doi:10.6342/NTU202201694

Define a map $f : P_n \to \mathbb{Z}_2^{2n}$ by $c \cdot (X^{i_1} Z^{j_1}) \otimes (X^{i_2} Z^{j_2}) \otimes ... \otimes (X^{i_n} Z^{j_n}) \mapsto (i_1, i_2, ..., i_n, j_1, j_2, ..., j_n)$. If we regard $\mathbb{Z}_2^{2n}$ as an additive group, then the map f is a group homomorphism. In the following, we view each element in $\mathbb{Z}_2^{2n}$ a row vector with entries in $\mathbb{Z}_2$.

Lemma 2.1. For $g_1, g_2 \in P_n$, $f(g_1)Rf(g_2)^t = 0$ or 1, with the outcome 0 occurs if and only if g_1 and g_2 commute, where $R = \begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}$ is a $2n \times 2n$ matrix.

Proof. Since g_1, g_2 acts individually on each component of the tensor product $\mathbb{C}^{2^n}$, it suffices to prove it for $n = 1$. Write $g_1 = c_1 X^i Z^j$, $g_2 = c_2 X^{i'} Z^{j'}$. It suffices to check all cases $(0, 0), (1, 0), (0, 1)$ and $(1, 1)$ for $(i, j), (i', j')$, which we omit here. $\square$

Definition. For the set $\{g_1, g_2, ..., g_{n-k}\}$, we define its *check matrix* to be the $(n - k) \times 2n$ matrix whose i-th row is $f(g_i)$.

That is to say, the i-th row of the check matrix represents the i-th check g_i, by recording whether it acts as X or Z on each qubits. For example, the repetition code has check matrix

$$\left[\begin{array}{ccc|ccc} 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right].$$

Lemma 2.2. Suppose $-I \notin \langle g_1, g_2, ..., g_{n-k} \rangle$. Then the elements $\{g_1, g_2, ..., g_{n-k}\}$ are independent if and only if the rows of its check matrix are linearly independent over $\mathbb{Z}_2$.

Proof. Since f is a group homomorphism, the rows of the check matrix are linearly dependent, say $\sum_{a_i} f(g_i) = 0$, if and only if the product $\prod_i g_i^{a_i} \in \ker(f)$. But $\ker(f) = \{I, -I, iI, -iI\}$, and since $-I \notin S$, nor do iI and $-iI$. Hence the above condition is equivalent to $\prod_i g_i^{a_i} = I$, which says $\{g_1, g_2, ..., g_k\}$ is not independent. $\square$

doi:10.6342/NTU202201694

Lemma 2.3. Suppose $S = \langle g_1, g_2, ..., g_{n-k} \rangle$ is generated by $n-k$ independent elements and $-I \notin S$. Then for each i, there is some $p_i \in P_n$ such that g_i commutes with p_i, and g_j anti-commutes with p_i if $j \neq i$.

Proof. By Lemma 2.2, the rows of H are linearly independent, hence $\operatorname{rank}(H) = n-k$, and thus for each $i = 1, 2, ..., n-k$, there is a column vector $x_i \in F_2^{2^n}$ such that $HRx_i = e_i$. (Here e_i is the i-th vector in the standard basis of $\mathbb{Z}_2^{n-k}$.) Pick $p_i \in P_n$ so that $f(p_i) = x_i^t$, then $HRx_i = e_i$ implies $f(g_j)Rf(p_i) = \delta_{ij}$, and by Lemma 2.1 this means g_i commutes with p_i, while g_j anti-commutes with p_i if $j \neq i$. □

Proof of Theorem 1. We have already showed that if S is a stabilizer of a nontrivial subspace, then its generators commute with each other and $-I \notin S$. For the converse, suppose that $S = \langle g_1, g_2, ..., g_{n-k} \rangle$ satisfies these two conditions. Let H be the check matrix of S. For each g_i, the operator $(I+g_i)/2$ (resp. $(I-g_i)/2$) is the projection onto the $+1$(resp.-1) eigenspace of g_i. Given a bit string $\mathbf{x} = (x_1, x_2, ..., x_{n-k})$ in $\mathbb{Z}_2^{n-k}$, consider

$$P^{\mathbf{x}} := \prod_{i=1}^{n-k} \frac{I + (-1)^{x_i} g_i}{2}.$$

Then since the g_i's commute, $P^{\mathbf{0}}$ is the projection onto the interchapter of all $+1$ eigenspaces of the g_i's, which is exactly the space $C(S)$ that S fixes. For any other $\mathbf{x}$, we choose $p_{\mathbf{x}} = \prod_i x_i p_i$, where p_i is given in Lemma 2.3, then $p_{\mathbf{x}} P^{\mathbf{0}} p_{\mathbf{x}}^\dagger = P^{\mathbf{x}}$, which shows that all $\operatorname{Im}(P^{\mathbf{x}})$'s have same dimensions. Finally, since the whole space $\mathbb{C}^{2^n}$ decomposes into direct sums of the $+1$ and -1 eigenspaces of the Pauli operators, we see that $\mathbb{C}^{2^n} = \bigoplus_{\mathbf{x} \in \mathbb{Z}_2^{n-k}} \operatorname{Im}(P^{\mathbf{x}})$. By counting the dimensions on both sides we see that $\dim(P^{\mathbf{x}}) = 2^k$ for all $\mathbf{x}$, and in particular we have $\dim C(S) = 2^k$. □

Definition. With S satisfying the requirement of Theorem 1, the subspace $C(S)$ is called a $[[n,k]]$ *stabilizer code.*

doi:10.6342/NTU202201694

As an example, we use the language of stabilizers to describe the repetition code. Let $n = 3$, and consider the two independent commuting elements Z_1Z_2 and Z_2Z_3. By Theorem 1, $S := \langle Z_2Z_3, Z_1Z_3\rangle$ defines a code with dimension $2^{3-2} = 2^1$, hence $k = 1$ and this is a $[3, 1]$ code.

Note that the stabilizer generators are not unique. For example, the group S can also be generated by $\langle Z_2Z_3, Z_1Z_3\rangle$.

2.4 Logical operators

There is still another object we are to describe - the encoded logical operators. In principal, the logical operators are the operators the act as Pauli Z and X operators on the logical qubits. In the quantum repetition code, we may choose the logical X to be $X_1X_2X_3$, and the logical Z to be Z_1. Again, the choices are not unique.

Definition. Given an $[[n, k]]$ code $C(S)$ with stabilizer $S = \langle g_1, g_2, ..., g_{n-k}\rangle$, the *logical operators of* $C(S)$ is a set of $2k$ coset representatives

$$\{\overline{Z_1}, \overline{Z_2}, ..., \overline{Z_k}, \overline{X_1}, \overline{X_2}, ..., \overline{X_k}\}$$

of $Z(S)/S$ with each element represents a different coset and not in S, such that for any $i \neq j$, we have $\overline{Z_iZ_j} = \overline{Z_jZ_i}$, $\overline{X_iX_j} = \overline{X_jX_i}$, $\overline{Z_iX_j} = \overline{X_jZ_i}$ and $\overline{Z_iX_i} = -\overline{X_iZ_i}$.

This definition follows from some physical requirements, such as a logical operator shouldn't break the code space $C(S)$. More precisely, we need a logical operator L for a stabilizer code $S = \langle g_1, g_2, ..., g_{n-k}\rangle$ to satisfy:

1. L must leave the code space $C(S)$ invariant;

2. L must commutes with all stabilizers in S. This is because if L anti-commutes with some $g \in S$, then for any state $|\psi\rangle \in C(S)$ we have

doi:10.6342/NTU202201694

$gL|\psi\rangle = -Lg|\psi\rangle = -L|\psi\rangle$, which implies that $L|\psi\rangle$ is no longer stabilized by g;

3. The logical Z and logical X operators satisfies the same commutation relation as the ordinary Z and X;

4. Since the code space is 2^k dimensional (it encodes a k-qubit system), we need k operators for the logical Z and k operators for the logical X;

5. For any $g \in S$, Lg acts identically as L.

Observation 5 leads to the coset notion in our definition. The following lemmas prove that the logical operators are well-defined.

Lemma 2.4. There are exactly $2k$ cosets of S in $Z(S)$. Equivalently, there are exactly $2k$ independent elements in P_n which commute with all stabilizer generators $\langle g_1, g_2, ..., g_{n-k}\rangle$.

Proof. Suppose $L \in P_n$ commute with all stabilizer generators. By Lemma 2.1 and Lemma 2.2, the requirement for L is equivalent to that $f(L)Rf(g_i)^t = 0$ for all $i = 1, 2, ..., n-k$, and $f(L)$ must be linearly independent to all $f(g_i)$'s. We may regard the former condition as a system of linear equations of $n-k$ equations in $\mathbb{Z}_2^{2n}$. By the theory of linear algebra, we see that the solution space for L is $2n-(n-k)-(n-k) = 2k$-dimensional, as desired. □

Lemma 2.5. The $2k$ coset representatives of S in $Z(S)$ can be chosen so that they satisfy the commuting relation.

Proof. Let the check matrix of $\{g_1, g_2, ..., g_{n-k}\}$ be $G = [G_1|G_2]$, where G_1, G_2 are both $(n-k)\times n$ matrices. Observe that elementary row operations to a check matrix does not change the code, since relabelling the generators and replacing g_i by g_ig_j with $j \neq i$ does no effect to the group that $\{g_1, g_2, ..., g_{n-k}\}$ generates. Also, simultaneously switching two columns of G_1, G_2 corresponds

doi:10.6342/NTU202201694

to relabelling the qubits, and hence does nothing to the code either. Thus we may perform Gauss elimination to G, turning it into the form

$$\begin{array}{r|cc|cc|}
 & r & n-r & r & n-r \\
r & I & A & B & C \\
n-k-r & 0 & 0 & X & Y
\end{array}$$

where $r = \operatorname{rank} G_1$. We further perform Gauss elimination to Y to get

$$\begin{array}{r|ccc|ccc|}
 & r & s & n-r-s & r & s & n-r-s \\
r & I & A_1 & A_2 & B & C_1 & C_2 \\
s & 0 & 0 & 0 & D & I & E \\
n-k-r-s & 0 & 0 & 0 & D' & 0 & 0
\end{array}$$

where $s = \operatorname{rank} Y$. Since the stabilizers must commute with each other, we must have D' to be the empty matrix, namely $s = n-k-r$, otherwise the last $n-k-r-s$ generators do not commute with the first r. Hence we arrive at

$$\begin{array}{r|ccc|ccc|}
 & r & n-k-r & k & r & n-k-r & k \\
r & I & A_1 & A_2 & B & C_1 & C_2 \\
n-k-r & 0 & 0 & 0 & D & I & E
\end{array}.$$

Finally, we use the I in the second row to eliminate C_1, so the check matrix results in the form

$$\begin{array}{r|ccc|ccc|}
 & r & n-k-r & k & r & n-k-r & k \\
r & I & A_1 & A_2 & B & 0 & C \\
n-k-r & 0 & 0 & 0 & D & I & E
\end{array}.$$

This is called the *standard form* of a check matrix. Now we choose the k

doi:10.6342/NTU202201694

logical Z and X operators so that they form the check matrix

$$\begin{array}{c|cccccc} & r & n-k-r & k & r & n-k-r & k \\ k & 0 & 0 & 0 & A_2^t & 0 & I \end{array},$$

and

$$\begin{array}{c|cccccc} & r & n-k-r & k & r & n-k-r & k \\ k & 0 & E^t & I & C^t & 0 & 0 \end{array},$$

respectively. By a direct computation (using Lemma 2.1), we can check that these operators do satisfy all the requirements in the definition of logical operators. □

Remark. The proof above gives an algorithm to determine the logical operators for any stabilizer codes. However, in most cases, an intuitive observation on the codes suffices.

2.5 Distance of a quantum code

As we mentioned earlier, a classical code with distance at least $2t + 1$ corrects an error on up to t bits. We have an analog idea in quantum codes. Recall that our error syndromes are the generators of S. An error that anti-commutes with the stabilizers can thus be detected. Hence the crucial issue is the operator commuting with the stabilizers, which is in $Z(S) - S$. Note that by the discussion above, $Z(S)$ is generated by S together with the logical operators we just defined.

Definition. For $g \in G$, the *weight* of g is the number of tensor components in g not equal to I. Denoted by $\mathrm{wt}(g)$.

For example, $\mathrm{wt}(Z_1Z_2) = 2, \mathrm{wt}(X_2Z_3Z_7) = 3$.

doi:10.6342/NTU202201694

Definition. The distance for an $[[n,k]]$ stabilizer code $C(S)$ is

$$d := \min\{\mathrm{wt}(L) | L \in Z(S) - S\}.$$

Then we say that $C(S)$ is an $[[n,k,d]]$ code. Intuitively, d is the minimal weight of error that will be undetected.

Remark. Some authors use $N(S)$ instead of $Z(S)$. This does not matter since for $-I \notin S$, one can show $N(S) = Z(S)$.

Theorem 2 below implies that a stabilizer code of at least $2t+1$ corrects an error on up to t qubits. Hence a major aim is to construct stabilizer codes with large distance; for example, $d = O(n)$. For years, the development of quantum codes faced a barrier $d = O(\sqrt{n}\,\mathrm{polylog}(n))$, which is broken in late 2020. [2]

Theorem 2 (Error-correction conditions for stabilizer codes)**.** *Let $C(S)$ be a stabilizer code with stabilizer S. Suppose $\{E_j\}$ is a set of operators in P_n such that $E_j^\dagger E_k \notin Z(S) - S$ for all j, k. Then $\{E_j\}$ is a correctable set of errors for $C(S)$.*

Proof. See appendix. □

Corollary. A quantum code with $d \geq 2t+1$ corrects an error up to t qubits.

Proof. Let C be such a code and $\{E_j\}$ be a set of error operations acting on at most t qubits. Then the weight of $E_j^\dagger E_k$ is at most $2t$ for any j, k, so $E_j^\dagger E_k \notin Z(S) - S$ by assumption. Hence Theorem 2 implies $\{E_j\}$ is a correctable set of errors for C. □

2.6 Summary, and the [[5,1,3]] code

Now we summarize the stabilizer formulation for quantum codes. An $[[n,k,d]]$ code is defined by a subgroup S generated by $n-k$ independent element $g_1, g_2, ..., g_{n-k}$ in P_n. The code space is then given by the subspace consisting of states fixed by all stabilizer generators, which is 2^k-dimensional. The set of logical operator is a set of coset representatives of $Z(S)/S$ which are not in the stabilizer group S and satisfy the commuting relations of Z and X. The distance d is the minimum weight of operators in $Z(S)$.

We close this chapter by giving an example of stabilizer code encoding $k = 1$ qubit. Let $n = 5$, and define the stabilizer generators to be

$$\begin{gathered} X_1Z_2Z_3X_4 \\ X_2Z_3Z_4X_5 \\ X_1X_3Z_4Z_5 \\ Z_1X_2X_4Z_5 \end{gathered}$$

We see directly that they form an independent set. Since $5-4 = 1$, this code has $k = 1$. The logical operator can be chosen to be $\overline{Z} = Z_1Z_2Z_3Z_4Z_5$ and $\overline{X} = X_1X_2X_3X_4X_5$. It's easy to see that $\overline{Z}$ and $\overline{X}$ commute with all stabilizer generators and they anti-commute with each other. To find the distance of the code, we are to compute all products of $\overline{Z}, \overline{X}$ with all stabilizers. The stabilizer group has order $2^4 = 16$, so $Z(S)$ has 32 possibilities. We will not list all of them here. By some inspection on those products, it is not hard to see that the distance is 3 (for example, $\text{wt}(X_1X_4Z_5) = 3$.

Also, this code is optimal in the sense that it is the smallest code that protects all 1-qubit errors. The reason is as follows. For a 5-qubit code, there are (at most) $1+3\times 5 = 16$ possibilities of errors. To distinguish them, each case must lead to an orthogonal 2-subspace. Since $2^5 = 32 = 2 \times 16$, this code does the job. For general n, a similar argument shows that we require $2^{n-1} \geq 1 + 3n$. The minimal n satisfying this inequality is $n = 5$.

doi:10.6342/NTU202201694

Chapter 3

Toric Code

In this chapter we introduce the toric code invented by Kitaev in 1997, as a motivation to the topological quantum code. We first give some background settings.

The 2-torus $\mathbb{T}_2$ is a "doughnut-like" 2-dimensional surface, which can be obtained by gluing edged of a square. To be precise, we define $\mathbb{T}_2 := \mathbb{R}^2/\sim$, where $(x,y) \sim (x,y+1) \sim (x+1,y)$ for any $x,y \in \mathbb{R}$.

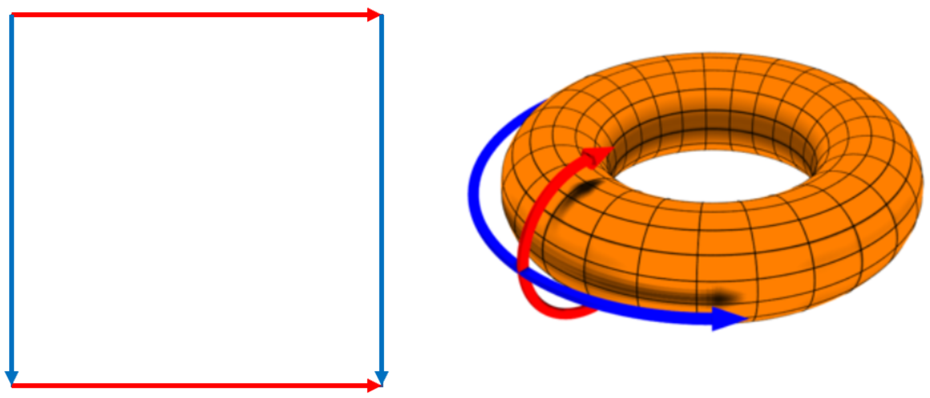


Figure 3.1: Torus obtained from square. The figure on the right is taken from wikipedia.

An $N \times N$ *lattice* on $\mathbb{T}_2$ is a square grid drawn on $\mathbb{T}_2$ with N^2 squares in total. Below is an example for $N = 5$.

doi:10.6342/NTU202201694

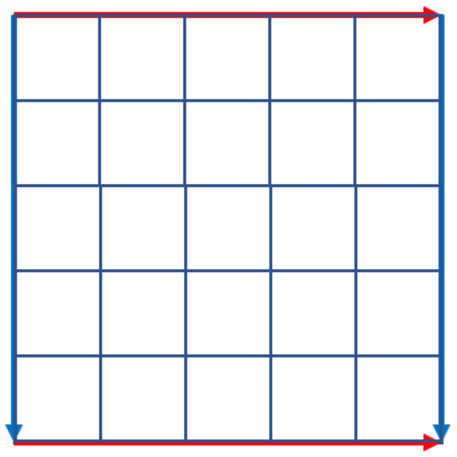


Figure 3.2: A 5×5 lattice on $\mathbb{T}_2$

We may regard the lattice as a graph G with vertex set V_G the collection of all vertices of each square, and with edge set E_G the collection of all edges of each square. Then a single square is a *face* of G. Thus an $N \times N$ lattice has N^2 vertices, $2N^2$ edges, and N^2 faces.

The *dual lattice* of a given lattice is its dual graph, namely obtained by switching the faces and vertices. The dashed line in Figure 3.3 shows the dual lattice of the 5×5 lattice. Since we are working on $\mathbb{T}_2$, which has no boundary, the dual lattice is just a shift of the original lattice.

doi:10.6342/NTU202201694

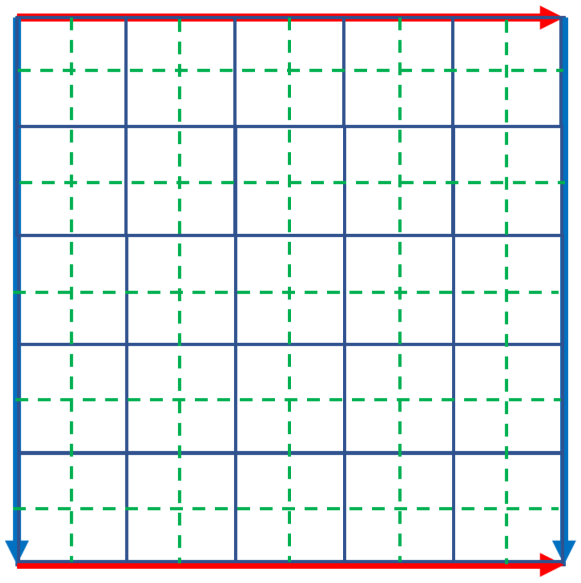


Figure 3.3: The dual lattice of the 5×5 lattice

3.1 Construction of the toric code

From now on, we fix an $N \times N$ lattice on $\mathbb{T}_2$ together with its dual lattice. We assign a qubit to each edge of the lattice. Hence this code has $n = 2N^2$. Next we describe the stabilizer of the code. For each face f of the lattice, define an operator Z_f called a *face operator* by applying Z to the four qubits on the edge of that face. For any vertex v, define an operator X_v called a *vertex operator* by applying X to the four qubits on the edge connecting v.

doi:10.6342/NTU202201694

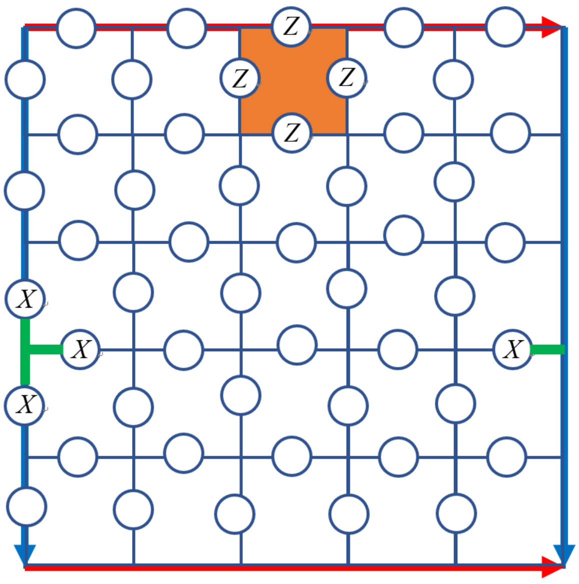


Figure 3.4: An example of face and vertex operator

When we multiply two or more face operators together, it results in applying Z to each qubit on the boundary of some combined faces. Figure 3.5 shows an example of combining 5 face operators.

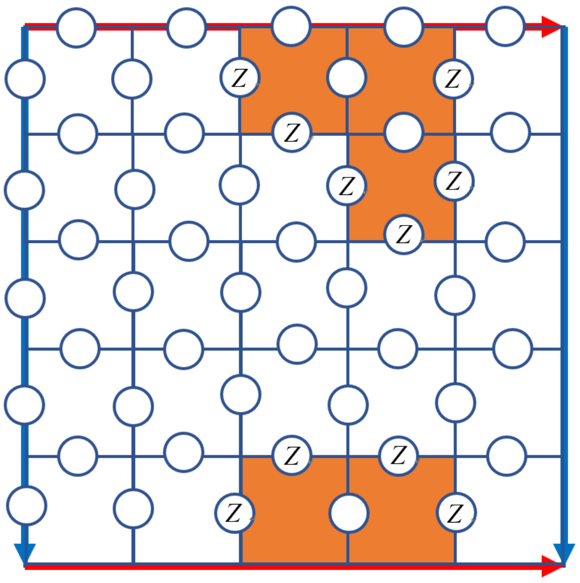


Figure 3.5: An example of a combined face operator

It's immediate that the face operators are not independent: the product

doi:10.6342/NTU202201694

of all face operators equals identity. However, once we remove one of them, they become independent, since any combination of the faces has non-trivial boundary, unless we combine all faces on the lattice. Thus we have $N^2 - 1$ independent face operators. Similarly that by removing one of the vertex operators, we obtain $N^2 - 1$ independent vertex operators. This can also be seen by regarding vertices as faces on the dual lattice.

Now it is also clear that all these $2N^2 - 2$ operators form an independent set and does not generate $-I$. It is also a commuting set since a face operator and a vertex operator shares either zero or two qubits, and we know X_iX_j commutes with Z_iZ_j for all i, j. Thus the code encodes $k = 2N^2 - (2N^2 - 2) = 2$ qubits.

Definition. Any choice of $N^2 - 1$ face operators and the $N^2 - 1$ vertex operators defines a set of stabilizer generator. The associated code is called the *toric code*.

We will not specify which operator we removed, since they only differ in a shift and leads to the same code.

3.2 Logical operators of the toric code

Recall that a logical operator must commute with all the stabilizer generators. First, we try to build one using Z. A Z acting on a single qubit is not a good choice - the vertex operator directly adjacent to that qubit does not commute with it. Similarly, Z acting on a string of qubits does not commute with the vertex operator at the boundary of that string. However, if that string has no boundary, namely if it is a *cycle*, then it does commute with all stabilizer generators. Since we require the operator not being in the stabilizer group, we shall choose cycles that is not contractible. Figure 3.6 shows an example, choosing a cycle in the horizontal direction. It is clear that any two cycles both connecting the left and right side of the torus (i.e.

doi:10.6342/NTU202201694

the two blue arrows) gives equivalent logical operator, since they differ by an element in the stabilizer group.

We choose $\overline{Z_1}$ to be one of the horizontal straight line just as in Figure 3.6. Similarly, we choose $\overline{Z_2}$ to be one of the vertical straight line. $\overline{X_1}, \overline{X_2}$ will be chosen in the same way using X connected by the edges of the dual lattice (see Figure 3.7). Since the code has $k = 2$, all logical operators have been found.

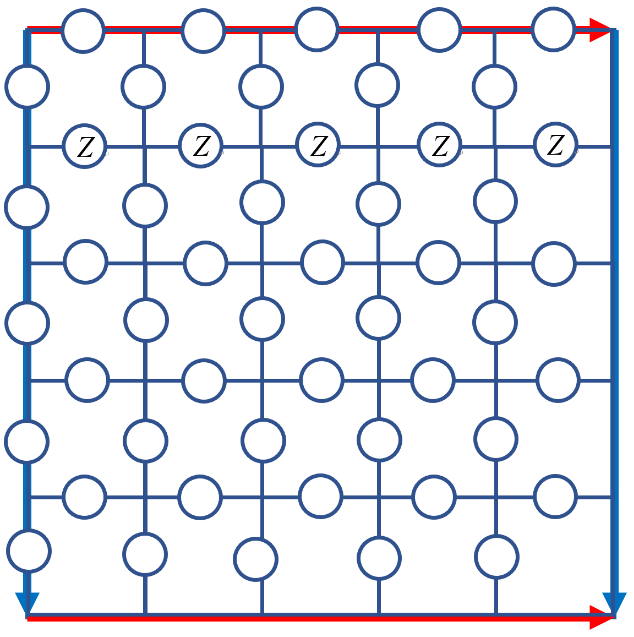


Figure 3.6: A logical Z operator

doi:10.6342/NTU202201694

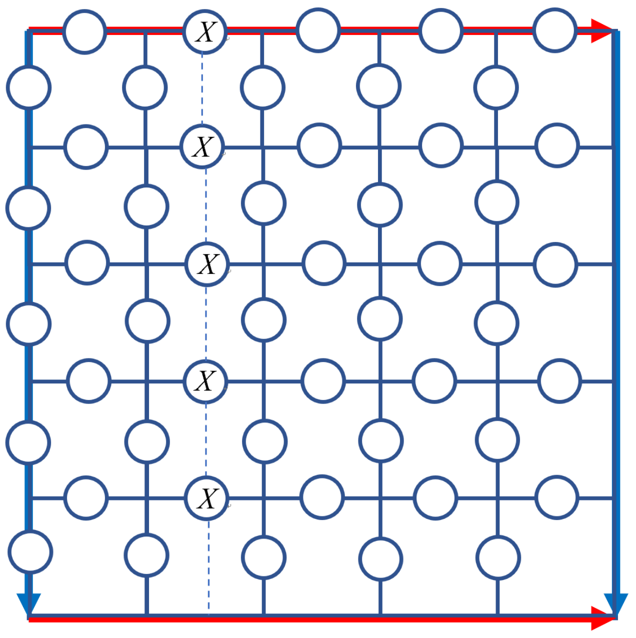


Figure 3.7: A logical X operator

Now the code distance is also clear. The shortest length of a non-contractible cycle on this torus is N, so $d = N$. The toric code is a $[[2N^2, 2, N]]$ code.

Note that the toric code is not just one code, but a *family* of codes: for each value of N we have a quantum code. Analogue to classical error correction theory, we wish to construct **Low-Density-Parity-Check(LDPC)** quantum codes. Also, to decrease the waste of resources, we also want the **coding rate** to be nice. We define these concepts as follows.

Definition. The *coding rate* of an $[[n, k, d]]$ quantum code is defined as k/n. A family $\{\mathcal{Q}_N\}_N$, where $\mathcal{Q}_N$ is an $[[n_N, k_N, d_N]]$ quantum code, is called an *LDPC quantum code* if $n_N \to \infty$ and the following two numbers are both bounded by a constant when $N \to \infty$:

- For each qubit, the number of stabilizer generators acting on it;
- For each stabilizer generator, the number of qubit it acts on.

doi:10.6342/NTU202201694

The toric code is am LDPC code, since each stabilizer generator acts exactly on four qubits, and each qubit is involved in four Z and four X stabilizer generators. The coding rate of the toric code is $1/N^2$, which is not good since it tend to 0 when $N \to \infty$. The ultimate goal of quantum error correction is to construct an LDPC quantum code, with coding rate $k/n \to R > 0$, and with linear distance $d = O(n)$.

3.3 Errors on the toric code

In this chapter we see how error correcting works on the toric code. Consider first a single qubit Z error. In this case, only the two vertex operators adjacent to that qubit will detect it, and hence it is easy to correct.

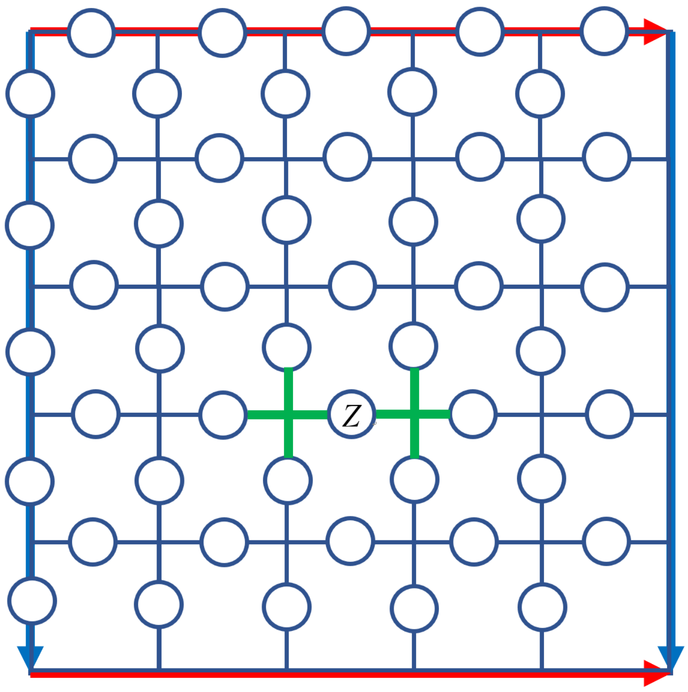


Figure 3.8: A single qubit Z error and the syndromes it affects

For multiple qubits Z error, if the affecting qubits form a connected subgraph, the syndromes detecting the error are those vertices at the endpoint

doi:10.6342/NTU202201694

of that graph. If they are not connected, just apply the argument on each connected components. Figure 3.9 is an example for a Z error on some qubits having 2 connected components. The yellow line shows the connectedness and the green cross is the syndrome detecting the error.

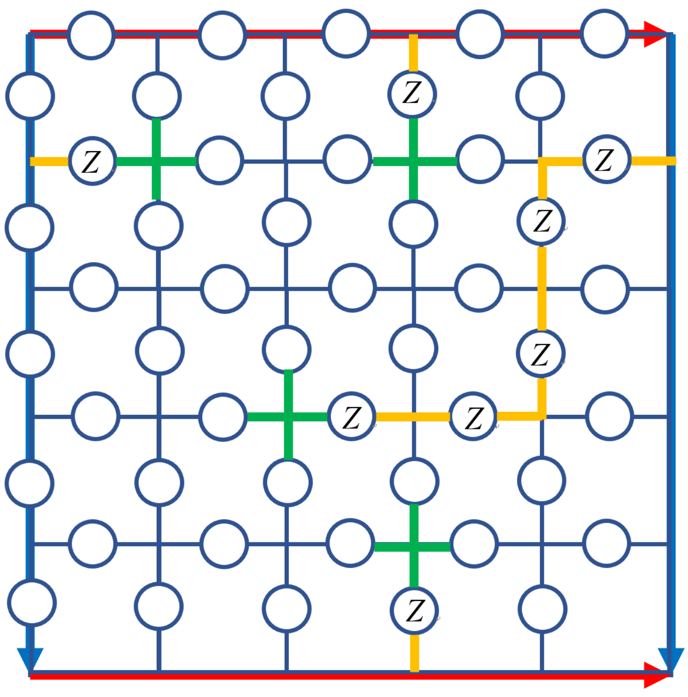


Figure 3.9: A multi-qubit Z error and the syndromes it affects

After detecting an error E, we only know where the endpoints of the error string are but not the string itself. If we choose an arbitrary string with the same endpoints. This may lead to another operator E' instead of E, but they differ by an element L in the centralizer $Z(S)$. More precisely, the string in E and E' together forms a cycle on the torus, and thus $E'E$ is either in the stabilizer, or belongs to the logical operators. To correct the error, we must choose E' to be a suitable one making $E'E$ in the stabilizer. This is a non-trivial work and depends heavily on the error model, and will not be discussed in this article.

Similarly, for X errors we can mimic the process above on the dual lattice.

doi:10.6342/NTU202201694

3.4 Relation to algebraic topology

From all the construction above, we see that this is strongly related to algebraic topology. The face, edge and vertex can be regarded as 2,1,0-simplices. The logical Z operators correspond to cycles, and the logical X operators correspond to cocycle. Equivalence of logical operators is thus translated into homology and cohomology groups. This is the main goal of the next chapter.

doi:10.6342/NTU202201694

Chapter 4

Topological Quantum Codes

Now we try to generalize the method we used in toric code to a general surface, manifold or topological space with a certain way to divide it into polygons (which is lattice in the toric code). While general theory in topology uses $\mathbb{Z}$, $\mathbb{R}$ or $\mathbb{C}$ as coefficients, we will use $\mathbb{Z}_2$ instead.

4.1 Homology and cohomology on 2-surfaces

Let M be a topological space. A *cellulation* of M is a method to divide M into "polygons". (For higher dimensional objects, it is defined by simplicial complexes). We will always assume M admits a cellulation, and M is connected (otherwise we just work on each connected components).

When M is a 2-dimensional surface, a cellulation on it gives the notion of *vertex, edge* and *face*. Denote the set of all vertices, edges and faces by V, E and F,resp. We form the chain groups C_i and the boundary maps ∂_i

doi:10.6342/NTU202201694

by these items. More precisely,

$$C_0 := \bigoplus_{v \in V} \mathbb{Z}_2 v$$

$$C_1 := \bigoplus_{e \in E} \mathbb{Z}_2 e$$

$$C_2 := \bigoplus_{f \in F} \mathbb{Z}_2 f$$

and the map $\partial_i : C_i \to C_{i-1}$ sends an element in C_i to the (formal) sum of its boundaries. An element in C_i is called a i-chain. For example, if f is a square (as in the toric code), then f is a 2-chain and $\partial_2(f)$ is the (formal) sum of its four sides (which is in C_1). Since we are working over $\mathbb{Z}_2$, the minus sign in classical homology theory is not needed. It is direct to see that $\partial_{i-1}\partial_i = 0$.

Now we will use the terminologies in homology theory. $Z_i := \ker \partial_i$ is the set of all *cycles* in C_i. $B_i := \operatorname{Im} \partial_{i+1}$ is the set of all *boundaries* in C_i. The quotient $H_i := Z_i/B_i$ is called the *i-th homology*. In most cases we concern, this space turns out to be finite dimensional, and $\beta_i := \dim_{\mathbb{Z}_2} H_i$ is called the i-th Betti number of X.

Next, we equip all the $\mathbb{Z}_2$ vector spaces with the standard scalar product. Though this is NOT an inner product, it is a non-degenerate bilinear form and so it induces an isomorphism between the space and its dual space. Denote C^i to be the dual space of C_i. The *coboundary map* ∂^i is the dual map of ∂_{i+1}, and similarly we have the set of *cocycles* $Z^i := \ker \partial^i$, the set of *coboundaries* $B^i := \operatorname{Im} \partial^{i-1}$, and the *$i$-th cohomology* $H^i : Z^i/B^i$. Since our spaces are isomorphic to their dual spaces, we have $\dim(H^i) = \dim(H_i) = \beta_i$.

Geometrically, the dual construction is basically the same as in the toric code. The cochains are the chains on the dual graph. From this point of view, the natural pairing $\langle \tilde{c}, c \rangle := \tilde{c}(c)$ on $C^i \times C_i$ is given by counting the number of interchapters of a chain and a cochain, modulo 2. Figure 4.1 is

doi:10.6342/NTU202201694

an example of a pairing on $C^0 \times C_0$, in the case of torus. The three red dots form a 0-chain, the colored squares form a 0-cochain. They meets at exactly 2 points and so the pairing of them equals 0.

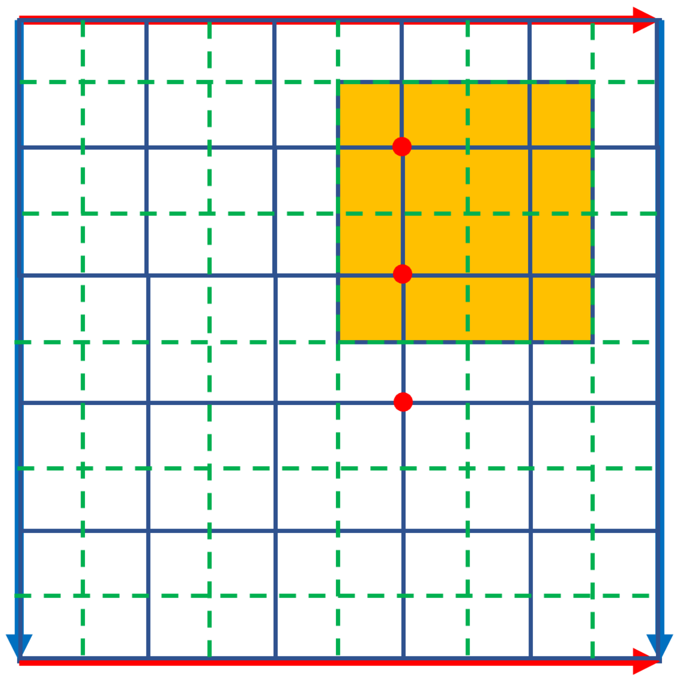


Figure 4.1: The pairing of a 0-chain and a 0-cochain on torus. The three red dots form a 0-chain, the colored squares form a 0-cochain. They meets at exactly 2 points and so the pairing of them equals 0.

In the (co)chain spaces, we have fixed their bases, namely V, E, F and their dual. We will call a basis elements a *(co)cell*. For a given i-(co)chain c, the *support* of c is the set of all i-(co)cells having nonzero coefficient in c, denoted by $\mathrm{supp}(c)$.

4.2 Code on a 2-surface

Having all these settings done, we are now in a position to define out code on the 2-surface X. In fact this is just the same as the toric code. For the given cellulation, we put a qubit on the interchapter of an 1-cell and 1-cocell.

doi:10.6342/NTU202201694

We will just identify the qubit with the (co)cell. For an 1-chain c, define the Z chain operator $Z(c)$ to be $\bigotimes_{x \in \mathrm{supp}(c)} Z_x$. (We use the convention that an empty tensor product equals the identity). That is, $Z(c)$ applies Z to each qubits on c. For an 1-cochain $\tilde{c}$, define the X cochain operator $X(\tilde{c})$ to be $\bigotimes_{\tilde{x} \in \mathrm{supp}(\tilde{c})} X_{\tilde{x}}$. A Z chain operator and an X cochain operator commutes if and only if the chain and the cochain intersects even number of times. In fact we have

$$X(\tilde{c})Z(c) = (-1)^{\langle \tilde{c}, c \rangle} Z(c)X(\tilde{c}).$$

The stabilizer generators of the code will be the face operators and the vertex operators as before. Given a 2-cell f, the *face operator* on f is $Z(\partial_2(f))$. Given a 0-cocell $\tilde{v}$, the *vertex operator* on $\tilde{v}$ is $X(\partial^0(\tilde{v}))$. These operators commute, since $\langle \partial_2(f), \partial^0(\tilde{v}) \rangle = \langle f, \partial^1 \partial^0(\tilde{v}) \rangle = \langle f, 0 \rangle = 0$. Also since they are constructed by tensor products of purely Z or X operators, it is clear that they don't generate $-I$. Again, by removing any single one Z and X operator, these operators become independent and thus defines a stabilizer code. Under this construction, the face (resp. vertex) operators are in 1-1 correspondence in B_1 (resp. B^1).

Next we shall identify the logical operators. Consider first the Z operators. We are to determine the set of all Z operators $Z(c)$ which commute with all stabilizer generators, especially the vertex operators. That is, we are looking for all 1-chains c satisfying $\langle \partial^0(\tilde{v}), c \rangle = 0$ for all 0-cocell $\tilde{v}$. But since $\langle \partial^0(\tilde{v}), c \rangle = \langle \tilde{v}, \partial_1(c) \rangle$, we see that the condition is equivalent to $c \in Z_1$. Similarly, the X operators could be regarded as Z^1.

Recall that two logical operators are equivalent if they differ by a stabilizer, which is identified with B_1 and B^1, we see that the object describing logical operators are exactly H_1 and H^1. To complete the construction, we choose an arbitrary set of representatives $\{c_1, c_2, ..., c_r\}$ for H_1 not lying in B_1, and take their dual vectors to be the representatives for H^1. Then the commutation relation for the logical operators is also satisfied. Moreover,

doi:10.6342/NTU202201694

since we know that the number of logical operators is $2k$, we see that $k = \beta_1$.

Lastly, the code distance is defined as follows. The *Z-distance* of the code d_Z is the minimum weight of all non-trivial Z-operators. Put another way, it is the minimum Hamming weight of elements in non-trivial classes in H_1. The *X-distance* is defined similarly. Then, the code distance d is $\min\{d_Z, d_X\}$.

We now summarize all the constructions:

Definition. Let M be a 2-surface with a fixed cellulation. Let V, E, F be the set of all 0,1,2-cells of the cellulation, respectively. Also we denote their dual by $\tilde{V}, \tilde{E}, \tilde{F}$. The **quantum code constructed on** M is built by:

- Each qubit corresponds to a 1-cell $c \in E$.
- The stabilizer generators are taken to be $\{Z(\partial_2(f)) | f \in F\} \cup \{X(\partial^0(\tilde{v})) | \tilde{v} \in \tilde{V}\}$, with one element removed from each of the two sets. They are in 1-1 correspondence with B_1 and B^1, respectively.
- The logical Z operators are identified with H_1, and the logical X operators are identified with H^1.
- The code distance $d = \min\{d_Z, d_X\}$, where $d_Z := \min \mathrm{wt}\{L \in Z_1 | L \notin B_1\}$, $d_X := \min \mathrm{wt}\{L \in Z^1 | L \notin B^1\}$.

Example. We construct Shor's 9-qubit code [3] on the real projective plane $\mathbb{RP}^2$ as follows.

doi:10.6342/NTU202201694

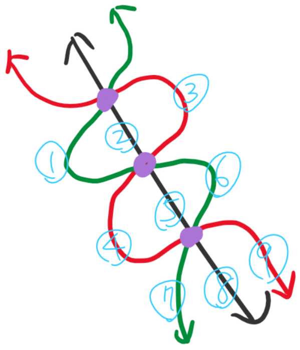


Figure 4.2: Shor code

The graph has 3 vertices, 9 edges and 7 faces. (Recall that on the projective plane, antipodal points are glued together). The 9 qubits are assigned to the 9 edges, as numbered in the graph. From the graph it is easy to recover the stabilizers of Shor code, which are

$$Z_1Z_2, Z_2Z_3, Z_4Z_5, Z_5Z_6, Z_7Z_8, Z_8Z_9,$$
$$X_1X_2X_3X_4X_5X_6, X_4X_5X_6X_7X_8X_9,$$

equalling to those we derived in chapter 1. The first Betti number of $\mathbb{RP}^2$ is $\beta_1 = 1$, so the code encodes 1 qubit and we have 1 logical Z and 1 logical X operator. The logical Z is a 1-cycle, we can take it to be the red line in the graph, i.e. $Z_3Z_6Z_9$. The logical X is a 1-cocycle, we can take it to be $X_1X_2X_3$, which is omitted in the figure. It is also clear that $d_Z = d_X = 3$, so the Shor code has code distance 3.

4.3 Code on higher dimensional objects

The construction above can be easily generalized to higher dimensional topological spaces. Instead of using 0,1,2-chains, for any $q \geq 2$, we may also define code using $q-1, q, q+1$-chains, which do the jobs of vertices, edges and faces, resp. Manifolds and differential geometry turns out to be helpful,

doi:10.6342/NTU202201694

which is beyond the scope of this article. Examples of such codes include Freedman-Meyer-Luo code [4], Haah code [5], etc.

4.4 Code constructed by chain complexes

Finally, we can define the code using purely algebraic approach. The homology of a topological space can be described by a chain complex. Thus a chain complex suffices to define codes.

Definition. A *chain complex* $\mathcal{C}_\cdot$ of $\mathbb{Z}_2$-vector spaces consists of a sequence of $\mathbb{Z}_2$-vector spaces $\{C_i\}_{i\geq 0}$ and a linear transformation $\partial_i : C_i \to C_{i-1}$ called *boundary map* for each i such that $\partial_{i+1}\partial_i = 0$.

$$\mathcal{C}_\cdot : \cdots \xrightarrow{\partial_{n+1}} C_n \xrightarrow{\partial_n} C_{n-1} \xrightarrow{\partial_{n-1}} \cdots \xrightarrow{\partial_1} C_0$$

We will only consider finite dimensional vector spaces. Taking dual of each space yields a *cochain complex*

$$\mathcal{C}^\cdot : \cdots \xleftarrow{\partial^{n+1}} C^n \xleftarrow{\partial^n} C^{n-1} \xleftarrow{\partial^{n-1}} \cdots \xleftarrow{\partial^1} C^0,$$

where $C^i = (C_i)^*$, $\partial^i = (\partial_i)^* = (\partial_i)^t$. Also we have $C^i \cong C_i$ via the non-degenerate bilinear form defined by dot product. The q-th homology and cohomology is given by

$$H_q := \ker(\partial_q)/\operatorname{Im}(\partial_{q+1}); H^q := \ker(\partial^{q+1})/\operatorname{Im}(\partial^q).$$

Given a chain complex, we take any three consecutive terms $C_{q+1} \xrightarrow{\partial_{q+1}} C_q \xrightarrow{\partial_q} C_{q-1}$ to build our code.

Definition. Let $\mathcal{C}_\cdot : C_{q+1} \xrightarrow{\partial_{q+1}} C_q \xrightarrow{\partial_q} C_{q-1}$ be a chain complex of $\mathbb{Z}_2$-vector spaces. The **quantum code constructed on** C is given by:

doi:10.6342/NTU202201694

- Let β_{q+1}, β_q and β_{q-1} be bases for the vector spaces C_{q+1}, C_q and C_{q-1}, respectively. A basis vector in C_k is called a k-cell.
- Each qubit corresponds to a q-cell, namely a vector in β_q.
- The stabilizer generators are taken to be $\bigotimes_{j\in\operatorname{supp}\partial_{q+1}c_{q+1}} Z_j$ for all $c_{q+1} \in \beta_{q+1}$ and $\bigotimes_{j\in\operatorname{supp}\partial^{q-1}c^{q-1}} X_j$ for all $c^{q-1} \in \beta^{q-1}$.
- The logical Z operators are identified with H_1, and the logical X operators are identified with H^1.
- The code distance $d = \min\{d_Z, d_X\}$, where $d_Z := \min \mathrm{wt}\{c \in Z_1 | L \notin B_1\}$, $d_X := \min \mathrm{wt}\{c \in Z^1 | L \notin B^1\}$. Here, the weight of a vector is computed via the chosen bases $\beta_\cdot$ and $\beta^\cdot$.

We close this chapter by introducing the CSS code as an example of chain complexes. CSS code is a type of quantum code built by classical linear code. Here we give the construction using chain complex formulation. Let C_1 be an $[n, k_1]$ classical linear code, C_2 be an $[n, k_2]$ classical linear code with $C_2^\perp \subseteq C_1$. Denote the parity check matrix and the generating matrix by H and G respectively. Consider

$$(\mathbb{Z}_2)^{k_1} \xrightarrow{H(C_1)^t} (\mathbb{Z}_2)^n \xrightarrow{H(C_2)} (\mathbb{Z}_2)^{k_2},$$

which is a chain complex since $H(C_2)H(C_1)^t = [H(C_1)G(C_2^\perp)]^t = 0$ by the assumption $C_2^\perp \subseteq C_1$. The first homology H_1 has dimension $\dim\ker(H(C_2)) - \dim \mathrm{Im}(H(C_1)^t) = \dim C_2 - (n - \dim C_1) = \dim C_1 - \dim C_2^\perp$.

Since a classical code is defined to be the kernel of the parity-check matrix H, we see that in the case of this CSS code, $Z_1 = \ker(H(C_2)) = C_2$ and $B_1 = C_1^\perp$. Similarly, $Z^1 = C_1$ and $B^1 = C_2^\perp$. Thus the distance of CSS code is $d = \min \mathrm{wt}((C_1 \backslash C_2^\perp) \cup (C_2 \backslash C_1^\perp))$.

doi:10.6342/NTU202201694

In terms of check matrices, the CSS code has check matrix

$$\begin{bmatrix} H(C_2) & 0 \\ 0 & H(C_1) \end{bmatrix}.$$

In this point of view, CSS code uses one classical code as Z check, and uses the other as X check.

doi:10.6342/NTU202201694

Chapter 5

Tensor Product Code

Modern development of quantum error correction introduces product constructions. Roughly speaking, a classical code uses one type of check, while quantum code uses two. The idea of products in mathematics allow us to build a quantum code from two classical ones, using one as Z check and the other as X check. In this chapter, we use a homological approach to introduce the product code invented by Jean-Pierre Tillich and Gilles Zémor[6].

Mathematically, given two chain complexes $\mathcal{C}.$ and $\mathcal{D}.$, we can construct their **tensor product complex** $(\mathcal{C} \otimes \mathcal{D}).$ by

$$\begin{aligned}
(\mathcal{C} \otimes \mathcal{D})_k &:= \bigoplus_{i+j=k} (C_i \otimes D_j) \\
\partial_k^{\mathcal{C}\otimes\mathcal{D}} &: \bigoplus_{i+j=k} (C_i \otimes D_j) \to \bigoplus_{i+j=k-1} (C_i \otimes D_j) \\
&\qquad c_i \otimes d_j \mapsto \partial_i^{\mathcal{C}}(c_i) \otimes d_j + c_i \otimes \partial_j^{\mathcal{D}}(d_j) \text{ (extended linearly)}
\end{aligned}$$

One checks by a direct computation that this is indeed a chain complex. Furthermore, the homology of tensor product complex can be derived by **Künneth formula**:

$$H_n((\mathcal{C} \otimes \mathcal{D}).) = \bigoplus_{i+j=n} H_i(\mathcal{C}.) \otimes H_j(\mathcal{D}.).$$

doi:10.6342/NTU202201694

Thus if we take tensor product of two complexes of length 2, it results in a complex with length 3. The idea here is to treat classical codes as complexes of length two. Recall that for a classical $[n, k]$ code C, we may choose an $r \times n$ parity-check matrix $H = H(C)$. This matrix thus induces a map $\mathbb{Z}_2^n \to \mathbb{Z}_2^r$, which is exactly a complex of length two. We are now in a position to construct a quantum code from two classical codes.

Definition. Let C_1, C_2 be two classical codes. The *tensor product code (or product code)*, denoted by $\mathcal{Q}(C_1 \otimes C_2)$, is the tensor product of the two complexes $\mathbb{Z}_2^{n_1} \xrightarrow{H_1} \mathbb{Z}_2^{r_1}$ and $\mathbb{Z}_2^{r_2} \xrightarrow{H_2^t} \mathbb{Z}_2^{n_2}$.

Let us investigate the homology of these complexes. Let H be an $r \times n$ parity-check matrix of an $[n, k]$ code C. Then $\ker H = C$, so $\dim \ker H = k$. By rank-nullity, we have $\operatorname{rank} H = n - k$. Thus the complex $\mathbb{Z}_2^n \xrightarrow{H} \mathbb{Z}_2^r$ has $\dim H_0 = r - (n - k)$ and $\dim H_1 = k$. When transposed, these two number interchanges. Thus by Künneth formula, the complex $\mathcal{Q}(C_1 \otimes C_2)$ defined above has

$$\dim H_1 = k_1 k_2 + (r_1 - (n_1 - k_1))(r_2 - (n_2 - k_2)).$$

If $H(C_1)$ and $H(C_2)$ are taken to be full-ranked, namely $r_i = n_i - k_i$, $i = 1, 2$, then this reduces to $\dim H_1 = k_1 k_2$. From where we see why the transposition is required: taking tensor without the transpose only leads to a complex with trivial homology H_1.

Next, if we write down directly the complex, we can read out the coding dimension:

$$\begin{aligned} C_1 &: \mathbb{Z}_2^{n_1} \xrightarrow{H_1} \mathbb{Z}_2^{r_1} \\ C_2^t &: \mathbb{Z}_2^{r_2} \xrightarrow{H_2^t} \mathbb{Z}_2^{n_2} \\ \mathcal{Q}(C_1 \otimes C_2) &: \mathbb{Z}_2^{n_1} \otimes \mathbb{Z}_2^{r_2} \longrightarrow \mathbb{Z}_2^{n_1} \otimes \mathbb{Z}_2^{n_2} \oplus \mathbb{Z}_2^{r_1} \otimes \mathbb{Z}_2^{r_2} \longrightarrow \mathbb{Z}_2^{r_1} \otimes \mathbb{Z}_2^{n_2} \end{aligned}$$

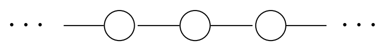
doi:10.6342/NTU202201694

That is, the code $\mathcal{Q}(C_1 \otimes C_2)$ has $n = n_1n_2 + r_1r_2$. Hence it is a $[[n_1n_2 + r_1r_2, k_1k_2 + (r_1-(n_1-k_1))(r_2-(n_2-k_2))]]$ code, or simply $[[n_1n_2+r_1r_2, k_1k_2]]$ when H_1, H_2 are full-ranked.

Example. Let C_1, C_2 both be the $[3, 1]$ repetition code, and choose the parity-check matrix to be

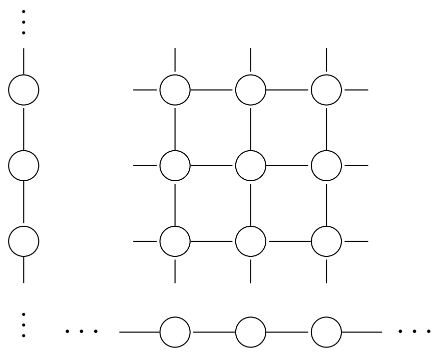
$$H = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

By our constructions above, the tensor product code $\mathcal{Q}(C_1 \otimes C_2)$ is a $[[18, 2]]$ code. If one write down precisely the maps, it could be recovered that this is exactly the toric code with $N = 3$. The detail is complicated, hence omitted. However, we remark here that this construction has a pretty geometry implication. We regard the $[3, 1]$ repetition code with parity-check matrix H as a *cycle graph* C_3, by treating vertices as bits and edges as parity-checks:



The dots on the two sides imply that they are glued together.

Then, the resulting tensor product code is as follows:



This graph gives a strong relation between the torus and the cycle. In fact, this is the original approach used by Jean-Pierre Tillich and Gilles Zémor. We now state this formalism and show that they are equivalent.

doi:10.6342/NTU202201694

Let C_1, C_2 be classical codes with parity-check matrices H_1, H_2. Consider their Tanner graphs $T(C_1) = (V_1, P_1)$ and $T(C_2) = (V_2, P_2)$. Consider the product graph $T(C_1) \times T(C_2)$, which is also a bipartite graph. We define two Tanner graphs as follows:

- T_X is the induced subgraph of $(V_1 \times V_2 \cup P_1 \times P_2, V_1 \times P_2)$ in $T(C_1) \times T(C_2)$.
- T_Z is the induced subgraph of $(V_1 \times V_2 \cup P_1 \times P_2, V_2 \times P_1)$ in $T(C_1) \times T(C_2)$.

As we mentioned before, any bipartite graph (V, P) could be regarded as a Tanner graph, hence defines a classical code. Finally, using the pair (T_X, T_Z) we may construct a CSS code, which is called the **product code**.

The condition $T_X^{\perp} \subseteq T_Z$ to construct CSS code is proved in proposition 4 of [6].

In this construction, the code is built on $V_1 \times V_2 \cup P_1 \times P_2$, showing it has $n = n_1 n_2 + r_1 r_2$. Also by observing the check matrix of CSS code and the checks in the chain complex construction, we see immediately this construction is equivalent to the tensor product construction. Also it is proven that the distance of the product code $d = \min(d(C_1), d(C_2))$ in [6].

doi:10.6342/NTU202201694

Chapter 6

Conclusion

We summarize some codes and their parameters we mentioned above:

Code on $n = 1$ qubits.

1. The Shor code is a [[9,1,3]] code, which could be constructed on $\mathbb{RP}^2$.
2. The Steane code is a [[7,1,3]] code.
3. The optimal code protecting one qubit is the [[5,1,3]] code.

Families of codes for large n.

1. The toric code is an $[[n, 2, O(\sqrt{n})]]$ code.
2. There is an $[[n, 2, \Omega(\log(n)^{1/4} n^{1/2})]]$ code building on hyperbolic surface [4].
3. CSS code is constructed via two classical codes, as well as the tensor product code.

Modern development of quantum error correction uses lots of idea from topology, geometry and also graph theory. For a summary and overview of known codes, see [7]. It is also quite inspiring to see that abstract tools from mathematics turns out to be useful in quantum mechanics.

doi:10.6342/NTU202201694

Chapter 7

Appendix

Proof of the error-correction conditions. We will first prove a more general condition in quantum codes, and the case of stabilizer code follows as a corollary.

Theorem 3 (Error-correction conditions for general quantum codes)**.** *Let $C \subseteq V = \mathbb{C}^{2^n}$ be a quantum code, P be the projection from V to C. Suppose $\mathcal{E}$ is a quantum operation with Kraus operators $\{E_i\}$, then there exists a quantum operation $\mathcal{R}$ correcting $\mathcal{E}$ if and only if $PE_i^\dagger E_j P = \alpha_{ij} P$ for some Hermitian matrix α.*

Proof. ($\Leftarrow$) Suppose $\mathcal{E}$ with Kraus operators $\{E_i\}$ satisfies the condition. Since α is Hermitian, it can be unitarily diagonalized, say $d = u^\dagger \alpha u$. Define $F_k = \sum_i u_{ik} E_i$. Then $\{F_k\}$ is also a set of Kraus operators for $\mathcal{E}$. Now

$$\begin{aligned} PF_k^\dagger F_l P &= \sum_{ij} u_{ki}^\dagger u_{jl} P E_i^\dagger E_j P \\ &= \sum_{ij} u_{ki}^\dagger u_{jl} \alpha_{ij} P \\ &= \sum_{ij} u_{ki}^\dagger \alpha_{ij} u_{jl} P \\ &= d_{kl} P. \end{aligned}$$

doi:10.6342/NTU202201694

That is to say, we simplified the condition by diagonalizing α. By polar decomposition applied to F_kP, we have $F_kP = U_k\sqrt{PF_k^\dagger F_kP} = \sqrt{d_{kk}}U_kP$ for some unitary U_k. Define projections $P_k = U_kPU_k^\dagger = F_kPU_k^\dagger/\sqrt{d_{kk}}$. Then since

$$P_lP_k = P_l^\dagger P_k = \frac{U_lPF_l^\dagger F_kPU_k^\dagger}{\sqrt{d_{ll}d_{kk}}} = 0$$

when $k \neq l$, the images of these projections are orthogonal. These projections along with $I - \sum_k P_k$ defines a measurement, which we choose to be the error syndrome. Error correction will be simply done by applying $U_k^\dagger$. Namely the recovery operation $\mathcal{R}$ of the system is given by Kraus operators $\{U_k^\dagger P_k\}$. A direct computation shows that

$$\begin{aligned} U_k^\dagger P_kF_l\sqrt{\rho} &= U_k^\dagger P_k^\dagger F_lP\sqrt{\rho} \\ &= \frac{U_k^\dagger U_kPF_k^\dagger F_lP\sqrt{\rho}}{\sqrt{d_{kk}}} \\ &= \delta_{kl}\sqrt{d_{kk}}P\sqrt{\rho} \\ &= \delta_{kl}\sqrt{d_{kk}}\sqrt{\rho}, \end{aligned}$$

and so

$$\begin{aligned} \mathcal{R}(\mathcal{E}(\rho)) &= \sum_{kl} U_k^\dagger P_kF_l\rho F_l^\dagger P_kU_k \\ &= \sum_{kl} \delta_{kl}d_{kk}\rho \\ &= C\rho, \end{aligned}$$

where C is a constant. This shows that the operation defined above does correct the error.

Conversely, suppose $\{E_i\}$ is a correctable set of errors, say by the operation $\mathcal{R}$ defined by $\{R_j\}$. Then since $P\rho P$ is in the code space for all ρ, $\mathcal{R}(\mathcal{E}(P\rho P)) = cP\rho P$ for some constant c. Both sides of this equation are linear in ρ, so c

doi:10.6342/NTU202201694

must not depend on ρ. Plugging in the definition yields $\sum_{ij} R_j E_i P\rho P E_i^\dagger R_j^\dagger = cP\rho P$, which shows that the quantum operation defined by Kraus operators $\{R_j E_i\}$ equals the one defined by a single operator $\sqrt{c}P$. The theorem of freedom in Kraus operator thus implies $R_k E_i P = c_{ki} P$ for some $c_{ki} \in \mathbb{C}$. Therefore $PE_i^\dagger R_k^\dagger R_k E_j P = c_{kj}^* c_{kj} P$. Also, since $\mathcal{R}$ is trace preserving, namely $\sum_k R_k^\dagger R_k = I$, showing that $PE_i^\dagger E_j P = \sum_k c_{ki}^* c_{kj} P$ by summing the previous equation over k. Now define $\alpha_{ij} = \sum_k c_{ki}^* c_{ki}$ and we arrive at the desiring result. □

Now, as a special case of this theorem, we prove the error correction condition of stabilizer code (Theorem 2).

Proof. Again we let the stabilizer generators be $g_1, g_2, ..., g_{n-k}$, and P be the projection onto the code space $C(S)$. So $P = \prod_{l=1}^{n-k} \frac{I+g_l}{2}$. By assumption, for any j, k we have $E_j^\dagger E_k \notin Z(S) - S$, so either $E_j^\dagger E_k \in S$ or $E_j^\dagger E_k \in P_n - Z(S)$. For the first case, $PE_j^\dagger E_k P = P$ since $C(S)$ is stabilized by elements in S. For the second case, since $E_j^\dagger E_k$ does not commute with all elements in S, it must anti-commute with some of them, say with g_1. Now $E_j^\dagger E_k P = \frac{I-g_1}{2} E_j^\dagger E_k \prod_{l=2}^{n-k} \frac{I+g_l}{2}$. Note that since $(I+g_1)(I-g_1) = 0$, so $P(I-g_1) = 0$, and thus $PE_j^\dagger E_k P = 0$ in this case. Thus $\{E_j\}$ satisfies the error correction condition given in Theorem 3. □

doi:10.6342/NTU202201694

Bibliography

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th ed. USA: Cambridge University Press, 2011.

[2] R. O. Matthew B. Hastings, Jeongwan Haah, "Fiber Bundle Codes: Breaking the $O(\sqrt{N}\text{polylog}(N))$ Barrier for Quantum LDPC Codes," 2020.

[3] P. Shor, "Scheme for reducing decoherence in quantum computer memory," *PHYSICAL REVIEW A*, vol. 52, no. R2493, 1995.

[4] F. L. Michael H. Freedman, David A. Meyer, "Z2-systolic freedom and quantum codes," *Mathematics of Quantum Computation*, 2002.

[5] J. Haah, "Algebraic Methods for Quantum Codes on Lattices." *Revista Colombiana de Matemáticas*, vol. 50, 2016.

[6] G. Z. Jean-Pierre Tillich, "Quantum LDPC codes with positive rate and minimum distance proportional to $n^{1/2}$," *arXiv:0903.0566*, 2013.

[7] J. N. E. Nikolas P. Breuckmann, "LDPC Quantum Codes," *PRX Quantum 2 (4)*, 2021.

doi:10.6342/NTU202201694