PREPARATION AND SIMULATION FOR GROUND STATES OF TOPOLOGICAL PHASES OF MATTER

by

Penghua Chen

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

Department of Physics and Astronomy

West Lafayette, Indiana

August 2024

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Xingshan Cui, Co-Chair
Department of Mathematics

Dr. Yuli Lyanda-Geller, Co-Chair
Department of Physics and Astronomy

Dr. Ruichao Ma
Department of Physics and Astronomy

Dr. Sabre Kais
Department of Chemistry

Approved by:
Dr. Gabor A. Csathy

TABLE OF CONTENTS

LIST OF FIGURES

1. INTRODUCTION

This chapter contains work from the article entitled "Ribbon operators in the generalized Kitaev quantum double model based on Hopf algebras" written by Bowen Yan, the author, and Shawn X. Cui published on Journal of Physics A [1], and the article entitled "Quantum circuits for toric code and X-cube fracton model" written by the author, Bowen Yan, and Shawn X. Cui published on Quantum [2], and the article entitled "Representing Arbitrary Ground States of Toric Code by Restricted Boltzmann Machine" written by the author, Bowen Yan, and Shawn X. Cui preprinted on arXiv [3].

In an era characterized by escalating technological complexity and an ever-increasing demand for computational power, the semiconductor industry confronts a formidable obstacle. Traditional chip development paradigms, reliant on relentless miniaturization, are increasingly strained by the unyielding physical limits of the atomic scale. At these microscopic dimensions, atoms themselves define the frontier of computational power, subject to the esoteric influences of quantum mechanical effects. Consequently, the quest for next-generation computing paradigms has taken on an unprecedented urgency. Quantum computing, boasting the potential to eclipse the computational capabilities of classical computers, has emerged as a promising contender. The shift toward quantum computing represents more than an option—it is a necessity, offering a vital lifeline for an industry grappling with its own physical boundaries.

However, the nascent field of quantum computing presents its own unique challenges. Notably, three promising approaches are currently at the cutting edge: Superconducting Circuits, Trapped Ions, and Topological Qubits. *Superconducting circuits* employ superconducting qubits that function as artificial atoms. Leveraging established silicon chip fabrication techniques, this approach affords significant scalability and has already demonstrated successful execution of select quantum algorithms. Yet, this method grapples with relatively short coherence times and high error rates, limiting the complexity of quantum computations that can be feasibly executed [4], [5]. The *trapped ions approach*, on the other hand, utilizes electromagnetic fields to confine ions, employing lasers to perform quantum operations. Although it offers longer coherence times and lower error rates compared to superconducting circuits, scaling trapped ion systems to the number of qubits required for practical quantum computation is a substantial challenge [6]. The *topological qubits* approach uses anyons in 2D systems to perform quantum operations. Theoretically, this system could offer the longest coherence time and the lowest error rate, as the information is stored non-locally, providing a kind of topological protection. However, as it stands, there are no reported instances of a successful creation of a topological qubit. Despite this, the robustness of topological states to local perturbations offers the potential to surmount current limitations, paving the way for the next generation of computational systems [7], [8].

The subject of topological phases of matter (TPMs) has has seen a surge of intensive research over the past few decades. Unlike conventional states described by Landau's theory of spontaneous symmetry breaking and local order parameters, topological phasesgapped spin liquids at low temperaturesare characterized by a new order, *topological order*. The ground states of a topological phase possess stable degeneracy and robust long range entanglement. Moreover, topological phases in 2D support quasi-particle excitations (aka anyons), and potentially non-Abelian exchange statistics. What sets TPMs apart is the presence of global degrees of freedom encoded in the ground states. These are resistant to local perturbations and which can be unitarily changed through non-trivial movements of quasi-particle excitations. These distinctive features position TPMs as ideal media for executing fault-tolerant quantum computing, namely, topological quantum computing [9], [10]. The theory of 2D TPMs can be equivalently depicted through either a $(2+1)$ topological quantum field theory or a unitary modular tensor category.

A large class of Topological Phases of Matter (TPM) in 2D can be realized by spin lattice models. Among these, the toric code stands out as one of the most recognized examples. This Abelian topological phase can also be interpreted through a $\mathbb{Z}_2$ gauge theory. The toric code is a special case of Kitaev's quantum double models which associate to each finite group G an exactly solvable lattice model[9]. When G is $\mathbb{Z}_2$, the theory reduces to the toric code, while a non-Abelian group G leads the model to realize a non-Abelian topological phase. In these models, the anyon types align with the irreducible representations of the Hopf algebra $D(G)$, the Drinfeld double (or quantum double) of the group algebra $\mathbb{C}[G]$. The quantum double model can be generalized by replacing G with a semi-simple $\mathbb{C}^*$ Hopf algebra H. Given such a Hopf algebra, the authors in [11] wrote down a frustration-free Hamiltonian consisting of pairwise commuting local projectors analogous to the original setup. We refer to this model as the generalized Kitaev quantum double model, which can be further generalized to a semi-simple weak Hopf algebra [12]. Another class of realizations are the Levin-Wen string-net models [13], based on unitary fusion categories. There is a close relationship between the string-net models and the quantum double models. Specifically, for a Hopf algebra H, it has been shown that the generalized quantum double model based on H is equivalent to the string-net model based on $\mathrm{Rep}(H)$, the category of representations of H [14], [15].

To effectively describe the creation, annihilation, and movement of anyons in the models mentioned above, the concept of ribbon operators (or string operators) plays a critical role. In the context of the toric code, these ribbon operators can be represented as either a string of Pauli Z operators on the lattice or a string of Pauli X operators on the dual lattice. However, for non-Abelian group G, these two types of string operators become 'entangled', necessitating the consideration of a 'thickened' string of operators, namely, operators on a ribbon. A ribbon can be broadly visualized as a strip in the lattice, with one side running along the lattice edges and the other along the dual lattice edges. Within the quantum double model, operators for two types of elementary ribbons (triangles) are initially defined, after which the definition is extended to longer ribbons via induction (see [9], [16] for details). The paper [11] briefly assertswithout offering proofsthat ribbon operators in the generalized quantum double model can be defined in a similar manner.

In the first part, we rigorously define ribbon operators in the generalized quantum double model that is based on a semi-simple $\mathbb{C}^*$ Hopf algebra, and we systematically study their properties. While we affirm several properties as expectedwhich might not be startling to expertsthe computations needed for proving these turn out to be considerably more intricate than those involving finite groups. This complexity partly arises from the challenges associated with dealing with general Hopf algebras rather than simply dealing with group algebras. Furthermore, we unravel some subtleties in the definition of ribbon operators. The literature [9], [16] only takes into account two types of elementary ribbons: the direct triangle and the dual triangle. Our study, however, broadens these into four types, introducing an extra characteristic, which we refer to as *local orientation*. This local orientation can also be applied to general ribbons, resulting in two types of ribbons based on their local orientation. It's important to note that the definition of ribbon operators must differ for each ribbon type. Should we fail to differentiate these two types of ribbons, certain expected common properties will not be upheld. For example, the ribbon operator would fail to commute with Hamiltonian terms away from the end points. We note that this issue is already present in the original quantum double model when the input group is non-Abelian, but it appears this concern hasn't been addressed in the literature to our best knowledge. Lastly, our ribbon operators' definition is explicit, contrasting with those in string-net models, where a set of consistency equations need to be resolved.

In recent years, there has been intensified investigation into 3D topological phases [17] and even more exotic 3D structures, known as fracton phases [18]–[20]. Similar to conventional TPMs, fractons possess stable ground state degeneracy and long-range entanglement. However, unlike TPMs, the ground state degeneracy of fractons is dependent on the system size and is therefore not a topological invariant. Additionally, the mobility of excitations within fractons is constrained, either moving within specific subsystems or not at all. Notable examples of fractons include the Haah code [18] and the X-cube model [20]. While regular topological phases can be characterized by topological quantum field theories, the mathematical characterization of fractons remains an open question. Since fractons also satisfy the conditions of topological order in the sense of [21], we classify the ground states of a fracton as topologically ordered states, aligning with those of conventional topological phases.

Realizing topological phases in physical systems continues to be a formidable challenge. However, we now have access to quantum processors based on a variety of platforms, such as superconducting qubits [22] and Rydberg atomic arrays[23], etc. These devices can support physical qubits on the scale of 10^2, a number that is projected to surge significantly in the near future. Therefore, simulating topological phases in quantum processors emerges as both a feasible and intriguing prospect. Given the inherent robustness of topological phases, such simulations are relatively immune to noise within current quantum processors. Furthermore, engineering topological phases in processors could provide us with greater insights. The toric code ground states have been realized in both superconducting-qubit-based systems [22] and

Rydberg-atom systems [24]. In [22], the authors proposed a quantum circuit comprised of Clifford gates to realize the ground states of the planar toric code (a.k.a. surface code [25]). Studies have also been conducted on quantum circuits realizing non-Abelian topological orders, such as the Levin-Wen string-net model and the Kitaev quantum double model. See for instance [26]–[31], though in these cases, the gates employed are no longer confined to the Clifford group and measurements are used.

In the second part, we construct quantum circuits that can realize the ground states for a variety of topological phases. While [22] only considered the planar toric code, where the lattice is defined on a planar surface, we extend their methodology to a large class of surfaces, both with and without boundaries, utilizing only Clifford gates. The method commences with the +1 eigenstate for all vertex terms, and the ground state is subsequently obtained by projecting this state to the +1 eigenstate of all plaquette operators. This process can be simulated through an appropriate combination of Hadamard and CNOT gates. The judicious selection of the sequence for the plaquettes to which quantum gates are applied is of utmost importance. Given that we are considering lattices on arbitrary surfaces, this issue becomes quite intricate. We outline an explicit algorithm to determine the sequence in which the plaquette operators are simulated. Further, we adapt this method to 3D phases including the 3D toric code and the X-cube fracton model. By comparison, using cluster states and measurements, the authors in [28] proposed an *approximate* realization of the model. Beyond the method using only quantum gates, we also suggest an alternative approachreferred to as the gluing methodfor realizing the same states, which offers a shorter circuit depth. Indeed, it is possible to obtain the ground state for the toric code or X-cube using only measurements. However, given that frequent measurements in near-term quantum processors can be costly, our method presents a trade-off between circuit depth and the extent of measurements.

Identifying the eigenstate of a specific Hamiltonian ranks among the most demanding tasks in condensed matter physics. This task becomes increasingly complex primarily because of the power scaling of the Hilbert space dimension, which inflates exponentially in relation to the system's size [32]. Nonetheless, it is often the case that the system's inherent physical properties, e.g. long-range entanglement, restrict the form of the ground states, and therefore the states corresponding to interesting quantum systems may only occupy a small portion of the exponentially large Hilbert space. This opens up the possibility of efficient representations of the wave function of many-body systems. Examples of efficient representations include matrix product states, projected entangled pair states, and more generally tensor networks.

A recent trend is the study of many-body quantum systems utilizing machine learning techniques, especially artificial neural networks. Restricted Boltzmann Machines (RBMs) are a generative stochastic artificial neural network [33]. Unlike other types of neural networks, RBMs have a unique two-layer architecture that consists of a visible input layer and a hidden layer. The 'restricted' part in the name refers to the lack of intra-layer connections; that is, nodes within the same layer do not interact with each other. RBMs have been used effectively in a variety of machine learning tasks, including dimensionality reduction, classification, regression, and even solving quantum many-body problems [34]–[39].

In 2017, Carleo and Troyer paved a novel path by applying RBM as a variational ansatz, utilizing it to represent ground states for Ising model [34]. This groundbreaking achievement catalyzed the development of numerous explicit RBM representations. Notably, substantial research efforts have been directed towards the examination of toric code [35], [36], graph states [37], and stabilizer code [38], [39], which is equivalent to a graph state under local Clifford operations [40]. While their topological properties and representational power [41], [42] have been extensively studied, there is still a need to explore feasible algorithms for specific models.

We start from the RBM representability of the toric code model as the first step, with the eventual goal of studying that for general topological phases. In [36], Deng and Li utilized a Further Restricted Restricted Boltzmann Machine (FRRBM), that allows only local connections, to numerically find a solution of the toric code model. However, toric code has degeneracy on non-trivial topology, and the ground state derived in the above manner always corresponds certain specific one. On the other hand, it is possible to achieve an arbitrary ground state by turning the toric code as a graph state [43] and transforming a graph state into an RBM [38]. Yet, this approach inevitably introduces non-local connections within each subgraph which adds to the complexity of the RBM.

In the third part, we initially apply stabilizer conditions to several specific configurations to analytically solve the FRRBM for the toric code, exploring its representational capacity. We factorize these solutions on square lattices of various sizes and find that different weights only alter the coefficients of the basis states forming the ground state by factors of ± 1. We then extended this approach to obtain an arbitrary ground state by strategically introducing several non-local connections into the RBM. While this generalization sacrifices the simplicity of local connections, it remains analytically solvable, enabling the simulation of arbitrary ground states in a clean manner. Additionally, we developed an efficient machine learning algorithm to verify the learnability of the models. We further generalize our approach from Z_2 to Z_n and outline potential directions for future research.

In this thesis, I focus on the preparation and simulation of ground states of topological phases of matter, with a particular emphasis on the preparation and simulation of arbitrary ground states. Typically, researchers are content with any ground state, as these states are sufficient to support the existence of anyons, which are pivotal for topological quantum computing. However, the ability to prepare arbitrary ground states is crucial for applications in quantum memory and error-corrected quantum computing. This work aims to advance the methods for achieving these states, thereby enhancing the robustness and reliability of quantum computational systems.

2. BACKGROUND

2.1 Hopf algebra

This chapter contains work from the article entitled "Ribbon operators in the generalized Kitaev quantum double model based on Hopf algebras" written by Bowen Yan, the author, and Shawn X. Cui published on Journal of Physics A [1], and the article entitled "Quantum circuits for toric code and X-cube fracton model" written by the author, Bowen Yan, and Shawn X. Cui published on Quantum [2], and the article entitled "Representing Arbitrary Ground States of Toric Code by Restricted Boltzmann Machine" written by the author, Bowen Yan, and Shawn X. Cui preprinted on arXiv [3].

Hopf algebras are important objects in various areas such as representation theory, tensor categories, algebraic topology, topological quantum field theories, etc. There exists an extensive literature covering different aspects of Hopf algebras. This section provides a brief review with the primary aim of fixing conventions. For detailed discussions, see for instance [44] [45]. A Hopf algebra over $\mathbb{C}$ is a vector space H endowed with the linear maps (called structure maps):

$$\mu\colon H \otimes H \to H, \quad \Delta\colon H \to H \otimes H, \tag{2.1}$$

$$\eta\colon \mathbb{C} \to H, \quad \epsilon\colon H \to \mathbb{C}, \quad S\colon H \to H, \tag{2.2}$$

satisfying several conditions to be specified in the following. Firstly, (μ, η) defines an (associative) algebra structure. That is, the multiplication μ is associative:

$$\mu\left[\mu(a \otimes b) \otimes c\right] = \mu\left[a \otimes \mu(b \otimes c)\right], \tag{2.3}$$

or briefly written as

$$(ab)c = a(bc). \tag{2.4}$$

The unit 1_H for the multiplication μ is given by $\eta(1)$. Secondly, (Δ, ϵ) defines a (coassociative) coalgebra structure with Δ and ϵ the comultiplication and counit, respectively. We will use the *Sweedler notation* for expressions involving comultiplications. For instance, we write

$$\Delta(a) = \sum_{(a)} a' \otimes a''. \tag{2.5}$$

The comultiplication map being coassociative means

$$(\Delta \otimes \mathrm{i}d) \circ \Delta = (\mathrm{i}d \otimes \Delta) \circ \Delta, \tag{2.6}$$

or in Sweedler notation,

$$\sum_{(a)} \left[\sum_{(a')} (a')' \otimes (a')'' \right] \otimes a'' = \sum_{(a)} a' \otimes \left[\sum_{(a'')} (a'')' \otimes (a'')'' \right]. \tag{2.7}$$

Due to the above equality, we simply write

$$(\Delta \otimes \mathrm{i}d) \circ \Delta(a) = \sum_{(a)} a' \otimes a'' \otimes a'''. \tag{2.8}$$

More generally, we use the Sweedler notation for

$$(\Delta \otimes \mathrm{i}d_{H^{\otimes(n-2)}}) \circ \cdots \circ (\Delta \otimes \mathrm{i}d) \circ \Delta(a) = \sum_{(a)} a^{(1)} \otimes \cdots \otimes a^{(n)}. \tag{2.9}$$

The counit ϵ satisfies

$$\sum_{(a)} \epsilon(a')a'' = \sum_{(a)} a'\epsilon(a'') = a. \tag{2.10}$$

Thirdly, Δ and ϵ are both required to be algebra morphisms. In particular, this implies ϵ defines a 1-dimensional representation of H. Lastly, S is called the antipode which is invertible in our consideration satisfying:

$$\sum_{(a)} a'S(a'') = \epsilon(a)1_H = \sum_{(a)} S(a')a''. \tag{2.11}$$

To emphasize on structure maps, we also denote a Hopf algebra by

$$(H; \mu, \eta, \Delta, \epsilon, S). \tag{2.12}$$

In this thesis, we focus solely on finite dimensional semisimple Hopf algebras. Over $\mathbb{C}$, semisimplicity is equivalent to the condition that S is involutory, namely, $S^2 = id$. Certain identities inherent to a finite dimensional Hopf algebra are implied:

$$S(ab) = S(b)S(a), \quad S(1_H) = 1_H, \quad \epsilon[S(a)] = \epsilon(a), \tag{2.13}$$

$$\sum_{(a)} S(a'') \otimes S(a') = \sum_{(S(a))} S(a)' \otimes S(a)''. \tag{2.14}$$

Given a Hopf algebra $(H; \mu, \eta, \Delta, \epsilon, S)$, there are several ways of constructing new Hopf algebras out of it. Taking H^* to be the linear dual of H, then

$$(H^*; \Delta^T, \epsilon^T, \mu^T, \eta^T, S^T) \tag{2.15}$$

defines a Hopf algebra structure on H^*, where f^T is the linear dual of map f[1]. And μ^T is a map from H^* to $H^* \otimes H^*$:

$$\mu^T(f)(a \otimes b) = f\left[\mu(a \otimes b)\right] = f(ab), \tag{2.16}$$

where $a, b \in H$, and $f \in H^*$. We can also define opposite Hopf algebra by

$$(H^{op}; \mu^{op}, \eta, \Delta, \epsilon, S^{-1}), \tag{2.17}$$

where $H^{op} = H$ acts as a vector space, and μ^{op} is defined as

$$\mu^{op}(a \otimes b) = \mu(b \otimes a) = ba. \tag{2.18}$$

[1]↑Another common notation for f^T is f^*. Here we use f^T since under appropriate bases, the matrix of f^T is the transpose of that of f. Another reason is to avoid confusion since we will introduce a $*$ operation below with a different meaning.

Similarly, the co-opposite Hopf algebra H^{cop} is defined by

$$(H^{cop}; \mu, \eta, \Delta^{cop}, \epsilon, S^{-1}), \tag{2.19}$$

where $H^{cop} = H$ acts as a vector space, and Δ^{cop} is defined as

$$\Delta^{cop}(a) = \sum_{(a)} a'' \otimes a'. \tag{2.20}$$

What's more, $(\cdot)^*$, $(\cdot)^{op}$, and $(\cdot)^{cop}$ are all involutive. It is direct to check $(H^*)^{cop} \simeq (H^{op})^*$ and $(H^*)^{op} \simeq (H^{cop})^*$.

For a semisimple Hopf algebra H, a (two-sided) integral is an element $h_0 \in H$ such that for all $a \in H$,

$$a h_0 = h_0 a = \epsilon(a) h_0. \tag{2.21}$$

The space of integrals is 1-dimensional subspace, and h_0 is uniquely defined if we require

$$h_0^2 = h_0, \quad \text{or equivalently} \quad \epsilon(h_0) = 1. \tag{2.22}$$

We call h_0 the Haar integral of H, which can be proved to be cocommutative, namely

$$\Delta(h_0) = \sum_{(h_0)} h_0' \otimes h_0'' = \sum_{(h_0)} h_0'' \otimes h_0'. \tag{2.23}$$

To make a Hopf algebra into Hilbert space, we introduce the $*$-structure, which is a conjugate-linear map $* : H \to H$ satisfying

$$(a^*)^* = a, \quad (ab)^* = b^* a^*, \quad 1^* = 1, \tag{2.24}$$

$$\sum_{(a)} (a')^* \otimes (a'')^* = \sum_{(a^*)} (a^*)' \otimes (a^*)''. \tag{2.25}$$

A Hopf algebra endowed with $*$-structure is called $\mathbb{C}^*$ Hopf algebra. For a Hopf algebra H, we denote the Haar integral of H^* by ϕ. Then $\langle\cdot,\cdot\rangle$ defines a Hermitian inner product on H:

$$\langle a, b\rangle = \phi(a^* b), \text{ for } a, b \in H. \tag{2.26}$$

Unless otherwise stated, throughout this thesis we will use letters h_0, ϕ to represent Haar integrals, a, b, c, x, y to denote general elements of H, and f, g, t for general elements of H^*. We adopt the following notation: $f(x?)$ represents an element of H^* such that $f(x?)(y) = f(xy)$.

2.2 Representations of semisimple Hopf algebras

The category of finite dimensional representations over $\mathbb{C}$ of a semisimple Hopf algebra H is a semisimple tensor category with duals. If V, W are two representations such that

$$\rho_V \colon H \to \mathrm{End}(V), \tag{2.27}$$

$$\rho_W \colon H \to \mathrm{End}(W), \tag{2.28}$$

then $V \otimes W$ is a representation with the action given by

$$a.(v \otimes w) := \big((\rho_V \otimes \rho_W)\Delta(a)\big)(v \otimes w), \quad a \in H, v \in V, w \in W. \tag{2.29}$$

And so is V^* with the action given by

$$a.f := f \circ \rho_V(S(a)), \quad a \in H, f \in V^*. \tag{2.30}$$

A representation V of H is irreducible if $\mathrm{End}_H(V) \simeq \mathbb{C}$. Denote by Irr_H the set of isomorphism classes of irreducible representations of H. Consider the regular representation H with the action given by left multiplication

$$L(a)(c) := ac, \tag{2.31}$$

or right multiplication using $S(\cdot)$:

$$R(a)(c) := cS(a). \tag{2.32}$$

These two actions commute and hence define an action of $H \otimes H$ on H as

$$(a \otimes b).c := acS(b). \tag{2.33}$$

We note that as a representation of $H \otimes H$, we have the isomorphism

$$H \simeq \bigoplus_{\mu \in \mathrm{Irr}_H} \mu^* \otimes \mu. \tag{2.34}$$

An explicit isomorphism is given as follows: For each $\mu \in \mathrm{Irr}_H$, we fix a basis $\{|\mathrm{i}\rangle \mid \mathrm{i} = 1, \cdots, dim(\mu)\}$ and denote the matrix of an element $a \in H$ under this basis by $D^\mu(a)$. Let $h_0 \in H$ be the Haar integral, as defined in Equations 2.21, 2.22. We then define the 'Fourier transformation' on H as [15]

$$|\nu\mathrm{ij}\rangle = \sqrt{\frac{dim(\nu)}{dim(H)}} \sum_{(h_0)} D^\nu(h_0')_{\mathrm{ij}} h_0'', \tag{2.35}$$

where $\nu \in \mathrm{Irr}_H$, and $\mathrm{i}, \mathrm{j} = 1, 2, \cdots, dim(\nu)$. For the sake of self-containedness, we verify in Appendix A.6 that the action of $H \otimes H$ on the subspace $\mathrm{span}\{|\nu\mathrm{ij}\rangle|\mathrm{i},\mathrm{j} = 1, \cdots, dim(\nu)\}$ is given by $\nu^* \otimes \nu$. This thereby defines the isomorphism in Equation 2.34. Lastly, the two representations L and R each induce a representation of H on H^*:

$$L(a)|f\rangle = |f[S(a)?]\rangle, \tag{2.36}$$

$$R(a)|f\rangle = |f(?a)\rangle, \quad |f\rangle \in H^*. \tag{2.37}$$

2.3 Drinfeld double of Hopf algebras

The Drinfeld double (or quantum double) $D(H)$ of a Hopf algebra H is a Hopf algebra

$$D(H) = \Big((H^*)^{cop} \otimes H; \mu_D, \eta_D, \Delta_D, \epsilon_D, S_D\Big), \tag{2.38}$$

which is constructed as a bicrossed product of H and $(H^*)^{cop}$. For $f, g \in H^*$ and $a, b \in H$, μ_D is defined by

$$\mu_D\left[(f \otimes a) \otimes (g \otimes b)\right] = \sum_{(a)} f\, g \left[S^{-1}(a''')?a'\right] \otimes a''b, \tag{2.39}$$

which is known as the *straightening equation*, and notice that

$$f \otimes a = (f \otimes 1)(1 \otimes a). \tag{2.40}$$

The remaining structure maps can be determined by the property that both $(H^*)^{cop}$ and H are sub Hopf algebras of $D(H)$. This is achieved through the inclusions $f \mapsto f \otimes 1$ and $a \mapsto \epsilon \otimes a$, respectively. For example, Δ_D is given by

$$\Delta_D(f \otimes a) = \sum_{(f),(a)} (f'' \otimes a') \otimes (f' \otimes a''), \tag{2.41}$$

where the Sweedler notation to f is applied, treating f as an element of H^* rather than $(H^*)^{cop}$. This convention will be used throughout the thesis. Specifically, for $a \in H^{cop}$, we use Δ rather than Δ^{cop} in the context of the Sweedler notation to define a', a'', and so forth. The definitions for the remaining structure maps are provided as follows:

$$\eta_D(1) = \epsilon \otimes 1, \tag{2.42}$$

$$\epsilon_D(f \otimes a) = f(1) \otimes \epsilon(a), \tag{2.43}$$

$$S_D(f \otimes a) = S(a)S^T(f). \tag{2.44}$$

2.4 Generalized Kitaev model based on Hopf algebras

In this section, H represents a semisimple $\mathbb{C}^*$ Hopf algebra. The original Kitaev model, as presented in [9], is constructed based on the group algebra $\mathbb{C}[G]$ of a finite group G. On the other hand, the generalized Kitaev model is rooted in a semisimple $\mathbb{C}^*$ Hopf algebra H. This generalized model was introduced in [11], which we review below.

To simplify our discussion, we establish the model on a square lattice $\Gamma = (V, E, P)$, where V, E and P denote the set of vertices, (directed) edges, and faces, respectively, as illustrated in Figure 2.1 (the solid grid)[2]. We also define the dual lattice $\Gamma^* = (P^*, E^*, V^*)$, where P^* corresponds to the vertices in Γ^* dual to the faces P in Γ, and E^* and V^* have similar interpretations. For any element $x \in V \cup E \cup P$, we use x^* to denote the corresponding element in $V^* \cup E^* \cup P^*$. The direction of the dual edge e^* of an edge $\mathrm{e} \in E$ is determined by rotating the direction of e counterclockwise by 90°. Lastly, a **site** $s = (v, p)$ is defined as a pair comprising a vertex v and an adjacent face p that contains v. We draw a segment connecting v and the dual vertex p^* to represent the site in Figure 2.1.

To each edge e of Γ, we attach a copy of the Hopf algebra (also a Hilbert space $\mathcal{H}_\mathrm{e} := H$). The model's total Hilbert space is then constructed as the tensor product of these associated Hilbert spaces over all edges:

$$\mathcal{H} := \bigotimes_{\mathrm{e}\in E} \mathcal{H}_\mathrm{e}. \tag{2.45}$$

$$L_+^a(x) = ax, \quad L_-^a(x) = xS(a). \tag{2.46}$$

$$T_+^f(x) = f(x'')x', \quad T_-^f(x) = f[S(x')]x''. \tag{2.47}$$

Upon the establishment of the oriented graph $\Gamma = (V, E, P)$, we can define the edge operators [11] illustrated in Figure 2.1 and the local operators $A_a(s)$ and $B_f(s)$ on a site $s = (v, p)$ illustrated in Figure 2.2 and Figure 2.3, respectively. For each edge e of the lattice and $f \in H^*$, $a \in H$, edge operators $T_\pm^f$ and $L_\pm^a$ act on $\mathcal{H}_\mathrm{e}$ as Equations 2.46-2.47.

To define $A_a(s)$ for $a \in H$, we start from the site s, go around the vertex v to apply edge operators $L_\pm^{a'}$, $L_\pm^{a''}$, $L_\pm^{a'''}$, $L_\pm^{a^{(4)}}$ to each edge adjacent to v in counterclockwise order as shown

[2]↑The edges in the lattice can be arbitrarily directed, and the physics of the model will be independent of those directions.

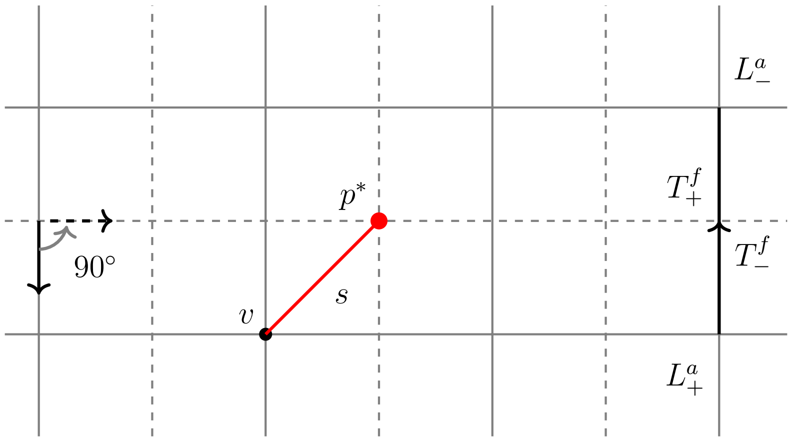


Figure 2.1. The solid grid connecting all vertices V represents the square lattice Γ, while the dashed grid connecting all dual vertices P^* represents the dual square lattice Γ^*. A site $s = (v, p)$ is represented by a segment connecting a vertex v and a dual vertex p^*. For $f \in H^*, a \in H$, the edge operators $T_\pm^f$ and $L_\pm^a$ act on the Hilbert space $\mathcal{H}_{\text{e}}$ of an edge e.

and explained in Figure 2.2. For example, when it is applied to the product state of $|x_1\rangle$, $|x_2\rangle$, $|x_3\rangle$, $|x_4\rangle$, the result is

$$A_a(s)|x_1\rangle|x_2\rangle|x_3\rangle|x_4\rangle = \sum |a'x_1\rangle|a''x_2\rangle|a'''x_3\rangle|a^{(4)}x_4\rangle. \tag{2.48}$$

$$A_a(s) = \sum L_+^{a'} \otimes L_+^{a''} \otimes L_+^{a'''} \otimes L_+^{a^{(4)}} \tag{2.49}$$

To define $B_f(s)$, $f \in H^*$, we start from the site s, go around the dual vertex p^* to apply edge operators $T_\pm^{f'}$, $T_\pm^{f''}$, $T_\pm^{f'''}$, $T_\pm^{f^{(4)}}$ to the edges on the boundary of p in counterclockwise order as shown and explained in Figure 2.3. When it is applied to the product state of $|x_1\rangle$, $|x_2\rangle$, $|x_3\rangle$, $|x_4\rangle$, the result is [3]

$$B_f(s)|x_1\rangle|x_2\rangle|x_3\rangle|x_4\rangle = \sum f(x_1''x_2''x_3''x_4'')|x_1'\rangle|x_2'\rangle|x_3'\rangle|x_4'\rangle. \tag{2.50}$$

$$B_f(s) = \sum T_+^{f'} \otimes T_+^{f''} \otimes T_+^{f'''} \otimes T_+^{f^{(4)}} \tag{2.51}$$

[3]↑To derive Equation 2.50, we use the fact that the comultiplication Δ_* in H^* is actually μ^T.

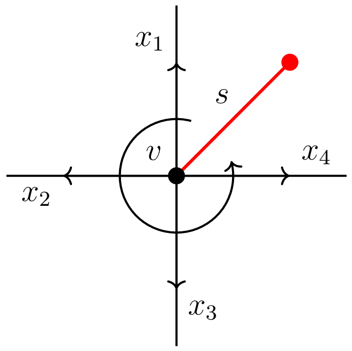


Figure 2.2. The convention for the local operator $A_a(s)$: for each edge, we choose $+$ sign for the edge operator $L^{a^{(n)}}$ if the edge leaves the vertex, and choose $-$ sign otherwise.

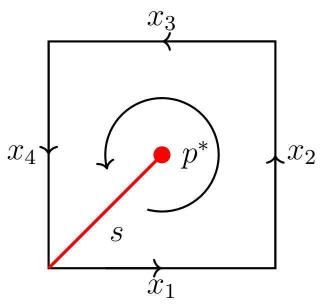


Figure 2.3. The convention for the local operator $B_f(s)$: for each edge, we choose $+$ sign for the edge operator $T^{f^{(n)}}$ if the direction of the edge coincides with the counterclockwise orientation of the boundary of p, and choose $-$ sign otherwise.

We remark that our convention for defining the operators $A_a(s)$ and $B_f(s)$ is opposite to that in [9] [11]. Explicitly, these operators on a lattice Γ will be the same as those of [11] on a lattice Γ' obtained from Γ by reversing the orientation of all edges. When the Hopf algebra is a group algebra, our convention is consistent with that in [16].

For each site $s = (v, p)$, we extend the definition of $A_a(s)$ and $B_f(s)$ for the entire Hilbert space $\mathcal{H}$. This is achieved by taking the tensor product with the identity operator on edges that are not adjacent to v or p. These local operators at s, denoted $A_a(s)$ and $B_f(s)$, define a representation of the Drinfeld double $D(H)$, mapping $f \otimes a$ to $B_f A_a$. A significant aspect of this representation is the straightening equation. To ensure the self containedness, we provide a verification of the equation for local operators in Appendix A.1.

Let $h_0 \in H$ and $\phi \in H^*$ represent the Haar integral. For a site $s = (v, p)$, it is verifiable that $A_{h_0}(v) := A_{h_0}(s)$ solely depends on v and $B_\phi(p) := B_\phi(s)$ exclusively relies on p. Furthermore, the operators in the set $\{A_{h_0}(v) : v \in V\} \cup \{B_\phi(p) : p \in P\}$ function as mutually commuting projectors. The (frustration-free) Hamiltonian of the model is given by

$$H = -\sum_{v \in V} A_{h_0}(v) - \sum_{p \in P} B_\phi(p), \tag{2.52}$$

and the ground states are simultaneously stabilized by all the terms in the Hamiltonian. Equivalently, the ground states space can be characterized as the subspace of $\mathcal{H}$ corresponding to the trivial representation of $D(H)$ on all sites s.

2.5 Toric code

The toric code represents the most elementary example of Kitaev's quantum double models where $G = \mathbb{Z}_2$. It is defined on a square lattice, topologically equivalent to a torus as shown in Figure 2.4. A spin 1/2 is located on each edge e of the lattice, and we also use e to denote the associated qubit. The lattice componentsvertices, faces, and edgesare denoted as V, F, and E, respectively. For each face $f \in F$, the set of surrounding edges is denoted by $s(f)$, and similarly, for each vertex $v \in V$, the set of surrounding edges is $s(v)$. The vertex operator A_v consists of tensor products of Pauli operators $\hat{\sigma}^x_{\mathrm{e}}$ acting on the edges e within $s(v)$. Similarly, the face operator B_f is formed from tensor products of Pauli operators $\hat{\sigma}^z_{\mathrm{e}}$ acting on the edges e within $s(f)$. The Hamiltonian of the toric code is defined as the sum of all vertex and face operators:

$$H = -\sum_{v \in V} A_v - \sum_{f \in F} B_f = -\sum_{v \in V} \prod_{\mathrm{e} \in s(v)} \hat{\sigma}^x_{\mathrm{e}} - \sum_{f \in F} \prod_{\mathrm{e} \in s(f)} \hat{\sigma}^z_{\mathrm{e}}. \tag{2.53}$$

For a Hamiltonian of the form

$$H = -\sum_{\mathrm{i}} P_{\mathrm{i}}, \tag{2.54}$$

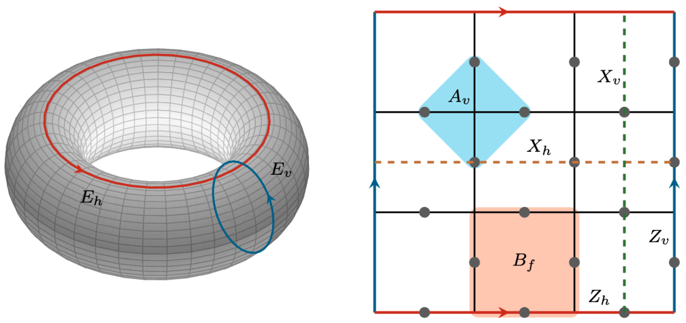


Figure 2.4. The torus on the left is cut along the edges E_v and E_h to get the square lattice shown on the right, with opposite edges identified. The 3×3 lattice shows stabilizer operators A_v within the blue range and B_f within the red range, logical operators X_v and X_h along the vertical and horizontal dashed loops, respectively, and logical operators Z_v and Z_h along the edges E_v and E_h.

where each P_i is a projector and all projectors are mutually commuting, the ground state $|GS\rangle$ can be derived from any arbitrary non-zero state $|\phi\rangle$:

$$|GS\rangle = \prod_{\mathrm{i}} P_\mathrm{i} |\phi\rangle. \tag{2.55}$$

This construction ensures that the ground state is the simultaneous eigenvector of all projectors. Given $A_v^2 = B_f^2 = 1$ and $[A_v, B_f] = 0$ for all $v \in V$ and $f \in F$, it can be confirmed that $\frac{1+A_v}{2}$ and $\frac{1+B_f}{2}$ act as projectors. Replacing A_v and B_f in the Hamiltonian with these projectors yields a equivalent form consistent with Equation 2.54. This equivalence, stemming from a one-to-one correspondence in their spectra, ensures that the state

$$|GS\rangle = \prod_{v \in V} \frac{1 + A_v}{2} \prod_{f \in F} \frac{1 + B_f}{2} |\phi\rangle \tag{2.56}$$

is a valid ground state as per the earlier defined criteria. Notice that the toric code model is inherently a stabilizer code, with the local Hamiltonian terms A_v and B_f acting as stabilizer operators. In the context of error-correcting codes, the ground states function as logical states. The operators that dissolve these ground states are known as logical operators[4]. As described in [9], the degeneracy of the ground states for a 2D toric code on a torus is identified as four distinct states: $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$. The logical operators X_v and X_h consist of strings of $\hat{\sigma}^x_{\mathrm{e}}$ acting along the vertical and horizontal loops, respectively, transforming $|00\rangle$ to $|01\rangle$ and $|10\rangle$. Similarly, the logical operators Z_v and Z_h consist of strings of $\hat{\sigma}^z_{\mathrm{e}}$ acting along E_v and E_h, respectively, distinguish $|00\rangle$ from $|01\rangle$ and $|10\rangle$.

2.6 Restricted Boltamann Machines

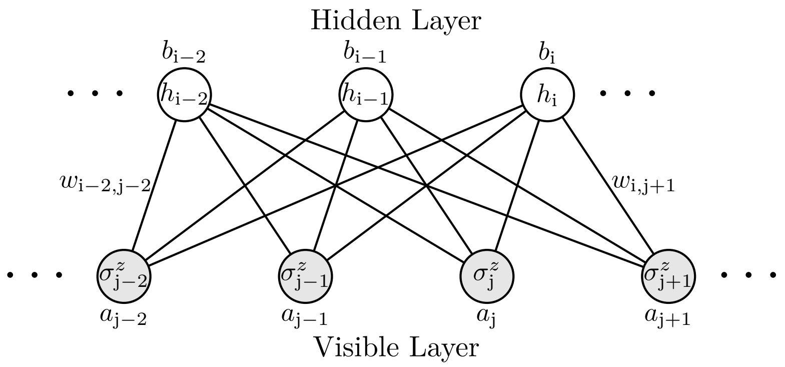


Figure 2.5. This diagram illustrates a RBM with visible neurons colored gray and hidden neurons colored white. The architecture ensures there are no intra-layer connections; instead, each hidden neuron is connected to all visible neurons. Each neuron and each connection is assigned a weight.

In the literature [34], Carleo and Troyer employ an Restricted Boltamann Machines (RBM) as a variational ansatz for the spin-half Ising model, as illustrated in Figure 2.5. The neural network consists of a layer of visible neurons corresponding to N physical spins in the configuration $S = (\sigma^z_1, \sigma^z_2, \ldots, \sigma^z_N)$ [5], and a single hidden layer containing M auxiliary spins represented as $M = (h_1, h_2, \ldots, h_M)$. The wave function for the configuration is expressed in the variational form:

$$\Psi_M(S;\mathcal{W}) = \sum_{\{h_{\mathrm{i}}\}} \mathrm{e}^{\sum_{\mathrm{j}} a_{\mathrm{j}}\sigma^z_{\mathrm{j}} + \sum_{\mathrm{i}} b_{\mathrm{i}} h_{\mathrm{i}} + \sum_{\mathrm{ij}} h_{\mathrm{i}} w_{\mathrm{i},\mathrm{j}} \sigma^z_{\mathrm{j}}}, \tag{2.57}$$

[4]↑Z_h and X_v, as well as Z_v and X_h, serve as pairs of logical X and Z operators for the two logical qubits, and are thus named as the logical operators.
[5]↑Throughout this paper, we use $\hat{\sigma}^z$ for operators and use σ^z for classical variables, $\sigma^z = \pm 1$.

where $\{h_{\mathrm{i}}\} = \{-1, 1\}^M$ represents all possible configurations of the hidden auxiliary spins. The network weights $\mathcal{W} = (a_{\mathrm{i}}, b_{\mathrm{j}}, w_{\mathrm{i,j}})$ can then be trained to optimize

$$|\Psi\rangle = \sum_S \Psi_M(S; \mathcal{W})|S\rangle \tag{2.58}$$

to best represent the ground state $|GS\rangle$. As RBM restricts intralayer interactions, we may trace out all hidden variables according to the chosen preferred basis to simplify the wave function:

$$\Psi_M(S; \mathcal{W}) = \mathrm{e}^{\sum_{\mathrm{j}} a_{\mathrm{j}} \sigma_{\mathrm{j}}^z} \prod_{\mathrm{i}} 2\cosh(b_{\mathrm{i}} + \sum_{\mathrm{j}} w_{\mathrm{i,j}} \sigma_{\mathrm{j}}^z). \tag{2.59}$$

3. RIBBON OPERATORS IN GENERALIZED KITAEV QUANTUM DOUBLE

This chapter contains work from the article entitled "Ribbon operators in the generalized Kitaev quantum double model based on Hopf algebras" written by Bowen Yan, the author, and Shawn X. Cui published on Journal of Physics A [1].

In this chapter, we rigorously define ribbon operators in the generalized Kitaev quantum double model based on a semi-simple $\mathbb{C}^*$ Hopf algebra. We systematically study their properties and demonstrate that the ribbon operators can be interpreted as representations of $D(H)^*$ or $D(H)^{*,op}$, where $D(H)$ represents the Drinfeld double of H. Additionally, we provide an explicit proof that when a ribbon is given, the ribbon operators on it commute with all terms in the Hamiltonian, except for those associated with the two ends of the ribbon. Consequently, ribbon operators create excitations exclusively at their ends. For a given ribbon τ, we denote by V_τ the space of states obtained by applying ribbon operators on τ to the ground state. The space V_τ represents the collection of 2-point excitations, wherein the excitations are localized at the ends of τ. It is shown that V_τ is naturally isomorphic to $D(H)^*$. Furthermore, local operators situated at the ends of τ act on V_τ through regular representations of $D(H)$. As a result, we establish a one-to-one correspondence between elementary excitation types and irreducible representations of $D(H)$.

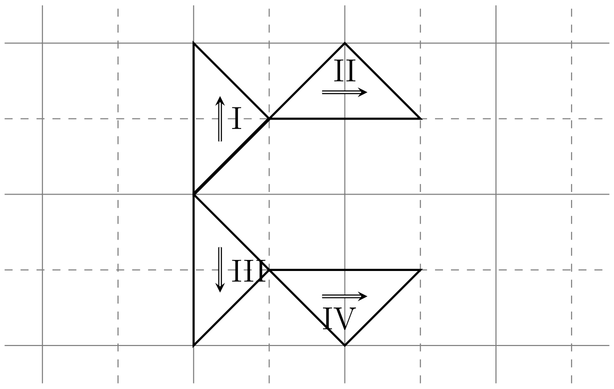


Figure 3.1. An illustration of elementary ribbons (dark solid triangles). The solid grid represents the lattice and the dashed grid represents the dual lattice.

Furthermore, we unravel some subtleties in the definition of ribbon operators. In the literature (e.g., [9], [16]), only two types of elementary ribbons are taken into account: the direct triangle and the dual triangle. For instance, in Figure 3.1, I and III are direct triangles, while II and IV are dual triangles. However, we show in Section 3.2 that I and III have to be treated differently when defining operators on them, and so do II and IV. The key aspect to note is that each pair of triangles possesses a distinguishing property, which we call *local orientation*. For instance, II is locally clockwise while IV is locally counterclockwise. Furthermore, the concept of local orientation can be extended to ribbons in a more general context as shown in Section 3.3. For a more comprehensive understanding of the technical intricacies, readers can refer to the appendices, where many of the detailed explanations can be found.

3.1 Directed ribbons

Let $s_0 = (v_0, p_0)$ and $s_1 = (v_1, p_1)$ be two distinct sites that share a common vertex (i.e., $v_0 = v_1$) or a common dual vertex (i.e., $p_0 = p_1$). There is a unique triangle τ whose sides are given by s_0, s_1, and an edge e_τ in the lattice or the dual lattice. See the bottom left two examples in Figure 3.2. The triangle τ is said to be of dual (resp. direct) **type** if e_τ is an edge in the dual (resp. direct) lattice, or equivalently, if $v_0 = v_1$ (resp. $p_0 = p_1$). We also assign a direction to τ, indicated by a double arrow inside the triangle, so that it points from s_0 to s_1. Denote by $s_\mathrm{i} = \partial_\mathrm{i}\tau,\ \mathrm{i} = 0, 1$. A *ribbon* is a sequence of mutually non-overlapping directed triangles $\tau = \tau_1\tau_2\cdots\tau_n$ such that $\partial_1\tau_\mathrm{i} = \partial_0\tau_{\mathrm{i}+1},\ \mathrm{i} = 1, \cdots, n-1$. Note that τ inherits a direction from its components, also indicated by a double arrow, and we call $\partial_0\tau := \partial_0\tau_1$ the initial site and $\partial_1\tau := \partial_1\tau_n$ the terminal site of τ. See Figure 3.2 for an illustration of several ribbons. By default, all ribbons are directed. A closed ribbon is one for which the initial site and terminal site coincide. Unless otherwise stated, ribbons considered in this thesis are not closed. Triangles are called elementary ribbons. We introduce a property, called **local orientation**, of directed ribbons which seems to be missing in the literature, but will turn out to be critical to coherently define ribbon operators.

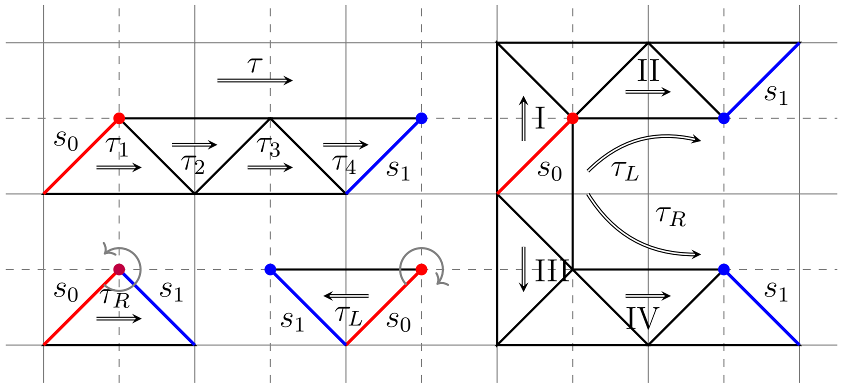


Figure 3.2. A ribbon τ is composed of triangles τ_{i} (i $= 1, 2 \cdots n$) with a direction from s_0 to s_1. A triangle is a component of a ribbon with inherited direction and also the shortest ribbon.

Let τ be a directed triangle (of dual or direct type) with initial site $s_0 = \partial_0\tau = (v_0, p_0)$. Then τ has clockwise (resp. counterclockwise) local orientation if a clockwise (resp. counterclockwise) rotation of s_0 around p_0^* immediately swipes through the interior of τ. We draw a clockwise/counterclockwise arrow around p_0^* to denote the local orientation of τ (See Figure 3.2).

An intuitive motivation for introducing local orientation is as follows. We can see that for a triangle of a given type, a choice of direction is not sufficient to uniquely determine the shape of the triangle. For example, the triangles II and IV in Figure 3.2 are both of dual type and directed to the right, but IV is an 'upside down' version of II, and as will be shown later, they have to be treated differently when we define ribbon operators on them. Local orientation can be used to distinguish those two since triangle II is locally clockwise while IV is locally counterclockwise.

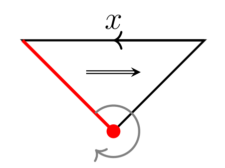
It is straightforward to see that changing the direction of a triangle will also change its local orientation. We note that a choice of direction is a *structure* on the triangle, while the type and local orientation are each a *property* of a directed triangle (though only the later depends on the direction). Thus, there are four classes of directed triangles according to different combinations of local orientation and type. In Figure 3.2, the triangles I-IV in increasing order are, respectively, clockwise direct, clockwise dual, counterclockwise direct, and counterclockwise dual.

Now let τ be a general directed ribbon. Clearly, its composite triangles can have different types (direct or dual). However, an important observation is that all of the triangles of τ must have the same local orientation. Hence, we can extend the notion of local orientation from triangles to general ribbons. Intuitively, if a ribbon aligns horizontally and directs from left to right, then turning it upside down will change its local orientation while keeping its direction. Reversing the direction alone will flip its local orientation as well. As a notation, we also denote a directed ribbon by τ_L if it is locally clockwise and by τ_R if it is locally counterclockwise (This notation is motivated by the left/right hand rule).

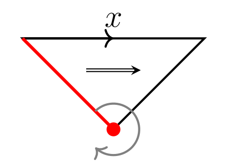
3.2 Definition of ribbon operators

For a directed ribbon τ and $h \otimes f \in H \otimes H^*$, we will define the ribbon operator $F^{h\otimes f}(\tau)$, also written as $F^{(h,f)}(\tau)$. The operators will act on the whole Hilbert space $\mathcal{H}$, but the action is non-trivial only on the edges contained in τ. Explicitly, for an elementary ribbon τ, let $\mathcal{H}_\tau := \mathcal{H}_{\mathrm{e}_\tau}$ if τ is direct, and $\mathcal{H}_\tau := \mathcal{H}_{\mathrm{e}^*_\tau}$ otherwise. For a general ribbon τ, decompose $\tau = \tau_1 \sqcup \tau_2$ so that $\partial_1\tau_1 = \partial_0\tau_2$ and define inductively $\mathcal{H}_\tau := \mathcal{H}_{\tau_1} \otimes \mathcal{H}_{\tau_2}$. Then $F^{(h,f)}(\tau)$ will only act non-trivially on the space $\mathcal{H}_\tau$. The definition of ribbon operators below is motivated by [9] [16] for group algebras and by [11] for Hopf algebras. However, none of the above references addresses the critical issue of local orientation, as to be discussed later.

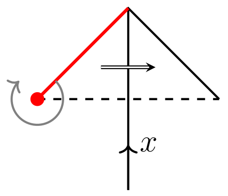
First, assume τ is an elementary directed ribbon, i.e., a triangle. There are four cases depending on its type and local orientation. Also, recall that the edges in the lattice as well as those in the dual lattice are directed. The direction of the edge e_τ and that of τ can be either parallel or opposite. Taking this into consideration, we distinguish eight cases in Equations 3.1a-3.1h, where Equations $(a)-(d)$ correspond to locally clockwise triangles and (e) $-(h)$ locally counterclockwise triangles.



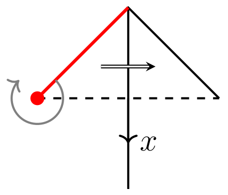
$$F^{(h,f)}(\tau_L)|x\rangle = \sum_{(x)} \epsilon(h) f[S(x'')]|x'\rangle \tag{3.1a}$$



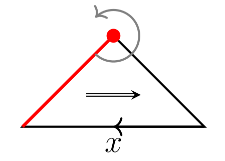
$$F^{(h,f)}(\tau_L)|x\rangle = \sum_{(x)} \epsilon(h) f(x')|x''\rangle \tag{3.1b}$$



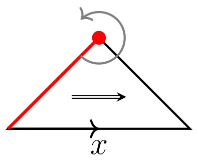
$$F^{(h,f)}(\tau_L)|x\rangle = \epsilon(f)|xS(h)\rangle \tag{3.1c}$$



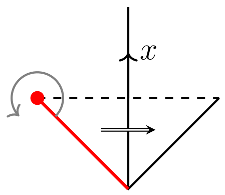
$$F^{(h,f)}(\tau_L)|x\rangle = \epsilon(f)|hx\rangle \tag{3.1d}$$



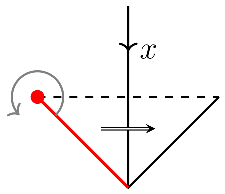
$$F^{(h,f)}(\tau_R)|x\rangle = \sum_{(x)} \epsilon(h) f[S(x')]|x''\rangle \tag{3.1e}$$



$$F^{(h,f)}(\tau_R)|x\rangle = \sum_{(x)} \epsilon(h) f(x'')|x'\rangle \tag{3.1f}$$



$$F^{(h,f)}(\tau_R)|x\rangle = \epsilon(f)|S(h)x\rangle \tag{3.1g}$$



$$F^{(h,f)}(\tau_R)|x\rangle = \epsilon(f)|xh\rangle \tag{3.1h}$$

For ribbons other than elementary triangles, we define the ribbon operators inductively. Let τ be an arbitrary ribbon. Decompose τ as $\tau = \tau_1 \sqcup \tau_2$, where the terminal site of τ_1 matches the initial site of τ_2, and they are disjoint otherwise. For $h \otimes f \in H \otimes H^*$, define

$$F^{h,f}(\tau) := \sum_{\mathrm{i},(\mathrm{i}),(h)} F^{h',g_{\mathrm{i}}}(\tau_1) F^{S(\mathrm{i}''')h''\mathrm{i}',f(\mathrm{i}''?)}(\tau_2), \tag{3.2}$$

where $\{\mathrm{i}\}$ is an orthogonal complete basis of H, and $g_{\mathrm{i}} = \langle \mathrm{i}, \ \rangle$ is the corresponding functional in H^*. The above definition is explicit, but a more intuitive way is as follows. For an element $h \otimes f \in D(H)^* \simeq H \otimes H^*$ where the isomorphism denotes a linear isomorphism between vector spaces,

$$\Delta(h \otimes f) = \sum_{(h \otimes f)} (h \otimes f)' \otimes (h \otimes f)''. \tag{3.3}$$

We apply the expansion to the construction of ribbon operators as

$$F^{h\otimes f}(\tau) := \sum_{(h\otimes f)} F^{(h\otimes f)'}(\tau_1)F^{(h\otimes f)''}(\tau_2). \tag{3.4}$$

It can be checked that Equations 3.2 and 3.4 are equivalent. The ribbon operators do not depend on how the ribbon is partitioned into shorter ones due to the coassociativity of the comultiplication in Hopf algebras.

3.3 Local orientation in original Kitaev model

In this subsection, we show that the distinction of local orientation is already necessary in the orignal Kitaev model. Note that, from Equations 3.1, $F^{(h,f)}(\tau)$ does not distinguish local orientations on direct triangles if H is cocommutative, and it does not distinguish local orientations on dual triangles if H is commutative. In particular, if H is the group algebra of an Abelian group (e.g., toric code), then local orientations are redundant. On the other hand, for the group algebra of a non-Abelian group in the original Kitaev model, the two local orientations on a dual triangle should support different ribbon operators according to our definitions. This distinction, however, has not been addressed in the literature, to the best of our knowledge. In [9], [16], the definition of ribbon operators on triangles coincide with that presented in Equations 3.1a-3.1d corresponding to locally clockwise orientation. We show below with an explicit example that ignoring local orientations can cause certain properties to fail.

For the rest of the subsection, let $H = \mathbb{C}[G]$ be the group algebra of a non-Abelian group G. Equation 3.5 is a commutation relation that is expected to hold between ribbon operators and plaquette operators, where s_0 is the initial site of a ribbon τ (see Equation (B42) in [16]), and $t, h, g \in G$.

$$B_t(s_0)F^{h,g}(\tau) = F^{h,g}(\tau)B_{th}(s_0). \tag{3.5}$$

In fact, we just need the above identity to hold when both sides act on the ground state.

Take τ to be the ribbon shown in Figure 3.3, which is a dual triangle and has locally counterclockwise orientation. In Appendix A.2, we show in detail that Equation 3.5 fails for

τ and any other ribbon that starts with τ if we use the old definition of ribbon operators on them. By recognizing τ with locally counterclockwise orientation and using the new definition (Equation 3.1g), we can resolve the issue, and obtain the following commutation relation,

$$B_t(s_0)F^{h,g}(\tau) = F^{h,g}(\tau)B_{ht}(s_0), \tag{3.6}$$

which is equivalent to Equation 3.5 when acting on the ground state since $\delta_{ht,\mathrm{e}} = \delta_{th,\mathrm{e}}$.

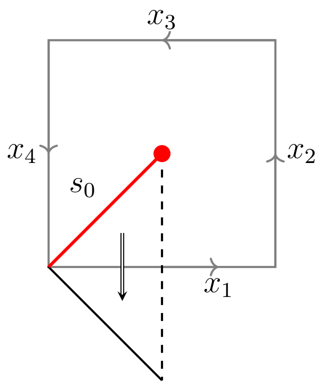


Figure 3.3. A counter-example of a ribbon for which Equation 3.5 fails in the original Kitaev model.

3.4 Properties of ribbon operators

In this section, we establish a few properties of ribbon operators. Recall that the ribbon operators $F^{h,f}(\tau)$ only act non-trivially on the Hilbert space $\mathcal{H}_\tau$. Let τ_L and τ_R be a locally clockwise and a locally counterclockwise ribbon, respectively:

$$F^{h_1,f_1}(\tau_L) \cdot F^{h_2,f_2}(\tau_L) = F^{h_1h_2,f_2f_1}(\tau_L), \tag{3.7}$$

$$F^{h_1,f_1}(\tau_R) \cdot F^{h_2,f_2}(\tau_R) = F^{h_2h_1,f_1f_2}(\tau_R). \tag{3.8}$$

In another words, the operators $F^{h,f}(\tau)$ define a representation of $D(H)^{*,op}$ on $\mathcal{H}_\tau$ if τ is locally clockwise, and a representation of $D(H)^*$ if τ is locally counterclockwise.

Proof. In Appendix A.3, we show in details that the above two equations hold for elementary ribbons. Then it can be proved inductively that they also hold for general ribbons using the compatibility condition between multiplication and comultiplication in a Hopf algebra. Notice that $D(H)^{*,op}$ and $D(H)^*$ share the same comultiplication. Below we only give the proof for τ_R since that of the other case is similar.

Let τ_R be a locally counterclockwise ribbon. Assume Equation 3.8 holds for any ribbon whose length is shorter than that of τ_R. Decompose τ_R as $\tau_R = \tau_1 \sqcup \tau_2$ such that $\partial_1\tau_1 = \partial_0\tau_2$:

$$
\begin{aligned}
& F^{h_1\otimes f_1}(\tau_R) \cdot F^{h_2\otimes f_2}(\tau_R) && (3.9)\\
= & \sum_{(h_1\otimes f_1)} F^{(h_1\otimes f_1)'}(\tau_1)F^{(h_1\otimes f_1)''}(\tau_2) \;\cdot \sum_{(h_2\otimes f_2)} F^{(h_2\otimes f_2)'}(\tau_1)F^{(h_2\otimes f_2)''}(\tau_2) && (3.10)\\
= & \sum_{(h_1\otimes f_1)} \sum_{(h_2\otimes f_2)} F^{(h_1\otimes f_1)'}(\tau_1)F^{(h_2\otimes f_2)'}(\tau_1)\; F^{(h_1\otimes f_1)''}(\tau_2)F^{(h_2\otimes f_2)''}(\tau_2) && (3.11)\\
= & \sum_{(h_1\otimes f_1)} \sum_{(h_2\otimes f_2)} F^{(h_1\otimes f_1)'(h_2\otimes f_2)'}(\tau_1)\; F^{(h_1\otimes f_1)''(h_2\otimes f_2)''}(\tau_2) && (3.12)\\
= & \sum_{(h_2h_1\otimes f_1f_2)} F^{(h_2h_1\otimes f_1f_2)'}(\tau_1)\; F^{(h_2h_1\otimes f_1f_2)''}(\tau_2) && (3.13)\\
= & F^{h_2h_1,f_1f_2}(\tau_R). && (3.14)
\end{aligned}
$$

In the above derivation, the first and the last equality are due to Equation 3.4, the third by induction, the fourth by the compatibility condition between multiplication and comultiplication in $D(H)^*$, and the second by the commutativity between ribbon operators on τ_1 and those on τ_2. $\square$

Next, we examine the commutation relation between ribbon operators and local operators. Let $|GS\rangle \in \mathcal{H}$ be the ground state[1]. Then at any site s, the local operators act on $|GS\rangle$ as follows,

$$A_a(s)|GS\rangle = |GS\rangle, \tag{3.15}$$

$$B_f(s)|GS\rangle = f(1)|GS\rangle, \quad a \in H, f \in H^*. \tag{3.16}$$

[1]↑To the interest of the current thesis, we can assume the lattice is defined on the sphere or the infinite plane, and so there is a unique ground state.

Let τ be a ribbon with initial site $s_0 = \partial_0\tau$ and terminal site $s_1 = \partial_1\tau$. Assume the length of τ, i.e., the number of triangles contained in τ, is greater than one. The following is a technical lemma concerning the commutation relation between ribbon operators on τ and local operators on its ends.

Lemma. Let τ_L and τ_R be a locally clockwise and a locally counterclockwise ribbon, respectively, as described above.

1. At s_0, we have

$$A_a(s_0)F^{(h,f)}(\tau_L) = \sum_{(a)} F^{\{a'hS(a'''),f[S(a'')?]\}}(\tau_L)A_{a^{(4)}}(s_0), \tag{3.17a}$$

$$A_a(s_0)F^{(h,f)}(\tau_R) = \sum_{(a)} F^{\{a''hS(a^{(4)}),f[S(a''')?]\}}(\tau_R)A_{a'}(s_0), \tag{3.17b}$$

$$B_t(s_0)F^{(h,f)}(\tau_L) = \sum_{(h)} F^{(h'',f)}(\tau_L)B_{t[?S(h')]}(s_0), \tag{3.17c}$$

$$B_t(s_0)F^{(h,f)}(\tau_R) = \sum_{(h)} F^{(h'',f)}(\tau_R)B_{t[S(h')?]}(s_0). \tag{3.17d}$$

2. At s_1, we have

$$A_a(s_1)F^{(h,f)}(\tau_L) = \sum_{(a)} F^{[h,f(?a'')]}(\tau_L)A_{a'}(s_1), \tag{3.18a}$$

$$A_a(s_1)F^{(h,f)}(\tau_R) = \sum_{(a)} F^{[h,f(?a')]}(\tau_R)A_{a''}(s_1), \tag{3.18b}$$

$$B_t(s_1)F^{(h,f)}(\tau_L) = \sum_{(\mathrm{i}),(h),\mathrm{i}} f(\mathrm{i}'')F^{(h',g_\mathrm{i})}(\tau_L)B_{t[S(\mathrm{i}''')h''\mathrm{i}'?]}(s_1), \tag{3.18c}$$

$$B_t(s_1)F^{(h,f)}(\tau_R) = \sum_{(\mathrm{i}),(h),\mathrm{i}} f(\mathrm{i}'')F^{(h',g_\mathrm{i})}(\tau_R)B_{t[?S(\mathrm{i}''')h''\mathrm{i}']}(s_1). \tag{3.18d}$$

In the above, $\{\mathrm{i}\}$ is an orthogonal complete basis of H, and $g_\mathrm{i} = \langle \mathrm{i}, \ \rangle$ is the corresponding functional in H^*.

Proof. For a detailed proof, see Appendix A.4. The idea is that we first prove the above equations for ribbons with shortest possible length, and then extend the equality to longer ribbons using the decomposition formula in Equation 3.4. The shortest possible ribbons for the equalities in Equation 3.17 are illustrated in Figure 3.4, and those in Equation 3.18 illustrated in Figure 3.5. □

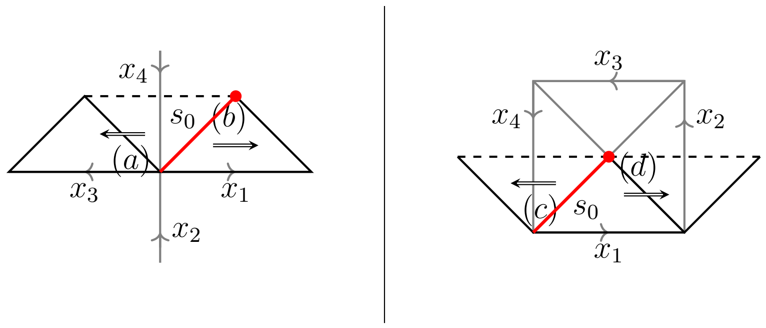


Figure 3.4. Ribbons marked with (a)-(d) correspond the Equation 3.17 a-d.

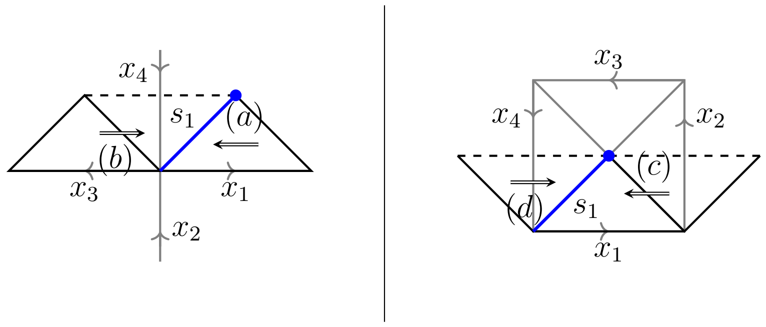


Figure 3.5. Ribbons marked with (a)-(d) correspond the Equation 3.18 a-d.

Using the lemma, we can also deduce that ribbon operators commute with all terms in the Hamiltonian except for those associated with the ends of the ribbon.

Let τ be a ribbon and s be a site on τ such that s has no overlap with $\partial_i \tau$. Denote the terms associated to s in the Hamiltonian by $A(s) = A_{h_0}(s), B(s) = B_\phi(s)$ where $h_0 \in H$ is the Haar integral of H and $\phi \in H^*$ is the Haar integral of H^*. Then,

$$A(s)F^{(h,f)}(\tau) = F^{(h,f)}(\tau)A(s), \tag{3.19a}$$

$$B(s)F^{(h,f)}(\tau) = F^{(h,f)}(\tau)B(s). \tag{3.19b}$$

Proof. See Appendix A.5 for a proof. □

The commutation relation between ribbon operators and local operators at the ends in the lemma may look complicated. However, if we restrict ribbon operators on the ground state, then those relations reduce to more compact formulas. Let V_τ be the Hilbert space of ribbon operators on τ acting on the ground state,

$$V_\tau = \mathrm{span}_{\mathbb{C}}\{|h \otimes f\rangle \equiv F^{h\otimes f}(\tau)|GS\rangle \ : \ h \otimes f \in D(H)^*\}. \tag{3.20}$$

Then, V_τ is naturally identified with the space $D(H)^*$. Recall from Equations 2.36 and 2.37, $D(H)$, as a Hopf algebra, has two natural representations on $D(H)^*$ denoted by L and R, where L is induced from the left multiplication of $D(H)$ on itself and R is induced from the right multiplication (precomposed by the antipode). Apparently, these two actions commute with each other.

Let τ be a ribbon of either local orientation with $s_\mathrm{i} = \partial_\mathrm{i}\tau$. Identify V_τ with $D(H)^*$. Then the local operators $B_t(s_0)A_a(s_0)$ define a representation of $D(H)$ on V_τ isomorphic to L, and $B_t(s_1)A_a(s_1)$ define a representation isomorphic to R.

Proof. The statement can be proved by restricting the identities in Equations 3.17 and 3.18 on the ground state. It is straightforward to see that, at s_0, the two identities in Equations 3.17a and 3.17b corresponding to the two cases of local orientations both reduce to,

$$A_a(s_0)|h \otimes f\rangle = \sum_{(a)} |a' h S(a'''), f\,[S(a'')?]\rangle, \tag{3.21}$$

which agrees with Equation 2.36, the action L on $D(H)^*$:

$$L(a)|(h \otimes f)\rangle = |(h \otimes f)(S(a)?)\rangle = \sum_{(a)} |a' h S(a''') \otimes f\,[S(a'')?]\rangle. \tag{3.22}$$

Similarly, at s_1, for either local orientation we have

$$A_a(s_1)|h \otimes f\rangle = |h, f(?a)\rangle, \tag{3.23}$$

which agrees with Equation 2.37, the action R on $D(H)^*$:

$$R(a)|h \otimes f\rangle = |(h \otimes f)(?a)\rangle = |h \otimes f(?a)\rangle. \tag{3.24}$$

We leave the verification for the actions of $B_f(s_0)$ and $B_f(s_1)$ as an exercise. □

To summarize, ribbon operators on a sufficiently long ribbon τ commute with all terms in the Hamiltonian except those associated with the ends of τ. Hence, ribbon operators create excitations only at the ends of a ribbon. When acting on the ground state, the space of ribbon operators on τ is naturally identified with $D(H)^*$. The action of local operators on $\partial_{\rm i}\tau$ preserve $D(H)^*$. Thus, $D(H)^*$ can be thought of as the space of elementary excitations. More specifically, the action on $\partial_0\tau$ define a representation of $D(H)$ on $D(H)^*$ coinciding with L, and that on $\partial_1\tau$ a representation of $D(H)$ on $D(H)^*$ coinciding with R. These two actions commute. By standard representation theory (see Equation 2.34), we have the decomposition,

$$D(H)^* \simeq \bigoplus_{\mu \in \mathrm{Irr}_{D(H)}} \mu \otimes \mu^*, \tag{3.25}$$

where L acts on the first factor and R acts on the second factor. Therefore, the local operators on the ends of τ can map a state in a sector $\mu^* \otimes \mu$ to any other state within the same sector, but cannot permute states of different sectors. This implies that the types of elementary excitations are labelled by irreducible representations of $D(H)$. Using Fourier transformation, it is not hard to find a specific basis $\{\langle \nu ab| : \nu \in \mathrm{Irr}_{D(H)}, a, b = 1, \cdots, dim(\nu)\}$ of $D(H)^*$ so that L acts only on the a index and R acts only on the b index (See Appendix A.6). That is, for $m \in D(H)$,

$$L(m)(\langle \nu ab|) = \sum_k D^\nu(m)_{ka} \langle \nu kb|, \tag{3.26}$$

$$R(m)(\langle \nu ab|) = \sum_k D^{\nu^*}(m)_{kb} \langle \nu ak|. \tag{3.27}$$

3.5 Conclusion and outlook

In this chapter, we provided a concrete definition of ribbon operators in the generalized Kitaev quantum double model, which is constructed over a semisimple Hopf algebra. We introduced the notion of local orientation on ribbons which we must distinguish in defining the operators on them. It was shown that even in the original Kitaev model based on non-Abelian groups, the issue of local orientation has to be addressed. Otherwise, certain properties of ribbon operators that are expected to hold would fail. We derived some properties of ribbon operators in the generalized model. For instance, they create quasi-particle excitations only at the end of the ribbon, and the types of the quasi-particles correspond to irreducible representations of the Drinfeld double of the input Hopf algebra. While these properties are a folklore, their derivations are technically complicated.

There are several future directions to proceed. Firstly, since this Hopf-algebra-model can be further replaced by a weak Hopf algebra (or quantum groupoid) [12], it will be interesting to define and study ribbon operators in that case. Secondly, the generalized Kitaev model may find applications in topological quantum computing. For example, which Hopf algebras support universal quantum computing? Lastly, the authors in [46] gave a Hamiltonian formulation for gapped boundaries in the original Kitaev model. It will be interesting to generalize the formulation to the case of Hopf algebras.

4. QUANTUM CIRCUITS FOR TORIC CODE AND X-CUBE FRACTON MODEL

This chapter contains work from the article entitled "Quantum circuits for toric code and X-cube fracton model" written by the author, Bowen Yan, and Shawn X. Cui published on Quantum [2].

In toric code, the Hamiltonian consists of two types of operators, the term A_v for each vertex v and the term B_p for each plaquette p. The key idea of constructing the ground state in [22] is as follows. Start with the product state $|\phi_0\rangle = \otimes|0\rangle$ which is the $+1$ eigenstate for all vertex terms. The ground state is then obtained by projecting $|\phi_0\rangle$ to the $+1$ eigenstate of all plaquette operators, that is,

$$|GS\rangle \sim \prod_p \frac{1+B_p}{2}|\phi_0\rangle. \tag{4.1}$$

The effect of $\frac{1+B_p}{2}$ acting on certain states can be simulated by an appropriate combination of the Hadamard gate and the CNOT gate. For this method to work, the control qubit for CNOT has to be in the $|0\rangle$ state prior to applying the Hadamard and CNOT. Hence, it is critical to choose the right sequence for the plaquettes so that, immediately before simulating the term corresponding to each plaquette p, there is always at least one edge on the boundary of p with the state $|0\rangle$.

Moreover, we also adapt this method to 3D phases including the 3D toric code and the X-cube fracton model. For the X-cube model, we again initialize the state to the product of $|0\rangle$ state and simulate the projectors corresponding to cube terms. A similar issue arises that we need to choose the correct sequence to simulate the cube terms. We note that the circuit we provide here realizes an *exact* ground state of the X-cube model. By comparison, using cluster states and measurements, the authors in [28] gave an *approximate* realization of the model.

4.1 Single plaquette

To systematically introduce our ground state simulation method, we initiate with the most elementary scenario: applying $\frac{1+B_p}{2}$ on a single plaquette, which is the *basic structure* in 2D toric code. A Hadamard gate H is naturally described by $\frac{X+Z}{\sqrt{2}}$, and CNOT gate $C_{\mathrm{i}\to\mathrm{j}}$ is defined as

$$C_{\mathrm{i}\to\mathrm{j}}|\mathrm{ij}\rangle = (\frac{1-Z_{\mathrm{i}}}{2}X_{\mathrm{j}} + \frac{1+Z_{\mathrm{i}}}{2})|\mathrm{ij}\rangle, \tag{4.2}$$

where i is the control qubit and j is the target qubit.

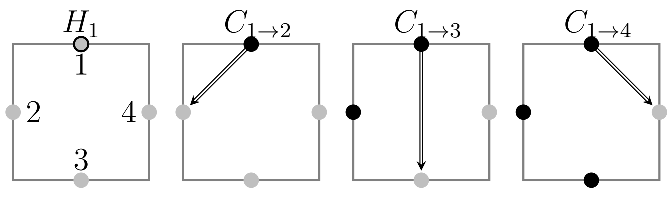


Figure 4.1. Initially, a qubit in the state $|0\rangle$ is situated at each gray dot. As quantum gates are applied to these qubits, their color changes to black. A circle positioned on a dot signifies the application of a Hadamard gate to the corresponding qubit, while an arrow indicates a CNOT gate, with the arrowhead pointing from the control qubit to the target qubit.

In the single plaquette shown in Figure 4.1, four qubits labeled $1, 2, 3$, and 4 are initialized to the state $|0\rangle$. Subsequently, we will systematically implement Hadamard and CNOT gates in a specific sequence, as outlined in the figure. After the application of H_1 and $C_{1\to2}$, we have

$$C_{1\to2}H_1|0000\rangle = (\frac{1-Z_1}{2}X_2 + \frac{1+Z_1}{2})\frac{X_1+Z_1}{\sqrt{2}}|0000\rangle = \frac{X_1X_2+1}{\sqrt{2}}|0000\rangle. \tag{4.3}$$

Explicitly, we can insert a $\frac{1+Z}{2}$ into the equation as $\frac{1+Z}{2}|0\rangle = |0\rangle$:

$$(\frac{1-Z_1}{2}X_2 + \frac{1+Z_1}{2})\frac{X_1+Z_1}{\sqrt{2}}\frac{1+Z_1}{2} \tag{4.4}$$

$$=\frac{1}{\sqrt{2}}\{(X_1\frac{1+Z_1}{2}X_2 + X_1\frac{1-Z_1}{2}) + (\frac{1-Z_1}{2}X_2 + \frac{1+Z_1}{2})\}\frac{1+Z_1}{2} \tag{4.5}$$

$$=\frac{1}{\sqrt{2}}(X_1X_2+1)\frac{1+Z_1}{2} \tag{4.6}$$

Notice $\frac{1+Z_1}{2}$ survives within { } in Equation 4.5. After reverting to the original expression and substituting $\frac{1+Z_1}{2}$ with 1, we verified the accuracy of Equation 4.3. Importantly, this equation holds for any quantum state $|\phi\rangle$:

$$C_{1\to 2}H_1|0\rangle \otimes |\phi\rangle = \frac{X_1X_2+1}{\sqrt{2}}|0\rangle \otimes |\phi\rangle, \tag{4.7}$$

since the key step only requires that the initial state must be the eigenstate of Z_1 with an eigenvalue $+1$. Finally, applying the other CNOT gates results in

$$\prod_{\mathrm{i}=2}^{4} C_{1\to\mathrm{i}}H_1|0000\rangle = \frac{X_1X_2X_3X_4+1}{\sqrt{2}}|0000\rangle = \frac{1+B_p}{\sqrt{2}}|0000\rangle, \tag{4.8}$$

which is the desired ground state. It is important to observe that this procedure remains effective as long as a qubit from $B_o(p)$ is initially in the state $|0\rangle$. We term such qubits as *free qubits*, and their presence is pivotal when considering scenarios involving multiple plaquettes.

4.2 Developing to a surface with boundary

Given a complicated lattice Γ in the state $|\phi_0\rangle$, we need to find a path (termed permissible order in [26]) through all plaquettes p_i, such that $\bigcup_\mathrm{i} p_\mathrm{i} = P$, using a sequence of edges $\mathrm{e_i} \in Bo(p_\mathrm{i})$ where $\mathrm{e_i} \notin \bigcup_{\mathrm{j}=1}^{\mathrm{i}-1} p_\mathrm{j}$. Each $\mathrm{e_i}$ is then utilized as a free qubit to apply the introduced basic structure, resulting in the accumulation of $\prod_\mathrm{i} \frac{1+B_{p_\mathrm{i}}}{\sqrt{2}}$ over $|0\cdots 0\rangle$, which represents the ground state of the toric code on lattice Γ. To illustrate the procedure, we take four plaquettes as an example depicted in Figure 4.2. A path featuring four free qubits e_1 to e_4 has been chosen, where $\mathrm{e_i}$ starts in the state $|0\rangle$ at the onset of every step. Upon completing the path, the desired ground state is eventually obtained.

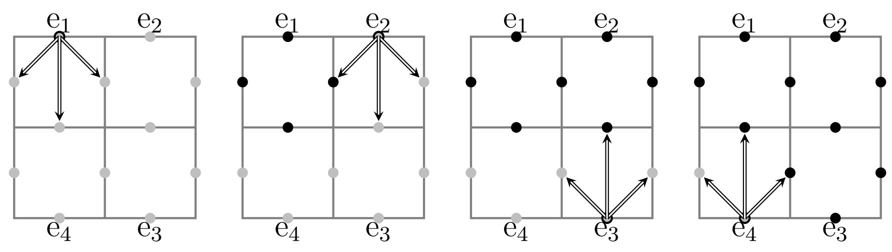


Figure 4.2. The procedure on the basics structure is applying Hadamard gate on any qubit at $|0\rangle$ first and CNOT gates to other qubits in any order.

4.3 Developing to a surface without boundary

The scenario shifts when dealing with a surface without boundary. While the initial state remains $|\phi_0\rangle$, it becomes impossible to locate a path with sufficient free qubits to cover the entire lattice. Fortunately, as every edge sides two plaquettes, the equation holds:

$$\prod_{p \in P} B_p = 1, \tag{4.9}$$

which implies that we can intentionally choose a specific B_p as redundant. Consequently, we can select the final plaquette as the redundant one, effectively terminating the path. To illustrate, consider the lattices on a torus shown in Figure 4.3, there is no need to apply Hadamard and CNOT gates to the bottom left plaquette, as we have previously simulated the toric code's ground state.

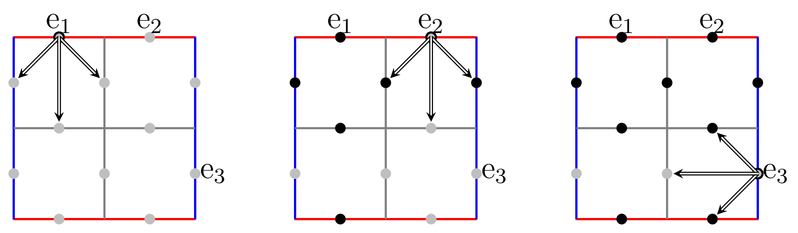


Figure 4.3. Boundaries with the same color are identified to represent a torus.

This method remains applicable to more intricate 2D surfaces with or without boundary, provided a suitable path can be identified. Additional examples are provided in Appendices B.1 and B.2. Furthermore, the gluing method detailed later empowers us to simulate ground states on arbitrary planar lattices.

4.4 Simulate arbitrary ground state

As stated in [9], the degeneracy of ground states for 2D toric code on torus is four: $|00\rangle$, $|01\rangle$, $|10\rangle$ and $|11\rangle$. The ground state $|00\rangle$, presented in Section 4.3, is simulated from the initial state ϕ_0. Due to the properties of logical operators, which can interchange ground states and commute with B_p, it is feasible to apply them to ϕ_0 to obtain the remaining ground states.

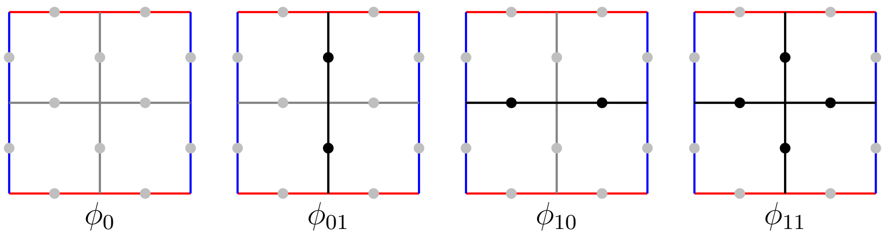


Figure 4.4. A qubit $|0\rangle$ is placed at each gray dot and the color changes to black when operator X flips the qubit from $|0\rangle$ to $|1\rangle$.

Illustrated in Figure 4.4, a vertical loop and a horizontal loop of X represent the two logical operators. They are capable of transforming ϕ_0 into ϕ_{01}, ϕ_{10} and ϕ_{11}, which correspond to the initial states of $|01\rangle$, $|10\rangle$ and $|11\rangle$, respectively. Subsequently, we can repeat the same procedure detailed in Section 4.3, but with a modification: utilize $X_{\mathrm{i}}C_{\mathrm{i}\to\mathrm{j}}X_{\mathrm{i}}$ instead of $C_{\mathrm{i}\to\mathrm{j}}$ when encountering a flipped qubit e_{i}.

One step further, in order to obtain an arbitrary ground state $\Phi = a\mathrm{e}^{\mathrm{i}\theta_a}|00\rangle + b\mathrm{e}^{\mathrm{i}\theta_b}|01\rangle + c\mathrm{e}^{\mathrm{i}\theta_c}|10\rangle + d\mathrm{e}^{\mathrm{i}\theta_d}|11\rangle$, we can implement the unitary operator U outlined in Equation 4.10 on an adjacent pair of vertical and horizontal edges of ϕ_0 and subsequently utilize CNOT gates to transmit vertically and horizontally to get ϕ. From there, we can proceed by repeating the aforementioned method and we must avoid qubits that have already been utilized, opting instead for free qubits.

$$U_1 = \begin{pmatrix} \frac{a}{\sqrt{a^2+b^2}} & \frac{-b}{\sqrt{a^2+b^2}} & 0 & 0 \\ \frac{b}{\sqrt{a^2+b^2}} & \frac{a}{\sqrt{a^2+b^2}} & 0 & 0 \\ 0 & 0 & \frac{c}{\sqrt{c^2+d^2}} & \frac{-d}{\sqrt{c^2+d^2}} \\ 0 & 0 & \frac{d}{\sqrt{c^2+d^2}} & \frac{c}{\sqrt{c^2+d^2}} \end{pmatrix} \begin{pmatrix} \sqrt{a^2+b^2} & 0 & -\sqrt{c^2+d^2} & 0 \\ 0 & \sqrt{a^2+b^2} & 0 & -\sqrt{c^2+d^2} \\ \sqrt{c^2+d^2} & 0 & \sqrt{a^2+b^2} & 0 \\ 0 & \sqrt{c^2+d^2} & 0 & \sqrt{a^2+b^2} \end{pmatrix},$$

$$U_2 = \begin{pmatrix} e^{i\theta_a} & 0 & 0 & 0 \\ 0 & e^{i\theta_b} & 0 & 0 \\ 0 & 0 & e^{i\theta_c} & 0 \\ 0 & 0 & 0 & e^{i\theta_d} \end{pmatrix} \qquad \textit{and} \qquad U = U_2 U_1. \tag{4.10}$$

4.5 Quantum circuit depth

To simulate a toric code with length L, using local unitary gates requires at least linear size $O(L)$ depth circuits [47], and constant depth is achievable if measurement operations are allowed [29]. A recent work provided a systematic method to simulate an unknown toric code in $3L + 2$ depth [48], [49]. In comparison, on a $L \times L$ square lattice over a torus, we can simulate a known toric code state like $|00\rangle$ in $2L + 2$ depth and an unknown toric code Φ in $\lceil 2L + 2 + log_2(d) + \frac{L}{2d} \rceil$ depth. Here, the quantum circuit is local and the $CNOT$ gate is restricted to be applied on two qubits with a distance d.

To simulate the state $|00\rangle$, we initiate the process with ϕ_0 and designate the plaquette at the bottom right corner as redundant. Subsequently, we proceed the quantum gates step by step, following the instructions outlined on the left side of Figure 4.5. On the other hand, as elaborated upon in Section 4.4, an unknown toric code state Φ can be attained by substituting ϕ_0 with ϕ, which is obtained from two logical qubits through a sequence of CNOT gates. This procedure demands $\lceil log_2(d) + \frac{L}{2d} \rceil$ steps [1], where d represents the maximum distance between the two qubits that CNOT gate could apply without breaking locality. Additionally, a slight variation in the order, as demonstrated on the right side of Figure 4.5, is essential to initiate with ϕ.

[1]↑For detailed discussion on the local CNOT gate, see Section B.3.

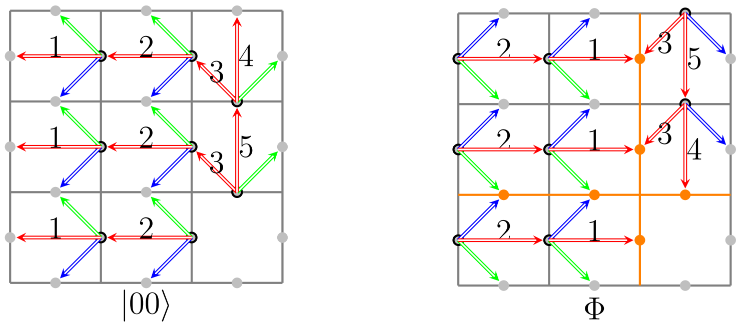


Figure 4.5. Opposite boundaries are identified and the case of L=3 is provided as an example. In both figures, the prescribed order for gate application is as follows: 1 Apply Hadamard gates to the qubits encircled by circles; 2 Execute CNOT gates, indicated by green arrows, followed by those with blue arrows; 3 Implement CNOT gates denoted by red arrows, following their numerical order. In the right figure, orange dots signify qubits that hold the information of Φ.

4.6 Gluing method for two single plaquettes

The method introduced in the preceding sections is efficient; however, it hinges on the selection of a suitable path. This choice could prove challenging for intricate surfaces. To address this concern, we propose a gluing method designed to overcome this complexity. To exemplify the essence of the gluing approach, we will commence with a straightforward example. To simulate the ground state of toric code on the two plaquettes in Figure 4.6, we can employ an ancilla qubit to partition it into two independent plaquettes p_1 and p_2. The edges within $Bo(p_1)$ are denoted by $1, 2, 3, 4$, while those within $Bo(p_2)$ are denoted by $5, 6, 7, 8$. We initiate the process with ϕ_0 and ignore the overall normalization constant to simplify subsequent calculations.

First apply $1 + B_{p_1}$ and $1 + B_{p_2}$ independently to get

$$(1 + X_1X_2X_3X_4)(1 + X_5X_6X_7X_8)|00...0\rangle. \tag{4.11}$$

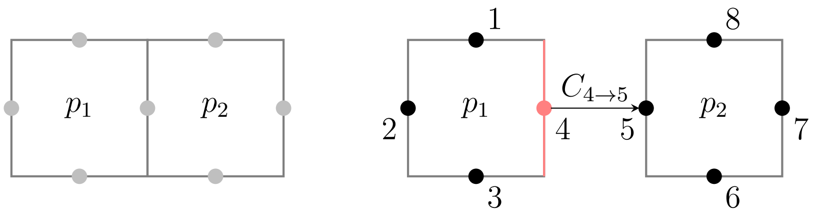


Figure 4.6. The lattice of two plaquettes is divided into two independent plaquettes by introducing the ancilla qubit in red.

Then apply $C_{4\to5}$ and notice this operator commutes with $1 + B_{p_2}$:

$$
\begin{aligned}
&(\frac{1-Z_4}{2}X_5 + \frac{1+Z_4}{2})(1 + X_1X_2X_3X_4)(1 + X_5X_6X_7X_8)|00...0\rangle \\
&= (1 + X_1X_2X_3X_4X_5)(1 + X_5X_6X_7X_8)|00...0\rangle. \qquad (4.12)
\end{aligned}
$$

Finally, make a measurement over the ancilla qubit with basis $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $|-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$. If we get +1, it is equivalent to applying $\frac{1+X_4}{2}$ and thus

$$
\begin{aligned}
&\frac{1+X_4}{2}(1 + X_1X_2X_3X_4X_5)(1 + X_5X_6X_7X_8)|00...0\rangle \\
&= (\frac{1+X_4}{2})(1 + X_1X_2X_3X_5)(1 + X_5X_6X_7X_8)|00...0\rangle. \qquad (4.13)
\end{aligned}
$$

The ancilla qubit is disentangled, leaving us with the ground state of the two plaquettes. Observing that, when two boundaries $\mathrm{e_i}$ and $\mathrm{e_j}$ are glued together, all plaquettes terms commute with each other and $C_{\mathrm{i}\to\mathrm{j}}$ commutes with all plaquette terms except $1 + B_{p_k}$ where $\mathrm{e_i} \in Bo(p_k)$. This observation underscores that the resultant combination remains a ground state even when multiple plaquettes are fused together concurrently.

On the other hand, if we get -1, it is equivalent to applying $\frac{1-X_4}{2}$ and thus

$$
\begin{aligned}
&\frac{1-X_4}{2}(1 + X_1X_2X_3X_4X_5)(1 + X_5X_6X_7X_8)|00...0\rangle \\
&= (\frac{1-X_4}{2})(1 - X_1X_2X_3X_5)(1 + X_5X_6X_7X_8)|00...0\rangle, \qquad (4.14)
\end{aligned}
$$

which is not the expected ground state. It is worth noting that the resulting state is an excited state if a magnetic charge exists at p_1. Fortunately, we can correct it by applying Z_1, Z_2 or Z_3, each of which is a short dual ribbon operator. In the subsequent section, we will establish a proof demonstrating that a correcting operator invariably exists for any planar lattice.

Our method, can be naturally extended to more general scenarios where projectors only involve tensor products of Pauli X (given that tensor products of Pauli Z operators are automatically satisfied by the state $|00...0\rangle$), such as 3D toric model or X cube model to be addressed below. For instance, let us consider two edges from different lattices, labeled as m and n (note that we abuse the notation referring to both edges and lattices). The state of these two lattices can be expressed as $\prod \frac{1+H_m}{2}|0\rangle_m \otimes |0\rangle_{res_m}$ or $\prod \frac{1+H_n}{2}|0\rangle_n \otimes |0\rangle_{res_n}$. Here, $res_{m(n)}$ signifies the remaining system of lattice $m(n)$, and $H_{m(n)}$ denotes the projector onto lattice $m(n)$. Given that $C_{m\to n}$ only relies on $|m\rangle$, expanding the product of H_m yields

$$C_{m\to n}\sum_{\mathrm{i}}(X\otimes 1\otimes A_{\mathrm{i}}+1\otimes 1\otimes B_{\mathrm{i}})\prod\frac{1+H_n}{2}|0\rangle_m|0\rangle_n|0\rangle_{res_m}|0\rangle_{res_n}, \tag{4.15}$$

where A_{i} and B_{i} acts only on res_m, and we have not expanded H_n since it has a trivial impact on m. Upon applying $C_{m\to n}$, we obtain

$$\sum_{\mathrm{i}}(X\otimes X\otimes A_{\mathrm{i}}+1\otimes 1\otimes B_{\mathrm{i}})\prod\frac{1+H_n}{2}|0\rangle_m|0\rangle_n|0\rangle_{res_m}|0\rangle_{res_n}. \tag{4.16}$$

Essentially, this signifies that the CNOT gate transfers all actions from m to n after disentangling m. Akin to Equation 4.13 and 4.14, we count the excitations and employ correction operators to obtain the ground state. Consequently, we can attain the expected ground state of the glued lattice by gluing the edges correspondingly , as long as the projectors consist of tensor products of Pauli X operators .

4.7 Gluing method for an arbitrary lattice

When transitioning from the gluing method's application on two single plaquettes to the broader context of numerous arbitrary plaquettes, our focus should not be on the edges measured +1, but rather on establishing a systematic method to correct address edges measured -1.

In the instance presented in Figure 4.7, if we apply $C_{\mathrm{i}\to\mathrm{j}}$ to glue two boundaries and get -1 after measuring qubit $\mathrm{e_i}$, the correcting operator must anti-commute with B_{p_i} and exhibit commutativity with everything else [2]. One intuitive approach is to apply a dual string operator starting at p_i and ending crossing a z-boundary.

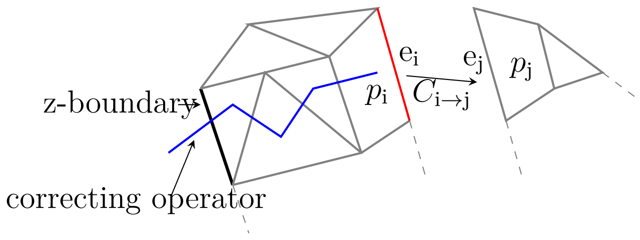


Figure 4.7. Correcting procedure after gluing two arbitrary plaquettes.

Expanding upon this notion, let us consider a situation involving any connected planar lattice $\gamma = \bigcup_{\mathrm{i}=1}^{n} p_\mathrm{i}$ with z-boundary e_0. For a series $T = p_\mathrm{i}$, $Bo(p_\mathrm{i}) \bigcap \bigcup_{\mathrm{j}=1}^{\mathrm{i}-1} Bo(p_\mathrm{j}) \neq \oslash$ for any $\mathrm{i} \in [2, n]$, we can insert ancilla qubits to separate T into multiple plaquettes and subsequently glue them back. To illustrate this idea, let us delve into an example consist of four plaquettes, as depicted in Figure 4.8.

First, we initiate by utilizing ancilla qubits to fragment the lattice into single plaquettes. For $\tau(\mathrm{e}_k)$ containing p_i and p_j, where $1 \leq \mathrm{i} < \mathrm{j} \leq n$ according to the series T, insert an ancilla edge e'_k into p_i while retaining e_k in p_j. Then we apply $1 + B_p$ to every single plaquette $p \in P$. Subsequently, glue them together piece by piece. For $p_\mathrm{i}, 1 < \mathrm{i} \leq n$, it becomes necessary to apply $C_{\mathrm{e}'\to\mathrm{e}}$ to all pairs of $\mathrm{e}' \in \bigcup_{\mathrm{j}=1}^{\mathrm{i}-1} Bo(p_\mathrm{j})$ and $\mathrm{e} \in Bo(p_\mathrm{i})$. Finally, we measure and disentangle e'. If we get -1, apply a dual string operator connecting p_i and the z-boundary of p_1 to correct it. It is noteworthy that all plaquettes can be glued simultaneously, allowing a dual string operator to annihilate two magnetic charges by connecting them. In this example, if we get -1 for e'_1 and e'_4 concurrently, the correcting operator will effectively nullify their impact.

[2]↑It is worth noting that this correcting operator effortlessly commutes with all vertex terms.

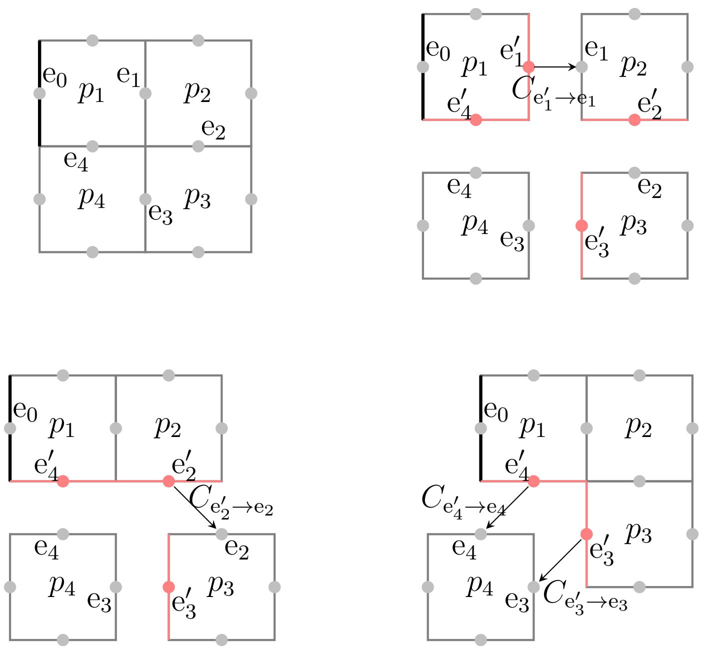


Figure 4.8. e$_0$ is a z-boundary and e$'$ in red represents an ancilla qubit.

In the case of a lattice without boundary, we can choose a specific plaquette p to be redundant, effectively transforming $Bo(p)$ into z-boundaries. Subsequently,this situation mirrors the scenario depicted in the lattice with boundaries, and further details are left for readers to explore. If the lattice solely contains x-boundaries, a viable solution is to consider the dual lattice of it. The process remains unchanged, except for the inversion of plaquette and vertex operators. Thus we can confidently assert that our method is universally capable of simulating the ground state of a toric code on any planar lattice configuration.

In the case of the 2D toric code, the gluing method might initially resemble a simple measurement process, especially when we break down the lattice into pieces, attach ancilla qubits, and then fuse them to obtain the ground state for the entire lattice. However, its capabilities extend significantly when we consider scenarios like 3D models, as discussed thoroughly in Section 4.10, or when we have two lattices in their ground states to be joined. In such cases, a stabilizer measurement can not glue two lattice and get the ground state of the glued lattice.

4.8 3D toric model

The 3D toric model bears strong resemblance to the 2D toric code and is established on an arbitrary 3D lattice. To enhance clarity, a cubic lattice is adopted, as depicted in Figure 4.9. Within this lattice, V represents the set of all vertices, while P corresponds to the set of all plaquettes; each edge accommodates a single qubit. Moreover, for the sake of convenience, we have affixed labels to each edge, denoting them as x, y, or z based on their alignment with the respective axis (i.e., parallel to the x, y, or z axis). Notice this labeling maintains consistency even when applied to a 3-dimensional torus. The Hamiltonian is defined as

$$H = -\sum_{v\in V} A_v - \sum_{p\in P} B_p, \tag{4.17}$$

where A_v pertains to the application of the Pauli operator X over six edges connected to the vertex v, and B_p pertains to the application of the Pauli operator Z over the four edges encompassing the plaquette p. It is straightforward to see these new-defined A_v and B_p operators also satisfy $A_v^2 = B_p^2 = 1$ and $[A_v, B_p] = 0$. So this 3D toric Hamiltonian is equivalent to the equation expressed as a summation of local projectors. We get the ground state

$$|GS\rangle = \prod_v \frac{1+A_v}{2}|\phi_0\rangle, \tag{4.18}$$

where $|\phi_0\rangle = |00...0\rangle$, and we drop $\frac{1+B_p}{2}$s since its action on $|\phi_0\rangle$ is $+1$. It is important to highlight that the constancy of ground state degeneracy endures with fluctuations in system size, a pivotal characteristic of topological phases of matter. Additionally, Figure 4.9 presents a comprehensive depiction of a pair of conjugate logical operators. Notice that, the definition of logical operators only depends on the nontrivial loop or non-cotractable planes. Consequently, we have three pairs of conjugate logical operators, each acting on edges labeled by x, y, or z respectively.

We can extend the method of 2D toric code to 3D toric model with boundary directly utilizing a plaquette as the basic structure. It is complicated yet straightforward, so its details are outlined in Section B.4. However, applying this approach to the 3D toric model without boundaries presents challenges, as the absence of free edges in the final step poses an

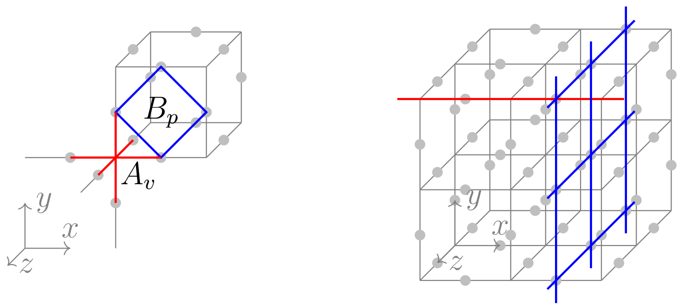


Figure 4.9. The left sub-figure illustrates the definitions of A_v and B_p operators. Meanwhile, the right sub-figure displays a pair of conjugate logical operators composed of edges labeled by x. The red string is a nontrivial circle parallel to x axis, and a logical Z operator is to apply Pauli Z over edges along the string. Conversely, the blue plane is a non-contractable plane perpendicular to x axis, and a logical X operator is to apply Pauli X over edges within the plane.

issue. To circumvent this challenge, we must adopt a basic structure, illustrated in Figure 4.10. We still initiate the process with $|\phi_0\rangle$. Then we execute the quantum circuits as illustrated in the figure to achieve the action of $\frac{1+A_v}{2}$.

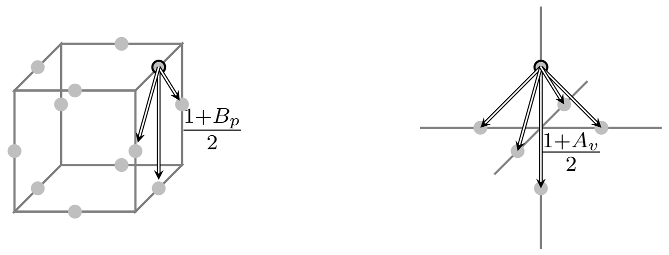


Figure 4.10. Comparison of two different basic structures: An example consisting of eight cubes with boundary is shown in Section B.4 and a similar example without boundary is shown in Section B.5.

Using this basic structure to develop the lattice vertex by vertex, we will end with a redundant vertex as

$$\prod_{v\in V} A_v = 1. \tag{4.19}$$

In Section B.5, we present a straightforward example comprising eight cubes to illustrate the method. To address the general case, we delineate the procedures required for constructing a quantum circuit for the 3D toric model on an $L \times L \times L$ lattice over a 3-dimensional torus in Figure 4.11. The process consists of several carefully orchestrated steps to efficiently realize the circuit, amounting to a total of $3L+8$ steps. The quantum circuit is purely local since all applied quantum gates are either acting on a single qubit or on nearest two qubits. Certain non-interacting gates offer the potential for further parallelization, but this would only result in a constant difference in circuit depth.

In a manner akin to the procedure detailed in Section 4.4, we employ certain qubits to generate a particular initial state that encodes information about the logical qubits, as depicted in Figure 4.12. However, it is important to note that these selected qubits are unnecessary when we opt for free qubits during the ground state preparation. In conclusion, our method can prepare an arbitrary ground state of the 3D toric model with linear depth.

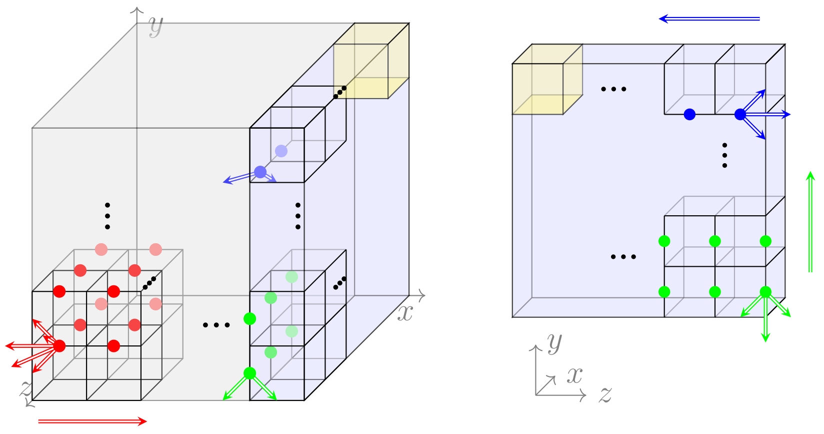


Figure 4.11. We initiate with slicing the 3D torus into layers along the x-direction and applying H gates to all the colored dots. Subsequently, we apply all CNOT gates from red dots to non-red dots simultaneously and between adjacent red dots layer by layer, which requires $L+3$ steps. The last layer needs special treatment, which simplifies into a 2D problem after applying 2 CNOT gates from all green and blue dots. Further progression involves applying CNOT gates concurrently from green dots to non-green dots and between adjacent green dots row by row, necessitating $L+1$ steps. Similarly, another $L+1$ steps applying CNOT gates from blue dots completes the procedure and leaves a redundant vertex in the yellow cube.

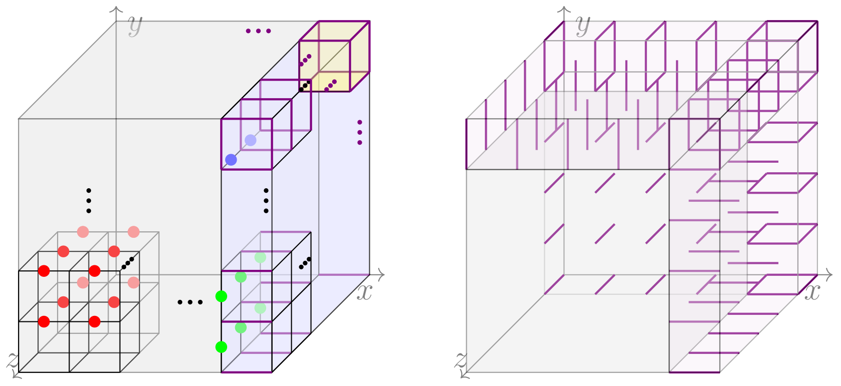


Figure 4.12. The left sub-figure highlights the employed qubits (indicated by violet edges) that are utilized for the specific initial state within the ground state preparation process. Meanwhile, the right sub-figure sketches all employed edges within a cubic lattice of size $L = 5$. All edges labeled by z in the back layer, x in the right layer, and y in the top layer are employed to encode arbitrary ground states. Importantly, the procedure we introduced earlier remains uninterrupted since we do not designate any of these edges as free qubits.

4.9 X-cube model

The X-cube model is a fracton model defined on a 3D cubic lattice, as visually depicted in Figure 4.13. Within this lattice, V represents the set of all vertices, while C corresponds to the set of all cubes; each edge accommodates a single qubit. For the sake of convenience, we also affixed labels to each edge, denoting them as x, y, or z based on their alignment with the respective axis. The Halmitonian is defined as

$$H = -\sum_{v\in V} A_v^x + A_v^y + A_v^z - \sum_{c\in C} B_c, \tag{4.20}$$

where $A_v^{\mathrm{i}}, \mathrm{i} = x, y, z$ is defined to implement Pauli operator Z across the four edges oriented vertically to the i axis and attached to vertex v, and B_c is designated to effectuate Pauli operator X across the twelve edges associated with cube c. Again these A_v^{i}s and B_cs operators satisfy $(A_v^{\mathrm{i}})^2 = B_c^2 = 1$ and $[A_v^{\mathrm{i}}, B_c] = 0$. We get the ground state

$$|GS\rangle = \prod_c \frac{1 + B_c}{2} |\phi_0\rangle, \tag{4.21}$$

where $|\phi_0\rangle = |00...0\rangle$, and we drop $\frac{1+A_v^{\mathrm{i}}}{2}$ since its action on $|\phi_0\rangle$ is $+1$. It is important to emphasize, however, that the ground state degeneracy experiences exponential growth alongside the system size. Additionally, Figure 4.13 presents a comprehensive depiction of a pair of logical operators of type W and T[3]. Similarly, we have three types of logical operator pairs, each acting on edges labeled by x, y, or z respectively. Notably, distinct non-trivial loops exhibit identical homotopy while differ in terms of logical operators. This distinction is a crucial hallmark distinguishing the fracton model from conventional topological orders.

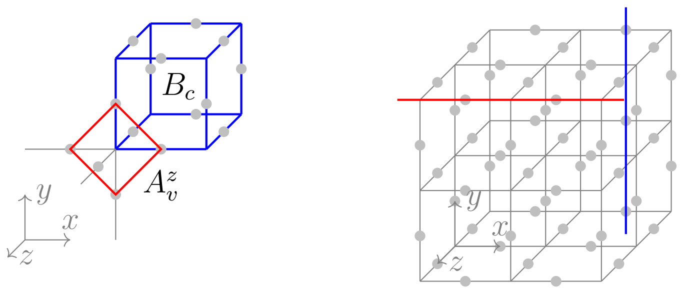


Figure 4.13. The left sub-figure illustrates the definitions of A_v and B_p operators. Meanwhile, the right sub-figure displays a pair of conjugate logical operators composed of edges labeled by x. The red string is of type W, a nontrivial circle parallel to x axis, and we apply Pauli X over edges along the string. Conversely, the blue string is of type T, a nontrivial circle perpendicular to x axis, and we apply Pauli Z over edges along the string.

To simulate the ground state for the X-cube model, we outline[4] the procedures required to construct a quantum circuit for the X-cube model on an $L \times L \times L$ lattice, over a 3-dimensional torus, in Figure 4.14. The initial state is $|\phi_0\rangle$, and our target is to find quantum

[3]↑The complete set of logical operators are given in [50], but we only use two types of them, which are not conjugate to each other.

[4]↑We also present a straightforward example comprising eight cubes to illustrate the method, as elaborated in Section B.6.

circuit to implement $\prod_c \frac{1+B_c}{2}$. We identify the redundancy by specifically selecting certain cubes, namely the three edges of the cubes in yellow, resulting from the requirement $\prod B_c = 1$ of the involved layer of cubes. Given that layers can be independently sliced in three distinct directions, this selective arrangement of yellow-colored structures [5] is achieved. To start, we strategically partition the cube into distinct components: a central $(L-1)\times(L-1)\times(L-1)$ cube (colored gray), three $(L-1)\times(L-1)\times 1$ layers of cubes (colored blue), and three rows of redundant cubes (colored yellow). Then we further slice the central cube into $(L-1)\times(L-1)\times 1$ layers. Notice each gray and blue layer has the same boundaries up to rotation. This occurs because those yellow cubes are redundant, and the blue cubes are intended for the application of projectors in other layers. Neither of them interferes with the preparation of the layer structure. Consequently, we treat each layer of cubes as having the same structure, and their corresponding quantum circuits are outlined in Figure 4.14.

After the initial 9 steps, we apply CNOT from (i,j) to $(\mathrm{i}-1,\mathrm{j})$ in the $(3k+10)$-th step, apply CNOT from (i,j) to $(\mathrm{i},\mathrm{j}-1)$ in the $(3k+11)$-th step, and apply CNOT from (i,j) to $(\mathrm{i}-1,\mathrm{j}-1)$ in the $(3k+12)$-th step, where $\mathrm{i}+\mathrm{j}=k+2$. This allows us to complete the layer structure in a total of $6L+6$ steps. These carefully orchestrated steps efficiently realize the circuit, requiring a total of $12L+11$ steps and the quantum circuit is purely local, similarly to the 3-dimensional toric code case. Certain non-interacting gates offer the potential for further parallelization, reducing the circuit depth by $2\lfloor\frac{2L-3}{2}\rfloor$. It is worth noting that this method can be readily extended to the X-cube model on a lattice of dimensions $L_1\times L_2\times L_3$.

The ground state degeneracy can be resolved by the complete set of logical operators [50]. We can readily attain all bases of the ground space of the X-cube model by replacing the initial product state, as demonstrated in Section 4.4 and 4.8. However, it is not straightforward to see whether our method can be applied to prepare arbitrary ground states of the X-cube model. The comprehensive encoding of arbitrary ground states still remains an open question and is left as a topic for future research directions.

[5]↑While there may exist more redundant cubes, our focus is solely on the chosen ones.

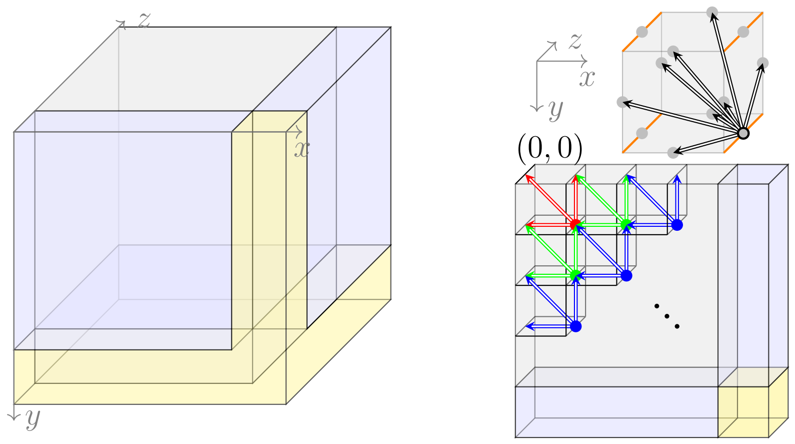


Figure 4.14. Treating each layer of cubes as the same structure with boundaries, we initiate H gates along the edges in the z-direction of each cube and apply CNOT gates from these edges to the others in the x and y directions, which requires 9 steps. Then we apply CNOT gates diagonally, row by row in different colors, necessitating $6L - 3$ steps. All gray layers are prepared simultaneously, followed by blue layers, leaving behind redundant cubes.

4.10 Gluing method for 3D models

Similar to the scenario in 2D toric code case, we can simulate the ground state of 3D toric model by breaking the lattice into basic structures, simulating on and gluing them back. This results in one redundant vertex term and the excitations are quasi-particles that are able to move freely. The situation is exactly the same as 2D toric code, so we can find correcting operators to annihilate all of the excitations, which is left to readers.

Different methods for gluing in the X-cube model exist, and an intuitive one is shown in Figure 4.15. In this method, the quantum circuits are applied to each of the individual pieces to obtain their respective ground states. Subsequently, CNOT gates are employed along the gluing plane to glue them together. It is essential to note that this process is not a simple measurement, as each edge is influenced by two cube terms. Disentanglement necessitates the implementation of measurements on all red edges and correction operators to eliminate potential excitations based on the measurement outcomes. However, the X-cube model poses

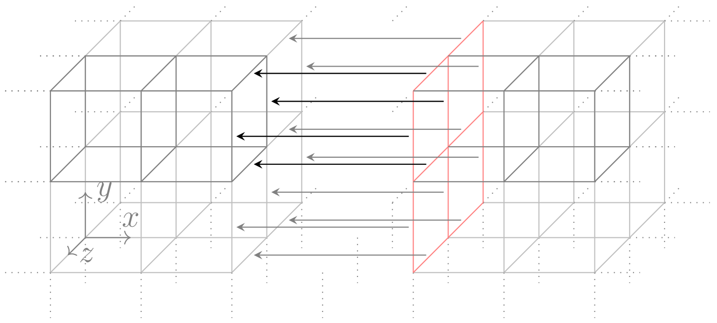


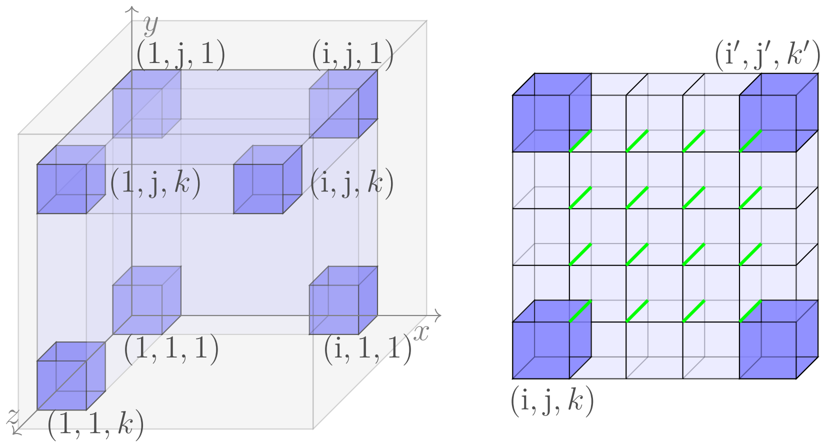
Figure 4.15. Following the preparation of ground states on the individual lattices, we designate all qubits on one side of the gluing plane as ancilla qubits (represented by red edges). Subsequently, we apply CNOT gates in parallel from these ancilla qubits to the opposite side. This process allows us to obtain the ground state of the fused lattice after appropriately disentangling the ancilla qubits.

greater complexity as the excitations are fractons. A systematic approach to find correcting operators is based on the following two facts:

1. There are three columns of redundant cubes as shown in Figure 4.14.

2. The excitation betraying cube terms is a fracton that are not able to move freely. While a *membrane operator* (see [51] for details) creates fractons on four corners of a rectangular.

Illustrated in Figure 4.16, each cube in the n^3 cubic lattice, underlying the 3D torus topology, is assigned Cartesian coordinates $(\mathrm{i}, \mathrm{j}, k)$, where $1 \leq \mathrm{i}, \mathrm{j}, k \leq n$. Redundant cubes are positioned along three columns, namely $(\mathrm{i}, 1, 1)$, $(1, \mathrm{i}, 1)$, and $(1, 1, \mathrm{i})$ for $\mathrm{i} = 1 \cdots n$. A membrane operator $\mathcal{M}[(\mathrm{i}, \mathrm{j}, k), (\mathrm{i}', \mathrm{j}', k')]$, consists of Z operators in the rectangle from $(\mathrm{i}, \mathrm{j}, k)$ to $(\mathrm{i}', \mathrm{j}', k')$ creates excitations at the four corners. For instance, when addressing an excitation at $(\mathrm{i}, \mathrm{j}, 1)$, applying $\mathcal{M}[(1, 1, 1), (\mathrm{i}, \mathrm{j}, 1)]$ leads to the annihilation of the excitation and the creation of excitations at redundant cubes, which are inconsequential. When dealing with a general excitation at $(\mathrm{i}, \mathrm{j}, k)$, with $\mathrm{i}, \mathrm{j}, k \neq 1$, a multi-step procedure comes into play. Initially, $\mathcal{M}[(1, \mathrm{j}, 1), (\mathrm{i}, \mathrm{j}, k)]$ is applied to eliminate the excitation, generating three additional excitations at $(1, \mathrm{j}, 1)$, $(1, \mathrm{j}, k)$, and $(\mathrm{i}, \mathrm{j}, 1)$. Disregarding the one at the redundant cube, the other two are subsequently eliminated by $\mathcal{M}[(1, 1, 1), (1, \mathrm{j}, k)]$ and $\mathcal{M}[(1, 1, 1), (\mathrm{i}, \mathrm{j}, 1)]$, respectively.



y
$(1, \mathrm{j}, 1)$
$(\mathrm{i}, \mathrm{j}, 1)$
$(1, \mathrm{j}, k)$
$(\mathrm{i}, \mathrm{j}, k)$
$(1, 1, 1)$
$(\mathrm{i}, 1, 1)$
x
z
$(1, 1, k)$
$(\mathrm{i}', \mathrm{j}', k')$
$(\mathrm{i}, \mathrm{j}, k)$

Figure 4.16. A membrane operator consisting of Z operators on green edges creates fractons at four corners; The correcting operator is a product of three membrane operators.

In essence, the product operator $\mathcal{M}[(1,1,1),(1,\mathrm{j},k)]\ \mathcal{M}[(1,1,1),(\mathrm{i},\mathrm{j},1)]\ \mathcal{M}[(1,\mathrm{j},1),(\mathrm{i},\mathrm{j},k)]$ is capable of annihilating general excitations.

4.11 Conclusion and outlook

In this paper, we propose a method to prepare the ground state of a Hamiltonian consisting of local commuting projectors composed solely of Pauli X and Pauli Z operators. Our approach involves finding an appropriate initial state that serves as the ground state of these projectors and applying a quantum circuit composed solely of Clifford gates to achieve the Hamiltonian's ground state. We demonstrate the effectiveness of our method on 2D toric codes with various surface conditions, both with and without boundaries, as well as on the 3D toric model and the X-cube model. Our method enables the preparation of arbitrary ground states for 2D and 3D toric model with a linear-depth circuit, meeting the lower bound for preparing ground states in topological phases. It also works for any basis of the ground state in the X-cube model using a linear-depth quantum circuit. We present these results on specific lattices, such as the 2D square lattice or 3D cubic lattice, and introduce a gluing method to facilitate ground state preparation on general 2D and 3D lattices. This gluing method provides a trade-off between measurement usage and circuit depth and can be applied to obtain the ground state of larger lattices by assembling ground states of smaller components.

There are several future directions to proceed from this work. One natural progression involves extending our method to other 3D models of interest. Furthermore, the applicability of our approach to the non-abelian Kitaev model presents a straightforward extension, offering the potential to broaden the scope of its application.

5. REPRESENTING ARBITRARY GROUND STATES OF TORIC CODE BY RESTRICTED BOLTZMANN MACHINE

This chapter contains work from the article entitled "Representing Arbitrary Ground States of Toric Code by Restricted Boltzmann Machine" written by the author, Bowen Yan, and Shawn X. Cui preprinted on arXiv [3].

5.1 Introduction

The research on topological phases of matter (TPMs) has significantly intensified in recent decades. These phases are characterized by topological order, setting them apart from conventional states. Topological phases feature ground states with stable degeneracy and robust long-range entanglement. In two dimensions, they support anyons and show resilience to local disruptions. These unique attributes render TPMs highly suitable for fault-tolerant quantum computing [9], [10]. In two dimensions, the underlying structure of TPMs can be described by either a (2+1)-dimensional topological quantum field theory or a unitary modular tensor category. Many topological phases can be realized on spin lattice models, with the toric code model standing out as one of the most notable examples. More generally, associated with each finite group G, Kitaev's quantum double model defines an exactly solvable lattice model realizing possibly non-Abelian anyons. When $G = \mathbb{Z}_2$, the theory simplifies to the toric code.

Identifying the eigenstates of the Hamiltonian of a topological phase, and more generally that of a many-body quantum system, ranks among the most demanding tasks in condensed matter physics. This task becomes increasingly complex primarily because of the power scaling of the Hilbert space dimension, which inflates exponentially in relation to the system's size [32]. Nonetheless, it is often the case that the system's inherent physical properties, e.g. long-range entanglement, restrict the form of the ground states, and therefore the states corresponding to interesting quantum systems may only occupy a small portion of the exponentially large Hilbert space. This opens up the possibility of efficient representations of the wave function of many-body systems. Examples of efficient representations include matrix product states, projected entangled pair states, and more generally tensor networks.

A recent trend is the study of many-body quantum systems utilizing machine learning techniques, especially artificial neural networks. Restricted Boltzmann Machines (RBMs) are a generative stochastic artificial neural network [33]. Unlike other types of neural networks, RBMs have a unique two-layer architecture that consists of a visible input layer and a hidden layer. The 'restricted' part in the name refers to the lack of intra-layer connections; that is, nodes within the same layer do not interact with each other. RBMs have been used effectively in a variety of machine learning tasks, including dimensionality reduction, classification, regression, and even solving quantum many-body problems [34]–[39].

In 2017, Carleo and Troyer paved a novel path by applying RBM as a variational ansatz, utilizing it to represent ground states for Ising model [34]. This groundbreaking achievement catalyzed the development of numerous explicit RBM representations. Notably, substantial research efforts have been directed towards the examination of toric code [35], [36], graph states [37], and stabilizer code [38], [39], which is equivalent to a graph state under local Clifford operations [40]. While their topological properties and representational power [41], [42] have been extensively studied, there is still a need to explore feasible algorithms for specific models.

We start from the RBM representability of the toric code model as the first step, with the eventual goal of studying that for general topological phases. In [36], Deng and Li utilized a Further Restricted Restricted Boltzmann Machine (FRRBM), that allows only local connections, to numerically find a solution of the toric code model. However, toric code has degeneracy on non-trivial topology, and the ground state derived in the above manner always corresponds certain specific one. On the other hand, it is possible to achieve an arbitrary ground state by turning the toric code as a graph state [43] and transforming a graph state into an RBM [38]. Yet, this approach inevitably introduces non-local connections within each subgraph which adds to the complexity of the RBM.

In this work, we initially apply stabilizer conditions to several specific configurations to analytically solve the FRRBM for the toric code, exploring its representational capacity. We factorize these solutions on square lattices of various sizes and find that different weights only alter the coefficients of the basis states forming the ground state by factors of ± 1. We then extended this approach to obtain an arbitrary ground state by strategically introducing several non-local connections into the RBM. While this generalization sacrifices the simplicity of local connections, it remains analytically solvable, enabling the simulation of arbitrary ground states in a clean manner. Additionally, we developed an efficient machine learning algorithm to verify the learnability of the models. We further generalize our approach from Z_2 to Z_n and outline potential directions for future research.

5.2 Further Restricted RBM

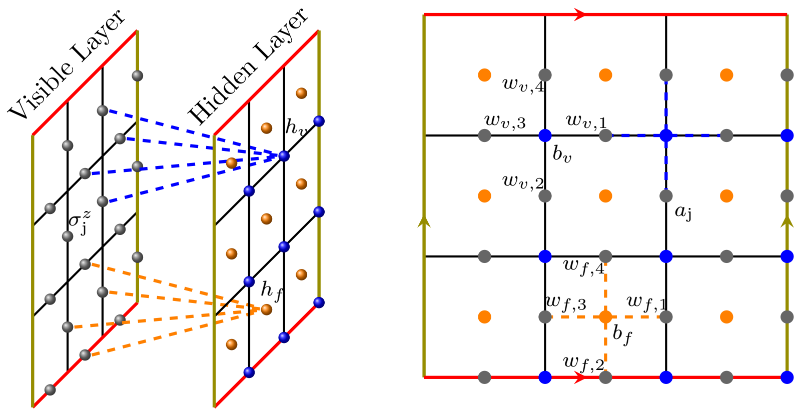


Figure 5.1. The right diagram results from collapsing the two layers shown in the left diagram. It illustrates a translation-invariant FRRBM with visible neurons colored gray and hidden neurons colored red and blue, corresponding to faces and vertices, respectively. The architecture ensures that each hidden neuron is connected only to the nearest visible neurons. Each neuron and each connection is assigned a weight.

To simulate the ground state of the 2D toric code, Deng and Li utilized a translation-invariant Further Restricted RBM (FRRBM), illustrated in Figure 5.1. This model, designed to permit only local connections, was employed to numerically find a solution [36]. For the physical spins $\{\sigma_j^z\}$ on the square lattice, a vertex-type hidden neuron h_v was assigned to each vertex and a face-type hidden neuron h_f to each face, with connections limited to the nearest visible neurons. Given the two types of hidden neurons, the wave function, as shown in Equation (2.59), is reformulated as follows:

$$\Psi_M(S;\mathcal{W}) = \mathrm{e}^{\sum_{\mathrm{j}} a_{\mathrm{j}}\sigma_{\mathrm{j}}^z} \prod_{v\in V} \Gamma_v(S;\mathcal{W}) \prod_{f\in F} \Gamma_f(S;\mathcal{W}), \tag{5.1}$$

$$\Gamma_v(S;\mathcal{W}) = 2\cosh(b_v + \sum_{\mathrm{j}\in s(v)} w_{v,\mathrm{j}}\sigma_{\mathrm{j}}^z), \tag{5.2}$$

$$\Gamma_f(S;\mathcal{W}) = 2\cosh(b_f + \sum_{\mathrm{j}\in s(f)} w_{f,\mathrm{j}}\sigma_{\mathrm{j}}^z). \tag{5.3}$$

They set weight $a_{\mathrm{j}} = 0$ for every visible neuron, choose $(b_f, w_{f,\mathrm{j}}) = (0, \frac{\pi}{4}\mathrm{i})$. Using the stabilizer conditions, they train the FRRBM to get the isotropic solution numerically, resulting in $(b_v, w_{v,\mathrm{j}}) = (0, \frac{\pi}{2}\mathrm{i})$. This FRRBM also naturally supports excited states if translation-invariant symmetry is broken and string operators are applied. Furthermore, this solution can be directly generalized to the 3D toric code.

5.3 Analytical solutions of FRRBM

However, the solution derived above is limited to describing only one specific ground state. To fully explore the representational capacity of the FRRBM, we aim to analytically solve it to identify all possible solutions. We begin with the translation-invariant wave function described above and also set $a_{\mathrm{j}} = 0$ for every visible neuron. Then we solve $\mathcal{W} = (b_f, w_{f,1-4}, b_v, w_{v,1-4})$ using the stabilizer conditions:

$$B_f|GS\rangle = \prod_{\mathrm{e}\in s(f)} \hat{\sigma}_{\mathrm{e}}^z|GS\rangle = |GS\rangle,\ \forall f \in F; \tag{5.4}$$

$$A_v|GS\rangle = \prod_{\mathrm{e}\in s(v)} \hat{\sigma}_{\mathrm{e}}^x|GS\rangle = |GS\rangle,\ \forall v \in V. \tag{5.5}$$

Solving the face stabilizer condition is straightforward, as the operator $\hat{\sigma}_{\mathrm{e}}^z$ does not alter the state of the qubit e. By substituting Equation (5.1) into Equation (2.58) and treating $|\Psi\rangle$ as the ground state $|GS\rangle$, we solve for $|\Psi\rangle$ under the constraints imposed by Equation (5.4). This process is applied within any single face to determine: $b_f = 0\,(mod\,\boldsymbol{\pi})$ and

$w_{f,\mathrm{j}} = \frac{\pi}{4}\mathrm{i}, \frac{3\pi}{4}\mathrm{i}\,(mod\,\pi)$, where an even number of the four $w_{f,\mathrm{j}}$ must be the same. Further calculation details are provided in Appendix C.1.

Unlike the face stabilizer condition, solving the vertex stabilizer condition is more complex. The operator $\hat{\sigma}^x_{\mathrm{e}}$ flips the state of qubit e. As shown in Figure 5.2, applying vertex operator to any vertex $v_0 \in V$ alters the configuration from $|S\rangle$ to $|h_0(S)\rangle$. Notably, applying the vertex operator twice will restore the original configuration. Applying the constraints outlined in Equation (5.5), we derive the following result:

$$\sum_S \prod_{v\in V} \Gamma_v(S;\mathcal{W}) \prod_{f\in F} \Gamma_f(S;\mathcal{W})|h_0(S)\rangle = \sum_S \prod_{v\in V} \Gamma_v(S;\mathcal{W}) \prod_{f\in F} \Gamma_f(S;\mathcal{W})|S\rangle. \tag{5.6}$$

By applying it twice, we can remove the sum to get

$$\prod_{v\in V} \Gamma_v(h_0(S);\mathcal{W}) \prod_{f\in F} \Gamma_f(h_0(S);\mathcal{W}) = \prod_{v\in V} \Gamma_v(S;\mathcal{W}) \prod_{f\in F} \Gamma_f(S;\mathcal{W}) \tag{5.7}$$

for any possible configuration S. However, there are many equal factors on both sides of the Equation (5.7). Canceling out them will reduce the configuration of interest from S to S' which only contains 16 qubits, giving a series of 2^{16} equations. Directly solving these equations is impossible. We can pick up particular configurations and apply one or more vertex operators on it to get independent restrictions. Full calculation details are provided in Appendix C.2, solving them out, we get [1]: $b_v = 0\,(mod\,\pi)$ and $w_{v,\mathrm{j}} = 0, \frac{\pi}{2}\mathrm{i}\,(mod\,\pi)$, where an even number of the four $w_{v,\mathrm{j}}$ must be the same; Otherwise $b_v = 0\,(mod\,\pi)$ and any three of the four $w_{v,\mathrm{j}}$ are equal to 0 or $\frac{\pi}{2}\,(mod\,\pi)$ while the other one is free.

5.4 Arbitrary ground state of RBM

To further elucidate the analytical solutions derived in the last section, we numerically factorize them on a 3×3 square lattice, as detailed in Appendix C.3). Setting $(a_{\mathrm{j}}, b_f, w_{f,\mathrm{j}}, b_v, w_{v,\mathrm{j}}) = (0, 0, \frac{\pi}{4}\mathrm{i}, 0, \frac{\pi}{2}\mathrm{i})$ isotropically results in the ground state $|GS\rangle = -|00\rangle + |01\rangle + |10\rangle - |11\rangle$. Conversely, if we change $w_{v,\mathrm{j}} = 0, 0, \frac{\pi}{2}\mathrm{i}, \frac{\pi}{2}\mathrm{i}$ for the respective directions, the ground state becomes $|GS\rangle = +|00\rangle + |01\rangle + |10\rangle + |11\rangle$. Different settings of $w_{v,\mathrm{j}}$ can

[1]↑We take $b_v = 0$ as $b_v \in \mathbb{C}$ will introduce superfluous freedom, discussed in Appendix C.2.

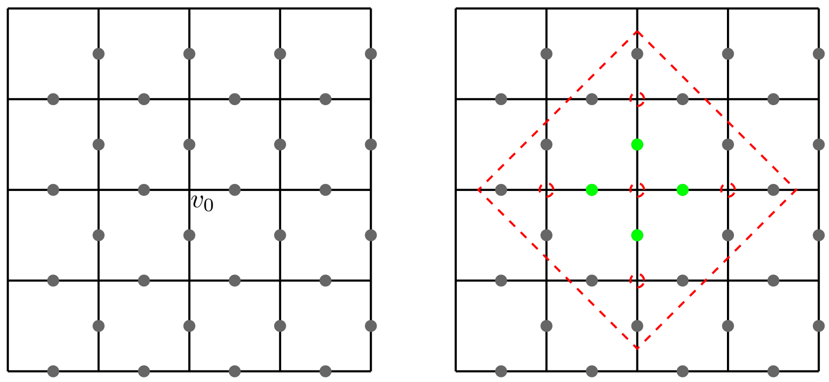


Figure 5.2. The left diagram presents a partial view of a configuration on a larger square lattice. The right diagram is obtained by applying a vertex operator to the vertex v_0. Green nodes indicate qubits that have flipped states, and the red dashed lines encircle nodes considered in the subsequent calculation.

alter the coefficients of the basis states forming the ground state, although these changes are confined to factors of ± 1. This limitation underscores the representational capacity of the FRRBM. Consequently, it prompts a natural question: how can one prepare an arbitrary ground state?

Inspired by the action of the logical operators Z_v and Z_h, we introduce three additional hidden neurons (h_x, h_y, and h_z) to the FRRBM, enabling it to encapsulate the topological information of ground states in a 2D toric code. These neurons have non-local connections as depicted in Figure 5.3. We demonstrate that the inclusion of h_x, h_y, and h_z allows for the simulation of any arbitrary ground state. The wave function in Equation (5.1) is modified as follows:

$$\Psi_M(S;\mathcal{W}) = e^{\sum_j a_j \sigma_j^z} \prod_{e\in\{x,y,z\}} \Gamma_e(S;\mathcal{W}) \prod_{v\in V} \Gamma_v(S;\mathcal{W}) \prod_{f\in F} \Gamma_f(S;\mathcal{W}), \tag{5.8}$$

$$\Gamma_e(S;\mathcal{W}) = 2\cosh(b_e + \sum_j w_e \sigma_j^z). \tag{5.9}$$

If we set the parameters $(a_{\mathrm{j}}, b_f, w_{f,\mathrm{j}}, b_v, w_{v,\mathrm{j}}, w_{x,y,z})$ to $(0, 0, \frac{\pi}{4}\mathrm{i}, 0, \frac{\pi}{2}\mathrm{i}, \frac{\pi}{4}\mathrm{i})$, the unnormalized ratio of the ground state on a 3×3 square lattice can be analytically derived:

$$\langle GS|00\rangle = -\cosh(b_x + \frac{\pi}{4}\mathrm{i})\cosh(b_y + \frac{\pi}{4}\mathrm{i})\cosh(b_z + \frac{\pi}{2}\mathrm{i}), \tag{5.10}$$

$$\langle GS|01\rangle = \cosh(b_x - \frac{\pi}{4}\mathrm{i})\cosh(b_y + \frac{\pi}{4}\mathrm{i})\cosh(b_z), \tag{5.11}$$

$$\langle GS|10\rangle = \cosh(b_x + \frac{\pi}{4}\mathrm{i})\cosh(b_y - \frac{\pi}{4}\mathrm{i})\cosh(b_z), \tag{5.12}$$

$$\langle GS|11\rangle = -\cosh(b_x - \frac{\pi}{4}\mathrm{i})\cosh(b_y - \frac{\pi}{4}\mathrm{i})\cosh(b_z - \frac{\pi}{2}\mathrm{i}). \tag{5.13}$$

For example, we can select the degeneracy state $|00\rangle$ by setting (b_x, b_y, b_z) to $(\frac{3\pi}{4}\mathrm{i}, \frac{3\pi}{4}\mathrm{i}, \frac{\pi}{2}\mathrm{i})$. Arbitrary ground states with amplitude ratios like $\langle GS|00\rangle : \langle GS|01\rangle : \langle GS|10\rangle : \langle GS|11\rangle = 1:2:3:4$ can be exactly solved. While an exact solution for certain ratios containing 0 may not exist, we can approximate these by setting the zeros to extremely small values. Further details are provided in Appendix C.4. While this generalization sacrifices the simplicity of local connections, it remains analytically solvable and enables the simulation of all possible ground states in a clean manner. It also retains the ability to manipulate string operators and has demonstrated both efficient and accurate performance when applied with machine learning techniques.

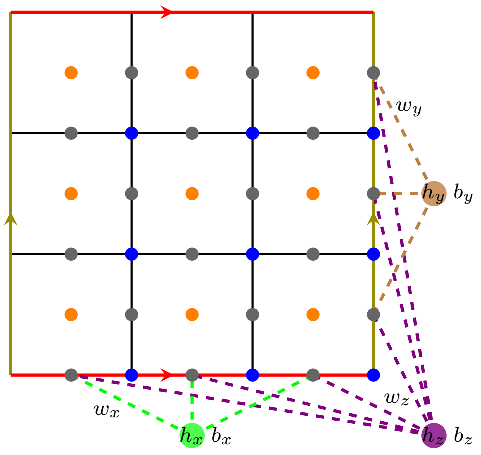


Figure 5.3. Three hidden neurons (h_x, h_y, h_z) are introduced into the FR-RBM to simulate an arbitrary ground state. h_x connects to visible neurons along a horizontal loop, h_y connects along a vertical loop, and h_z connects to all neurons connected by h_x and h_y. Each connection type from a specific hidden neuron is uniformly weighted (w_x, w_y, w_z).

5.5 Efficiency and Learnability of the RBM

In the literature [36], Deng and Li analytically derived a solution for the face terms and then trained the FRRBM using a vertex stabilizer condition on a portion of a larger square lattice, employing a large number of configurations. In contrast, our study analytically derives both face and vertex terms and numerically verifies their learnability on square lattices of various sizes using a significantly reduced set of only 50 configurations20 selected for the degeneracy basis and 30 random configurations. This approach is both efficient and accurate compared to approaches that use large numbers of random configurations, as demonstrated in Figure 5.4.

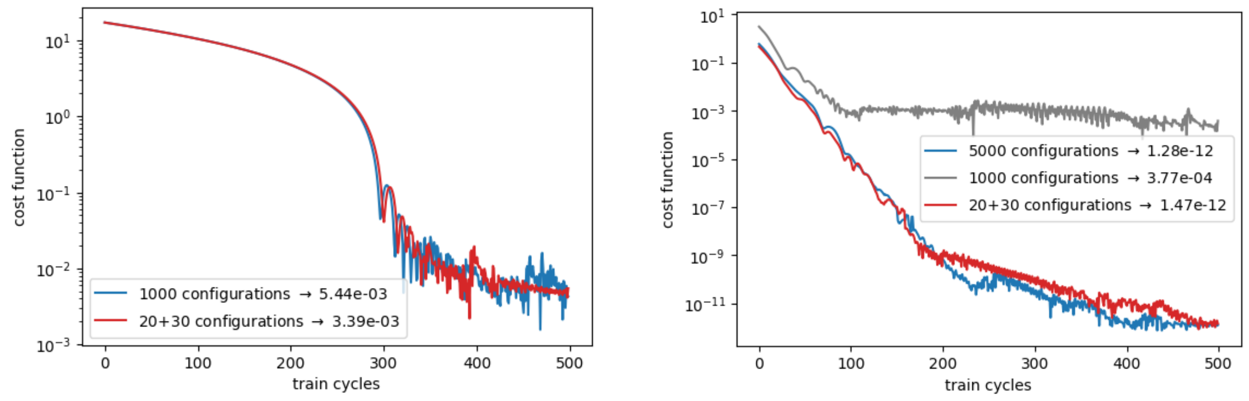


Figure 5.4. On a 3×3 square lattice, we separately train the face terms and vertex terms using face stabilizer and vertex stabilizer conditions, respectively. The left plot compares the training efficiencies of different configurations for face terms, while the right plot does the same for vertex terms.

As learnability is influenced by the initial settings, we randomly select 10,000 settings for $b_v \in \mathbb{C}$ and $w_v \in \mathbb{C}$ to numerically search for solutions. Despite this extensive search, the presence of *Barren Plateaus*, illustrated in Figure 5.5, limited us to only 20 solutions. Although the search parameters b_v and w_v were complex, we found solutions only where both b_v and w_v are purely imaginary. Barren plateaus are regions in the optimization landscape where gradients vanish, impeding any significant learning progress. This phenomenon explains why our search procedure, with limited training time, only yielded a few solutions.

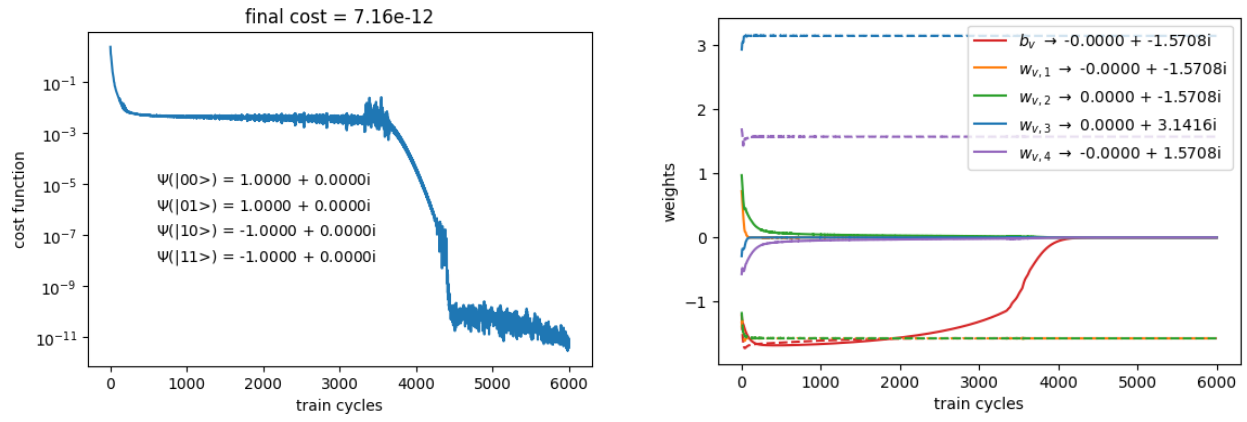


Figure 5.5. On a 3×3 square lattice, this example demonstrates the existence of barren plateaus, characterized by a sudden drop in performance after prolonged training.

5.6 Generalization from $\mathbb{Z}_2$ to $\mathbb{Z}_n$

In previous sections, we determined the weights of the RBMs analytically and numerically to assess their representational capabilities. Notably, B_f selects configurations with trivial flux, while A_v ensures all configurations have uniform weight across a logical state. Among the solutions we found, a 'trivial solution' $(b_f, w_{f,\mathrm{j}}, b_v, w_{v,\mathrm{j}}) = (0, \frac{\pi}{4}\mathrm{i}, 0, 0)$ emerged, where each survived configuration possess equal weight. By foregoing the manipulation of string operators, this solution can be generalized to implement the *Kitaev quantum double model* associated with the group Z_N[2]. The model, set on an oriented lattice with a $|G|$-dimensional qudit on each edge labeled by a group element g, follows the convention in [1]. Though the Hamiltonian resembles Equation (2.53), A_v and B_f are defined in a different manner. As shown in Figure 5.6, we focus exclusively on the action of B_f:

$$B_f|v_1\ v_2\ v_3\ v_4\rangle = \delta_{p_g,1_g}|v_1\ v_2\ v_3\ v_4\rangle, \tag{5.14}$$

where 1_g is the identity element of the group G, and p_g is the group product of states on each edge bordering the face counterclockwise[3]. If an edges direction aligns with the orientation,

[2]↑For $N = 2$, the model corresponds to the toric code with a basis change to represent qubits.
[3]↑We need to pick up a start-up vertex, though it turns out to be insignificant.

we include $v_{\rm i}$; otherwise, we use $v_{\rm i}^{-1}$. Thus, $p_g = \prod_{\rm i} v_{\rm i}^{a_{\rm i}}$, where $a_{\rm i} = \pm 1$ reflects this alignment. Specifically, for $G = Z_N$, the state of the N-dimensional qudit is labelled by $0, 1, \ldots N-1$. In this setting, the group product is arithmetic summation, the identity element 1_g is 0, and each element is its own inverse. Then the action of B_f is significantly simplified:

$$B_f|v_1\ v_2\ v_3\ v_4\rangle = \delta_{\sum_{\rm i} v_{\rm i},0}|v_1\ v_2\ v_3\ v_4\rangle. \tag{5.15}$$

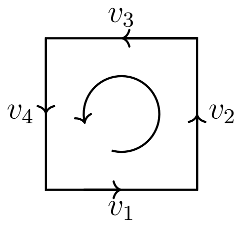


Figure 5.6. Convention for the local operator: Edges are ordered counter-clockwise as v_1, v_2, v_3, v_4, with directions indicated by arrows on each edge.

We utilize a natural RBM to implement the above action, corresponding to the condition $\sum v_{\rm i} = 0\,(mod\,N)$. This is achieved using an N-dimensional invisible qudit u with the setting of isotropic weights $(b, w_{\rm i}) = (0, \frac{2\pi{\rm i}}{N})$:

$$\sum_{u=0}^{N-1} \exp(b + \sum_{\rm i} u w_{\rm i} v_{\rm i}) = \frac{1 - \exp(2\pi{\rm i}(b + \sum_{\rm i} v_{\rm i}))}{1 - \exp\left(2\pi{\rm i}(b + \sum_{\rm i} v_{\rm i})/N\right)} = \begin{cases} N & \text{if } b + \sum_{\rm i} v_{\rm i} = 0\,(mod\,N). \\ 0 & \text{otherwise.} \end{cases} \tag{5.16}$$

The action of A_v is safely neglected here, as the RBM already simulates a ground state that is an equal superposition of all logical bases. And this expression explicitly ensures the flux-free requirement. It offers a natural method for creating fluxions by setting $b \neq 0$, though such creation is not arbitrary on a closed manifold due to global constraints. They are elementary magnetic excitations, since each element of Z_N represents a unique conjugacy class. Furthermore, a complete basis of the ground state can be found in the same manner as illustrated in Figure 5.3.

This method can be generalized to other lattice model with frustration-free Hamiltonian composed of two types of terms: one constraining local flux and the other enabling gauge transformations. Applying the flux-free RBM achieves an equal superposition of all flux-free configurations, automatically satisfying gauge transformation terms and resulting in the superposition of all logical states. For instance, this approach is applicable to Kitaev model associated with abelian group, X-cube model, checkerboard model, Haar-A and Haar-B codes and so on. However, the generalization to Kitaev quantum double model associated with a non-abelian group remains unclear.

5.7 Conclusion and further work

We analytically resolved the FRRBM proposed for the toric code, determining all possible ground states to assess the model's capabilities. We then modified this model to support arbitrary ground states through the integration of non-local connections. This enhanced model remains analytically solvable and can also be efficiently solved using machine learning techniques. We then extend our work to Kitaev quantum double model associated with abelian group Z_N. Our ongoing research aims to investigate feasible RBM implementations for more specific codes, including those for the Double Semion [13], Fibonacci Anyon [13], [52], and Kitaev quantum double model associated with a non-abelian group.

REFERENCES

[1] B. Yan, P. Chen, and S. X. Cui, "Ribbon operators in the generalized kitaev quantum double model based on hopf algebras," *Journal of Physics A: Mathematical and Theoretical*, vol. 55, no. 18, p. 185 201, 2022.

[2] P. Chen, B. Yan, and S. X. Cui, "Quantum circuits for toric code and x-cube fracton model," *Quantum*, vol. 8, p. 1276, 2024.

[3] P. Chen, B. Yan, and S. X. Cui, "Representing arbitrary ground states of toric code by restricted boltzmann machine," *arXiv preprint arXiv:2407.01451*, 2024.

[4] F. Arute, K. Arya, R. Babbush, *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[5] Y. Kim, A. Eddins, S. Anand, *et al.*, "Evidence for the utility of quantum computing before fault tolerance," *Nature*, vol. 618, no. 7965, pp. 500–505, 2023.

[6] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, "Trapped-ion quantum computing: Progress and challenges," *Applied Physics Reviews*, vol. 6, no. 2, p. 021 314, 2019.

[7] C. Nayak, S. H. Simon, A. Stern, M. Freedman, and S. D. Sarma, "Non-abelian anyons and topological quantum computation," *Reviews of Modern Physics*, vol. 80, no. 3, p. 1083, 2008.

[8] D. I. Pikulin, B. van Heck, T. Karzig, *et al.*, "Protocol to identify a topological superconducting phase in a three-terminal device," *arXiv preprint arXiv:2103.12217*, 2021.

[9] A. Y. Kitaev, "Fault-tolerant quantum computation by anyons," *Annals of Physics*, vol. 303, no. 1, pp. 2–30, 2003.

[10] M. H. Freedman, M. Larsen, and Z. Wang, "A modular functor which is universal for quantum computation," *Communications in Mathematical Physics*, vol. 227, no. 3, pp. 605–622, 2002.

[11] O. Buerschaper, J. M. Mombelli, M. Christandl, and M. Aguado, "A hierarchy of topological tensor network states," *Journal of Mathematical Physics*, vol. 54, no. 1, p. 012 201, 2013.

[12] L. Chang, "Kitaev models based on unitary quantum groupoids," *Journal of Mathematical Physics*, vol. 55, no. 4, p. 041 703, 2014.

[13] M. A. Levin and X.-G. Wen, "String-net condensation: A physical mechanism for topological phases," *Physical Review B*, vol. 71, no. 4, p. 045 110, 2005.

[14] O. Buerschaper and M. Aguado, "Mapping Kitaev's quantum double lattice models to Levin and Wen's string-net models," *Physical Review B*, vol. 80, no. 15, p. 155 136, 2009.

[15] O. Buerschaper, M. Christandl, L. Kong, and M. Aguado, "Electric–magnetic duality of lattice systems with topological order," *Nuclear Physics B*, vol. 876, no. 2, pp. 619–636, 2013.

[16] H. Bombin and M. Martin-Delgado, "Family of non-Abelian Kitaev models on a lattice: Topological condensation and confinement," *Physical Review B*, vol. 78, no. 11, p. 115 421, 2008.

[17] K. Walker and Z. Wang, "(3+ 1)-TQFTs and topological insulators," *Frontiers of Physics*, vol. 7, no. 2, pp. 150–159, 2012.

[18] J. Haah, "Local stabilizer codes in three dimensions without string logical operators," *Physical Review A*, vol. 83, no. 4, p. 042 330, 2011.

[19] S. Vijay, J. Haah, and L. Fu, "A new kind of topological quantum order: A dimensional hierarchy of quasiparticles built from stationary excitations," *Physical Review B*, vol. 92, no. 23, p. 235 136, 2015.

[20] S. Vijay, J. Haah, and L. Fu, "Fracton topological order, generalized lattice gauge theory, and duality," *Physical Review B*, vol. 94, no. 23, p. 235 157, 2016.

[21] S. Bravyi, M. B. Hastings, and S. Michalakis, "Topological quantum order: Stability under local perturbations," *Journal of mathematical physics*, vol. 51, no. 9, p. 093 512, 2010.

[22] K. Satzinger, Y.-J. Liu, A. Smith, *et al.*, "Realizing topologically ordered states on a quantum processor," *Science*, vol. 374, no. 6572, pp. 1237–1241, 2021.

[23] S. Ebadi, T. T. Wang, H. Levine, *et al.*, "Quantum phases of matter on a 256-atom programmable quantum simulator," *Nature*, vol. 595, no. 7866, pp. 227–232, 2021.

[24] R. Verresen, M. D. Lukin, and A. Vishwanath, "Prediction of toric code topological order from Rydberg blockade," *Physical Review X*, vol. 11, no. 3, p. 031 005, 2021.

[25] S. B. Bravyi and A. Y. Kitaev, "Quantum codes on a lattice with boundary," *arXiv preprint quant-ph/9811052*, 1998.

[26] Y.-J. Liu, K. Shtengel, A. Smith, and F. Pollmann, "Methods for simulating string-net states and anyons on a digital quantum computer," *arXiv:2110.02020*, 2021.

[27] N. Tantivasadakarn, R. Thorngren, A. Vishwanath, and R. Verresen, "Long-range entanglement from measuring symmetry-protected topological phases," *arXiv:2112.01519*, 2021.

[28] R. Verresen, N. Tantivasadakarn, and A. Vishwanath, "Efficiently preparing Schrödingers cat, fractons and non-Abelian topological order in quantum devices," *arXiv:2112.03061*, 2021.

[29] S. Bravyi, I. Kim, A. Kliesch, and R. Koenig, "Adaptive constant-depth circuits for manipulating non-Abelian anyons," *arXiv:2205.01933*, 2022.

[30] N. Tantivasadakarn, R. Verresen, and A. Vishwanath, "The shortest route to non-Abelian topological order on a quantum processor," *arXiv:2209.03964*, 2022.

[31] N. Tantivasadakarn, A. Vishwanath, and R. Verresen, "A hierarchy of topological order from finite-depth unitaries, measurement and feedforward," *arXiv:2209.06202*, 2022.

[32] T. J. Osborne, "Hamiltonian complexity," *Reports on progress in physics*, vol. 75, no. 2, p. 022 001, 2012.

[33] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[34] G. Carleo and M. Troyer, "Solving the quantum many-body problem with artificial neural networks," *Science*, vol. 355, no. 6325, pp. 602–606, 2017.

[35] D.-L. Deng, X. Li, and S. D. Sarma, "Quantum entanglement in neural network states," *Physical Review X*, vol. 7, no. 2, p. 021 021, 2017.

[36] D.-L. Deng, X. Li, and S. D. Sarma, "Machine learning topological states," *Physical Review B*, vol. 96, no. 19, p. 195 145, 2017.

[37] X. Gao and L.-M. Duan, “Efficient representation of quantum many-body states with deep neural networks,” *Nature communications*, vol. 8, no. 1, p. 662, 2017.

[38] S. Lu, X. Gao, and L.-M. Duan, “Efficient representation of topologically ordered states with restricted boltzmann machines,” *Physical Review B*, vol. 99, no. 15, p. 155 136, 2019.

[39] Z.-A. Jia, Y.-H. Zhang, Y.-C. Wu, L. Kong, G.-C. Guo, and G.-P. Guo, “Efficient machine-learning representations of a surface code with boundaries, defects, domain walls, and twists,” *Physical Review A*, vol. 99, no. 1, p. 012 307, 2019.

[40] M. Van den Nest, J. Dehaene, and B. De Moor, “Graphical description of the action of local clifford transformations on graph states,” *Physical Review A*, vol. 69, no. 2, p. 022 316, 2004.

[41] N. Le Roux and Y. Bengio, “Representational power of restricted boltzmann machines and deep belief networks,” *Neural computation*, vol. 20, no. 6, pp. 1631–1649, 2008.

[42] Y. Huang, J. E. Moore, *et al.*, “Neural network representation of tensor network and chiral states,” *Physical Review Letters*, vol. 127, no. 17, p. 170 601, 2021.

[43] P. Liao and D. L. Feder, “Graph-state representation of the toric code,” *Physical Review A*, vol. 104, no. 1, p. 012 432, 2021.

[44] D. E. Radford, *Hopf algebras*. World Scientific, 2011, vol. 49.

[45] C. Kassel, *Quantum groups*. Springer Science & Business Media, 2012, vol. 155.

[46] I. Cong, M. Cheng, and Z. Wang, “Hamiltonian and algebraic theories of gapped boundaries in topological phases of matter,” *Communications in Mathematical Physics*, vol. 355, no. 2, pp. 645–689, 2017.

[47] S. Bravyi, M. B. Hastings, and F. Verstraete, “Lieb-robinson bounds and the generation of correlations and topological quantum order,” *Physical review letters*, vol. 97, no. 5, p. 050 401, 2006.

[48] O. Higgott, M. Wilson, J. Hefford, *et al.*, “Optimal local unitary encoding circuits for the surface code,” *Quantum*, vol. 5, p. 517, 2021.

[49] M. Aguado and G. Vidal, “Entanglement renormalization and topological order,” *Physical review letters*, vol. 100, no. 7, p. 070 404, 2008.

[50] K. Slagle and Y. B. Kim, “Quantum field theory of x-cube fracton topological order and robust degeneracy from geometry,” *Physical Review B*, vol. 96, no. 19, p. 195 139, 2017.

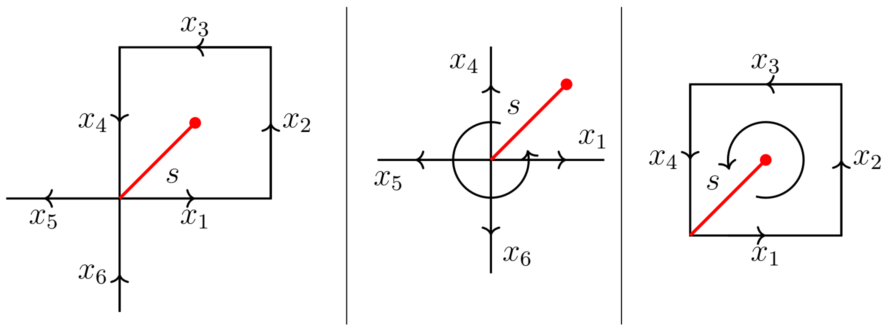
[51] A. Prem, J. Haah, and R. Nandkishore, “Glassy quantum dynamics in translation invariant fracton models,” *Physical Review B*, vol. 95, no. 15, p. 155 133, 2017.

[52] C.-H. Lin, M. Levin, and F. J. Burnell, “Generalized string-net models: A thorough exposition,” *Physical Review B*, vol. 103, no. 19, p. 195 155, 2021.

A. SUPPLEMENTAL MATERIAL FOR CHAPTER 3

This chapter contains work from the article entitled "Ribbon operators in the generalized Kitaev quantum double model based on Hopf algebras" written by Bowen Yan, the author, and Shawn X. Cui published on Journal of Physics A [1].

A.1 Straightening equation of A_a and B_f



This equation holds no matter how the edges are oriented. We check the case above:

$$A_a(s)B_f(s)|x_1\ x_2\ x_3\ x_4\ x_5\ x_6\rangle \tag{A.1}$$

$$= A_a(s)\sum_{(x_{\mathrm{i}})} f(x_1''x_2''x_3''x_4'')|x_1'\ x_2'\ x_3'\ x_4'\ x_5\ x_6\rangle \tag{A.2}$$

$$= \sum_{(x_{\mathrm{i}}),(a)} f(x_1''x_2''x_3''x_4'')|a^{(4)}x_1'\ x_2'\ x_3'\ x_4'S(a')\ a''x_5\ x_6S(a''')\rangle \tag{A.3}$$

$$= \sum_{(x_{\mathrm{i}}),(a)} f[S(a^{(8)})a^{(7)}x_1''x_2''x_3''x_4''S(a'')a'] \tag{A.4}$$

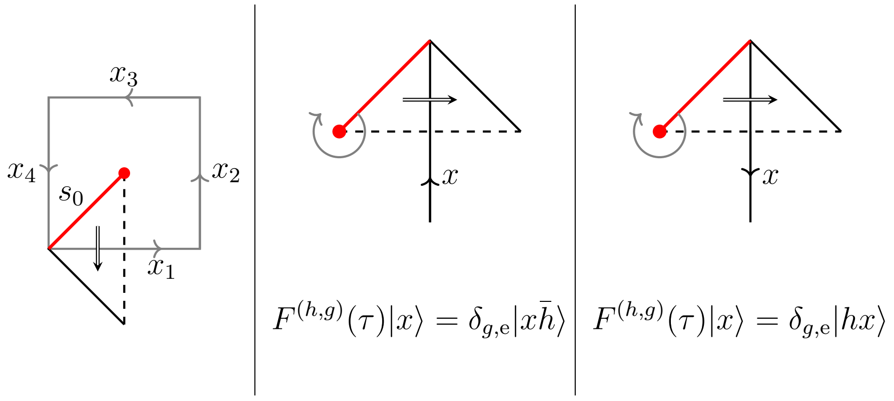
$$|a^{(6)}x_1'\ x_2'\ x_3'\ x_4'S(a''')\ a^{(4)}x_5'\ x_6S(a^{(5)})\rangle \tag{A.5}$$

$$= \sum_{(x_{\mathrm{i}}),(a)} B_{f[S(a^{(6)})?a']}(s)|a^{(5)}x_1\ x_2\ x_3\ x_4S(a'')\ a'''x_5\ x_6S(a^{(4)})\rangle \tag{A.6}$$

$$= \sum_{(a)} B_{f[S(a''')?a']}(s)A_{a''}(s)|x_1\ x_2\ x_3\ x_4\ x_5\ x_6\rangle \tag{A.7}$$

This is exactly the straightening equation.

A.2 Violation and correction in group algebra

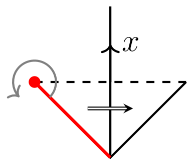


$F^{(h,g)}(\tau)|x\rangle = \delta_{g,\mathrm{e}}|x\bar{h}\rangle$ $F^{(h,g)}(\tau)|x\rangle = \delta_{g,\mathrm{e}}|hx\rangle$

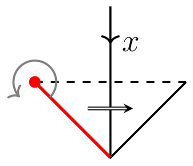
We show Equation 3.5 is violated for the ribbon τ in the first figure above for the original Kitaev model where H is taken to be the group algebra of a non-Abelian group G. In [16], only two formulas are provided for dual triangles as shown in the second and third figure above. However, we can not get the desired commutation relation using either of them:

$$B_{h'}(s_0)F^{(h,g)}(\tau)|x_1\ x_2\ x_3\ x_4\rangle \tag{A.8}$$
$$= B_{h'}(s_0)\delta_{g,\mathrm{e}}|x_1\bar{h}\ x_2\ x_3\ x_4\rangle \tag{A.9}$$
$$= \delta_{h',x_1\bar{h}x_2x_3x_4}\delta_{g,\mathrm{e}}|x_1\bar{h}\ x_2\ x_3\ x_4\rangle \tag{A.10}$$
$$\neq \delta_{g,\mathrm{e}}\delta_{hh',x_1x_2x_3x_4}|x_1\bar{h}\ x_2\ x_3\ x_4\rangle \tag{A.11}$$
$$= F^{(h,g)}(\tau)\delta_{hh',x_1x_2x_3x_4}|x_1\ x_2\ x_3\ x_4\rangle \tag{A.12}$$
$$= F^{(h,g)}(\tau)B_{hh'}(s_0)|x_1\ x_2\ x_3\ x_4\rangle \tag{A.13}$$
$$\tag{A.14}$$
$$B_{h'}(s_0)F^{(h,g)}(\tau)|x_1\ x_2\ x_3\ x_4\rangle \tag{A.15}$$
$$= B_{h'}(s_0)\delta_{g,\mathrm{e}}|hx_1\ x_2\ x_3\ x_4\rangle \tag{A.16}$$
$$= \delta_{h',hx_1x_2x_3x_4}\delta_{g,\mathrm{e}}|hx_1\ x_2\ x_3\ x_4\rangle \tag{A.17}$$
$$\neq \delta_{g,\mathrm{e}}\delta_{hh',x_1x_2x_3x_4}|hx_1\ x_2\ x_3\ x_4\rangle \tag{A.18}$$
$$= F^{(h,g)}(\tau)\delta_{hh',x_1x_2x_3x_4}|x_1\ x_2\ x_3\ x_4\rangle \tag{A.19}$$
$$= F^{(h,g)}(\tau)B_{hh'}(s_0)|x_1\ x_2\ x_3\ x_4\rangle \tag{A.20}$$

Moreover, the issue can not be removed by making τ longer. Roughly, this is because for the current τ, the initial site and terminal site already lie in different plaquettes, and thus lengthening it will not affect the action of the plaquette operator at the initial site. To resolve the issue, we recognize that τ has locally counterclockwise orientation, and hence we need to apply the following formulas for the ribbon operators:



$$F^{(h,g)}(\tau)|x\rangle = \delta_{g,\mathrm{e}}|\bar{h}x\rangle$$



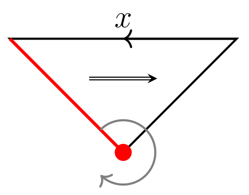
$$F^{(h,g)}(\tau)||x\rangle = \delta_{g,\mathrm{e}}|xh\rangle$$

With the new formula above, we have

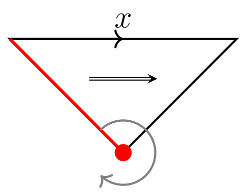
$$\begin{aligned}
& B_{h'}(s_0)F^{(h,g)}(\tau)|x_1\ x_2\ x_3\ x_4\rangle \\
= \ & B_{h'}(s_0)\delta_{g,\mathrm{e}}|\bar{h}x_1\ x_2\ x_3\ x_4\rangle \\
= \ & \delta_{h',\bar{h}x_1x_2x_3x_4}\delta_{g,\mathrm{e}}|\bar{h}x_1\ x_2\ x_3\ x_4\rangle \\
= \ & \delta_{g,\mathrm{e}}\delta_{hh',x_1x_2x_3x_4}|\bar{h}x_1\ x_2\ x_3\ x_4\rangle \\
= \ & F^{(h,g)}(\tau)\delta_{hh',x_1x_2x_3x_4}|x_1\ x_2\ x_3\ x_4\rangle \\
= \ & F^{(h,g)}(\tau)B_{hh'}(s_0)|x_1\ x_2\ x_3\ x_4\rangle
\end{aligned}$$

A.3 Multiplication of ribbon operators on elementary ribbons

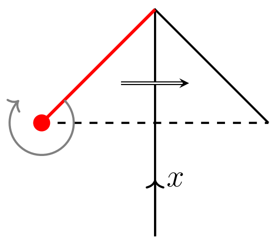
A.3.1 For locally clockwise ribbons τ_L



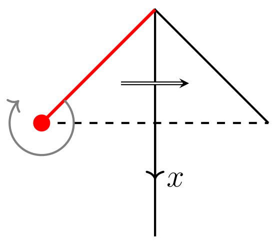
$$\begin{aligned}
& F^{(h_1,f_1)}(\tau_L)F^{(h_2,f_2)}(\tau_L)|x\rangle \\
= & \sum_{(x)} F^{(h_1,f_1)}(\tau_L)\epsilon(h_2)f_2[S(x'')]|x'\rangle \\
= & \sum_{(x)} \epsilon(h_2)\epsilon(h_1)f_2[S(x''')]f_1[S(x'')]|x'\rangle \\
= & \sum_{(x)} \epsilon(h_1h_2)\langle f_2\otimes f_1, \Delta[S(x'')]\rangle|x'\rangle \\
= & F^{(h_1h_2,f_2f_1)}(\tau_L)|x\rangle
\end{aligned}$$



$$\begin{aligned}
& F^{(h_1,f_1)}(\tau_L)F^{(h_2,f_2)}(\tau_L)|x\rangle \\
= & \sum_{(x)} F^{(h_1,f_1)}(\tau_L)\epsilon(h_2)f_2(x')|x''\rangle \\
= & \sum_{(x)} \epsilon(h_2)\epsilon(h_1)f_2(x')f_1(x'')|x'''\rangle \\
= & \sum_{(x)} \epsilon(h_1h_2)\langle f_2\otimes f_1, \Delta(x')\rangle|x''\rangle \\
= & F^{(h_1h_2,f_2f_1)}(\tau_L)|x\rangle
\end{aligned}$$

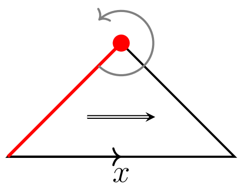


$$
\begin{aligned}
& F^{(h_1,f_1)}(\tau_L)F^{(h_2,f_2)}(\tau_L)|x\rangle \\
= \; & F^{(h_1,f_1)}(\tau_L)\epsilon(f_2)|xS(h_2)\rangle \\
= \; & \epsilon(f_2)\epsilon(f_1)|xS(h_2)S(h_1)\rangle \\
= \; & \epsilon(f_2f_1)|xS(h_1h_2)\rangle \\
= \; & F^{(h_1h_2,f_2f_1)}(\tau_L)|x\rangle
\end{aligned}
$$

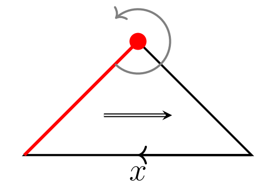


$$
\begin{aligned}
& F^{(h_1,f_1)}(\tau_L)F^{(h_2,f_2)}(\tau_L)|x\rangle \\
= \; & F^{(h_1,f_1)}(\tau_L)\epsilon(f_2)|h_2x\rangle \\
= \; & \epsilon(f_2)\epsilon(f_1)|h_1h_2x\rangle \\
= \; & F^{(h_1h_2,f_2f_1)}(\tau_L)|x\rangle
\end{aligned}
$$

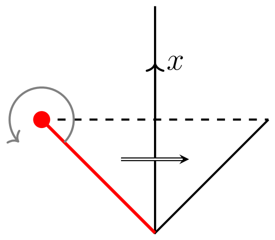
A.3.2 For locally counterclockwise ribbons τ_R



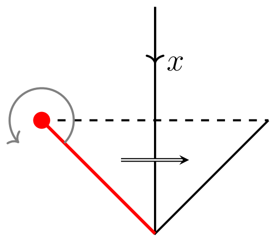
$$
\begin{aligned}
& F^{(h_1,f_1)}(\tau_R)F^{(h_2,f_2)}(\tau_R)|x\rangle \\
= \; & \sum_{(x)} F^{(h_1,f_1)}(\tau_R)\epsilon(h_2)f_2(x'')|x'\rangle \\
= \; & \sum_{(x)} \epsilon(h_2)\epsilon(h_1)f_2(x''')f_1(x'')|x'\rangle \\
= \; & \sum_{(x)} \epsilon(h_2h_1)\langle f_1 \otimes f_2, \Delta(x'')\rangle|x'\rangle \\
= \; & F^{(h_2h_1,f_1f_2)}(\tau_R)|x\rangle
\end{aligned}
$$



$$
\begin{aligned}
& F^{(h_1,f_1)}(\tau_R)F^{(h_2,f_2)}(\tau_R)|x\rangle \\
= & \sum_{(x)} F^{(h_1,f_1)}(\tau_R)\epsilon(h_2)f_2[S(x')]|x''\rangle \\
= & \sum_{(x)} \epsilon(h_2)\epsilon(h_1)f_2[S(x')]f_1[S(x'')]|x'''\rangle \\
= & \sum_{(x)} \epsilon(h_2h_1)\langle f_1\otimes f_2, \Delta[S(x')]\rangle|x''\rangle \\
= & F^{(h_2h_1,f_1f_2)}(\tau_R)|x\rangle
\end{aligned}
$$



$$
\begin{aligned}
& F^{(h_1,f_1)}(\tau_R)F^{(h_2,f_2)}(\tau_R)|x\rangle \\
= & F^{(h_1,f_1)}(\tau_R)\epsilon(f_2)|S(h_2)x\rangle \\
= & \epsilon(f_2)\epsilon(f_1)|S(h_1)S(h_2)x\rangle \\
= & \epsilon(f_1f_2)|S(h_2h_1)x\rangle \\
= & F^{(h_2h_1,f_1f_2)}(\tau_R)|x\rangle
\end{aligned}
$$



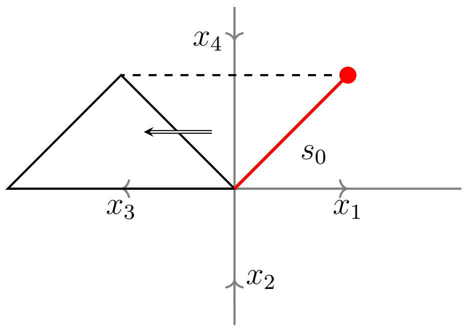
$$
\begin{aligned}
& F^{(h_1,f_1)}(\tau_R)F^{(h_2,f_2)}(\tau_R)|x\rangle \\
= & F^{(h_1,f_1)}(\tau_R)\epsilon(f_2)|xh_2\rangle \\
= & \epsilon(f_2)\epsilon(f_1)|xh_2h_1\rangle \\
= & \epsilon(f_1f_2)|xh_2h_1\rangle \\
= & F^{(h_2h_1,f_1f_2)}(\tau_R)|x\rangle
\end{aligned}
$$

A.4 Proof of Lemma for local operator at ends

The idea is to first prove the equations in the lemma for ribbons as short as possible, and then extend them to longer ribbons. It turns out that the shortest ribbon for some of the equations to hold is a triangle (direct or dual), while for others is a 2-triangle. For example, see the ribbon in Subsection A.4.1. Equation 3.17a does not hold for the rightmost triangle alone. This is roughly because for that triangle, its initial site and terminal site share the same vertex so that $A_a(s_0)$ would also act on s_1, which is unexpected. As will be shown below, the equation does hold as long as we make the triangle a bit longer. This is not a problem since we are only interested in properties of sufficiently long ribbons.

Subsections A.4.1-A.4.8 each addresses an identity in Equations 3.17a - 3.18d for the shortest possible ribbon. For each of the eight equations, there are two types of triangles (direct or dual) to consider. To avoid lengthy calculations, we only present the details for one of the two types for each equation. The proof for the other cases is similar. If a triangle does not work, then we lengthen it to a 2-triangle. In Subsection A.4.9 we extend the results to longer ribbons for Equations 3.17b and 3.17c while leave the other six cases as an exercise (whose proof is similar as well).

A.4.1 Equation 3.17a for short ribbons



$$A_a(s_0)F^{(h,f)}(\tau_L)|x_1\ x_4\ x_3\ x_2\rangle \tag{A.21}$$

$$= A_a(s_0) \sum_{(\mathrm{i}),\mathrm{i},(h)} F^{(h',g_{\mathrm{i}})}(\tau_1)F^{[S(\mathrm{i}''')h''\mathrm{i}',f(\mathrm{i}''?)]}(\tau_2)|x_1\ x_4\ x_3\ x_2\rangle \tag{A.22}$$

$$= A_a(s_0) \sum_{(h),(\mathrm{i}),\mathrm{i},(x_3)} F^{(h',g_{\mathrm{i}})}(\tau_1)\epsilon[S(\mathrm{i}''')h''\mathrm{i}']f(\mathrm{i}''x_3')|x_1\ x_4\ x_3''\ x_2\rangle \tag{A.23}$$

$$= A_a(s_0) \sum_{(h),(\mathrm{i}),\mathrm{i},(x_3)} \epsilon(g_{\mathrm{i}})\epsilon[S(\mathrm{i}''')h''\mathrm{i}']f(\mathrm{i}''x_3')|x_1\ x_4S(h')\ x_3''\ x_2\rangle \tag{A.24}$$

$$= A_a(s_0) \sum_{(\mathrm{e}),(x_3),(h)} \epsilon(\mathrm{e}''')\epsilon(h'')\epsilon(\mathrm{e}')f(\mathrm{e}''x_3')|x_1\ x_4S(h')\ x_3''\ x_2\rangle \tag{A.25}$$

$$= \sum_{(a),(x_3),(h)} f(\mathrm{e}x_3')|a^{(4)}x_1\ x_4S(h')S(a')\ a''x_3''\ x_2S(a''')\rangle \tag{A.26}$$

$$= \sum_{(a),(x_3)} f[\epsilon(a'')x_3']|a^{(5)}x_1\ x_4\epsilon(a^{(6)})S(h)S(a')\ a'''x_3''\ x_2S(a^{(4)})\rangle \tag{A.27}$$

$$= \sum_{(a),(x_3)} f[S(a'')\mathrm{e}a'''x_3']|a^{(6)}x_1\ x_4S(h)S(a')\ a^{(4)}x_3''\ x_2S(a^{(5)})\rangle \tag{A.28}$$

$$= \sum_{(\mathrm{i}),\mathrm{i},(h),(x_3),(a)} \epsilon(g_{\mathrm{i}})\epsilon(\mathrm{i}''')\epsilon\{[a''h''S(a^{(4)})]\}\epsilon(\mathrm{i}')f[S(a''')\mathrm{i}''a^{(7)}x_3'] \tag{A.29}$$

$$|a^{(10)}x_1\ x_4S(a^{(6)})S\{[a'h'S(a^{(5)})]\}\ a^{(8)}x_3''\ x_2S(a^{(9)})\rangle \tag{A.30}$$

$$= \sum_{(\mathrm{i}),\mathrm{i},(x_3),(a),(x_3)} F^{\{[a'hS(a''')]',g_{\mathrm{i}}\}}(\tau_1)\epsilon\{S(\mathrm{i}''')[a'hS(a''')]''\mathrm{i}'\}f[S(a'')\mathrm{i}''(a''')'x_3'] \tag{A.31}$$

$$|a^{(7)}x_1\ x_4S(a^{(4)})\ (a^{(5)})''x_3''\ x_2S(a^{(6)})\rangle \tag{A.32}$$

$$= \sum_{(\mathrm{i}),\mathrm{i},(a),(a'hS(a'''))} F^{\{[a'hS(a''')]',g_{\mathrm{i}}\}}(\tau_1)F^{\{S(\mathrm{i}''')[a'hS(a^{(3)})]''\mathrm{i}',f[S(a'')\mathrm{i}''?]\}}(\tau_2) \tag{A.33}$$

$$|a^{(7)}x_1\ x_4S(a^{(4)})\ a^{(5)}x_3\ x_2S(a^{(6)})\rangle \tag{A.34}$$

$$= \sum_{(\mathrm{i}),(a)} F^{\{a'hS(a'''),f[S(a'')?]\}}(\tau_L)|a^{(7)}x_1\ x_4S(a^{(4)})\ a^{(5)}x_3\ x_2S(a^{(6)})\rangle \tag{A.35}$$

$$= \sum_{(\mathrm{i}),(a)} F^{\{a'hS(a^{(3)}),f[S(a'')?]\}}(\tau_L)A_{a^{(4)}}(s_0)|x_1\ x_4\ x_3\ x_2\rangle \tag{A.36}$$

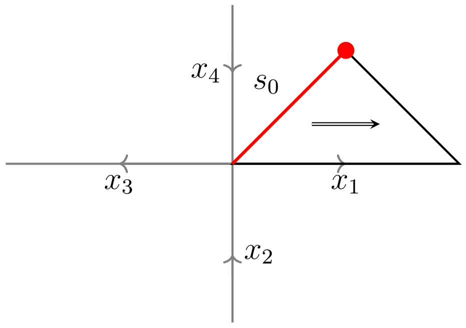
From the fourth line to the fifth line above, we used $\epsilon(g_{\mathrm{i}}) = g_{\mathrm{i}}(\mathrm{e})$ and

$$\sum_{(\mathrm{i}),\mathrm{i}} g_{\mathrm{i}}(a)\mathrm{i}'f(\mathrm{i}'') = \sum_{(a)} a'f(a'').$$

To derive the above equality, note that,

$$\sum_{(a)} a' f(a'') = (Id \otimes f)\Delta(a) = (Id \otimes f)\Delta\left(\sum_{\mathrm{i}} g_{\mathrm{i}}(a)\mathrm{i}\right).$$

A.4.2 Equation 3.17b for short ribbons



$$A_a(s_0)F^{(h,f)}(\tau_R)|x_1\ x_4\ x_3\ x_2\rangle \tag{A.37}$$

$$= A_a(s_0)\sum_{(x_1)} \epsilon(h)f(x_1'')|x_1'\ x_4\ x_3\ x_2\rangle \tag{A.38}$$

$$= \sum_{(x_1),(a)} \epsilon(h)f(x_1'')|a^{(4)}x_1'\ x_4S(a')\ a''x_3\ x_2S(a''')\rangle \tag{A.39}$$

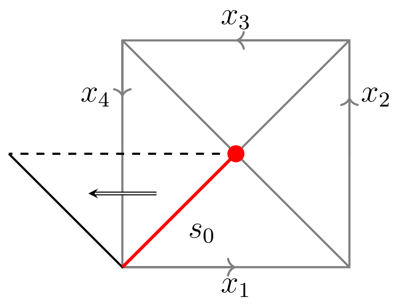
$$= \sum_{(x_1),(a)} \epsilon(a^{(6)})\epsilon(h)\epsilon(a^{(8)})f[S(a^{(7)})a^{(5)}x_1'']|a^{(4)}x_1'\ x_4S(a')\ a''x_3\ x_2S(a''')\rangle \tag{A.40}$$

$$= \sum_{(x_1),(a)} \epsilon[a^{(5)}hS(a^{(7)})]f[S(a^{(6)})(a^{(4)}x_1)'']|(a^{(4)}x_1)'\ x_4S(a')\ a''x_3\ x_2S(a''')\rangle \tag{A.41}$$

$$= \sum_{(a)} F^{\{a^{(5)}hS(a^{(7)}),f[S(a^{(6)})?]\}}(\tau_R)|a^{(4)}x_1\ x_4S(a')\ a''x_3\ x_2S(a''')\rangle \tag{A.42}$$

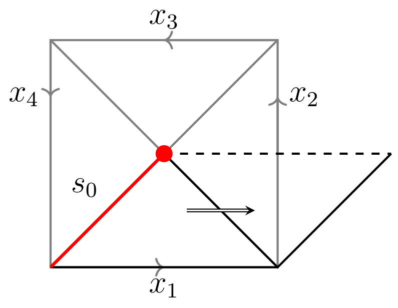
$$= \sum_{(a)} F^{\{a''hS(a^{(4)}),f[S(a''')?]\}}(\tau_R)A_{a'}(s_0)|x_1\ x_4\ x_3\ x_2\rangle \tag{A.43}$$

A.4.3 Equation 3.17c for short ribbons



$$
\begin{aligned}
& B_t(s_0)F^{(h,f)}(\tau_L)|x_1\ x_2\ x_3\ x_4\rangle && \text{(A.44)}\\
&= B_t(s_0)\epsilon(f)|x_1\ x_2\ x_3\ x_4 S(h)\rangle && \text{(A.45)}\\
&= \sum_{(x_{\mathrm{i}}),(h)} \epsilon(f)t[x''_1x''_2x''_3x''_4S(h')]|x'_1\ x'_2\ x'_3\ x'_4S(h'')\rangle && \text{(A.46)}\\
&= \sum_{(x_{\mathrm{i}}),(h)} F^{(h'',f)}(\tau_L)t[x''_1x''_2x''_3x''_4S(h')]|x'_1\ x'_2\ x'_3\ x'_4\rangle && \text{(A.47)}\\
&= \sum_{(h)} F^{(h'',f)}(\tau_L)B_{t[?S(h')]}(s_0)|x_1\ x_2\ x_3\ x_4\rangle && \text{(A.48)}
\end{aligned}
$$

A.4.4 Equation 3.17d for short ribbons



$$B_t(s_0)F^{(h,f)}(\tau_R)|x_1\ x_2\ x_3\ x_4\rangle \tag{A.49}$$

$$= B_t(s_0)\sum_{(h),\mathrm{i},(\mathrm{i})} F^{(h',g_\mathrm{i})}(\tau_1)F^{[S(\mathrm{i}''')h''\mathrm{i}',f(\mathrm{i}''?)]}(\tau_2)|x_1\ x_2\ x_3\ x_4\rangle \tag{A.50}$$

$$= B_t(s_0)\sum_{(h),\mathrm{i},(\mathrm{i})} F^{(h',g_\mathrm{i})}(\tau_1)\epsilon[f(\mathrm{i}''?)]|x_1\ S[S(\mathrm{i}''')h''\mathrm{i}']x_2\ x_3\ x_4\rangle \tag{A.51}$$

$$= B_t(s_0)\sum_{(h),\mathrm{i},(\mathrm{i}),(x_1)} \epsilon(h')g_\mathrm{i}(x_1'')f(\mathrm{i}'')|x_1'\ S(\mathrm{i}')S(h'')\mathrm{i}'''x_2\ x_3\ x_4\rangle \tag{A.52}$$

$$= B_t(s_0)\sum_{(x_1)} f(x_1''')|x_1'\ S(x_1'')S(h)x_1^{(4)}x_2\ x_3\ x_4\rangle \tag{A.53}$$

$$= \sum_{(h),(x_\mathrm{i})} f(x_1^{(5)})t[x_1''S(x_1''')S(h')x_1^{(7)}x_2''x_3''x_4'']|x_1'\ S(x_1^{(4)})S(h'')x_1^{(6)}x_2'\ x_3'\ x_4'\rangle \tag{A.54}$$

$$= \sum_{(h),(x_\mathrm{i})} f(x_1^{(4)})t[\epsilon(x_1'')S(h')x_1^{(6)}x_2''x_3''x_4'']|x_1'\ S(x_1''')S(h'')x_1^{(5)}x_2'\ x_3'\ x_4'\rangle \tag{A.55}$$

$$= \sum_{(h),(x_\mathrm{i})} f(x_1''')t[S(h)x_1^{(5)}x_2''x_3''x_4'']|x_1'\ S(x_1'')S(h'')x_1^{(4)}x_2'\ x_3'\ x_4'\rangle \tag{A.56}$$

$$= \sum_{(h),(x_\mathrm{i}),\mathrm{i},(\mathrm{i})} \epsilon(h'')g_\mathrm{i}(x_1'')f(\mathrm{i}'')t[S(h')x_1''x_2''x_3''x_4'']|x_1'\ S(\mathrm{i}')S(h''')\mathrm{i}'''x_2'\ x_3'\ x_4'\rangle \tag{A.57}$$

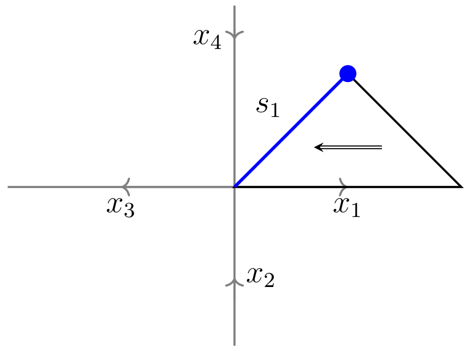
$$= \sum_{(h),(x_\mathrm{i}),\mathrm{i},(\mathrm{i})} F^{(h'',g_\mathrm{i})}(\tau_1)\epsilon[f(\mathrm{i}''?)]t[S(h')x_1''x_2''x_3''x_4'']|x_1'\ S[S(\mathrm{i}''')h'''\mathrm{i}']x_2'\ x_3'\ x_4'\rangle \tag{A.58}$$

$$= \sum_{(h),(x_\mathrm{i}),\mathrm{i},(\mathrm{i})} F^{(h'',g_\mathrm{i})}(\tau_1)F^{[S(\mathrm{i}''')h'''\mathrm{i}',f(\mathrm{i}''?)]}(\tau_2)t[S(h')x_1''x_2''x_3''x_4'']|x_1'\ x_2'\ x_3'\ x_4'\rangle \tag{A.59}$$

$$= \sum_{(h),(x_\mathrm{i})} F^{(h'',f)}(\tau_R)t[S(h')x_1''x_2''x_3''x_4'']|x_1'\ x_2'\ x_3'\ x_4'\rangle \tag{A.60}$$

$$= \sum_{(h)} F^{(h'',f)}(\tau_R)B_{t[S(h')?]}(s_0)|x_1\ x_2\ x_3\ x_4\rangle \tag{A.61}$$

A.4.5 Equation 3.18a for short ribbons



$$A_a(s_1)F^{(h,f)}(\tau_L)|x_1\ x_4\ x_3\ x_2\rangle \tag{A.62}$$

$$= A_a(s_1)\sum_{(x_1)}\epsilon(h)f(x_1'')|x_1'\ x_4\ x_3\ x_2\rangle \tag{A.63}$$

$$= \sum_{(x_1),(a)}\epsilon(h)f[S(x_1'')]|a^{(4)}x_1'\ x_4S(a')\ a''x_3\ x_2S(a''')\rangle \tag{A.64}$$

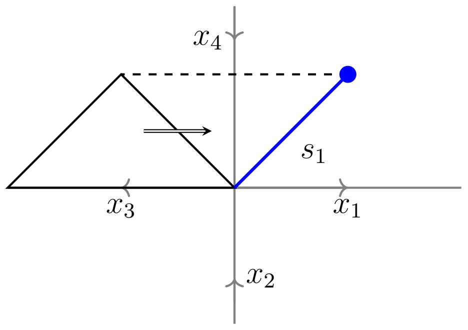
$$= \sum_{(x_1),(a)}\epsilon(h)f[S(x_1'')\epsilon(a^{(5)})]|a^{(4)}x_1'\ x_4S(a')\ a''x_3\ x_2S(a''')\rangle \tag{A.65}$$

$$= \sum_{(x_1),(a)}\epsilon(h)f[S(x_1'')S(a^{(5)})a^{(6)}]|a^{(4)}x_1'\ x_4S(a')\ a''x_3\ x_2S(a''')\rangle \tag{A.66}$$

$$= \sum_{(a)}F^{[h,f(?a^{(5)})]}(\tau_L)|a^{(4)}x_1\ x_4S(a')\ a''x_3\ x_2S(a''')\rangle \tag{A.67}$$

$$= \sum_{(a)}F^{[h,f(?a'')]}(\tau_L)A_{a'}(s_1)|x_1\ x_4\ x_3\ x_2\rangle \tag{A.68}$$

A.4.6 Equation 3.18b for short ribbons



$$A_a(s_1)F^{(h,f)}(\tau_R)|x_1\ x_4\ x_3\ x_2\rangle \tag{A.69}$$

$$= A_a(s_1) \sum_{(\mathrm{i}),\mathrm{i},(h)} F^{(h',g_\mathrm{i})}(\tau_1)F^{[S(\mathrm{i}''')h''\mathrm{i}',f(\mathrm{i}''?)]}(\tau_2)|x_1\ x_4\ x_3\ x_2\rangle \tag{A.70}$$

$$= A_a(s_1) \sum_{(\mathrm{i}),\mathrm{i},(h)} F^{(h',g_\mathrm{i})}(\tau_1)\epsilon[f(\mathrm{i}'?)]|x_1\ x_4 S(\mathrm{i}''')h''\mathrm{i}'\ x_3\ x_2\rangle \tag{A.71}$$

$$= A_a(s_1) \sum_{(\mathrm{i}),\mathrm{i},(h),(x_3)} \epsilon(h')g_\mathrm{i}[S(x_3')]f(\mathrm{i}'')|x_1\ x_4 S(\mathrm{i}''')h''\mathrm{i}'\ x_3''\ x_2\rangle \tag{A.72}$$

$$= \sum_{(x_3)} A_a(s_1)f[S(x_3'')]|x_1\ x_4 x_3' h S(x_3''')\ x_3^{(4)}\ x_2\rangle \tag{A.73}$$

$$= \sum_{(a),(x_3)} f[S(x_3'')]|a^{(4)}x_1\ x_4 x_3' h S(x_3''')S(a')\ a'' x_3^{(4)}\ x_2 S(a''')\rangle \tag{A.74}$$

$$= \sum_{(a),(x_3)} f[S(x_3'')S(a'')a']|a^{(6)}x_1\ x_4 x_3' h S(x_3''')S(a''')\ a^{(4)} x_3^{(4)}\ x_2 S(a^{(5)})\rangle \tag{A.75}$$

$$= \sum_{(a),(x_3)} f[S(a^{(4)}x_3'')a']|a^{(8)}x_1\ x_4 S(a'')a''' x_3' h S(x_3''')S(a^{(5)})\ a^{(6)} x_3^{(4)}\ x_2 S(a^{(7)})\rangle \tag{A.76}$$

$$= \sum_{(a),(\mathrm{i}),\mathrm{i},(h),(x_3)} \epsilon(h')g_\mathrm{i}[S(a''' x_3')]\epsilon[f(\mathrm{i}''?a')] \tag{A.77}$$

$$|a^{(6)}x_1\ x_4 S(a'')S(\mathrm{i}''')h''\mathrm{i}'\ a^{(4)} x_3''\ x_2 S(a^{(5)})\rangle \tag{A.78}$$

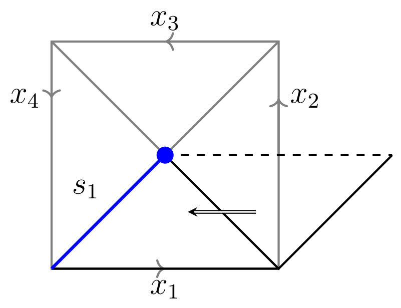
$$= \sum_{(a),(\mathrm{i}),\mathrm{i},(h)} F^{(h',g_\mathrm{i})}(\tau_1)\epsilon[f(\mathrm{i}''?a')]|a^{(5)}x_1\ x_4 S(a'')S(\mathrm{i}''')h''\mathrm{i}'\ a''' x_3\ x_2 S(a^{(4)})\rangle \tag{A.79}$$

$$= \sum_{(a),(\mathrm{i}),\mathrm{i},(h)} F^{(h',g_\mathrm{i})}(\tau_1)F^{[S(\mathrm{i}''')h''\mathrm{i}',f(\mathrm{i}''?a')]}(\tau_2)|a^{(5)}x_1\ x_4 S(a'')\ a''' x_3\ x_2 S(a^{(4)})\rangle \tag{A.80}$$

$$= \sum_{(a)} F^{[h,f(?a')]}(\tau_R)|a^{(5)}x_1\ x_4 S(a'')\ a''' x_3\ x_2 S(a^{(4)})\rangle \tag{A.81}$$

$$= \sum_{(a)} F^{[h,f(?a')]}(\tau_R)A_{a''}(s_1)|x_1\ x_4\ x_3\ x_2\rangle \tag{A.82}$$

A.4.7 Equation 3.18c for short ribbons



$$B_t(s_1)F^{(h,f)}(\tau_L)|x_1\ x_2\ x_3\ x_4\rangle \tag{A.83}$$

$$= B_t(s_1) \sum_{(h),\mathrm{i},(\mathrm{i})} F^{(h',g_{\mathrm{i}})}(\tau_1)F^{[S(\mathrm{i}''')h''\mathrm{i}',f(\mathrm{i}''?)]}(\tau_2)|x_1\ x_2\ x_3\ x_4\rangle \tag{A.84}$$

$$= B_t(s_1) \sum_{(h),\mathrm{i},(\mathrm{i}),(x_1)} F^{(h',g_{\mathrm{i}})}(\tau_1)\epsilon[S(\mathrm{i}''')h''\mathrm{i}']f(\mathrm{i}''x_1')|x_1''\ x_2\ x_3\ x_4\rangle \tag{A.85}$$

$$= B_t(s_1) \sum_{(h),\mathrm{i},(\mathrm{i}),(x_1)} \epsilon(g_{\mathrm{i}})\epsilon(\mathrm{i}''')\epsilon(h'')\epsilon(\mathrm{i}')f(\mathrm{i}''x_1')|x_1''\ x_2S(h')\ x_3\ x_4\rangle \tag{A.86}$$

$$= \sum_{(x_1)} B_t(s_1)f(x_1')|x_1''\ x_2S(h)\ x_3\ x_4)\rangle \tag{A.87}$$

$$= \sum_{(x_{\mathrm{i}}),(h)} f(x_1')t[S(x_1'')h''S(x_2')S(x_3')S(x_4')]|x_1'''\ x_2''S(h')\ x_3''\ x_4''\rangle \tag{A.88}$$

$$= \sum_{(x_{\mathrm{i}}),(h)} f(x_1''')t[S(x_1^{(4)})h''x_1''S(x_1')S(x_2')S(x_3')S(x_4')]|x_1^{(5)}\ x_2''S(h')\ x_3''\ x_4''\rangle \tag{A.89}$$

$$= \sum_{(x_{\mathrm{i}}),\mathrm{i},(\mathrm{i}),(h)} f(\mathrm{i}'')g_{\mathrm{i}}(x_1'')t[S(\mathrm{i}''')h''\mathrm{i}'S(x_1')S(x_2')S(x_3')S(x_4')]|x_1'''\ x_2''S(h')\ x_3''\ x_4''\rangle \tag{A.90}$$

$$= \sum_{(x_{\mathrm{i}}),\mathrm{i},(\mathrm{i}),\mathrm{j},(\mathrm{j}),(h)} f(\mathrm{i}'')\epsilon(g_{\mathrm{i}})\epsilon(\mathrm{j}''')\epsilon(h'')\epsilon(\mathrm{j}')g_{\mathrm{i}}(\mathrm{j}''x_1'') \tag{A.91}$$

$$t[S(\mathrm{i}''')h''\mathrm{i}'S(x_1')S(x_2')S(x_3')S(x_4')]|x_1'''\ x_2''S(h')\ x_3''\ x_4''\rangle \tag{A.92}$$

$$= \sum_{(x_{\mathrm{i}}),\mathrm{i},(\mathrm{i}),\mathrm{j},(\mathrm{j}),(h)} f(\mathrm{i}'')F^{(h',g_{\mathrm{i}})}(\tau_1)\epsilon[S(\mathrm{j}''')h''\mathrm{j}']g_{\mathrm{i}}(\mathrm{j}''x_1'') \tag{A.93}$$

$$t[S(\mathrm{i}''')h''\mathrm{i}'S(x_1')S(x_2')S(x_3')S(x_4')]|x_1'''\ x_2''\ x_3''\ x_4''\rangle \tag{A.94}$$

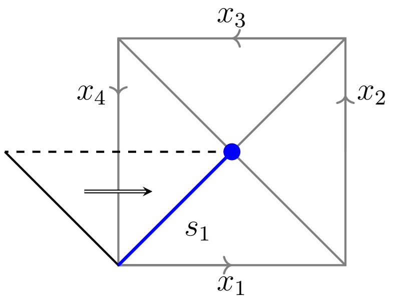
$$= \sum_{(x_{\mathrm{i}}),\mathrm{i},(\mathrm{i}),\mathrm{j},(\mathrm{j}),(h)} f(\mathrm{i}'')F^{(h',g_{\mathrm{i}})}(\tau_1)F^{[S(\mathrm{j}''')h''\mathrm{j}',g_{\mathrm{i}}(\mathrm{j}''?)]}(\tau_2) \tag{A.95}$$

$$t[S(\mathrm{i}''')h''\mathrm{i}'S(x_1')S(x_2')S(x_3')S(x_4')]|x_1''\ x_2''\ x_3''\ x_4''\rangle \tag{A.96}$$

$$= \sum_{(x_{\mathrm{i}}),\mathrm{i},(\mathrm{i}),(h)} f(\mathrm{i}'')F^{(h',g_{\mathrm{i}})}(\tau_L)t[S(\mathrm{i}''')h''\mathrm{i}'S(x_1')S(x_2')S(x_3')S(x_4')]|x_1''\ x_2''\ x_3''\ x_4''\rangle \tag{A.97}$$

$$= \sum_{\mathrm{i},(\mathrm{i}),(h)} f(\mathrm{i}'')F^{(h',g_{\mathrm{i}})}(\tau_L)B_{t[S(\mathrm{i}''')h''\mathrm{i}'?]}(s_1)|x_1\ x_2\ x_3\ x_4\rangle \tag{A.98}$$

A.4.8 Equation 3.18d for short ribbons



$$B_t(s_1)F^{(h,f)}(\tau_R)|x_1\ x_2\ x_3\ x_4\rangle \tag{A.99}$$

$$= B_t(s_1)\epsilon(f)|x_1\ x_2\ x_3\ x_4S(h)\rangle \tag{A.100}$$

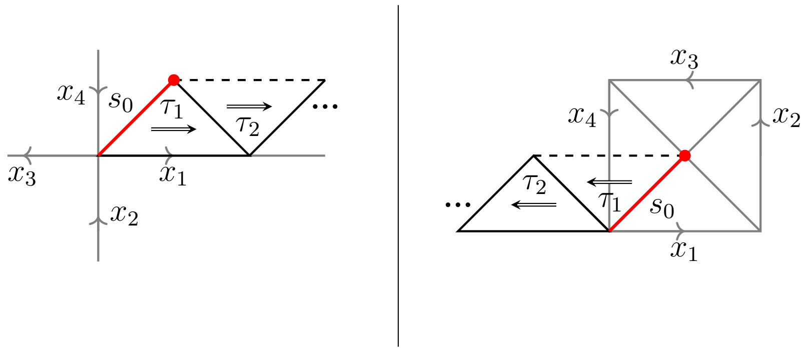
$$= \sum_{(x_\mathrm{i}),(h)} f(\mathrm{e})t(x_1''x_2''x_3''x_4''h'')|x_1'\ x_2'\ x_3'\ x_4'h'\rangle \tag{A.101}$$

$$= \sum_{(x_\mathrm{i}),(h),\mathrm{i},(\mathrm{i})} f(\mathrm{i}'')\epsilon(g_\mathrm{i})t[x_1''x_2''x_3''x_4''S(\mathrm{i}''')h''\mathrm{i}'](s_1)|x_1'\ x_2'\ x_3'\ x_4'h'\rangle \tag{A.102}$$

$$= \sum_{(x_\mathrm{i}),(h),\mathrm{i},(\mathrm{i})} f(\mathrm{i}'')F^{(h',g_\mathrm{i})}(\tau_R)t[x_1''x_2''x_3''x_4''S(\mathrm{i}''')h''\mathrm{i}'](s_1)|x_1'\ x_2'\ x_3'\ x_4'\rangle \tag{A.103}$$

$$= \sum_{(h),\mathrm{i},(\mathrm{i})} f(\mathrm{i}'')F^{(h',g_\mathrm{i})}(\tau_R)B_{t[?S(\mathrm{i}''')h''\mathrm{i}']}(s_1)|x_1\ x_2\ x_3\ x_4\rangle \tag{A.104}$$

A.4.9 Equations 3.17b and 3.17c for long ribbons



For the left figure above, we have,

$$
\begin{aligned}
&A_a(s_0)F^{(h,f)}(\tau_R) && \text{(A.105)}\\
&= \sum_{(h),\mathrm{i},(\mathrm{i})} A_a(s_0)F^{(h',g_{\mathrm{i}})}(\tau_1)F^{[S(\mathrm{i}''')h''\mathrm{i}',f(\mathrm{i}''?)]}(\tau_2) && \text{(A.106)}\\
&= \sum_{(a),(h),\mathrm{i},(\mathrm{i})} F^{\{a''h'S(a^{(4)}),g_{\mathrm{i}}[S(a''')?]\}}(\tau_1)A_{a''}(s_0)F^{[S(\mathrm{i}''')h''\mathrm{i}',f(\mathrm{i}''?)]}(\tau_2) && \text{(A.107)}\\
&= \sum_{(a),(h),\mathrm{i},(\mathrm{i})} F^{\{a''h'S(a^{(4)}),g_{\mathrm{i}}[S(a''')?]\}}(\tau_1)F^{[S(\mathrm{i}''')h''\mathrm{i}',f(\mathrm{i}''?)]}(\tau_2)A_{a'}(s_0) && \text{(A.108)}\\
&= \sum_{(a),(h),\mathrm{i},(\mathrm{i}),\mathrm{j}} F^{\{a''h'S(a^{(4)}),g_{\mathrm{i}}[S(a''')\mathrm{j}]g_{\mathrm{j}}(?)\}}(\tau_1)F^{[S(\mathrm{i}''')h''\mathrm{i}',f(\mathrm{i}''?)]}(\tau_2)A_{a''}(s_0) && \text{(A.109)}\\
&= \sum_{(a),(h),\mathrm{j},(\mathrm{j})} F^{[a''h'S(a^{(6)}),g_{\mathrm{j}}]}(\tau_1)F^{\{S(\mathrm{j}''')a'''h''S(a^{(5)})\mathrm{j}',f[S(a^{(4)})\mathrm{j}''?]\}}(\tau_2)A_{a'}(s_0) && \text{(A.110)}\\
&= \sum_{(a),(h)} F^{\{a''hS(a^{(4)}),f[S(a''')?]\}}(\tau_R)A_{a'}(s_0) && \text{(A.111)}
\end{aligned}
$$

From the forth line to the fifth line in the above equation, we need to use

$$
g_{\mathrm{i}}(a\ b) = g_{\mathrm{i}}\left[a\sum_{\mathrm{j}} g_{\mathrm{j}}(b)\mathrm{j}\right] = \sum_{\mathrm{j}} g_{\mathrm{i}}(a\ \mathrm{j})g_{\mathrm{j}}(b) \tag{A.112}
$$

$$
\Longrightarrow g_{\mathrm{i}}(a\ ?) = \sum_{\mathrm{j}} g_{\mathrm{i}}(a\ \mathrm{j})g_{\mathrm{j}}(?). \tag{A.113}
$$

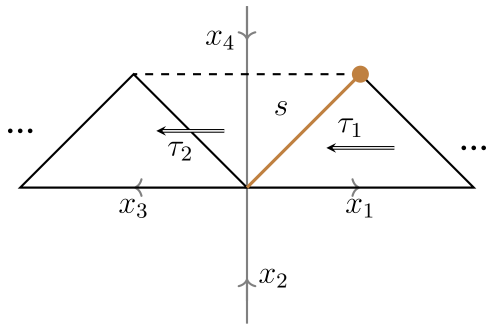
For the right figure above,

$$
\begin{aligned}
&B_t(s_0)F^{(h,f)}(\tau_L) && \text{(A.114)}\\
&= \sum_{(h),\mathrm{i},(\mathrm{i})} B_t(s_0)F^{(h',g_{\mathrm{i}})}(\tau_1)F^{[S(\mathrm{i}''')h''\mathrm{i}',f(\mathrm{i}''?)]}(\tau_2) && \text{(A.115)}\\
&= \sum_{(h),\mathrm{i},(\mathrm{i})} F^{(h'',g_{\mathrm{i}})}(\tau_1)B_{t[?S(h')]}(s_0)F^{[S(\mathrm{i}''')h'''\mathrm{i}',f(\mathrm{i}''?)]}(\tau_2) && \text{(A.116)}\\
&= \sum_{(h),\mathrm{i},(\mathrm{i})} F^{(h'',g_{\mathrm{i}})}(\tau_1)F^{[S(\mathrm{i}''')h'''\mathrm{i}',f(\mathrm{i}''?)]}(\tau_2)B_{t[?S(h')]}(s_0) && \text{(A.117)}\\
&= \sum_{(h)} F^{(h'',f)}(\tau_L)B_{t[?S(h')]}(s_0) && \text{(A.118)}
\end{aligned}
$$

A.5 Proof of Ribbon operator in middle

We are going to talk about Hamiltonian terms where a is the Haar integral of H and t is the Haar integral of H^* temporarily. Notice that the Haar integral is cocomutative, and so we can cyclically rotate the components a', a'', a''', etc. Below we prove the commutation relation for locally clockwise ribbons, and leave the details for locally counterclockwise ribbons to the reader.

A.5.1 Equation 3.19a



$$A_a(s)F^{h,f}(\tau_L) \tag{A.119}$$

$$= \sum_{(h),\mathrm{i},(\mathrm{i})} A_a(s)F^{(h',g_{\mathrm{i}})}(\tau_1)F^{[S(\mathrm{i}''')h''\mathrm{i}',f(\mathrm{i}''?)]}(\tau_2) \tag{A.120}$$

$$= \sum_{(h),\mathrm{i},(\mathrm{i}),(a)} F^{[h',g_{\mathrm{i}}(?a'')]}(\tau_1)A_{a'}(s)F^{[S(\mathrm{i}''')h''\mathrm{i}',f(\mathrm{i}''?)]}(\tau_2) \tag{A.121}$$

$$= \sum_{(h),\mathrm{i},(\mathrm{i}),(a)} F^{[h',g_{\mathrm{i}}(?a^{(5)})]}(\tau_1)F^{\{a'S(\mathrm{i}''')h''\mathrm{i}'S(a'''),f[\mathrm{i}''S(a'')?]\}}(\tau_2)A_{a^{(4)}}(s) \tag{A.122}$$

$$= \sum_{(h),\mathrm{i},(\mathrm{i}),(a),\mathrm{j}} F^{[h',g_{\mathrm{i}}(\mathrm{j}a^{(5)})g_{\mathrm{j}}(?)]}(\tau_1)F^{\{a'S(\mathrm{i}''')h''\mathrm{i}'S(a'''),f[\mathrm{i}''S(a'')?]\}}(\tau_2)A_{a^{(4)}}(s) \tag{A.123}$$

$$= \sum_{(h),(a),\mathrm{j},(\mathrm{j})} F^{(h',g_{\mathrm{j}})}(\tau_1)F^{\{a'S(a^{(7)})S(\mathrm{j}''')h''\mathrm{j}'a^{(5)}S(a'''),f[\mathrm{j}''a^{(6)}S(a'')?]\}}(\tau_2)A_{a^{(4)}}(s) \tag{A.124}$$

$$= \sum_{(h),(a),\mathrm{j},(\mathrm{j})} F^{(h',g_{\mathrm{j}})}(\tau_1)F^{\{S(\mathrm{j}''')h''\mathrm{j}'a^{(4)}S(a''),f[\mathrm{j}''a^{(5)}S(a')?]\}}(\tau_2)A_{a'''}(s) \tag{A.125}$$

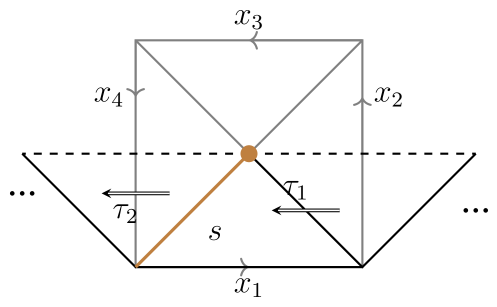
$$= \sum_{(h),(a),\mathrm{j},(\mathrm{j})} F^{(h',g_\mathrm{j})}(\tau_1)F^{\{S(\mathrm{j}''')h''\mathrm{j}'a^{(3)}S(a'),f[\mathrm{j}''?]\}}(\tau_2)A_{a''}(s) \tag{A.126}$$

$$= \sum_{(h),\mathrm{j},(\mathrm{j})} F^{(h',g_\mathrm{j})}(\tau_1)F^{\{S(\mathrm{j}''')h''\mathrm{j}',f[\mathrm{j}''?]\}}(\tau_2)A_a(s) \tag{A.127}$$

$$= F^{h,f}(\tau_L)A_a(s) \tag{A.128}$$

From the sixth line to the end in the above equation, we used the cocomutative condition of $a \in H$, the Haar integral of H. So we can rotate a' to $a^{(n_{max})}$ and $a^{(n)}$ to $a^{(n-1)}$ for $n > 1$. After the rotation, we obtain $\epsilon(a^{(n)})$ to lower the maximum order step by step.

A.5.2 Equation 3.19b



$$B_t(s)F^{(h,f)}(\tau_L) \tag{A.129}$$

$$= \sum_{(h),\mathrm{i},(\mathrm{i})} B_t(s)F^{(h',g_\mathrm{i})}(\tau_1)F^{[S(\mathrm{i}''')h''\mathrm{i}',f(\mathrm{i}''?)]}(\tau_2) \tag{A.130}$$

$$= \sum_{(h),\mathrm{i},(\mathrm{i}),\mathrm{j},(\mathrm{j})} g_\mathrm{i}(\mathrm{j}'')F^{(h',g_\mathrm{j})}(\tau_1)B_{t[S(\mathrm{j}''')h''\mathrm{j}'?]}(S)F^{[S(\mathrm{i}''')h'''\mathrm{i}',f(\mathrm{i}''?)]}(\tau_2) \tag{A.131}$$

$$= \sum_{(h),\mathrm{i},(\mathrm{i}),\mathrm{j},(\mathrm{j})} g_\mathrm{i}(\mathrm{j}'')F^{(h',g_\mathrm{j})}(\tau_1)F^{[S(\mathrm{i}^{(4)})h^{(4)}\mathrm{i}'',f(\mathrm{i}'''?)]}(\tau_2) \tag{A.132}$$

$$B_{t[S(\mathrm{j}''')h''\mathrm{j}'?S(\mathrm{i}')S(h''')\mathrm{i}^{(5)}]}(s) \tag{A.133}$$

$$= \sum_{(h),\mathrm{j},(\mathrm{j})} F^{(h',g_\mathrm{j})}(\tau_1)F^{[S(\mathrm{j}^{(5)})h^{(4)}\mathrm{j}''',f(\mathrm{j}^{(4)}?)]}(\tau_2)B_{t[S(\mathrm{j}^{(7)})h''\mathrm{j}'?S(\mathrm{j}'')S(h''')\mathrm{j}^{(6)}]}(s) \tag{A.134}$$

$$= \sum_{(h),\mathrm{j},(\mathrm{j})} F^{(h',g_{\mathrm{j}})}(\tau_1)F^{[S(\mathrm{j}^{(5)})h^{(4)}\mathrm{j}''',f(\mathrm{j}^{(4)}?)]}(\tau_2)B_{t[S(\mathrm{j}'')S(h''')\mathrm{j}^{(6)}S(\mathrm{j}^{(7)})h''\mathrm{j}'?]}(s) \tag{A.135}$$

$$= \sum_{(h),\mathrm{j},(\mathrm{j})} F^{(h',g_{\mathrm{j}})}(\tau_1)F^{[S(\mathrm{j}''')h''\mathrm{j}',f(\mathrm{j}''?)]}(\tau_2)B_t(s) \tag{A.136}$$

$$= F^{(h,f)}(\tau_L)B_t(s) \tag{A.137}$$

Similarly, from the last third line to the last second line, we used the cocomutative condition of $t \in H^*$, the Haar integral of H^*.

A.6 Fourier transformation of H^*

Let H be any finite dimensional $\mathbb{C}^*$ Hopf algebra. First, we define a Fourier transformation on H [15]:

$$|\nu ab\rangle = \sqrt{\frac{dim(\nu)}{dim(H)}} \sum_{(h_0)} D^\nu(h_0')_{ab}h_0'', \quad \nu \in \mathrm{Irr}_H, a, b = 1, \cdots, dim(\nu),$$

where Irr_H is the set of irreducible representations of H, and $D^\nu(h_0')_{ab}$ is the matrix entry of h_0' for the representation ν under a chosen (fixed) basis.

Recall from Section 2.2 that there are two commuting actions, L and R, of H on itself corresponding to multiplication on the left and multiplication on the right by $S(\cdot)$, respectively. We check the form of the two actions under the Fourier basis.

For an element $m \in H$, the action $L(m)$ is

$$L(m)|\nu ab\rangle = \sqrt{\frac{dim(\nu)}{dim(H)}} \sum_{(h_0)} D^\nu(h_0')_{ab}mh_0'' \tag{A.138}$$

$$= \sqrt{\frac{dim(\nu)}{dim(H)}} \sum_{(h_0),(m)} D^\nu(h_0')_{ab}m'\epsilon(m'')h_0'' \tag{A.139}$$

$$= \sqrt{\frac{dim(\nu)}{dim(H)}} \sum_{(h_0),(m)} D^\nu(h_0')_{ab}m'\epsilon[S(m'')]h_0''. \tag{A.140}$$

As $xh_0 = \epsilon(x)h_0$, we have $\sum_{(x),(h_0)} x'h_0' \otimes x''h_0'' = \epsilon(x)\sum_{(h_0)} h_0' \otimes h_0''$. Applying the above identity for $x = S(m'')$, we obtain

$$L(m)|\nu ab\rangle = \sqrt{\frac{dim(\nu)}{dim(H)}} \sum_{(h_0),(m)} D^\nu[S(m''')h_0']_{ab} m' S(m'')h_0'' \tag{A.141}$$

$$= \sqrt{\frac{dim(\nu)}{dim(H)}} \sum_{(h_0)} D^\nu[S(m)h_0']_{ab} h_0'' \tag{A.142}$$

$$= \sqrt{\frac{dim(\nu)}{dim(H)}} \sum_{(h_0),k} D^\nu[S(m)]_{ak} D^\nu(h_0')_{kb} h_0'' \tag{A.143}$$

$$= \sum_k D^\nu[S(m)]_{ak} |\nu kb\rangle \tag{A.144}$$

$$= \sum_k D^{\nu^*}[m]_{ka} |\nu kb\rangle. \tag{A.145}$$

Similarly, we can obtain the action of $R(m)$:

$$R(m)|\nu ab\rangle = \sum_k D^\nu(m)_{kb} |\nu ak\rangle.$$

Now, take the dual basis $\{\langle \nu ab|\}$ in H^*. L and R each induces a representation on H^*, still denoted by the same letter. Then on the dual basis, the two actions are given by,

$$L(m)(\langle \nu ab|) = \sum_k D^\nu(m)_{ka} \langle \nu kb|, \tag{A.146}$$

$$R(m)(\langle \nu ab|) = \sum_k D^{\nu^*}(m)_{kb} \langle \nu ak|. \tag{A.147}$$

Applying the above dual basis to $D(H)$, we get the basis for Equations 3.26, 3.27.

B. SUPPLEMENTAL MATERIAL FOR CHAPTER 4

This chapter contains work from the article entitled "Quantum circuits for toric code and X-cube fracton model" written by the author, Bowen Yan, and Shawn X. Cui published on Quantum [2].

B.1 2D toric code on sphere

Similar with the example of genus 1 torus, we identify different qubit pairs to change the four plaquettes into a sphere as shown in Figure B.1. The bottom right plaquette is chosen to be redundant and two steps will complete the procedure.

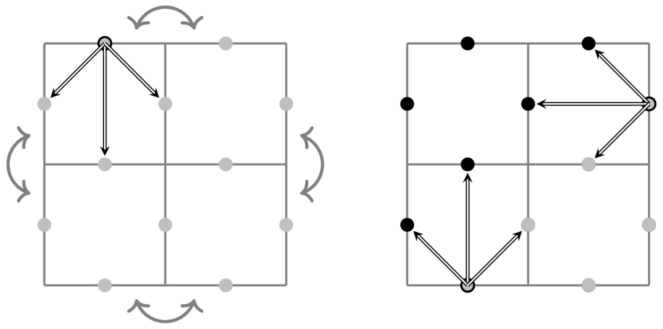


Figure B.1. Boundary edges are identified according to the double-headed arrows.

B.2 2D toric code on genus n surface

Figure B.2 shows a genus n surface which is a disk enclosed by a ribbon with identified edges. Beginning with $|\phi_0\rangle$, we develop a disk from inside and leave the ribbon with all identified edges undeveloped. Then we choose one edge in the ribbon to apply the method of basic structure and repeat in clockwise direction. After $2n - 1$ steps for a genus n torus, we will get the ground state of the closed surface.

B.3 Local CNOT operation

In the preparation of arbitrary state of 2D toric code, we use $CNOT$ to transmit the logical states vertically and horizontally. If we employ non-local CNOT gates, as illustrated

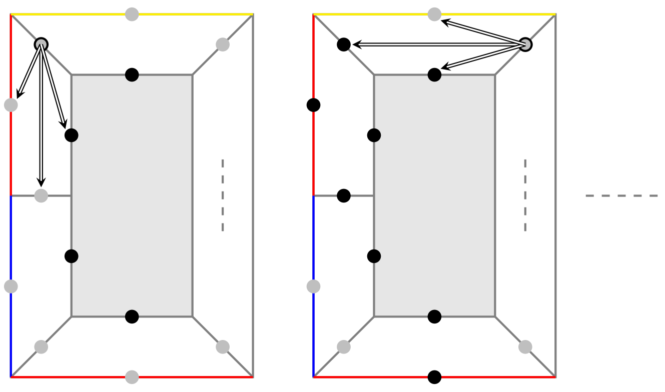


Figure B.2. The shaded area represents the developed disk; Boundaries with the same color are identified to change the plaquettes into a genus n torus.

on the left side of Figure B.3, it takes $\lceil log_2(L) \rceil$ steps. However, when CNOT gates are constrained to constant distances d, this procedure requires $\lceil log_2(d) + \frac{L}{2d} \rceil$ steps, as shown on the right side of Figure B.3. The distance d is defined such that two qubits are considered to be d apart if the shortest path connecting them contains $d - 1$ qubits.

B.4 3D toric model with boundary

The generation from 2D toric code to 3D toric model with boundary is complicated but direct. We can continue to use a plaquette as the basic structure but consider four different types of cubes. Let us take the eight cubes in Figure B.4 as an example. We begin with the red cube and develop it into pink cubes. Orange cubes are the next and the yellow cube completes the model. In the following, we will divide the method into four steps, each step describes one type of cubes.

To develop the qubits in the beginning red cube, we need to develop five rather than six faces as the cube is a closed surface with one redundant face. As shown in Figure B.5, we develop a face first and choose the four qubits on the opposite face to repeat the basic structure. After that, considering the pink cube shares a face with developed cube, we only need to develop four more faces as the second cube is also a closed surface. We choose the four qubits on the face opposite to the developed cube to repeat the basic structure.

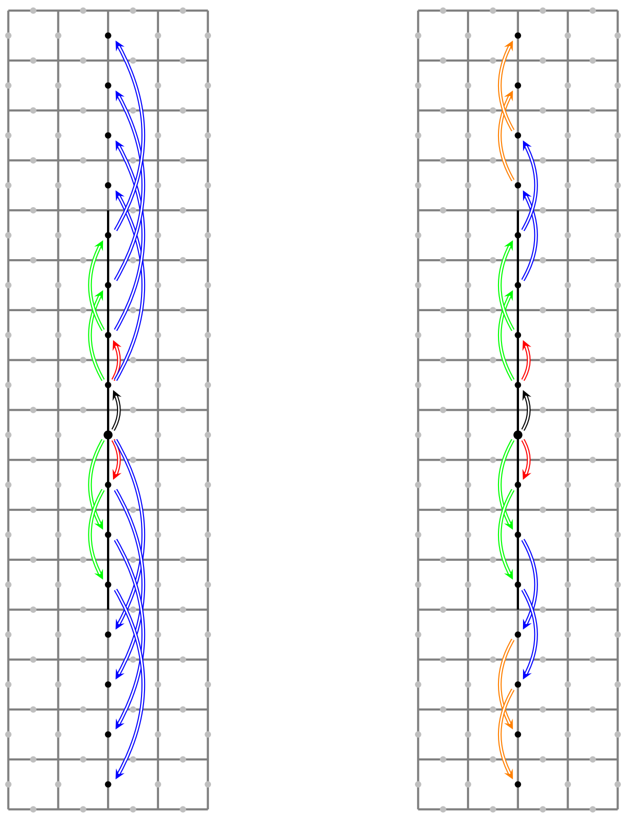


Figure B.3. For the case $L = 16$, we illustrate the utilization of CNOT gates to vertically transmit the logical states in the sequence: black, red, green, blue, orange. On the left-hand side, there exists no constraint on the distance d, permitting the use of non-local CNOT gates, resulting in $log_2(16) = 4$ steps. On the right-hand side, with the restriction of $d = 2$, the process requires $log_2(2) + \frac{16}{4} = 5$ steps.

Similarly, we need to develop three faces for the orange cubes and two faces for the yellow cube as shown in Figure B.6. The four steps complete the procedure to simulate the ground state of toric model on the eight cubes lattices. And we are able to develop any size cubes with boundary using the method described above.

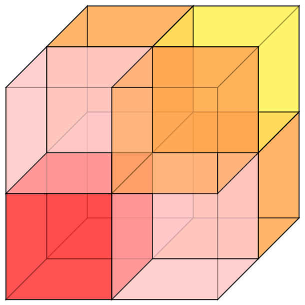


Figure B.4. The beginning cube is colored red. The pink, orange and yellow cube represent the cubes connected with one, two or three faces developed.

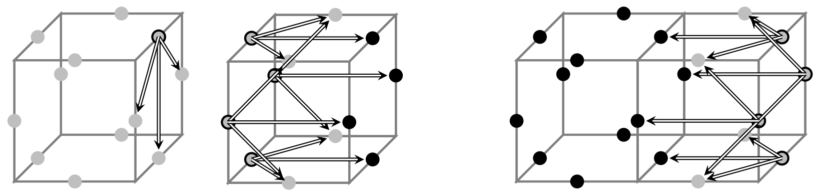


Figure B.5. The left two cubes describe the first step to develop the red cube. The right cubes describes the second step to develop the pink cube.

B.5 3D toric model without boundary

In Figure B.7, the opposite faces are identified together to represent the 3D torus. We begin with $|\phi_0\rangle$ and choose four free qubits in the lower layer to take the procedure in basic structure. After this step and identification of opposite faces, we get the lattice with the middle untouched. Finally, choose three more free qubits to repeat the basic structure and leave a vertex redundant.

B.6 X-cube model simple example

To illustrate the method, we take the eight cubes case as a simple example shown in Figure B.8. Considering the redundant cubes in yellow, we only need to develop four cubes left. The initial state is $|\phi_0\rangle$, and we begin with the cube at the right front higher corner to apply the basic structure. After this step and identifying opposite faces, we get the result

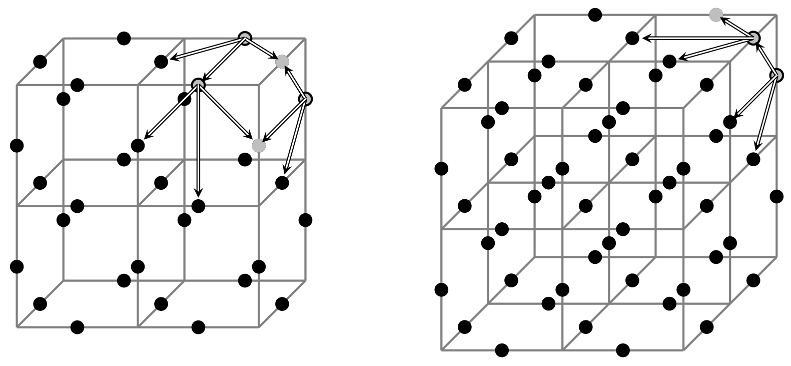


Figure B.6. The left figure describes the step of orange cubes, and we need to develop the face in front first. The right figure describes the final step to develop the yellow cube, and we need to develop the face above first.

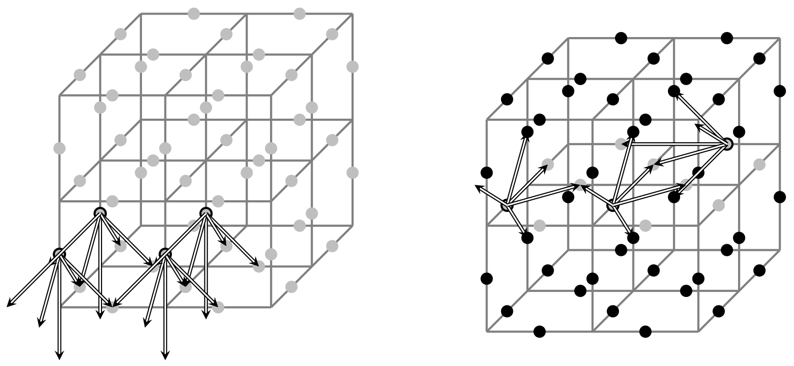


Figure B.7. A qubit $|0\rangle$ is placed at each gray dot at the beginning. The color changes to black when a quantum gate is applied on the qubit.

on the right-hand side of Figure B.8. Then we choose three more free qubits from each cube connecting with the developed cube to repeat the procedure of basic structure and the ground state is completed.

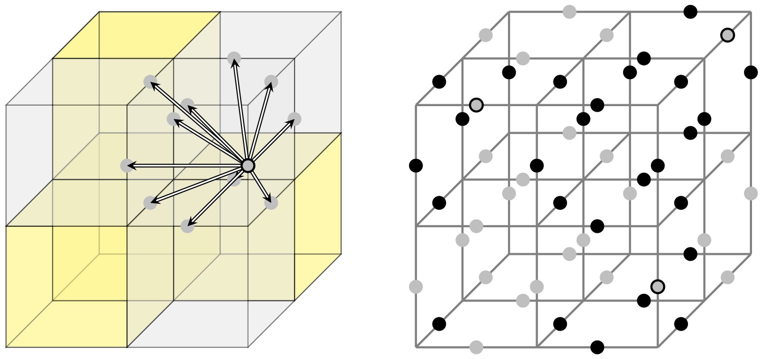


Figure B.8. The left figure is an example of X-cube model with opposite faces identified. The right figure shows the result after the first step and the free qubits for next step are circled.

C. SUPPLEMENTAL MATERIAL FOR CHAPTER 5

This chapter contains work from the article entitled "Representing Arbitrary Ground States of Toric Code by Restricted Boltzmann Machine" written by the author, Bowen Yan, and Shawn X. Cui preprinted on arXiv [3].

C.1 Analytical solution of b_f, $w_{f,\mathrm{j}}$ in the FRRBM

To optimize $|\Psi\rangle = \sum_S \Psi_M(S;\mathcal{W})|S\rangle$ to best represent the ground state $|GS\rangle$, consider the following expression:

$$\Psi_M(S;\mathcal{W}) = \mathrm{e}^{\sum_\mathrm{j} a_\mathrm{j}\sigma^z_\mathrm{j}} \prod_{v\in V} \Gamma_v(S;\mathcal{W}) \prod_{f\in F} \Gamma_f(S;\mathcal{W}), \tag{C.1}$$

$$\Gamma_v(S;\mathcal{W}) = 2\cosh(b_v + \sum_{\mathrm{j}\in s(v)} w_{v,\mathrm{j}}\sigma^z_\mathrm{j}), \tag{C.2}$$

$$\Gamma_f(S;\mathcal{W}) = 2\cosh(b_f + \sum_{\mathrm{j}\in s(f)} w_{f,\mathrm{j}}\sigma^z_\mathrm{j}). \tag{C.3}$$

Setting $a_\mathrm{j} = 0$, we treat $|\Psi\rangle$ as $|GS\rangle$:

$$\begin{aligned}|GS\rangle = \sum_S \mathrm{e}^{\sum_\mathrm{j} a_\mathrm{j}\sigma^z_\mathrm{j}} \prod_{v\in V} 2\cosh(b_v + \sum_{\mathrm{j}\in s(v)} w_{v,\mathrm{j}}\sigma^z_\mathrm{j})\\ \prod_{f\in F} 2\cosh(b_f + \sum_{\mathrm{j}\in s(f)} w_{f,\mathrm{j}}\sigma^z_\mathrm{j})|S\rangle.\end{aligned} \tag{C.4}$$

The stabilizer condition of face operator is examined next:

$$B_f|GS\rangle = \prod_{\mathrm{e}\in s(f)} \hat{\sigma}^z_\mathrm{e}|GS\rangle = |GS\rangle,\ \forall f. \tag{C.5}$$

As the configuration $|S\rangle$ remains unchanged by $\hat{\sigma}^z_\mathrm{e}$, we get:

$$\begin{aligned}&\prod_{\mathrm{e}\in s(f)} \hat{\sigma}^z_\mathrm{e} \mathrm{e}^{\sum_\mathrm{j} a_\mathrm{j}\sigma^z_\mathrm{j}} \prod_{v\in V} \Gamma_v(S;\mathcal{W}) \prod_{f'\in F} \Gamma_{f'}(S;\mathcal{W})\\ &=\mathrm{e}^{\sum_\mathrm{j} a_\mathrm{j}\sigma^z_\mathrm{j}} \prod_{v\in V} \Gamma_v(S;\mathcal{W}) \prod_{f'\in F} \Gamma_{f'}(S;\mathcal{W}),\ \forall f,\ \forall S.\end{aligned} \tag{C.6}$$

All irrelevant terms on both sides are then cancelled:

$$\prod_{\mathrm{e}\in s(f)} \hat{\sigma}^z_{\mathrm{e}} \cosh(b_f + \sum_{\mathrm{j}\in s(f)} w_{f,\mathrm{j}}\sigma^z_{\mathrm{j}}) = \cosh(b_f + \sum_{\mathrm{j}\in s(f)} w_{f,\mathrm{j}}\sigma^z_{\mathrm{j}}),\ \forall f,\ \forall S. \tag{C.7}$$

Due to translation invariance, it is unnecessary to repeat the calculation for all faces. Instead, the possible configurations in a single face contribute 2^4 equations, as illustrated in Figure C.1:

$$\cosh(b - w_1 + w_2 + w_3 + w_4) = 0 \tag{C.8}$$

$$\cosh(b + w_1 - w_2 - w_3 - w_4) = 0 \tag{C.9}$$

$$\cosh(b - w_1 - w_2 + w_3 + w_4) \neq 0 \tag{C.10}$$

$$\cosh(b - w_1 - w_2 - w_3 - w_4) \neq 0 \tag{C.11}$$

$$\vdots$$

Solving these equations yields the complete set of solutions for the face terms b_f, $w_{f,\mathrm{j}}$: $b_f = 0\,(mod\,\pi)$ and $w_{f,\mathrm{j}} = \frac{\pi}{4}\mathrm{i}, \frac{3\pi}{4}\mathrm{i}\,(mod\,\pi)$, where an even number of the four $w_{f,\mathrm{j}}$ must be the same. Since the function of the face terms selectively excludes some configurations, any solution set is valid and can be chosen without loss of generality. In the main article, we choose the isotropic solution $(b_f, w_{f,\mathrm{j}}) = (0, \frac{\pi}{4}\mathrm{i})$.

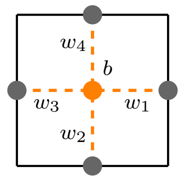


Figure C.1. This lattice diagram represents a translation-invariant structure for a face-type hidden neuron, using simplified notation without the subscript f.

C.2 Analytical solution of b_v, $w_{v,\mathrm{j}}$ in the FRRBM

The face terms B_f typically rule out certain configurations without trivial flux, while the vertex terms A_v ensures all configurations in the same logical state are uniformly weighted. In this appendix, we continue from Equation (5.7) discussed in the main article, focusing on the configurations with trivial flux illustrated in Figures C.3 through C.7. We extract relevant independent equations (C.13) through (C.16) to analytically solve for b_v and $w_{v,\mathrm{j}}$. To simplify notation further in the calculation, we replace cosh with cos and divide all weights by i. We treat b_v as a redundant parameter, similar to a_j, and set $b_v = 0$, as allowing $b_v \in \mathbb{C}$ would introduce superfluous freedom. Further elaboration on this issue is provided at the end.

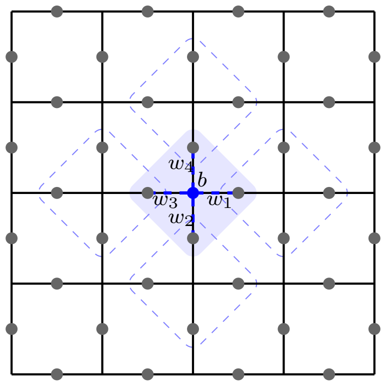


Figure C.2. This lattice diagram represents a translation-invariant structure for a vertex-type hidden neuron, using simplified notation without the subscript v and $(b_v, w_{v,\mathrm{j}}) = \mathrm{i} * (b, w_\mathrm{j})$. If we flip the four qubits surrounding the central vertex, qubits contributing to the phase difference are circled for clarity.

Equation (C.12) defines the often-used phase factor A:

$$\cos(w_1 + w_2 + w_3 + w_4) := A. \tag{C.12}$$

Equation (C.13), the most discussed criterion, is abstracted from Figure C.3:

$$\begin{aligned}\cos(-w_1+w_2+w_3+w_4)\cos(w_1-w_2+w_3+w_4)&\\ \cos(w_1+w_2-w_3+w_4)\cos(w_1+w_2+w_3-w_4)&=A^4.\end{aligned}\tag{C.13}$$

Equations (C.14, C.15, C.16) describe squared conditions, while Equations (C.17, C.18) specify additional criteria. All these equations are derived from the configurations shown in Figure C.4 through C.7:

$$\begin{aligned}&[\cos(-w_1-w_2+w_3+w_4)\cos(w_1+w_2-w_3-w_4)]^L=A^{2L}\text{ for any }L\\ &\Rightarrow \cos^2(-w_1-w_2+w_3+w_4)=A^2.\end{aligned}\tag{C.14}$$

$$\begin{aligned}&[\cos(-w_1+w_2+w_3-w_4)\cos(w_1-w_2-w_3+w_4)]^L=A^{2L}\text{ for any }L\\ &\Rightarrow \cos^2(w_1-w_2-w_3+w_4)=A^2.\end{aligned}\tag{C.15}$$

$$\cos(-w_1+w_2-w_3+w_4)\cos(-w_1+w_2+w_3+w_4)\cos(w_1+w_2-w_3+w_4)=A^3$$
$$\cos(w_1-w_2+w_3-w_4)\cos(w_1-w_2+w_3+w_4)\cos(w_1+w_2+w_3-w_4)=A^3$$
$$\Rightarrow \cos^2(w_1-w_2+w_3-w_4)=A^2.\tag{C.16}$$
$$\Rightarrow \cos^2(w_1-w_2+w_3+w_4)\cos^2(w_1+w_2+w_3-w_4)=A^4.\tag{C.17}$$
$$\Rightarrow \cos^2(-w_1+w_2+w_3+w_4)\cos^2(w_1+w_2-w_3+w_4)=A^4.\tag{C.18}$$

Next, we need to solve and discuss Equations (C.12) through (C.18):

$$\begin{aligned}
&\text{Equation (C.12)+(C.14):}\quad w_1+w_2=0 \ \text{ or } \ w_3+w_4=0\\
&\text{Equation (C.12)+(C.15):}\quad w_1+w_4=0 \ \text{ or } \ w_2+w_3=0\\
&\text{Equation (C.12)+(C.16):}\quad w_1+w_3=0 \ \text{ or } \ w_2+w_4=0\\
&\Rightarrow -w_1=w_2=w_3=w_4 \, (\text{mod}\, \frac{\pi}{2}) \text{ and alternations.} \qquad \text{(C.19)}\\
&\text{or} \quad w_1=w_2=w_3=0 \text{ or } \frac{\pi}{4} \, (\text{mod}\, \frac{\pi}{2}) \text{ and alternations.} \qquad \text{(C.20)}
\end{aligned}$$

In the first scenario derived in Equation (C.19), without loss of generality, we can set $w_1=-w+\frac{\pi}{2}m_1$, $w_2=w+\frac{\pi}{2}m_2$, $w_3=w+\frac{\pi}{2}m_3$, and $w_4=w+\frac{\pi}{2}m_4$, where m_1, m_2, m_3, m_4 $\in \mathbb{N}$. Subsequently, the criteria in Equation (C.13) is rewritten as

$$\begin{aligned}
\cos\,[4w+\frac{\pi}{2}(M-2m_1)]\cos\,[\frac{\pi}{2}(M-2m_2)]&\\
\cos\,[\frac{\pi}{2}(M-2m_3)]\cos\,[\frac{\pi}{2}(M-2m_4)]&=\cos^4(2w+\frac{\pi}{2}M), \qquad \text{(C.21)}
\end{aligned}$$

where $M := m_1+m_2+m_3+m_4$. And $\cos\,[\frac{\pi}{2}(M-2m_2)]\neq 0 \ \Rightarrow \ M$ is even, thus the terms with $\frac{\pi}{2}$ above could be rearranged as follows:

$$\begin{aligned}
\cos\,[4w+\frac{\pi}{2}(M-2m_1)] &= \cos(4w)\cos\left[\frac{\pi}{2}(M-2m_1)\right]-\sin(4w)\sin\left[\frac{\pi}{2}(M-2m_1)\right]\\
&= \cos(4w)\cos\left[\frac{\pi}{2}(M-2m_1)\right],
\end{aligned}$$

or vice versa:

$$\cos\,[\frac{\pi}{2}(M-2m_2)]\cos\,[\frac{\pi}{2}(M-2m_3)] = \cos\,[\frac{\pi}{2}(2M-2m_2-2m_3)].$$

Then Equation (C.21) is simplified as

$$\begin{aligned}
\cos(4w)\cos\,[\frac{\pi}{2}(4M-2M)] &= [\,\cos(2w)\cos(\frac{\pi}{2}M)]^4\\
\cos(4w) &= \cos^4(2w). \qquad \text{(C.22)}
\end{aligned}$$

Solving Equation (C.22), we get $\cos(4w) = 1$, implying $w = 0 \,(\mathrm{mod}\, \frac{\pi}{2})$. Considering the definition of A in Equation (C.12), which is not 0, we conclude: **even number of w are $\mathbf{0 \,(mod\, \pi)}$ and the others are $\boldsymbol{\frac{\pi}{2} \,(\mathrm{mod}\, \pi)}$. For these solutions, $\boldsymbol{A}$ is either $\mathbf{+1}$ or $\mathbf{-1}$.**

In the second scenario derived in Equation (C.20), without loss of generality, we can set $w_1 = w_2 = w_3 = w$, $w = 0$ or $\frac{\pi}{4} \,(\mathrm{mod}\, \frac{\pi}{2})$ and w_4 is free. By inserting all possible values of w_1, w_2 and w_3 into Equations (C.13) and (C.17, C.18), we can find the allowed solutions: **three of w equal to $\mathbf{0 \,(mod\, \frac{\pi}{2})}$ and the other serves as a normalization parameter such that $\boldsymbol{\cos(3w + w_4) = A}$.**

In the calculation above, we treat b as a redundant parameter and set $b = 0$. Reader can practice by setting $b = \frac{\pi}{2}$ to obtain another set of solutions, which yields results similar to those above. To understand how allowing $b \in \mathbb{C}$ introduces superfluous freedom, one can abstract Equation (C.23) from Figure C.8 and incorporate b into Equation (C.12). By re-deriving equations (C.13) through (C.16) and substituting them into Equation (C.23), one can obtain Equation (C.24), which represents the restriction on b. There are infinite many possibilities, and above two learnable solutions emerge through the training process introduced in the next section.

$$\begin{aligned}&\cos(b-w_1-w_2+w_3+w_4)\cos(b+w_1-w_2-w_3+w_4)\cos(b+w_1+w_2-w_3-w_4)\\&\cos(b-w_1+w_2+w_3-w_4)\cos^2(b+w_1-w_2-w_3-w_4)\cos^2(b-w_1+w_2-w_3-w_4)\\&\qquad\cos^2(b-w_1-w_2+w_3-w_4)\cos^2(b-w_1-w_2-w_3+w_4) = A^{12}.\end{aligned}\tag{C.23}$$

$$\cos^4(b - w_1 - w_2 - w_3 - w_4) = \cos^4(b + w_1 + w_2 + w_3 + w_4).\tag{C.24}$$

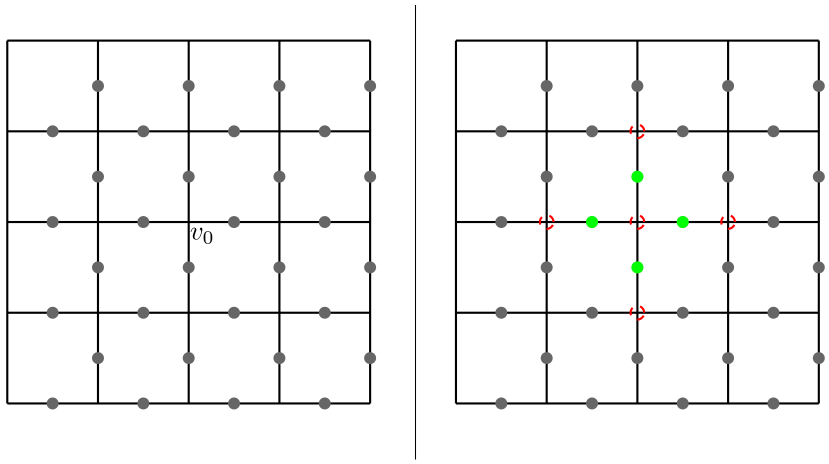


Figure C.3. In this configuration, each black dot represents a qubit in state $|-1\rangle$, while each green dot indicates a qubit in state $|+1\rangle$. Starting with the initial configuration on the left, a vertex operator is applied at vertex v_0 to flip the adjacent four qubits. The resulting configuration, displayed on the right, features five vertices encircled in red that contribute to the phase difference. By comparing the phase contributions from these vertices in both configurations, we derive Equation (C.13), which is a crucial criterion for our calculation.

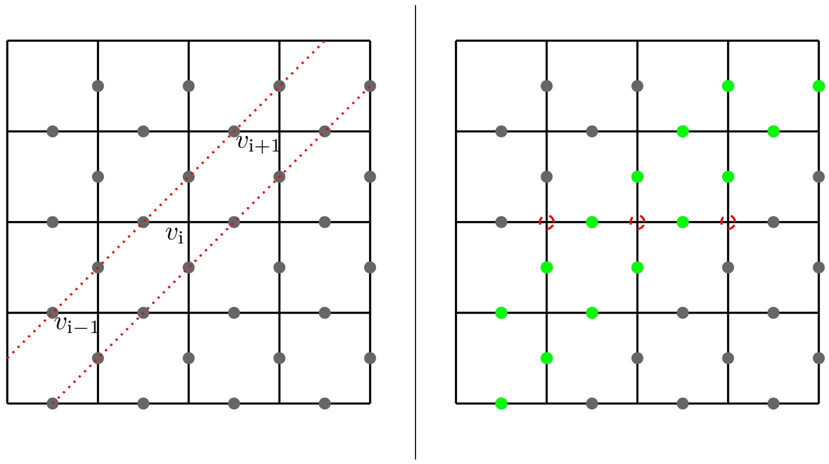


Figure C.4. Starting with the initial configuration illustrated on the left, we apply vertex operators diagonally at vertices v_0. The resultant configuration, showcased on the right, exhibits translational symmetry horizontally. Notably, the three vertices encircled in red contribute to the phase difference. By analyzing the phase contributions from these three vertices in both configurations, we deduce Equation (C.14). This equation represents one of the three pivotal square conditions essential for our calculation.

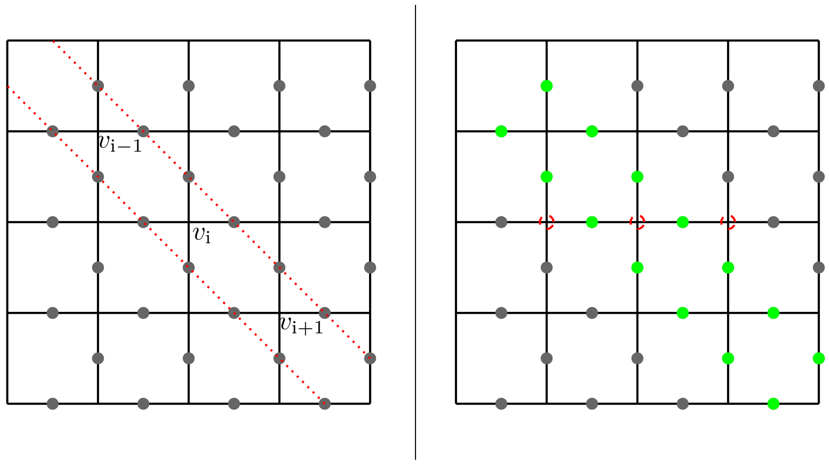


Figure C.5. Similarly, applying vertex operators diagonally in the perpendicular direction, we obtain Equation (C.15), another pivotal square condition.

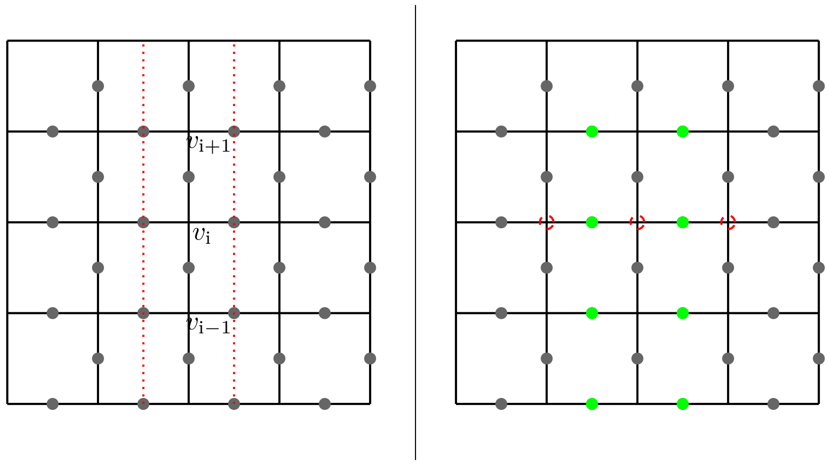


Figure C.6. This configuration corresponds to the first equation to derive the Equation (C.16), the last pivotal square condition.

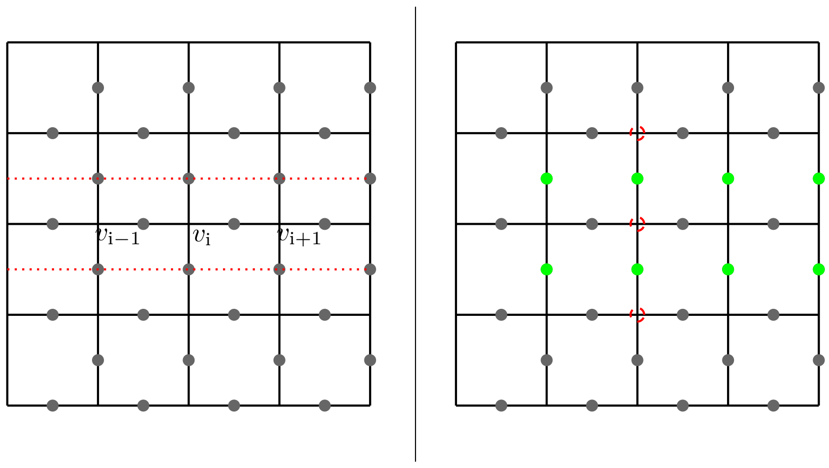


Figure C.7. This configuration corresponds to the second equation to derive the Equation (C.16), the last pivotal square condition.

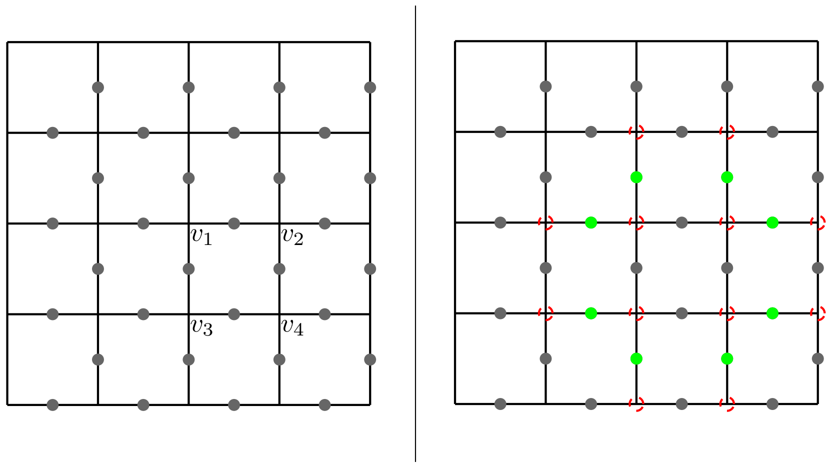


Figure C.8. This configuration corresponds to Equation (C.23), the condition to find the restriction on b_v.

C.3 Machine Learning of the FRRBM

To further elucidate the analytical solutions derived in the main article for the FRRBM illustrated in Figure 5.1, we numerically determine the ground state solution from Equations (2.58) and (5.1) by applying a vertex stabilizer condition on square lattices of various sizes. Namleluy, Figures C.9 and C.10 show their factorization on 3×3 and 4×4 square lattices with different initial settings, where the common setting is $(a_{\mathrm{j}}, b_f, w_{f,\mathrm{j}}, b_v) = (0, 0, \frac{\pi}{4}\mathrm{i}, 0)$, with variations in $w_{v,\mathrm{j}}$ making the difference.

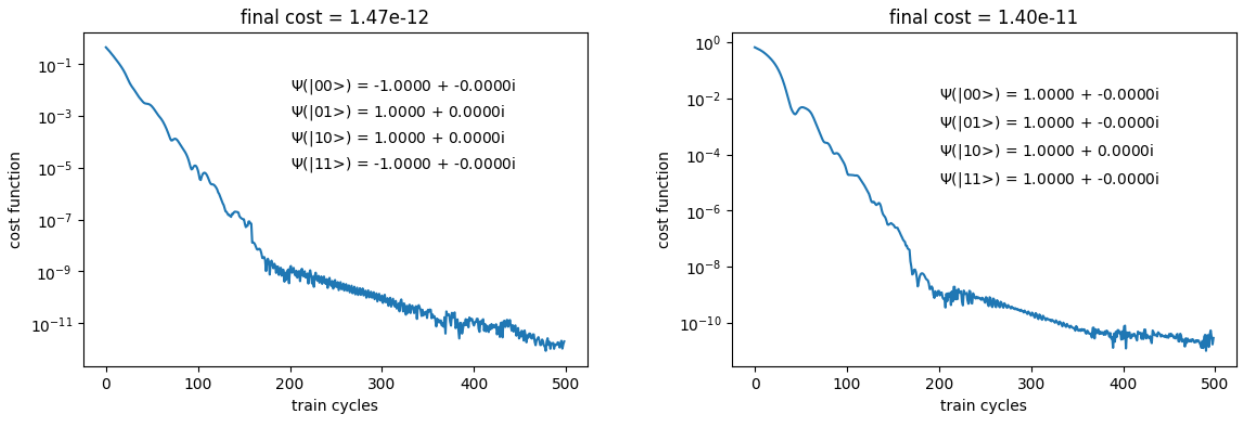
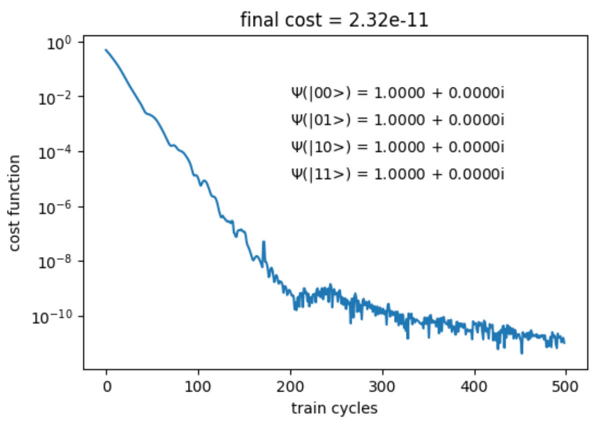


Figure C.9. On a 3×3 lattice, the left plot shows the training result for the isotropic setting $w_{v,\mathrm{j}} = \frac{\pi}{2}\mathrm{i}$, and right for the anisotropic setting $w_{v,\mathrm{j}} = 0, 0, \frac{\pi}{2}\mathrm{i}, \frac{\pi}{2}\mathrm{i}$.



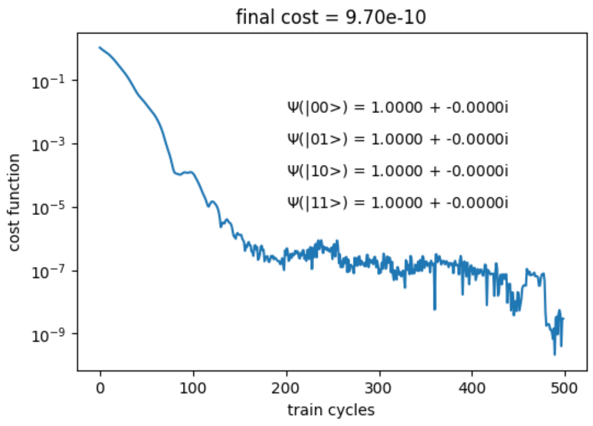


Figure C.10. On a 4×4 lattice, the left plot shows the training result for the isotropic setting $w_{v,\mathrm{j}} = \frac{\pi}{2}\mathrm{i}$, and right for the anisotropic setting $w_{v,\mathrm{j}} = 0, 0, \frac{\pi}{2}\mathrm{i}, \frac{\pi}{2}\mathrm{i}$. Despite the varied interaction settings, both configurations yield identical ground states. This outcome contrasts with the results shown in Figure C.9, where the ground states differ significantly.

C.4 Machine Learning of the RBM

Illustrated in Figure 5.3, we pick the isotropic setting $(a_\mathrm{j}, b_f, w_{f,\mathrm{j}}, b_v, w_{v,\mathrm{j}}) = (0, 0, \frac{\pi}{4}\mathrm{i}, 0, \frac{\pi}{2}\mathrm{i})$ and uniformly weighted every new connection ($w_{x,y,z} = \frac{\pi}{4}\mathrm{i}$). Then three hidden neurons (h_x, h_y, h_z) are introduced into the FRRBM to simulate an arbitrary ground state. Reader can verify that the inclusion of h_x and h_y (inspired by the logical operators Z_v and Z_h) allows for the simulation of any specific degeneracy state, while h_z enables the representation of any arbitrary ground state as a linear combination within the degeneracy basis. Then, using Equations from (5.10) to (5.13), we can analytically solve the weights for arbitrary ground state.

On the other hand, illustrated in C.11, the configurations S_1, S_2, S_3, S_4 are chosen from the equi-positioned configurations of the states $|00\rangle$, $|01\rangle$, $|10\rangle$, $|11\rangle$, respectively. Employing the condition $\langle GS|S1\rangle : \langle GS|S2\rangle : \langle GS|S3\rangle : \langle GS|S4\rangle = \langle GS|00\rangle : \langle GS|01\rangle : \langle GS|10\rangle : \langle GS|11\rangle$, we can numerically train the weights for arbitrary ground state according to the ratio conditions.

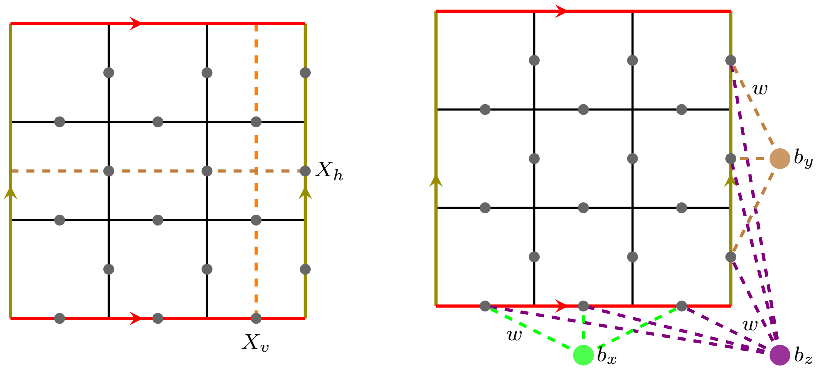


Figure C.11. On the square lattice displayed on the left, we identify four distinct qubit configurations: $S1$, where all qubits are in the $|-1\rangle$ state; $S2$, with qubits only on the vertical dashed loop in the $|1\rangle$ state, namely $|S2\rangle = X_v|S1\rangle$; similarly $|S3\rangle = X_h|S1\rangle$; and $|S4\rangle = X_hX_v|S1\rangle$. The weights of interest are illustrated on the right.

Let us consider a straightforward example involving the degeneracy state $|00\rangle$. We employ the condition $\langle GS|S1\rangle : \langle GS|S2\rangle : \langle GS|S3\rangle : \langle GS|S4\rangle = 1:0:0:0$ to analytically determine the weights, yielding in $(b_x, b_y, b_z) = (\frac{3\pi}{4}\mathrm{i}, \frac{3\pi}{4}\mathrm{i}, \frac{\pi}{2}\mathrm{i})$. Subsequently, we verify the learnability of the RBM, as illustrated in Figure C.12, ensuring that it can accurately and efficiently represent the specified state characteristics.

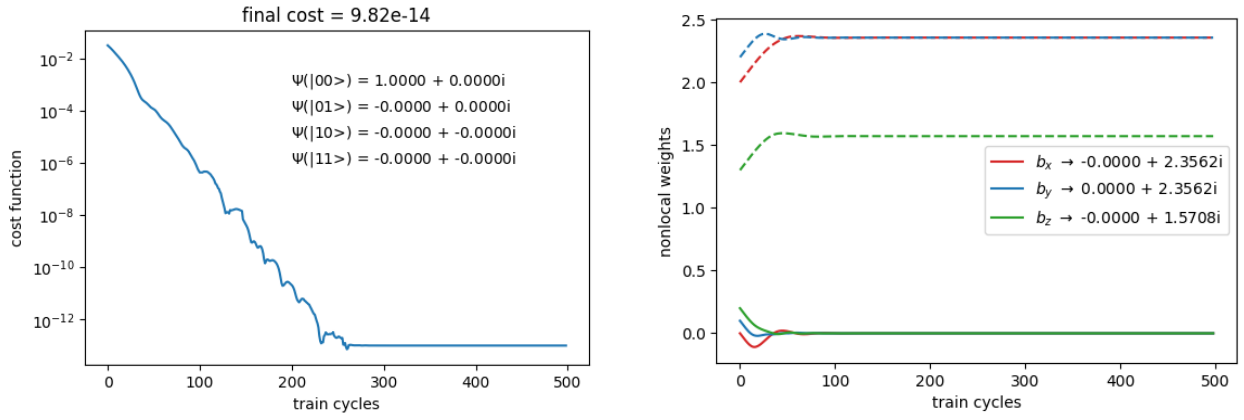


Figure C.12. Training results on a 3×3 lattice for $|GS\rangle = |00\rangle$.

Similarly, another example with amplitude ratios $\langle GS|00\rangle : \langle GS|01\rangle : \langle GS|10\rangle : \langle GS|11\rangle = 1 : 2 : 3 : 4$ results in the solution $(b_x, b_y, b_z) = (\coth^{-1}(2\sqrt{2/3}) + \frac{\pi}{4}\mathrm{i},\ \coth^{-1}(\sqrt{6}) +$

$\frac{\pi}{4}$i, $\coth^{-1}(\sqrt{3/2})$). We then verify the learnability of the RBM, as illustrated in Figure C.13. We find that finer results can be achieved with smaller training step sizes, though extending training time does not lead to significant improvements. Finally, we present an example that can only be approximated, as shown in Figure C.14. We observe that finer results are achievable with smaller training step sizes, and unlike the previous case, longer training times also contribute to better outcomes.

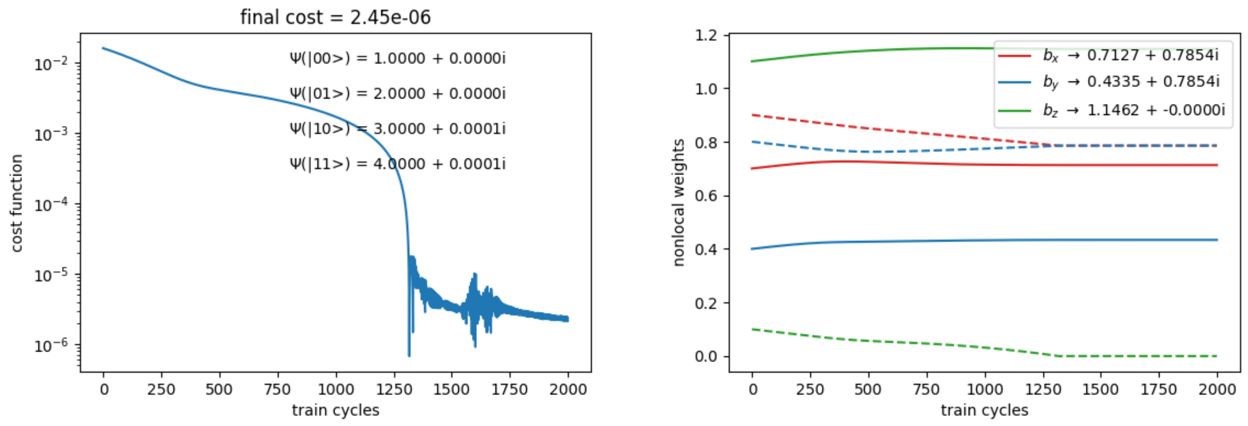


Figure C.13. Training results on a 3×3 lattice for $|GS\rangle = |00\rangle + 2|01\rangle + 3|10\rangle + 4|11\rangle$.

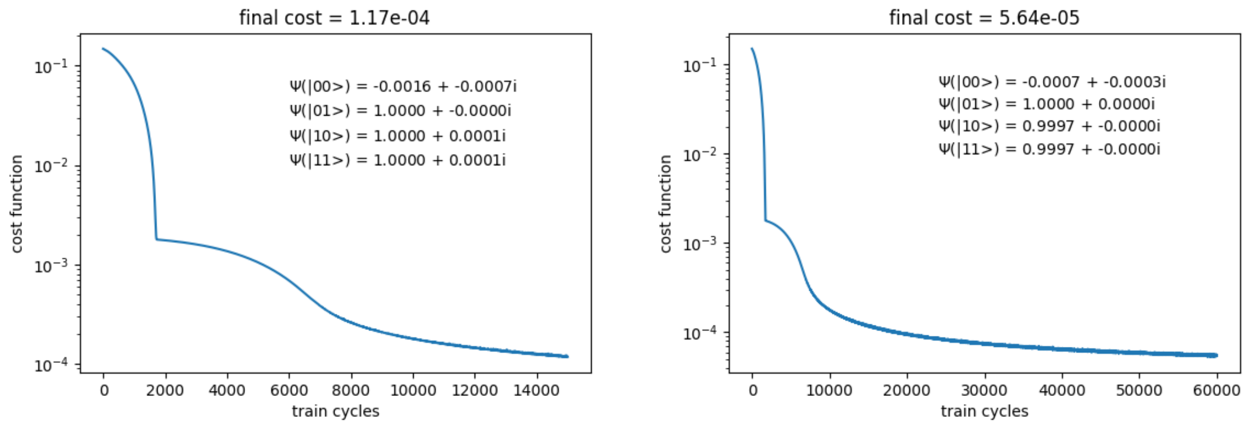


Figure C.14. Training results on a 3×3 lattice for $|GS\rangle = |01\rangle + |10\rangle + |11\rangle$.

C.5 Python code for 2D the FRRBM

```
# This code trains bv and wv in FRRBM
import torch
import numpy as np
import matplotlib.pyplot as plt
import time

# 20 selected for the degeneracy basis and 30 random configurations.
S = np.load('S20_30.npy')
SS = np.array([[-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1],
               [-1, +1, -1, -1, - 1, -1, -1, +1, -1, -1, -1, -1, -1, +1, -1, -1, -1, -1],
               [-1, -1, -1, -1, -1, -1, -1, -1, -1, +1, +1, +1, -1, -1, -1, -1, -1, -1],
               [-1, +1, -1, -1, -1, -1, -1, +1, -1, +1, +1, +1, -1, +1, -1, -1, -1, -1]])


def pr(b, w, s1, s2, s3, s4):
    result = torch.cosh(
        (b[0] + 1j * b[1]) + (w[0] + 1j * w[4]) * s1 + (w[1] + 1j * w[5]) * s2 + (
                w[2] + 1j * w[6]) * s3 + (w[3] + 1j * w[7]) * s4)
    return result


def prr(b, w, s1, s2, s3, s4):
    result = torch.cosh(b[0] + 1j * (w[0] * s1 + w[1] * s2 + w[2] * s3 + w[3] * s4))
    return result


# ph is the phase function.
def ph(bv, wv, bf, wf, s):
    result = (pr(bv, wv, s[0], s[3], s[2], s[15]) * pr(bv, wv, s[1], s[4], s[0], s[16])
              * pr(bv, wv, s[2], s[5], s[1], s[17]) * pr(bv, wv, s[6], s[9], s[8], s[3])
              * pr(bv, wv, s[7], s[10], s[6], s[4]) * pr(bv, wv, s[8], s[11], s[7], s[5])
              * pr(bv, wv, s[12], s[15], s[14], s[9]) * pr(bv, wv, s[13], s[16], s[12], s[10])
              * pr(bv, wv, s[14], s[17], s[13], s[11]) * prr(bf, wf, s[4], s[6], s[3], s[0])
              * prr(bf, wf, s[5], s[7], s[4], s[1]) * prr(bf, wf, s[3], s[8], s[5], s[2])
              * prr(bf, wf, s[10], s[12], s[9], s[6]) * prr(bf, wf, s[11], s[13], s[10], s[7])
              * prr(bf, wf, s[9], s[14], s[11], s[8]) * prr(bf, wf, s[16], s[0], s[15], s[12])
              * prr(bf, wf, s[17], s[1], s[16], s[13]) * prr(bf, wf, s[15], s[2], s[17], s[14]))
    return result


# The 9 vertex operators flip qubits.
```

```
A1 = np.diag([-1, +1, -1, -1, +1, +1, +1, +1, +1, +1, +1, +1, +1, +1, +1, -1, +1, +1])
A2 = np.diag([-1, -1, +1, +1, -1, +1, +1, +1, +1, +1, +1, +1, +1, +1, +1, +1, -1, +1])
A3 = np.diag([+1, -1, -1, +1, +1, -1, +1, +1, +1, +1, +1, +1, +1, +1, +1, +1, +1, -1])
A4 = np.diag([+1, +1, +1, -1, +1, +1, -1, +1, -1, -1, +1, +1, +1, +1, +1, +1, +1, +1])
A5 = np.diag([+1, +1, +1, +1, -1, +1, -1, -1, +1, +1, -1, +1, +1, +1, +1, +1, +1, +1])
A6 = np.diag([+1, +1, +1, +1, +1, -1, +1, -1, -1, +1, +1, -1, +1, +1, +1, +1, +1, +1])
A7 = np.diag([+1, +1, +1, +1, +1, +1, +1, +1, +1, -1, +1, +1, -1, +1, -1, -1, +1, +1])
A8 = np.diag([+1, +1, +1, +1, +1, +1, +1, +1, +1, +1, -1, +1, -1, -1, +1, +1, -1, +1])
A9 = np.diag([+1, +1, +1, +1, +1, +1, +1, +1, +1, +1, +1, -1, +1, -1, -1, +1, +1, -1])
S1 = np.dot(S, A1)
S2 = np.dot(S, A2)
S3 = np.dot(S, A3)
S4 = np.dot(S, A4)
S5 = np.dot(S, A5)
S6 = np.dot(S, A6)
S7 = np.dot(S, A7)
S8 = np.dot(S, A8)
S9 = np.dot(S, A9)


def criterion(bv, wv, bf, wf):
    V = torch.empty((18 * N,), dtype=torch.complex64)
    nor = torch.empty((N,), dtype=torch.complex64)
    for k in range(N):
        sub = ph(bv, wv, bf, wf, S[k])
        nor[k] = sub
        V[k] = ph(bv, wv, bf, wf, S1[k]) - sub
        V[1 * N + k] = ph(bv, wv, bf, wf, S2[k]) - sub
        V[2 * N + k] = ph(bv, wv, bf, wf, S3[k]) - sub
        V[3 * N + k] = ph(bv, wv, bf, wf, S4[k]) - sub
        V[4 * N + k] = ph(bv, wv, bf, wf, S5[k]) - sub
        V[5 * N + k] = ph(bv, wv, bf, wf, S6[k]) - sub
        V[6 * N + k] = ph(bv, wv, bf, wf, S7[k]) - sub
        V[7 * N + k] = ph(bv, wv, bf, wf, S8[k]) - sub
        V[8 * N + k] = ph(bv, wv, bf, wf, S9[k]) - sub
        V[9 * N + k] = (S[k][4] * S[k][6] * S[k][3] * S[k][0] - 1) * sub
        V[10 * N + k] = (S[k][5] * S[k][7] * S[k][4] * S[k][1] - 1) * sub
        V[11 * N + k] = (S[k][3] * S[k][8] * S[k][5] * S[k][2] - 1) * sub
        V[12 * N + k] = (S[k][10] * S[k][12] * S[k][9] * S[k][6] - 1) * sub
        V[13 * N + k] = (S[k][11] * S[k][13] * S[k][10] * S[k][7] - 1) * sub
        V[14 * N + k] = (S[k][9] * S[k][14] * S[k][11] * S[k][8] - 1) * sub
        V[15 * N + k] = (S[k][16] * S[k][0] * S[k][15] * S[k][12] - 1) * sub
        V[16 * N + k] = (S[k][17] * S[k][1] * S[k][16] * S[k][13] - 1) * sub
        V[17 * N + k] = (S[k][15] * S[k][2] * S[k][17] * S[k][14] - 1) * sub
    v = torch.norm(V, p=2) / torch.norm(nor, p=1)
    return v


# initial setting
(ss, tt, N) = (0.01, 500, len(S))
BF = torch.tensor([0.0])
WF = torch.tensor([np.pi / 4, np.pi / 4, np.pi / 4, np.pi / 4])
BV = torch.tensor([0.2, -0.2], requires_grad=True)
WV = torch.tensor([-0.3, -0.1, 0.1, 0.3, 1.2, 1.4, 1.6, 1.8], requires_grad=True)
optimizer = torch.optim.Adam([BV, WV], lr=ss)

# Stochastic gradient descent
start_time = time.time()
COST = []
ABV = [[], []]
AWV = [[], [], [], [], [], [], [], []]
for h in range(tt):
    cost = criterion(BV, WV, BF, WF)
    COST.append(cost.tolist())
    for p in range(2):
        ABV[p].append(BV[p].tolist())
    for q in range(8):
        AWV[q].append(WV[q].tolist())
    cost.backward()
    optimizer.step()
    optimizer.zero_grad()
end_time = time.time()
execution_time = end_time - start_time

# Visualize training cycles up to t
t = (tt - 1)
Final = COST[t]
BVt = torch.tensor([ABV[0][t], ABV[1][t]])
WVt = torch.tensor([AWV[0][t], AWV[1][t], AWV[2][t], AWV[3][t],
                    AWV[4][t], AWV[5][t], AWV[6][t], AWV[7][t]])

fig0 = plt.figure(figsize=(6, 4))
plt.plot(COST)
plt.title("final cost = %.2e" % Final)
plt.text(0.6 * t, 0.8 * COST[0], "20+30 configurations")
plt.text(0.6 * t, 0.7 * COST[0], "$b_{f} = 0$, $w_{f} = \pi/4 i$")
plt.text(0.6 * t, 0.6 * COST[0], "Adam step size: %.2f" % ss)
plt.text(0.6 * t, 0.5 * COST[0], "Execution time: %is" % execution_time)
plt.xlabel("train cycles")
plt.ylabel("cost function")
plt.show()

fig1 = plt.figure(figsize=(6, 4))
plt.plot(COST[0:t])
plt.title("final cost = %.2e" % Final)
plt.text(200, 10 ** (-2), "$\Psi$(|00>) = %.4f + %.4fi" % (
    ph(BVt, WVt, BF, WF, SS[0]).data.item().real, ph(BVt, WVt, BF, WF, SS[0]).data.item().imag))
plt.text(200, 10 ** (-3), "$\Psi$(|01>) = %.4f + %.4fi" % (
    ph(BVt, WVt, BF, WF, SS[1]).data.item().real, ph(BVt, WVt, BF, WF, SS[1]).data.item().imag))
plt.text(200, 10 ** (-4), "$\Psi$(|10>) = %.4f + %.4fi" % (
    ph(BVt, WVt, BF, WF, SS[2]).data.item().real, ph(BVt, WVt, BF, WF, SS[2]).data.item().imag))
plt.text(200, 10 ** (-5), "$\Psi$(|11>) = %.4f + %.4fi" % (
    ph(BVt, WVt, BF, WF, SS[3]).data.item().real, ph(BVt, WVt, BF, WF, SS[3]).data.item().imag))
plt.xlabel("train cycles")
plt.ylabel("cost function")
plt.yscale('log')
plt.show()

fig2 = plt.figure(figsize=(6, 4))
plt.plot(ABV[0][0:t], color='tab:red', label=r"$b_v$ $\rightarrow$ %.4f + %.4fi" % (
    ABV[0][t], ABV[1][t]))
plt.plot(AWV[0][0:t], color='tab:orange', label=r"$w_{v,1}$ $\rightarrow$ %.4f + %.4fi" % (
    AWV[0][t], AWV[4][t]))
plt.plot(AWV[1][0:t], color='tab:green', label=r"$w_{v,2}$ $\rightarrow$ %.4f + %.4fi" % (
    AWV[1][t], AWV[5][t]))
plt.plot(AWV[2][0:t], color='tab:blue', label=r"$w_{v,3}$ $\rightarrow$ %.4f + %.4fi" % (
    AWV[2][t], AWV[6][t]))
plt.plot(AWV[3][0:t], color='tab:purple', label=r"$w_{v,4}$ $\rightarrow$ %.4f + %.4fi" % (
    AWV[3][t], AWV[7][t]))
plt.plot(ABV[1][0:t], '--', color='tab:red')
plt.plot(AWV[4][0:t], '--', color='tab:orange')
plt.plot(AWV[5][0:t], '--', color='tab:green')
plt.plot(AWV[6][0:t], '--', color='tab:blue')
plt.plot(AWV[7][0:t], '--', color='tab:purple')
plt.xlabel("train cycles")
plt.ylabel("weights")
plt.legend()
plt.show()
```

C.6 Python code for searching

```
# This code search solutions from random bv and wv for FRRBM
import torch
import numpy as np
import time

# 20 selected for the degeneracy basis and 30 random configurations.
S = np.load('S20_30.npy')
SS = np.array([[-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1],
               [-1, +1, -1, -1, -1, -1, -1, +1, -1, -1, -1, -1, -1, +1, -1, -1, -1, -1],
               [-1, -1, -1, -1, -1, -1, -1, -1, -1, +1, +1, +1, -1, -1, -1, -1, -1, -1],
               [-1, +1, -1, -1, -1, -1, -1, +1, -1, +1, +1, +1, -1, +1, -1, -1, -1, -1]])

# 10k sets of variables for 10 variables.
Variables = np.load('Variables10_10k.npy')


# pr is the shortcut for cosh function.
def pr(b, w, s1, s2, s3, s4):
    result = torch.cosh((b[0] + 1j * b[1]) + 1j * ((w[0] + 1j * w[4]) * s1 + (
            w[1] + 1j * w[5]) * s2 + (w[2] + 1j * w[6]) * s3 + (w[3] + 1j * w[7]) * s4))
    return result


def prr(b, w, s1, s2, s3, s4):
    result = torch.cosh(b[0] + 1j * (w[0] * s1 + w[1] * s2 + w[2] * s3 + w[3] * s4))
    return result


# ph is the phase function.
def ph(bv, wv, bf, wf, s):
    result = (pr(bv, wv, s[0], s[3], s[2], s[15]) * pr(bv, wv, s[1], s[4], s[0], s[16])
              * pr(bv, wv, s[2], s[5], s[1], s[17]) * pr(bv, wv, s[6], s[9], s[8], s[3])
              * pr(bv, wv, s[7], s[10], s[6], s[4]) * pr(bv, wv, s[8], s[11], s[7], s[5])
              * pr(bv, wv, s[12], s[15], s[14], s[9]) * pr(bv, wv, s[13], s[16], s[12], s[10])
              * pr(bv, wv, s[14], s[17], s[13], s[11]) * prr(bf, wf, s[4], s[6], s[3], s[0])
              * prr(bf, wf, s[5], s[7], s[4], s[1]) * prr(bf, wf, s[3], s[8], s[5], s[2])
              * prr(bf, wf, s[10], s[12], s[9], s[6]) * prr(bf, wf, s[11], s[13], s[10], s[7])
              * prr(bf, wf, s[9], s[14], s[11], s[8]) * prr(bf, wf, s[16], s[0], s[15], s[12])
              * prr(bf, wf, s[17], s[1], s[16], s[13]) * prr(bf, wf, s[15], s[2], s[17], s[14]))
    return result

```

```

# nor is the normalize function.
def nor(bv, wv, bf, wf):
    result = 0
    for i in range(len(S)):
        result += abs(ph(bv, wv, bf, wf, S[i]))
    return result


# The 9 vertex operators flip qubits.
A1 = np.diag([-1, +1, -1, -1, +1, +1, +1, +1, +1, +1, +1, +1, +1, +1, +1, -1, +1, +1])
A2 = np.diag([-1, -1, +1, +1, -1, +1, +1, +1, +1, +1, +1, +1, +1, +1, +1, +1, -1, +1])
A3 = np.diag([+1, -1, -1, +1, +1, -1, +1, +1, +1, +1, +1, +1, +1, +1, +1, +1, +1, -1])
A4 = np.diag([+1, +1, +1, -1, +1, +1, -1, +1, -1, -1, +1, +1, +1, +1, +1, +1, +1, +1])
A5 = np.diag([+1, +1, +1, +1, -1, +1, -1, -1, +1, +1, -1, +1, +1, +1, +1, +1, +1, +1])
A6 = np.diag([+1, +1, +1, +1, +1, -1, +1, -1, -1, +1, +1, -1, +1, +1, +1, +1, +1, +1])
A7 = np.diag([+1, +1, +1, +1, +1, +1, +1, +1, +1, -1, +1, +1, -1, +1, -1, -1, +1, +1])
A8 = np.diag([+1, +1, +1, +1, +1, +1, +1, +1, +1, +1, -1, +1, -1, -1, +1, +1, -1, +1])
A9 = np.diag([+1, +1, +1, +1, +1, +1, +1, +1, +1, +1, +1, -1, +1, -1, -1, +1, +1, -1])
S1 = np.dot(S, A1)
S2 = np.dot(S, A2)
S3 = np.dot(S, A3)
S4 = np.dot(S, A4)
S5 = np.dot(S, A5)
S6 = np.dot(S, A6)
S7 = np.dot(S, A7)
S8 = np.dot(S, A8)
S9 = np.dot(S, A9)


# lam is the multiplier to amplify the choice of ground state
def criterion(bv, wv, bf, wf):
    V = torch.empty((18 * N,), dtype=torch.complex64)
    for k in range(N):
        sub = ph(bv, wv, bf, wf, S[k])
        V[k] = ph(bv, wv, bf, wf, S1[k]) - sub
        V[1 * N + k] = ph(bv, wv, bf, wf, S2[k]) - sub
        V[2 * N + k] = ph(bv, wv, bf, wf, S3[k]) - sub
        V[3 * N + k] = ph(bv, wv, bf, wf, S4[k]) - sub
        V[4 * N + k] = ph(bv, wv, bf, wf, S5[k]) - sub
        V[5 * N + k] = ph(bv, wv, bf, wf, S6[k]) - sub
        V[6 * N + k] = ph(bv, wv, bf, wf, S7[k]) - sub
        V[7 * N + k] = ph(bv, wv, bf, wf, S8[k]) - sub
        V[8 * N + k] = ph(bv, wv, bf, wf, S9[k]) - sub
        V[9 * N + k] = (S[k][4] * S[k][6] * S[k][3] * S[k][0] - 1) * sub
        V[10 * N + k] = (S[k][5] * S[k][7] * S[k][4] * S[k][1] - 1) * sub
        V[11 * N + k] = (S[k][3] * S[k][8] * S[k][5] * S[k][2] - 1) * sub
        V[12 * N + k] = (S[k][10] * S[k][12] * S[k][9] * S[k][6] - 1) * sub
        V[13 * N + k] = (S[k][11] * S[k][13] * S[k][10] * S[k][7] - 1) * sub
        V[14 * N + k] = (S[k][9] * S[k][14] * S[k][11] * S[k][8] - 1) * sub
        V[15 * N + k] = (S[k][16] * S[k][0] * S[k][15] * S[k][12] - 1) * sub
        V[16 * N + k] = (S[k][17] * S[k][1] * S[k][16] * S[k][13] - 1) * sub
        V[17 * N + k] = (S[k][15] * S[k][2] * S[k][17] * S[k][14] - 1) * sub
    v = torch.norm(V) / nor(bv, wv, bf, wf)
    return v


start_time = time.time()
(ss, tt, lam, N) = (0.02, 50, 1.0, len(S))
BF = torch.tensor([0.0])
WF = torch.tensor([np.pi / 4, np.pi / 4, np.pi / 4, np.pi / 4])
All = []
Select = []
for k in range(0, 100):
    IN = Variables[k]
    BV = torch.tensor([IN[0], IN[1]], requires_grad=True)
    WV = torch.tensor([IN[2], IN[3], IN[4], IN[5], IN[6], IN[7], IN[8], IN[9]], requires_grad=True)
    optimizer = torch.optim.Adam([BV, WV], lr=ss)
    for h in range(tt):
        cost = criterion(BV, WV, BF, WF)
        cost.backward()
        optimizer.step()
        optimizer.zero_grad()
        All.append(IN.tolist())
    print(
        "%d: (%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f)->"
        "(%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f,%.2f), cost=%.2e" % (
            k, IN[0].tolist(), IN[1].tolist(), IN[2].tolist(), IN[3].tolist(),
            IN[4].tolist(), IN[5].tolist(), IN[6].tolist(), IN[7].tolist(),
            IN[8].tolist(), IN[9].tolist(), BV[0].tolist(), BV[1].tolist(),
            WV[0].tolist(), WV[1].tolist(), WV[2].tolist(), WV[3].tolist(),
            WV[4].tolist(), WV[5].tolist(), WV[6].tolist(), WV[7].tolist(), cost))
    if cost < 0.01:
        Select.append(k)
end_time = time.time()
execution_time = end_time - start_time
```

```
print(f"Execution time: {execution_time} seconds")
print(f"Select:{Select}")
```

C.7 Python code for the RBM

```
# This code trains RBM to get basis ratios 1:2:3:4
import torch
import numpy as np
import matplotlib.pyplot as plt
import time

# 20 selected for the degeneracy basis and 30 random configurations.
S = np.load('S20_30.npy')
SS = np.array([[-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1],
               [-1, +1, -1, -1, -1, -1, -1, +1, -1, -1, -1, -1, -1, +1, -1, -1, -1, -1],
               [-1, -1, -1, -1, -1, -1, -1, -1, -1, +1, +1, +1, -1, -1, -1, -1, -1, -1],
               [-1, +1, -1, -1, -1, -1, -1, +1, -1, +1, +1, +1, -1, +1, -1, -1, -1, -1]])


def pr(b, w, s1, s2, s3, s4):
    result = torch.cosh(
        (b[0] + 1j * b[1]) + (w[0] + 1j * w[4]) * s1 + (w[1] + 1j * w[5]) * s2 + (
                w[2] + 1j * w[6]) * s3 + (w[3] + 1j * w[7]) * s4)
    return result


def prr(br, bi, s1, s2, s3):
    result = torch.cosh((br + 1j * bi) + 1j * (np.pi / 4) * (s1 + s2 + s3))
    return result


def prrr(br, bi, s1, s2, s3, s4, s5, s6):
    result = torch.cosh((br + 1j * bi) + 1j * (np.pi / 4) * (s1 + s2 + s3 + s4 + s5 + s6))
    return result


def ph(gl, nl, bv, wv, bf, wf, s):
    result = (np.exp(gl[0] + 1j * gl[1])
              * prr(nl[0], nl[3], s[0], s[1], s[2]) * prr(nl[1], nl[4], s[3], s[9], s[15])
              * prrr(nl[2], nl[5], s[0], s[1], s[2], s[3], s[9], s[15])
              * pr(bv, wv, s[0], s[3], s[2], s[15]) * pr(bv, wv, s[1], s[4], s[0], s[16])
              * pr(bv, wv, s[2], s[5], s[1], s[17]) * pr(bv, wv, s[6], s[9], s[8], s[3])
```

```
            * pr(bv, wv, s[7], s[10], s[6], s[4]) * pr(bv, wv, s[8], s[11], s[7], s[5])
            * pr(bv, wv, s[12], s[15], s[14], s[9]) * pr(bv, wv, s[13], s[16], s[12], s[10])
            * pr(bv, wv, s[14], s[17], s[13], s[11]) * pr(bf, wf, s[4], s[6], s[3], s[0])
            * pr(bf, wf, s[5], s[7], s[4], s[1]) * pr(bf, wf, s[3], s[8], s[5], s[2])
            * pr(bf, wf, s[10], s[12], s[9], s[6]) * pr(bf, wf, s[11], s[13], s[10], s[7])
            * pr(bf, wf, s[9], s[14], s[11], s[8]) * pr(bf, wf, s[16], s[0], s[15], s[12])
            * pr(bf, wf, s[17], s[1], s[16], s[13]) * pr(bf, wf, s[15], s[2], s[17], s[14]))
    return result


# The 9 vertex operators flip qubits.
A1 = np.diag([-1, +1, -1, -1, +1, +1, +1, +1, +1, +1, +1, +1, +1, +1, +1, -1, +1, +1])
A2 = np.diag([-1, -1, +1, +1, -1, +1, +1, +1, +1, +1, +1, +1, +1, +1, +1, +1, -1, +1])
A3 = np.diag([+1, -1, -1, +1, +1, -1, +1, +1, +1, +1, +1, +1, +1, +1, +1, +1, +1, -1])
A4 = np.diag([+1, +1, +1, -1, +1, +1, -1, +1, -1, -1, +1, +1, +1, +1, +1, +1, +1, +1])
A5 = np.diag([+1, +1, +1, +1, -1, +1, -1, -1, +1, +1, -1, +1, +1, +1, +1, +1, +1, +1])
A6 = np.diag([+1, +1, +1, +1, +1, -1, +1, -1, -1, +1, +1, -1, +1, +1, +1, +1, +1, +1])
A7 = np.diag([+1, +1, +1, +1, +1, +1, +1, +1, +1, -1, +1, +1, -1, +1, -1, -1, +1, +1])
A8 = np.diag([+1, +1, +1, +1, +1, +1, +1, +1, +1, +1, -1, +1, -1, -1, +1, +1, -1, +1])
A9 = np.diag([+1, +1, +1, +1, +1, +1, +1, +1, +1, +1, +1, -1, +1, -1, -1, +1, +1, -1])
S1 = np.dot(S, A1)
S2 = np.dot(S, A2)
S3 = np.dot(S, A3)
S4 = np.dot(S, A4)
S5 = np.dot(S, A5)
S6 = np.dot(S, A6)
S7 = np.dot(S, A7)
S8 = np.dot(S, A8)
S9 = np.dot(S, A9)


# lam is the multiplier to amplify the choice of ground state
def criterion(gl, nl, bv, wv, bf, wf):
    V = torch.empty((18 * N + 3,), dtype=torch.complex64)
    nor = torch.empty((N,), dtype=torch.complex64)
    for k in range(N):
        one = ph(gl, nl, bv, wv, bf, wf, SS[0])
        sub = ph(gl, nl, bv, wv, bf, wf, S[k])
        nor[k] = sub
        V[k] = ph(gl, nl, bv, wv, bf, wf, S1[k]) - sub
        V[1 * N + k] = ph(gl, nl, bv, wv, bf, wf, S2[k]) - sub
        V[2 * N + k] = ph(gl, nl, bv, wv, bf, wf, S3[k]) - sub
        V[3 * N + k] = ph(gl, nl, bv, wv, bf, wf, S4[k]) - sub
        V[4 * N + k] = ph(gl, nl, bv, wv, bf, wf, S5[k]) - sub
        V[5 * N + k] = ph(gl, nl, bv, wv, bf, wf, S6[k]) - sub
        V[6 * N + k] = ph(gl, nl, bv, wv, bf, wf, S7[k]) - sub
        V[7 * N + k] = ph(gl, nl, bv, wv, bf, wf, S8[k]) - sub
        V[8 * N + k] = ph(gl, nl, bv, wv, bf, wf, S9[k]) - sub
        V[9 * N + k] = (S[k][4] * S[k][6] * S[k][3] * S[k][0] - 1) * sub
        V[10 * N + k] = (S[k][5] * S[k][7] * S[k][4] * S[k][1] - 1) * sub
        V[11 * N + k] = (S[k][3] * S[k][8] * S[k][5] * S[k][2] - 1) * sub
        V[12 * N + k] = (S[k][10] * S[k][12] * S[k][9] * S[k][6] - 1) * sub
        V[13 * N + k] = (S[k][11] * S[k][13] * S[k][10] * S[k][7] - 1) * sub
        V[14 * N + k] = (S[k][9] * S[k][14] * S[k][11] * S[k][8] - 1) * sub
        V[15 * N + k] = (S[k][16] * S[k][0] * S[k][15] * S[k][12] - 1) * sub
        V[16 * N + k] = (S[k][17] * S[k][1] * S[k][16] * S[k][13] - 1) * sub
        V[17 * N + k] = (S[k][15] * S[k][2] * S[k][17] * S[k][14] - 1) * sub
        V[18 * N + 0] = lam * (ph(gl, nl, bv, wv, bf, wf, SS[1]) - (bb / aa) * one)
        V[18 * N + 1] = lam * (ph(gl, nl, bv, wv, bf, wf, SS[2]) - (cc / aa) * one)
        V[18 * N + 2] = lam * (ph(gl, nl, bv, wv, bf, wf, SS[3]) - (dd / aa) * one)
    v = torch.norm(V, p=2) / torch.norm(nor, p=1)
    return v


# initial setting with the basis ratio (aa, bb, cc, dd)
(aa, bb, cc, dd, lam) = (1, 2, 3, 4, 1.0)
(ss, tt, N) = (0.0001, 2000, len(S))
BF = torch.tensor([0.0, 0.0])
WF = torch.tensor([0.0, 0.0, 0.0, 0.0, np.pi / 4, np.pi / 4, np.pi / 4, np.pi / 4])
BV = torch.tensor([0.0, 0.0])
WV = torch.tensor([0.0, 0.0, 0.0, 0.0, np.pi / 2, np.pi / 2, np.pi / 2, np.pi / 2])
# global constant GL change the overall phase to modify aa, bb, cc, dd
GL = torch.tensor([0.7135588, -1.570752])
# NL stores all nonlocal weights bx, by, bz in C, wx=wy=wz=pi/4 i
# NL = [Re(bx),Re(by),Re(bz),Im(bx),Im(by),Im(bz)]
NL = torch.tensor([0.7, 0.4, 1.1, 0.9, 0.8, 0.1], requires_grad=True)
optimizer = torch.optim.Adam([NL], lr=ss)

# Stochastic gradient descent
start_time = time.time()
COST = []
ANL = [[], [], [], [], [], []]
for h in range(tt):
    cost = criterion(GL, NL, BV, WV, BF, WF)
    COST.append(cost.tolist())
    for p in range(6):
        ANL[p].append(NL[p].tolist())
    cost.backward()
    optimizer.step()
    optimizer.zero_grad()
end_time = time.time()
execution_time = end_time - start_time

# Visualize training cycles up to t
t = (tt - 1)
Final = COST[t]
NLt = torch.tensor([ANL[0][t], ANL[1][t], ANL[2][t], ANL[3][t], ANL[4][t], ANL[5][t]])

fig0 = plt.figure(figsize=(6, 4))
plt.plot(COST[0:t])
plt.title("final cost = %.2e" % Final)
plt.text(0.6 * t, 0.85 * COST[0], "GL: %.4f + %.4fi" % (GL[0], GL[1]))
plt.text(0.6 * t, 0.79 * COST[0], "$b_{f} = 0$, $w_{f} = \pi/4 i$")
plt.text(0.6 * t, 0.73 * COST[0], "$b_{v} = 0$, $w_{v} = \pi/2 i$")
plt.text(0.6 * t, 0.67 * COST[0], "$w_{x,y,z} = \pi/4 i$")
plt.text(0.6 * t, 0.61 * COST[0], "Execution time: %is" % execution_time)
plt.xlabel("train cycles")
plt.ylabel("cost function")
plt.show()

fig1 = plt.figure(figsize=(6, 4))
plt.plot(COST[0:t])
plt.title("final cost = %.2e" % Final)
plt.text(0.4 * t, 10 ** (-2), "$\Psi$(|00>) = %.4f + %.4fi" % (
    ph(GL, NLt, BV, WV, BF, WF, SS[0]).data.item().real,
    ph(GL, NLt, BV, WV, BF, WF, SS[0]).data.item().imag))
plt.text(0.4 * t, 10 ** (-2.5), "$\Psi$(|01>) = %.4f + %.4fi" % (
    ph(GL, NLt, BV, WV, BF, WF, SS[1]).data.item().real,
    ph(GL, NLt, BV, WV, BF, WF, SS[1]).data.item().imag))
plt.text(0.4 * t, 10 ** (-3), "$\Psi$(|10>) = %.4f + %.4fi" % (
    ph(GL, NLt, BV, WV, BF, WF, SS[2]).data.item().real,
    ph(GL, NLt, BV, WV, BF, WF, SS[2]).data.item().imag))
plt.text(0.4 * t, 10 ** (-3.5), "$\Psi$(|11>) = %.4f + %.4fi" % (
    ph(GL, NLt, BV, WV, BF, WF, SS[3]).data.item().real,
    ph(GL, NLt, BV, WV, BF, WF, SS[3]).data.item().imag))
plt.xlabel("train cycles")
plt.ylabel("cost function")
plt.yscale('log')
plt.show()
```

```

fig2 = plt.figure(figsize=(6, 4))
plt.plot(ANL[0][0:t], color='tab:red', label=r"$b_x$ $\rightarrow$ %.4f + %.4fi"
                                             % (ANL[0][t], ANL[3][t]))
plt.plot(ANL[1][0:t], color='tab:blue', label=r"$b_y$ $\rightarrow$ %.4f + %.4fi"
                                              % (ANL[1][t], ANL[4][t]))
plt.plot(ANL[2][0:t], color='tab:green', label=r"$b_z$ $\rightarrow$ %.4f + %.4fi"
                                               % (ANL[2][t], ANL[5][t]))
plt.plot(ANL[3][0:t], '--', color='tab:red')
plt.plot(ANL[4][0:t], '--', color='tab:blue')
plt.plot(ANL[5][0:t], '--', color='tab:green')
plt.xlabel("train cycles")
plt.ylabel("nonlocal weights")
plt.legend()
plt.show()
```