Compact Fermion to Qubit Mappings for Quantum Simulation

Charles Derby

Supervisor: Toby Cubitt

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

University College London.

Department of Computer Science

University College London

February 15, 2023

I, Charles Derby, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

Fermions are one of two types of particles that make up matter in the universe, characterised by many-body wavefunctions that are antisymmetric under particle exchange. Electrons, which underpin many physical systems of interest, are included in this group, so the ability to accurately simulate fermionic physics would be a great asset to research. However, the antisymmetric nature of these particles means that classical simulation of systems of multiple fermions is, in general, infeasible due to sign problems. This infeasibility extends even to simplified systems such as the Fermi-Hubbard model on a 2D grid. Simulation of fermions on a quantum device would avoid this problem entirely.

A requisite step in simulating fermions on a quantum computer is mapping a many-body fermionic system onto qubits through a fermionic encoding. Significant properties of fermionic encodings include their qubit to fermionic mode ratio and the weight of their encoded fermionic interaction operators. Both affect the runtime of quantum simulation algorithms so it is ideal to minimise these quantities. This thesis presents the novel "compact" encoding which outperforms all previous local encodings in these metrics. The construction of the encoding is shown for a number of interaction graph structures and its general properties are explored. Special attention is given to a remarkable feature where low weight undetectable noise on the encoding corresponds to a natural noise process on fermionic systems, indicating that it may have utility in simulation even on imperfect, noisy quantum devices.

An interesting feature of the compact encoding and others is an apparent link to topological error correcting codes like the toric code. Inspection of the compact encoding for a cubic lattice reveals a link to an apparently novel 3D topological

code with some unusual properties. This size of its codespace and code distance are calculated and the exact form of its logical operators and syndromes are shown. Excitations with fermionic character exist in this code, consistent with the other codes linked with fermionic encodings, pointing to a possible unifying picture for local fermionic encodings.

Impact Statement

Quantum computers have the potential for a wide reaching impact. The development of a fault tolerant quantum computer would allow hitherto infeasible problems to be tackled computationally. A significant application, which has been a primary motivator for the field since its inception, is the simulation of other quantum systems. Systems containing many fermions are of particular interest. Not only because they are fundamentally difficult to simulate with normal computers but because they include systems of electrons, the particles which underpin almost all of chemistry. Simulating these systems on a quantum computer is not a simple task however, as there must be a procedure to map the physics of many indistinguishable fermions onto the physics of stationary qubits, two fundamentally different systems. This procedure is called a *fermionic encoding* and the main subject of this thesis is an example of this.

The content of this thesis could benefit researchers in a number of fields. It adds to the rich zoo of fermionic encodings and may provide inspiration for further results in the field, it also highlights a possible link between the seemingly disparate local fermionic encodings which may pave the way to a more unified general theory of representing fermions on qubits. The encoding presented in this work has favourable properties for simulation on noisy devices so it may benefit research groups working on near term quantum hardware by providing the means to perform interesting fermionic simulation experiments. The content of the last chapter may also be of interest to the error correction community as it provides an example of an apparently unclassified topological code, this may lead to the development of new classes of code.

This research may also yield benefits outside of academia. The quantum simulation of electronic systems would lead to greater understanding of chemical reactions such as Nitrogen fixing and materials such as superconductors and batteries. This understanding could lead to improvements in efficiency or the development of new substances which would be invaluable to industries including agriculture, transportation and battery production.

Acknowledgements

First of all, thanks are due to my supervisor Toby Cubitt, for taking me on as one of his few PhD students and for his guidance and encouragement throughout my studies. His attentive supervision and helpful advice on research and communication were indispensable for a young scientist such as myself. I would like to thank Phasecraft Ltd. for the company's financial support of my studies and for the opportunity to work with so many talented researchers.

Special thanks go to Phasecraft employee Joel Klassen for countless insightful discussions, always hearing me out and for keeping me honest in my writings. I would like to thank Phasecraft employees Raul Santos and Evan Sheridan for many helpful conversations about unfamiliar subjects and for strengthening my confidence in the background material which underpin my work. Phasecraft intern Riley Chien also deserves thanks for patiently listening as I feverishly brainstormed proof ideas for chapter 5 at him and for other interesting fermion based discussions. I am grateful to fellow Phasecraft affiliated PhD student Laura Clinton for her constant support and companionship. I also want to thank my Quantum Technologies CDT colleague Tom Scruby, former CDT student Mike Vasmer, UCL research fellow Niko Breuckmann and Stanford post-doc Dominic Williamson for interesting discussions about topological codes and braiding.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Background

In a keynote speech in 1982 [1], often cited at the beginning of quantum computation publications, Richard Feynman discussed the problem of simulating physics, in particular, quantum physics on computers. He concluded that a deterministic classical computer would encounter unavoidable exponential explosions in resource requirements and that a probabilistic classical machine would be unable to replicate the non-local behaviour of quantum systems. The only alternative he left the audience with was to remove all the difficulty of simulating quantum behaviour by using a machine made of quantum systems itself. This was shown to be a real possibility in 1996 by Lloyd [2] who presented the first quantum algorithm for simulating the evolution of a (local) quantum system via Trotter decomposition. Since then the field of quantum simulation has progressed substantially, with the development of algorithms with improved scaling in simulated evolution time and error [3, 4, 5], including methods which scale optimally in both regards [6, 7]; although it has been argued that approaches closer to Lloyd's original protocol will be among the first to produce useful results on real hardware [8].

Better quantum simulation algorithms alone, however, do not immediately allow us to efficiently simulate the entire quantum world as we please. In fact, in Feynman's original talk he admitted he was unsure if a machine composed of local spin-$\frac{1}{2}$ particles could efficiently simulate fermions, the family of particles containing electrons, the objects at the heart of any chemical process one might want to simulate. Indeed this is a non-trivial problem as one must find a suitable relationship between

a many body system of dynamic fermions and a system of stationary qubits before even beginning to think about simulation algorithms. Such relationships between fermions and qubits have existed since the early 20th century with the Jordan-Wigner transformation produced by its namesake duo in 1928 [9] which maps a system of N fermionic modes onto N qubits[1] with fermionic Hamiltonian terms mapping to Pauli operators of weight $O(N)$. In complexity theoretic terms, this already is enough to simulate fermionic systems "efficiently". In fact shortly after the publication of Lloyd's first quantum simulation result, he and Abrams presented an algorithm to simulate the Fermi-Hubbard model with runtime scaling quadratically with the system size [10] using only Trotter decomposition and the Jordan-Wigner transformation. This does not however concern itself with the complexities of working on real, noisy hardware.

Quantum computing devices are currently in what is referred to as the NISQ era (Noisy Intermediate Scale Quantum) [11]. NISQ devices are characterised by modest qubit counts (typically < 100), error rates currently above fault tolerant thresholds (for schemes compatible with their qubit counts) and lack of full error correcting capabilities. Attempting to execute useful quantum algorithms on these machines then becomes a race against the clock as irreversible noise gradually creeps in and renders the output meaningless after a relatively short time. Naturally, it is in the interest of anyone wanting to get anything interesting from a NISQ device to do everything they can to minimise resource use, in particular runtime and qubit count. In the context of simulating fermionic systems these savings can be made by mapping the system onto qubits in a more efficient manner.

Local fermionic encodings are fermion to qubit mappings which use a greater number of qubits in order to map fermionic interaction terms to low weight qubit operators which do not scale with system size. Early examples of these date to the early 2000s [12, 13] but the field has received a renewed interest in the past few years [14, 15, 16, 17] with new encodings achieving lower operator weights and qubit counts, both of which are beneficial for simulation on NISQ devices.

[1] The original work actually frames this as representing a system of stationary spins on a fermionic system but the mapping reverses easily.

The first two chapters of this thesis concern a new local fermion encoding which outperforms all others in qubit count and operator weight when simulating locally interacting fermionic modes on a square lattice, potentially bringing a quantum simulation of the 2D Fermi-Hubbard model closer to reality. The encoding on a 2D grid is detailed in chapter 2, generalisations to further 2D lattices and a 3D cubic lattice are presented in chapter 3 along with some general theoretical results about fermion encodings.

The remaining two chapters of content concern results that stemmed from these novel encodings. Chapter 4 discusses the error mitigating capabilities of the encodings and shows that their low weight undetectable errors map to physically meaningful noise on the encoded fermionic system. In chapter 5, it is shown that an interesting topological error correcting code emerges from the cubic lattice encoding presented in chapter 3 and its logical qubit count, logical operators and code distance are found for all possible configurations. Furthermore the code is found to exhibit excitations that are fermionic in nature. As will be discussed later, numerous encodings in the literature display a link to the fermionic excitations in Kitaev's toric code [18], the results in chapter 5 motivate further investigation into a more general connection between fermionic encodings and topological codes with fermions.

The remainder of this chapter will review relevant background material, including some of the technical information required to understand the results in this work.

1.1 Fermions

Fundamental particles of a given type, such as electrons, are indistinguishable. More precisely, this means that no physical observable yields information that allows one to differentiate between any two of them. This has some interesting implications. Consider a 2 particle spatial wavefunction $\Psi(x_1, x_2)$. The probability of finding particle 1 at position x_1 and particle 2 at x_2 is given by the absolute square of the wavefunction $|\Psi(x_1, x_2)|^2$. In principle, this probability can be physically observed

so it must be impossible to tell the particles apart through its measurement. This means that if the particles labels 1 and 2 are switched then the the probability density will remain unchanged, i.e.

$$|\Psi(x_1,x_2)|^2 = |\Psi(x_2,x_1)|^2. \tag{1.1}$$

This property of the squared wavefunction means that the wavefunction itself must be either symmetric or antisymmetric under particle exchange, meaning that it is either unchanged or picks up a minus sign after two particles are switched:

$$\Psi(x_1,x_2) = \pm\Psi(x_2,x_1). \tag{1.2}$$

These two behaviours neatly divide all fundamental particles into two camps, bosons (symmetric) and fermions (antisymmetric). An equivalent definition of the groups exists in the fact that bosons have integer spin and fermions have half integer spin although the explicit connection between these two notions is beyond the scope of this work, the intrigued reader is directed to [19] and the references therein for more in depth discussion.

Fermions are of particular interest as they include electrons, the particles at the heart of all chemical processes, so it is important to understand their behaviour. A good place to start is from the fact that a group of fermions can only exist in an antisymmetric wavefunction, and the implications which follow from this restricted form. An arbitrary wavefunction of N fermions $\Psi(x_1,\ldots,x_N)$ can be made antisymmetric (up to a normalisation factor) via the operation

$$\mathcal{A}[\Psi(x_1,\ldots,x_N)] := \sum_{p\in P}(-1)^p\Psi(x_{p(1)},\ldots,x_{p(N)}), \tag{1.3}$$

where P denotes the set of all unique permutations of N labels and $(-1)^p$ denotes the parity of a permutation p that maps $i \to p(i)$ ($+1$ if it is equivalent to an even number of exchanges, -1 otherwise). The normalisation factor depends on the overlaps between the permuted wavefunctions. A basis for antisymmetric wavefunctions can

be constructed from *Slater determinants*, these are antisymmetrised product states of N fermions each occupying different orthonormal states $\{\alpha_1, \ldots, \alpha_N\}$ (often referred to as *orbitals* or *modes*) so called because they can be expressed as a determinant

$$\Phi_{\alpha_1,\ldots,\alpha_N}(x_1,\ldots,x_N) := \frac{1}{\sqrt{N!}} \begin{vmatrix} \alpha_1(x_1) & \alpha_2(x_1) & \cdots & \alpha_N(x_1) \\ \alpha_1(x_2) & \alpha_2(x_2) & \cdots & \alpha_N(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1(x_N) & \alpha_2(x_N) & \cdots & \alpha_N(x_N) \end{vmatrix} \tag{1.4}$$

$$\equiv \frac{1}{\sqrt{N!}} \mathcal{A}[\alpha_1(x_1)\alpha_2(x_2)\cdots\alpha_N(x_N)].$$

Slater determinants are unchanged under unitary transforms between their single particle states up to a phase (multiplication by a unitary only adds a phase to any determinant) so are really defined over an N-dimensional subspace spanned by the orthonormal states. Note that the Pauli exclusion principle (one fermion per mode) is enforced by the wavefunction being 0 if any states are repeated. If the fermions live in a M-dimensional space then the space of all antisymmetric N fermion states is

$$f_M^N = \text{span}\left(\Phi_{\alpha_{i_1},\ldots,\alpha_{i_N}} \;:\; \forall\{i_1,\ldots,i_N\} \subseteq [M]\right), \tag{1.5}$$

where $\Phi_{\alpha_{i_1},\ldots,\alpha_{i_n}}$ is a Slater determinant as defined in eq. (1.4) with the position variables dropped. The dimension of f_M^N is $\binom{M}{N}$. The total space of possible fermionic states on an M-dimensional system is the direct sum of the above spaces for all particle numbers 0 through M

$$\mathcal{F}_M = \bigoplus_{n=0}^{M} f_M^n. \tag{1.6}$$

This is called the *Fock space* and has total dimension $\sum_{n=0}^{M}\binom{M}{n} = 2^M$.

This gets very complicated when dealing with large numbers of particles, fortunately things can be made much simpler via *second quantised notation*. In multi-fermion wavefunctions, the only point of distinction is whether a mode is

occupied by a fermion or not so they can be specified purely in these terms. For example, the Slater determinant over the first N modes of a basis $\{\alpha_1,\ldots,\alpha_M\}$ can be written as

$$\Phi_{\alpha_1,\ldots,\alpha_N} \to |1_1,1_2,\ldots,1_N,0_{N+1},\ldots,0_M\rangle \tag{1.7}$$

where 1_i and 0_i denote occupied and unoccupied modes indexed by i. The main idea of second quantisation is to define these states via the action of *creation operators* on the *vacuum state* which is simply the case where no fermions exist in a system. An arbitrary Slater determinant over a given set of modes can then be defined as

$$\begin{gathered} |b_1,b_2,\ldots,b_M\rangle = (c_1^\dagger)^{b_1}(c_2^\dagger)^{b_2}\cdots(c_M^\dagger)^{b_M}|\Omega\rangle\,, \quad b_i\in\{0,1\} \\ \text{where} \quad |\Omega\rangle := |0,0,\ldots,0\rangle \end{gathered} \tag{1.8}$$

where $c_i^\dagger$ is the creation operator for mode i. Considering a one particle state $|1_i\rangle = c_i^\dagger|\Omega\rangle$ (where unoccupied have been omitted), its overlap with itself must be non-zero so $\langle 1_i|1_i\rangle = \langle\Omega|c_ic_i^\dagger|\Omega\rangle > 0$. This implies that the action of c_i, the adjoint of $c_i^\dagger$, is to remove a fermion from mode i, c_i are therefore named *annihilation operators* and the vacuum state is fixed such that $\langle\Omega|\Omega\rangle = 1$ ensuring that states created by $c_i^\dagger$ are normalised. Two more simple rules fully define the operators: $\{c_i^\dagger,c_j^\dagger\} = \{c_i,c_j\} = 0$ captures the antisymmetry of wavefunctions under particle exchange by ensuring that $c_i^\dagger c_j^\dagger|\Omega\rangle = -c_j^\dagger c_i^\dagger|\Omega\rangle$ (equivalent to particle exchange, take particle 1 to be the first created by an operator) and $\{c_i,c_j^\dagger\} = \delta_{ij}$ preserves the orthonormality of different basis states, i.e. $\langle 1_i|1_j\rangle = \langle\Omega|c_ic_j^\dagger|\Omega\rangle = \delta_{ij}\langle\Omega|\Omega\rangle - \langle\Omega|c_i^\dagger c_j|\Omega\rangle = \delta_{ij}$.

One finds that these operators defined for a given basis and the rules

$$\langle\Omega|\Omega\rangle = 1 \tag{1.9}$$

$$c_i|\Omega\rangle = 0, \tag{1.10}$$

$$\{c_i,c_j\} = \{c_i^\dagger,c_j^\dagger\} = 0 \tag{1.11}$$

$$\{c_i,c_j^\dagger\} = \delta_{ij}. \tag{1.12}$$

completely describe the physics across the entire Fock space.

An important operator in this formalism is the *number operator*, defined as

$$n_i = c_i^\dagger c_i, \tag{1.13}$$

which simply yields the occupation number of the mode i in a given many-body state. One other important thing to note is that the ordering of the creation operators in eq. (1.8) is arbitrary up to a global sign and an ordering of the modes must be chosen and stuck to for calculations to remain consistent. The algebra generated by the creation and annihilation operators is called the *fermionic algebra*.

The precise relationship between the first and second quantised formalisms is somewhat nuanced and has been glossed over for brevity. A more in depth yet still accessible discussion can be found in [20], on which the material just presented is largely based. For some peace of mind, some consistencies between the two formalisms can be easily shown. For instance, the Pauli Exclusion principle is enforced by $c_i^\dagger c_i^\dagger = 0$, just as it is in a first quantised Slater determinant. As well as this, the invariance of Slater determinants under a unitary transform over their subspace is also reproduced in this formalism. Defining creation operators in a new basis as

$$\sum_{j=1}^{N} d_i^\dagger = U_{ij} c_j^\dagger \tag{1.14}$$

for some unitary matrix U, it follows from the rules that

$$\prod_{i=1}^{N} d_i^\dagger |\Omega\rangle = \det(U) \prod_{i=1}^{N} c_i^\dagger |\Omega\rangle \tag{1.15}$$

amounting to merely a difference in phase, just as in the first quantised formalism. Note that this only applies if the transform U exclusively applies within the subspace occupied by the Slater determinant, a transform which maps a single mode to a superposition of two modes will clearly transform a single particle Slater determinant to a different many body state. This new state will still be a Slater determinant, but over a different basis.

Being comfortable with the idea that this formalism does in fact completely

describe many body fermionic systems one can write multi-fermion Hamiltonians (of closed systems) like

$$H = \sum_{ij} M^{(1)}_{ij} c^\dagger_i c_j + \sum_{klmn} M^{(2)}_{klmn} c^\dagger_k c^\dagger_l c_m c_n + \cdots \tag{1.16}$$

where the first sum contains one body terms (e.g. kinetic energy and external potential contributions), the second contains two body terms (interactions between pairs of fermions). From a fundamental point of view, these two terms capture the behaviour of electrons in most situations. Hamiltonians with higher order interactions can be defined, but these amount to including higher order processes, which are usually less likely.

In Hamiltonians of the above form there is an annihilation operator for every creation operator in each term, meaning that particle number is conserved. More general Hamiltonians can exist with terms that do not conserve this quantity such as $(c_i c_j + c^\dagger_j c^\dagger_i)$, however they may only contain *even* fermionic operators, that is sums of even products of creation and annihilation operators. This is because fermionic Hamiltonians must follow the *parity superselection rule*. A superselection rule prohibits superpositions between particular types of quantum states to exist and therefore prohibits physical observables (including Hamiltonians) from coupling them. The fermionic parity superselection rule prohibits the superposition between many-body fermionic states with an odd and even particle number, limiting fermionic Hamiltonians to sums of even fermionic operators which preserve parity. Some intuition can be gained for the reason behind this superselection rule by considering a single mode fermionic system the superposition state

$$|\psi\rangle = \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle) \tag{1.17}$$

in theory one could rotate the lab frame of reference by one full turn and the $|1\rangle$ state would pick up a phase due to fermions having half-integer spin, amounting to an observable change in the total state without interaction with the system which would lead to causality violation. This is by no means a rigorous argument. They can be

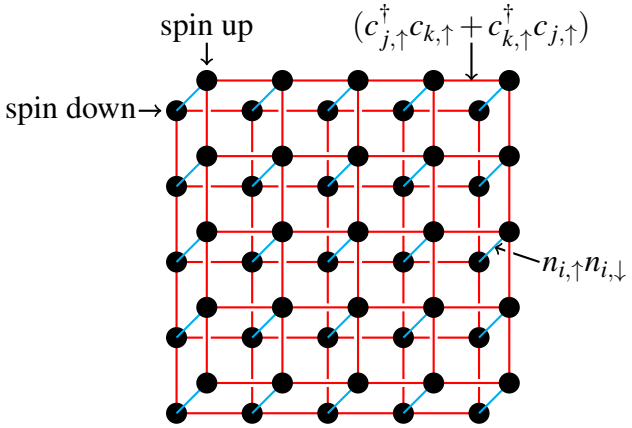


Figure 1.1: Graph representation of the Fermi Hubbard Hamiltonian on a 5×5 2D lattice. Red lines denote hopping interactions between fermionic sites within a given spin sector and blue lines denote on-site interactions between spin sectors.

found in [21, 22].

Now it is hopefully clear that many body fermionic systems can be described in simple terms via the powerful second quantised formalism. However, despite their simple description, even seemingly basic fermionic Hamiltonians can have rich properties.

1.2 The Fermi-Hubbard Model

1.2.1 The Hamiltonian

A well known and well studied fermionic Hamiltonian is the Fermi-Hubbard model [23]. It is considered to be an approximate model of strongly correlated fermionic systems, in particular, cuprates which are of interest due to their unusual symmetry breaking behaviour and high temperature superconductivity [24, 25]. The system is defined on a lattice of M sites with adjacent sites $\langle i, j\rangle$ and where each site i has two fermionic modes of spin-up and spin-down associated with it indexed by $i,\uparrow$ and $i,\downarrow$, there are then $N = 2M$ modes. The Hamiltonian is given by

$$H_{FH} = \underbrace{-t \sum_{\langle i,j\rangle,\sigma} \left(c^\dagger_{i,\sigma}c_{j,\sigma} + c^\dagger_{j,\sigma}c_{i,\sigma}\right)}_{H_T} + \underbrace{U \sum_i n_{i,\uparrow}n_{i,\downarrow}}_{H_U}. \tag{1.18}$$

The H_T part, or the kinetic energy part, is a sum of *hopping terms* over all

adjacent pairs of modes with the same spin. See fig. 1.1 for an illustration of the Hamiltonian on a 2D lattice. The value of the *hopping energy* t is usually taken to be positive, but regardless of sign the term can be interpreted as making it energetically favourable for fermions to be delocalised between lattice sites. Take the sea of delocalised valence electrons in a metal for example.

The H_U part (interaction part) is a sum of on-site interactions between pairs of fermions on the same lattice site. The repulsive (attractive) case where the interaction strength $U > 0$ ($U < 0$) induces an energy cost (benefit) when both modes at a lattice cite are filled, i.e. $n_{i,\uparrow} = n_{i,\downarrow} = 1 \implies n_{i,\uparrow} n_{i,\downarrow} = 1$. These interactions can approximate Coulomb repulsion by only considering the energy cost at close range, alternatively it has been suggested that the $U < 0$ case can approximate the effective attraction between cooper pairs of electrons in BCS superconductivity [26, 27].

In the weakly interacting case where $U = 0$, H_{FH} can be written as

$$H_{FH}(U=0) = \frac{1}{2}\mathbf{c}^\dagger M \mathbf{c} + E \tag{1.19}$$

where E is a constant, $\mathbf{c} = (c_1, \ldots, c_M, c_1^\dagger, \ldots, c_M^\dagger)^T$ and

$$M = \begin{pmatrix} A & 0 \\ 0 & -A^* \end{pmatrix} \tag{1.20}$$

with $A = A^\dagger$ and $\frac{1}{2}\mathrm{Tr}[A] = E$. As A is Hermitian then one can easily find a real diagonal matrix D and unitary matrix V such that $D = V^\dagger A V$. Following from this we may rewrite the Hamiltonian as

$$H_{FH}(U=0) = \frac{1}{2}\sum_i \varepsilon_i b_i^\dagger b_i \tag{1.21}$$

where $\varepsilon_i = D_{ii}$ and $b_i^\dagger = \sum_j V_{ij} c_j^\dagger$. Since U is unitary, the b_i and $b_i^\dagger$ are creation and annihilation operators over a new basis. The eigenstates of the weakly interacting model are then simply Slater determinants over this basis.

In the strongly interacting case where $t = 0$ the lattice sites are decoupled and the

system effectively becomes a collection of single site Hamiltonians. The eigenstates are then just Slater determinants over the lattice position basis with eigenenergies as a multiple of doubly occupied lattice sites.

The Fermi-Hubbard model naturally acts on many-body states with any number of particles but the *filling* ($f = N/M$ where N is the number of particles) of the system can be controlled by adding an extra term to the Hamiltonian

$$H_{FH} - \mu N = H_{FH} - \mu \sum_i (n_{i,\uparrow} + n_{i,\downarrow}), \tag{1.22}$$

where $N = \sum_i (n_{i,\uparrow} + n_{i,\downarrow})$ is operator giving the total number of particles in the system and μ is the chemical potential, which can be interpreted as the energy cost / benefit to adding a particle to the system. Setting the value of μ determines the number of particles in the system's thermal equilibrium state. More precisely, the value of μ determines the expected value of the filling in the thermal Gibbs state $g(\beta)$

$$\langle f \rangle_{g(\beta)} = \frac{1}{MZ} \mathrm{Tr}\left[(n_\uparrow + n_\downarrow) e^{-\beta(H_{FH} - \mu N)} \right]. \tag{1.23}$$

where $Z = \mathrm{Tr}[e^{-\beta(H_{FH} - \mu N)}]$ and $\beta = 1/k_B T$ where k_B is the Boltzmann constant and T is the temperature. For example, setting $\mu = U/2$ puts the system in a state of half filling where M electrons are present in the system. Note that changing the values of t and U would also change the expected filling but altering them is effectively altering the system itself.

1.2.2 Classical and Quantum Simulation of the Model

Despite the ease with which the separate parts of the Hubbard model can be solved, the total model has resisted solution[2] for decades, with the exception of the exactly solved case of a 1D chain lattice [28]. Exact numerical diagonalisation has been performed on the model at higher dimensions but this has been limited to a system of about 20 sites [29, 30]. Approximate numerical simulations have also been attempted via the Quantum Monte-Carlo method presented in [31] but they run into so called

[2] "Solution" here can refer to knowledge of the model's full spectrum and corresponding eigenstates or simply knowledge of the state at thermal equilibrium for temperature T (where the equilibrium state at $T = 0$ is simply the ground state).

sign problems [32].

The full procedure is very involved so it will only be described superficially here, a thorough explanation can be found in [33]. The aim is to estimate the expectation of some operator O on a many body system in thermal equilibrium i.e. find an approximate value for

$$\langle O \rangle_{g(\beta)} = \frac{\mathrm{Tr}\left[Oe^{-\beta(H-\mu N)}\right]}{\mathrm{Tr}\left[e^{-\beta(H-\mu N)}\right]}. \tag{1.24}$$

This expression can be written as

$$\langle O \rangle_{g(\beta)} = \frac{\sum_x \rho(x) O(x)}{\sum_x \rho(x)}, \tag{1.25}$$

where $\{x\}$ is some ensemble of system configurations, $O(x)$ is the value of O in that configuration and $\rho(x)$ weight the configurations in the sum. If these weights are all positive, the Monte-Carlo algorithm samples over this ensemble with each configuration having the probability

$$P(x) = \frac{\rho(x)}{\sum_x \rho(x)}, \tag{1.26}$$

yielding an estimate for the expectation of $O(x)$ over the probability distribution P, $\langle O \rangle_P = \langle O \rangle_{g(\beta)}$.

In fermionic systems, these weights can be negative and may not map so easily onto probabilities, if this is the case then the estimation of O suffers from a sign problem. One can still use the Monte-Carlo algorithm by writing

$$\rho(x) = |\rho(x)| S(x) \tag{1.27}$$

where $S(x) = \pm 1$ and then defining probabilities

$$P'(x) = \frac{|\rho(x)|}{\sum_x |\rho(x)|}. \tag{1.28}$$

After which eq. (1.25) can be rewritten as

$$\langle O \rangle_{g(\beta)} = \frac{\langle OS \rangle_{P'}}{\langle S \rangle_{P'}}. \tag{1.29}$$

The sign problem arises when the value of $\langle S \rangle_{P'}$ is very small meaning that small statistical fluctuations in its estimation can cause the estimated value of $\langle O \rangle_{g(\beta)}$ to vary wildly, making its accurate numerical estimation very costly [32].

In 2004, Troyer and Wiese [34] showed that the sign problem is NP-hard. They achieved this by constructing a fermionic system in which the estimation of the thermal energy (which has a sign problem) in polynomial time would constitute the solution to the NP-complete problem of determining whether a classical spin glass system has a state energy below a given threshold. The bosonic equivalent of the system they construct has no sign problem and is efficiently soluble, implying that the sign problem is the source of the hardness. This result makes a generic efficient solution to the sign problem unlikely. Some recent developments have been made to avoid the problem but the improvements only apply to a limited set of models [35].

For the 2D square lattice Fermi-Hubbard model, if the system is set at half filling ($\langle f \rangle = 1$, $\mu = U/2$) then the sign problem does not cause issues. In fact, the phase diagram of U/t vs T for the system at half filling is well understood, with numerical simulations yielding a rich phase diagram in this regime (see fig. 1.2 based on a figure originally published in [27]). However, as the filling diverges from this value (apart from a few specific cases [33]) the sign problem makes the numerical solution for even relatively small 2D models numerically intractable. To illustrate this point consider the fact that in [30], the authors report that a solution a 22 site model filled with 17 fermions required over 7TB of memory and 13 TFLOPS on a 512-node supercomputer.

With this grim outlook for classical simulation of the model and the recent developments in quantum algorithms and devices, researchers have begun to seriously consider the prospect of using these to solve the Hubbard model [36, 25]. One proposal is to probe the ground state of the Hubbard model for a superconducting phase by preparing a Slater determinant ground state of a well understood supercon-

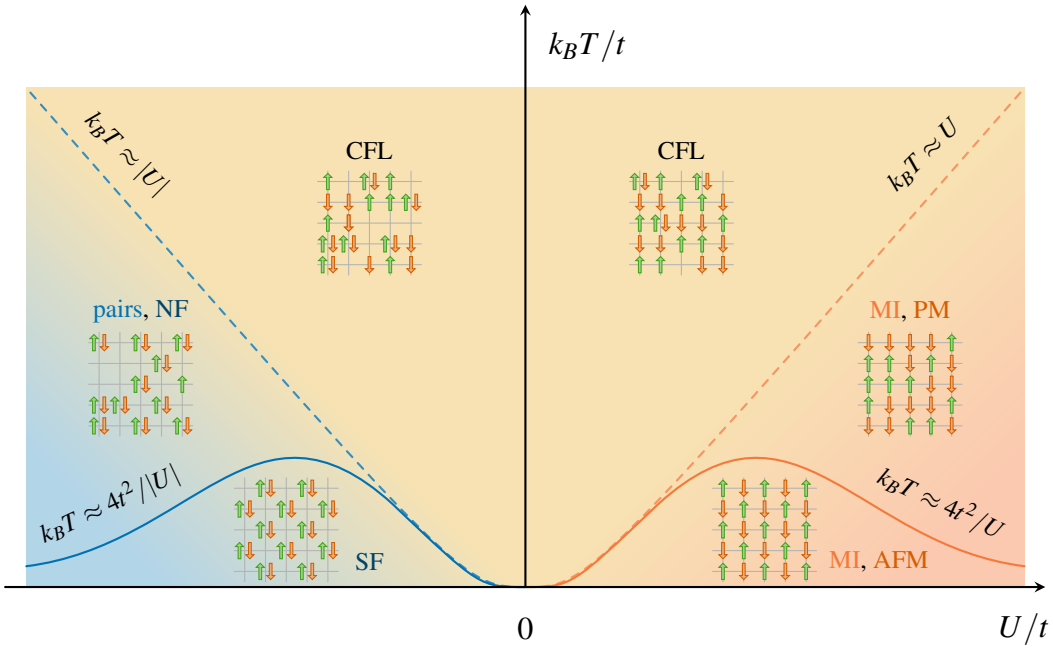


Figure 1.2: The phase diagram of the Fermi-Hubbard model on a 3D cubic lattice at half filling over values of temperature T and interaction strength U in units of hopping energy t. At low interaction strengths relative to temperature ($|U| < k_bT$, upper centre) the particles exist as a correlated Fermi liquid with density fluctuations as in metallic materials (CFL). For strong repulsive interactions ($U > 0$, $U > k_bT$, lower right), the system becomes a Mott insulator (MI) with particles restricted to individual sites. Within this regime a second order phase transition (solid red line) separates paramagnetic (PM) and anti-ferromagnetic (AFM) phases at high and low temperatures. For strong attractive interactions ($U < 0$, $|U| > k_bT$, lower right), particles form spin pairs with a second order phase transition (solid blue line) separating normal fluid (NF) and superfluid (SF) states at high and low temperatures.

ducting Hamiltonian, simulating the slow change of the Hamiltonian to a weakly perturbed Hubbard model[3] such that the state adiabatically evolves to a state close to the ground state of this perturbed Hubbard Model. Following this, as proposed in [37], a phase estimation algorithm [38] would be performed using the resulting state as input and simulated unitary evolution under the (perturbed) Hubbard Hamiltonian as the unitary after which the measurement of the phase will project the state into an eigenstate of this Hamiltonian, which, if the overlap is large enough, is likely to be the ground state. With a ground state prepared the expectations of numerous interesting operators could then be measured, electron density correlations and Green's

[3]The weak perturbation ensures that the gap of the Hamiltonian does not close during the change, without biasing a possible tendency towards superconductivity, see [36] for further details.

functions which would provide evidence of phases with long range orders or indeed lack thereof.

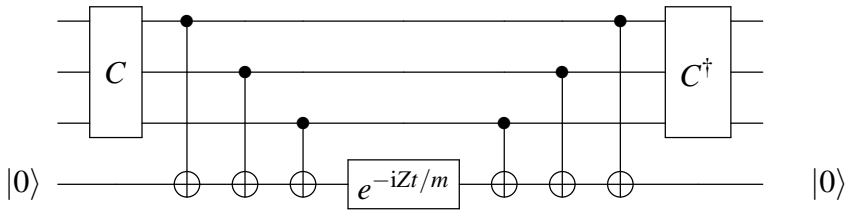
The above method requires the representation of fermionic Hamiltonians and states and simulation of unitary dynamics of Hamiltonians on systems of qubits. The latter can inform the approach for the former so both topics will be reviewed in the following sections.

1.3 Hamiltonian Simulation

The first published algorithm for Hamiltonian dynamics simulation due to Lloyd [2] is based off the Suzuki-Trotter expansion. The basic idea is that for a Hamiltonian that can be expressed as $H = \sum_{i=1}^{N} h_i$ where h_i are k-local terms then the unitary dynamics of the Hamiltonian can be approximated as

$$e^{-\mathrm{i}Ht} = \left(e^{-\mathrm{i}h_1 t/m} e^{-\mathrm{i}h_2 t/m} \cdots e^{-\mathrm{i}h_N t/m}\right)^m + \varepsilon, \tag{1.30}$$

where $\varepsilon \sim O(t^2/m)$. To simulate the unitary evolution within arbitrarily small error ε, the number of required steps m scales with $O(t^2/\varepsilon)$. The $e^{-\mathrm{i}h_i t/m}$ steps themselves are implemented "manually" via small scale circuits; usually one breaks the Hamiltonian down into k-local Pauli terms which can be simulated with a circuit of this form ($k = 3$)



where C denotes a layer of single qubit Clifford gates. On most quantum devices the 2-qubit gate part of this circuit will dominate the runtime setting the real-time scaling of the algorithm to $O(kt^2/\varepsilon)$.

A more general result due to Aharonov yielded a more general algorithm which could simulate the dynamics of any sparse Hamiltonian with running time scaling as

$O(\text{poly}(d)t^{3/2}/\sqrt{\varepsilon})$ where d is the sparsity[4] of the Hamiltonian matrix. This method is less easily stated than Lloyd's algorithm so the reader is instead directed to the original publication and the explanation presented in this essay [39]. Simulation algorithms which scale with sparsity rather than locality are desirable because the Hamiltonian's locality is not necessarily restricted, however a relationship exists between a Hamiltonian's locality k and sparsity d

$$d \leq 2^k N \tag{1.31}$$

where N is the number of terms in the Hamiltonian. With this being the case, reducing the Hamiltonian's locality will reduce the upper bound on the runtime of such an algorithm.

Subsequent improvements in Hamiltonian simulation have been made based on the quantum walk formalism due to Childs [4]. At the heart of these methods is a quantum walk type operator W with eigenvectors $|\mu_\pm\rangle$ and corresponding eigenvalues $\mu_\pm = \pm e^{\pm \arcsin \lambda}$, where λ are eigenvectors of $\frac{H}{\|\text{abs}(H)\|}$ with $\text{abs}(H) = \sum_{ij} |H_{ij}|$. Child's original paper presents a simulation algorithm based on a modified version of W with H replaced by δH for some small δ. Repeated application of this modified walk (conjugated by some isometry) approximates evolution under H by exploiting the fact that $\arcsin \lambda \approx \lambda$ for small values. The resulting algorithm scales as $O(t^{3/2}/\sqrt{\varepsilon})$, simulating the evolution with fidelity $(1-\varepsilon)$. A further improvement was made by combining the walk operator with quantum phase estimation to produce a scaling of $O(t/\sqrt{\varepsilon})$ which is optimal in time due to the no fast forwarding theorem [40] which precludes a generic Hamiltonian simulation algorithm with sub-linear scaling in t. A full exposition of the above and further discussion can be found in the original publication [4].

Berry et al [5] devised another quantum walk based algorithm for the d-sparse

[4] maximum number of non-zero elements per row

case. They show that

$$\sum_{m=-\infty}^{\infty} J_m(-td\|H\|_{\max})\mu_{\pm}^m = e^{-i\lambda t}, \tag{1.32}$$

where J_m are Bessel functions of the first kind and $\|H\|_{\max}$ denotes the maximum absolute value of the elements of H. Noticing that this sum yields the same value regardless of the associated sign of $\mu_{\pm}$, one sees that

$$\sum_{m=-\infty}^{\infty} J_m(-td\|H\|_{\max})W^m \tag{1.33}$$

effectively applies the time evolution operator of H over the subspaces spanned by the pairs $|\mu_{\pm}\rangle$. Using the linear combination of unitaries method shown in [41] they implement the truncated sum

$$\sum_{m=-k}^{k} J_m(-td\|H\|_{\max})W^m. \tag{1.34}$$

Choosing the cutoff k that approximates the evolution witin error ε results in scaling $O(\tau\frac{\log(\tau/\varepsilon)}{\log\log(\tau/\varepsilon)})$ with $\tau = td\|H\|_{\max}$. This is close to the optimal scaling which is shown to be $\Theta(t + \frac{\log 1/\varepsilon}{\log\log 1/\varepsilon})$ in the same paper.

This optimal scaling was reached by Low with algorithms presented in [6, 7] which use *quantum signal processing* [42] a method through which the eigenphases of a unitary can be transformed like

$$U = e^{i\theta}|u_\theta\rangle\langle u_\theta| \to \tilde{U} = e^{ih(\theta)}|u_\theta\rangle\langle u_\theta| \tag{1.35}$$

for some real function h using only controlled-U operations, a single ancilla and single qubit rotations. The optimal scaling of $\Theta(\tau + \frac{\log 1/\varepsilon}{\log\log 1/\varepsilon})$ is reached by using this method to apply the function $h(\theta) = -\tau\sin(\theta)$ to the eigenphases of the quantum walk operator W, thus transforming its eigenvalues to $e^{-i\lambda t}$ for both $|\lambda_{\pm}\rangle$ as in the previous algorithm.

Childs compares the above methods of Hamiltonian simulation (among others)

in [8] to investigate which may be the most likely to yield the first example of a quantum speed-up for this problem. He concludes that, while signal processing had the best rigorous long-term scaling and guarantees on accuracy, the Trotter expansion method may be more likely to show results on small near term devices (albeit with looser, empirical error bounds based off classical simulations of small instances) due to the large multiplicative constants on the running time of the more sophisticated methods rendering them out of reach in the near term.

In many Hamiltonian simulation algorithms, including the top picks in [8], the scaling depends on the Hamiltonian locality or the sparsity, which is itself upper bounded by the locality of Hamiltonian terms. When simulating fermionic systems on qubits, the locality is a quantity that one has some degree of control over as section 1.5 will discuss.

1.4 Stabilizer Codes: A Brief Review

Before discussing fermionic encodings, some essential background on stabilizer codes should be covered.

Stabilizer codes are a class of *quantum error correcting codes* described by the stabilizer formalism, introduced by Gottesman in [43]. Quantum error correcting codes are a method to protect quantum information against noise by storing it in a subspace of a larger system. A stabilizer code on n qubits is defined by a stabilizer group $\mathcal{S}$[5] which is an Abelian subgroup of the Pauli group

$$\mathcal{P}_n = \langle \mathrm{i}, X_j, Y_j, Z_j \ : \ j \in \{1,\ldots,n\} \rangle, \tag{1.36}$$

where $-I \notin \mathcal{S}$. Let m be the size of a minimal generating set for $\mathcal{S}$, this is the *rank* of $\mathcal{S}$. As all elements of $\mathcal{S}$ commute, they all share the same eigenstates, furthermore it can be shown that there exists a subspace on which all elements have an eigenvalue of $+1$. Let the *codespace* be

$$C_{\mathcal{S}} := \{ |\psi\rangle \in \mathbb{C}^n \ : \ S|\psi\rangle = |\psi\rangle \ \ \forall S \in \mathcal{S} \}, \tag{1.37}$$

[5]The symbol $\mathcal{S}$ will refer to the stabilizer group of a code and the code itself interchangeably.

this subspace has dimension 2^{n-m}, equivalent to $n-m$ qubits. Then the codespace of $\mathcal{S}$ encodes $k = n-m$ *logical qubits* in the space of n *physical qubits*. A state on a logical qubit is denoted by a tilde $|\tilde{\psi}\rangle \in \mathbb{C}^k$ with a non-tilded state being on the physical qubits $|\psi\rangle \in \mathbb{C}^n$, the physical qubit states that represent logical qubit states are often called *codewords*.

The states of the logical qubits encoded by $\mathcal{S}$ can be manipulated and differentiated via *logical operators*. These are elements of the *normalizer* $N(\mathcal{S})$ of the stabilizer group which is the set of operators that commute with $\mathcal{S}$ and therefore preserve the codespace but do not necessarily leave states within unchanged. The most trivial example of these are the elements of the stabilizer group $S \in \mathcal{S}$ which by definition (see eq. (1.37)) leave the elements of $C_{\mathcal{S}}$ completely unchanged, each S can then be thought of as a logical identity. Non-trivial logical operators map between different elements of $C_{\mathcal{S}}$ and these are used to define different logical states. For instance, if $2k$ Pauli operators are found such that each one anticommutes with one other and commutes with the rest then these may be identified as logical $\tilde{X}_j$ and $\tilde{Z}_j$ operators for all k logical qubits, through which the logical qubit states can be identified (eigenstates of the $\tilde{Z}_j$). Here the tilde denotes encoded operators. As with the "logical identities", there exist multiple versions of non-trivial logical operators. For a single $\tilde{X}_j$ the set of equivalent operators is simply $\{\tilde{X}_j S \ : \ \forall S \in \mathcal{S}\}$. By eq. (1.37) these must all have the same action on elements of $C_{\mathcal{S}}$. The minimum weight[6] of any non-trivial logical operator is called the *code distance*.

The stabilizer group of a stabilizer code is also used to detect errors. As the codespace is defined as the +1 eigenspace of $\mathcal{S}$ then any $S \in \mathcal{S}$ measured as an observable will yield a +1 result on a state in the codespace. Conversely, an S measurement with a -1 results confirms that a state is out of the codespace. To see this consider a codeword $|\phi\rangle$ affected by a Pauli error E that anticommutes with S, so the final state is $E|\phi\rangle$. By these properties

$$SE|\phi\rangle = -ES|\phi\rangle = -E|\phi\rangle \tag{1.38}$$

[6] Weight here refers to the number of qubits on which an operator acts non-trivially.

so clearly E removes $|\phi\rangle$ from the codespace. Errors of this form are called *detectable errors* as they can be detected by "flipped" stabilizer measurements. Consider now a Pauli error E' which commutes with $\mathcal{S}$, such an error will flip no stabilizers and is therefore *undetectable*. These errors can either be stabilizer elements, having no effect on the encoded state, or they can be logical operators that change the information encoded by $\mathcal{S}$.

The stabilizer group also aids in the correction of errors. Any detectable error flips some set of stabilizer elements making their measurement outcome -1, the set of stabilizer measurement outcomes after an error called the *syndrome* of the error. This syndrome can be expressed as simply the results from measuring the elements of a generating set of $\mathcal{S}$. A syndrome of all +1 measurements is called a *trivial syndrome*, stabilizers and logical operators induce trivial syndromes. Error correction on a stabilizer code is then performed by periodically measuring some generating set of stabilizers, noting the syndrome, deciding which error is most likely to have caused it and then applying a *correction operator* which multiplies with the error to form a stabilizer element. This procedure is performed by a classical algorithm called a *decoder*.

One may wonder why the above analysis was restricted to Pauli errors when a quantum system will in general be subject to a continuum of possible errors. Consider the fact that any coherent error on a qubit system is a linear combination of Paulis which transforms the system into a superposition of states affected by different Pauli errors. Coherent measurement of the stabilizers will cause this superposition to collapse into a sector associated with Paulis that give a specific syndrome. If the error is weight $\leq w$ and the code can distinguish Pauli errors up to weight w then the collapsed state will be affected by a single Pauli error which can then be corrected. This is known as error discretisation, and it applies to more general quantum processes which can be seen as probabilistic combinations of coherent processes.

An error correcting code's capabilities can be summarised by the triple $[[n,k,d]]$, where n is the number of physical qubits use by the code, k is the number of logical

qubits encoded and d is the code distance, the weight of the lowest weight logical operator. The ratio between n and k gives a good measure on the efficiency of the code's use of resources and the code distance is an important property for quantifying a code's error correcting capabilities. The code distance d quantifies the code's error correcting ability. To see this recall that in most noise models low weight errors are more likely. Most decoders will then assume that a syndrome indicates the smallest possible error that can cause it. Consider a code with a logical operator $L = L_1 L_2$ with $w(L_1) > w(L_2)$ where w is the weight of an operator. L_1 and L_2 will have the same syndrome as their product is a logical operator but since $w(L_1) > w(L_2)$, if an L_1 error occurs on the code then a decoder will usually assume that L_2 has actually occurred and attempt to correct by applying an operator of the form SL_2 for $S \in \mathcal{S}$. The result of this erroneous correction will then be a logical operator SL and while the state of the system will be returned to the codespace, the encoded information will have changed. If a code has a code distance d then the lowest weight error for which this can happen is $\lceil \frac{d}{2} \rceil$, so if a code has distance d then it can reliably correct errors of weight $\leq \lceil \frac{d}{2} \rceil - 1$.

Another important metric for an error correcting code is its *threshold*. This term comes from the *quantum threshold theorem* [44, 45] which states that if errors occur on a quantum device below a certain constant threshold probability p_{th} then quantum circuits can be performed fault-tolerantly to arbitrary depth. Each error correcting code will have its own value of p_{th} for a given noise model.

1.4.1 A Simple Example: 3-qubit Repetition Code

This code encodes $k = 1$ logical qubit using $n = 3$ physical qubits, the logical states are

$$\left|\tilde{0}\right\rangle = |000\rangle, \quad \left|\tilde{1}\right\rangle = |111\rangle, \tag{1.39}$$

and logical operators can be defined

$$\tilde{X} = X_1 X_2 X_3, \quad \tilde{Z} = Z_1 Z_2 Z_3. \tag{1.40}$$

This code can also be described with the stabilizer formalism, its stabilizer

group is

$$\mathcal{S} = \langle Z_1Z_2,\, Z_2Z_3 \rangle = \{I,\, Z_1Z_2,\, Z_2Z_3,\, Z_1Z_3\}. \tag{1.41}$$

The minimal generator shown has 2 elements so $\mathcal{S}$ is rank 2 and the code encodes $3-2=1$ qubits. Operators XXX and ZZZ commute with the stabilizer and mutually anticommute so they can be chosen as $\tilde{X}$ and $\tilde{Z}$ respectively, the logical states $|\tilde{0}\rangle$ and $|\tilde{1}\rangle$ can then be chosen as the ± 1 eigenstates of $\tilde{Z}$ within the codespace. The code distance can be found by considering the full sets of logical operators

$$\begin{aligned} \{\tilde{X}\} &= \{X_1X_2X_3,\, -Y_1Y_2X_3,\, -X_1Y_2Y_3,\, -Y_1X_2Y_3\}, \\ \{\tilde{Z}\} &= \{Z_1Z_2Z_3,\, Z_1,\, Z_2,\, Z_3\}, \end{aligned} \tag{1.42}$$

the lowest weight logical Pauli that can be constructed is weight 1 so the code distance is $d=1$. This is not a particularly good code because it cannot reliably correct errors above weight 0. It can detect errors that flip less than 3 bits as they anticommute with stabilizers but it cannot detect any phase-flip errors (Pauli Z errors) as they are all logical operators.

1.4.2 A More Complex Example: The Toric Code

The toric code, introduced in [18], is a good example of the stabilizer formalism's powerful ability to concisely describe complex code structures.

Consider an $L \times L$ square lattice with periodic boundary conditions (i.e. the lattice lives on the surface of a 2-torus) with sets of vertices, edges and faces V, E and F. Place a qubit on every edge of the lattice such that a qubit on edge e is indexed by e and define the stabilizer

$$\mathcal{S} = \langle \prod_{e:v\in e} X_e,\, \prod_{e\in f} Z_e \;:\; \forall v \in V,\, \forall f \in F \rangle. \tag{1.43}$$

In words, for every vertex v a "star" operator is defined as X on the qubits of its four surrounding edges and for every face f a "plaquette" operator is defined as Z on the qubits of its four surrounding edges. The stabilizer $\mathcal{S}$ is generated by these star and plaquette operators (see fig. 1.3). A minimal generating set for $\mathcal{S}$ can be made from

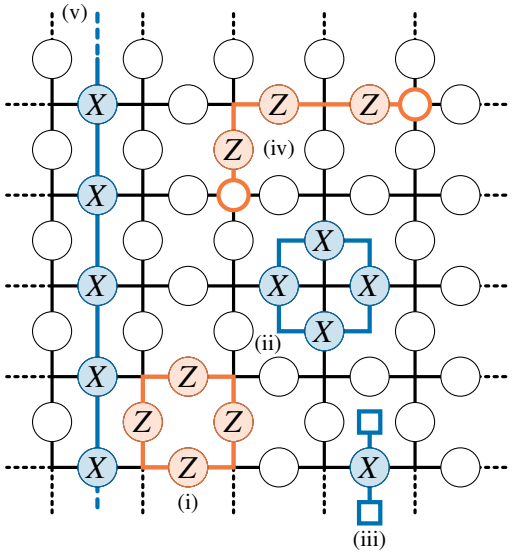


Figure 1.3: Operators on the toric code on a 5×5 torus. (i) A plaquette stabilizer (ii) a star stabilizer (iii) a single X error with the flipped plaquette stabilizers highlighted by squares (iv) a string of Z operators with the flipped star stabilizers highlighted by circles (v) a logical operator. Both (iii) and (iv) also provide illustrations for pairs of m and e quasiparticles respectively.

stars for all but one vertex and plaquettes for all but one face, the rank of $\mathcal{S}$ is then $|V|+|F|-2 = |E|-2$ and the code then encodes $n-k = |E|-|E|+2 = 2$ logical qubits regardless of lattice size.

To find the logical operators of the code it is helpful to first consider the error syndromes. A Z error on a qubit will flip the star operators at either end of its associated edge, a product of Z errors on 2 edges that share a vertex will flip the two stars at the vertices they don't share but their shared star operator will be flipped twice leaving its outcome as +1. One finds that if Z errors are chained together to form strings connecting vertices of the lattice, the product will flip the star operators at each end of each string. More precisely, star operators will be flipped if an odd number of their incident edge qubits have a Z acting on them and will otherwise not be flipped. X errors have a similar property only their strings run from face to face across edges and flip plaquette stabilizers at each end (see fig. 1.3). If a string of X or Z errors forms a closed loop then there are no end points for flipped stabilizers and the syndrome is trivial, this means that closed loops of X and Z operators are either logical operators or stabilizers. It can be shown that *trivial cycles*, loops that

are contractable, correspond to stabilizers and *non-trivial cycles* are logical operators (see fig. 1.3). There are 4 non-trivial cycles up to multiplication by stabilizers and these can be chosen as the $\tilde{X}$ and $\tilde{Z}$ operators for the 2 logical qubits which can then define the logical qubit states. The code distance is then the total width of the torus as that is the length of the shortest non-trivial cycle.

The toric code belongs to a class of error correcting codes called *topological codes*, named as such because their properties can be described through a topological lens as in the above discussion.

1.4.2.1 Excitations on the Toric Code

The codespace of the toric code is equivalent to the degenerate ground state of the Hamiltonian

$$H_{\text{toric}} = -\sum_S \Pi_S - \sum_P \Pi_P. \tag{1.44}$$

where Π_S are star operators and Π_P are plaquette operators which generate the stabilizer. A Pauli error (with a syndrome) on the ground state will send the system into an excited state, increasing the energy by 2 for every Π_S or Π_P that is flipped. A useful interpretation of this excited state is as quasiparticle excitations on the vertices and faces of the lattice. These quasiparticles can be divided into two types: "electric" e particles which appear on vertices where Π_S operators are flipped and magnetic m particles which appear on faces and are flagged similarly by Π_P operators (see fig. 1.3 for illustration).

The notion of these particles makes for an intuitive picture for understanding error syndromes and corrections. Particles are created in pairs by Z and X errors, a Z (X) error creates a pair of e (m) particles on the vertices (faces) adjacent to the qubit's corresponding edge. If two particles are created at a given point, this corresponds to a stabilizer being flipped twice so no particle will be detected, this implies that the e and m particles are their own antiparticle. With this property the pair creation operators also function as "hopping" operators for the particles. For instance a Z operator on an edge adjacent to an e particle will annihilate it and create another at a new vertex. With this intuition, one can interpret the resolution of syndromes on the

toric code as moving pairs of particles together and annihilating them to bring the system back to the vacuum state.

Another interesting aspect of these particle excitations is that they exhibit exchange statistics. They can be identified using the formalism for hard-core[7] particles provided by Levin and Wen in [46]. The formalism uses "hopping operators" t_{ij} which remove a hard-core particle from state j and replace it in state i, with the property

$$[t_{ij}, t_{kl}] = 0, \quad i \neq j \neq k \neq l. \tag{1.45}$$

The statistics of identical hard-core particles are then found by finding some sequence of hops where a re-ordering of the operators amounts to an exchange of two particles. In particular, one ordering will move a particle from position A to B and another from C to D where the re-ordering will move from A to D and C to B. The end state will be identical up to the phase picked up by the exchange of the two particles. Levin and Wen show that this phase $e^{i\theta}$ can be expressed as

$$t_{il}t_{ki}t_{ij} = e^{i\theta}t_{ij}t_{ki}t_{il} \tag{1.46}$$

where j,, k, l are distinct neighbours of i (ordered in the clockwise direction for the 2D case). One can use the fermionic hopping operator from the Fermi-Hubbard Hamiltonian as the hopping operator $t_{ij} = c_i^\dagger c_j$ and confirm that $e^{i\theta} = -1$. As pointed out in [46], this formalism also applies to quasiparticle excitations such as the e and m particles on the toric code, with Z and X operators being their respective hopping operators. Indeed the e particles display bosonic statistics ($e^{i\theta} = 1$) with each other, as do m particles. A composite particle ε can also be formed by taking an e and m together, one can show by the above formalism that these exhibit fermionic statistics.

1.5 Fermionic Encodings

Most quantum computing architectures are composed of distinguishable qubits as opposed to a system of indistinguishable fermions. These are fundamentally different

[7] Hard-core means that only one particle may occupy a state, effectively an enforced Pauli exclusion principle that may apply to non-fermions.

systems so if one wants to simulate the latter on the former then a mapping must be found between states and operators on a fermionic Fock space to the tensor product Hilbert space of qubits; this is known as a *fermionic encoding*. This section will review two types of encoding,: *N-to-N encodings*, the earliest examples of such mappings, that represent a system of N fermionic modes on N qubits but result in operators which grow with system size, and *local encodings* which have a qubit to mode ratio > 1 but can map local fermionic interactions to local qubit interactions of constant weight, a desirable property for simulation as explained in the previous section. Recently there have been developments in encodings which map systems of N modes to $< N$ qubits by exploiting symmetries such as particle number conservation, generally at the trade-off of non-local encoded operators, these will not be reviewed here for brevity but the interested reader may wish to browse the existing literature [47, 48, 49].

1.5.1 *N*-to-*N* Encodings

The first instance of a fermionic encoding, known as the Jordan-Wigner transformation (JW) [9], was actually proposed as the reverse, a mapping of spin degrees of freedom onto fermions. This found use for the exact solution of 1D spin-chain models like the Ising model long before anyone even conceived of a quantum computer. The transform sees more attention nowadays for its inverse function and despite its performance issues its simplicity still make it an attractive choice as the encoding used in proposals for fermionic simulations on quantum devices [36, 25, 50].

The JW transformation is the most direct fermion encoding in a sense. The starting point is simply assigning a mode to each qubit and having a $|1\rangle$ on a qubit signify an occupied mode. In general a fermionic state maps as

$$(c_1^\dagger)^{b_1}(c_2^\dagger)^{b_2}\cdots(c_M^\dagger)^{b_M}\,|\Omega\rangle \to |b_1, b_2, \ldots, b_M\rangle\,, \quad b_i \in \{0,1\} \tag{1.47}$$

with ordering $(1,2,\ldots,M)$ chosen for the modes. The action of a creation operator

$\tilde{c}_i^\dagger$ on an arbitrary occupation state can be found using the anticommutation relations

$$\begin{aligned} c_i^\dagger (c_1^\dagger)^{b_1} \cdots (c_i^\dagger)^{b_i} \cdots (c_M^\dagger)^{b_M} |\Omega\rangle &= \prod_{j<i} (-1)^{b_j} (c_1^\dagger)^{b_1} \cdots c_i^\dagger (c_i^\dagger)^{b_i} \cdots (c_M^\dagger)^{b_M} |\Omega\rangle \\ &= \delta_{b_i,0} \prod_{j<i} (-1)^{b_j} (c_1^\dagger)^{b_1} \cdots c_i^\dagger \cdots (c_M^\dagger)^{b_M} |\Omega\rangle . \end{aligned} \tag{1.48}$$

In the second line, the $c_i^\dagger$ has been commuted through to the ith place in the product, picking up a minus sign every time it commutes past a creation operator earlier in the ordering and the third line uses the fact that $c_i^\dagger c_i^\dagger = 0$. The action of an *encoded* creation operator $\tilde{c}_i^\dagger$ must be the same on equivalent states, i.e.

$$\begin{aligned} \tilde{c}_i^\dagger |b_1, \ldots, b_i, \ldots, b_M\rangle &= \delta_{b_i,0} \prod_{j<i} (-1)^{b_j} |b_1, \ldots, 1_i, \ldots, b_M\rangle \\ &= \left(\prod_{j<i} Z_j \right) \sigma_i^+ |b_1, \ldots, b_i, \ldots, b_M\rangle . \end{aligned} \tag{1.49}$$

So a creation operator on mode i maps to a $\sigma^+ = (X - \mathrm{i}Y)/2$ operator on the corresponding qubit, multiplied by Z operators on every qubit preceding it in the ordering. These Z parts are sometimes called *Jordan-Wigner strings*. The corresponding encoded annihilation operator is simply the conjugate as in the fermionic system.

These operator mappings are easy to follow but do not maintain operator locality well when used to simulate all but the simplest of systems. Consider a Fermi-Hubbard model, shown again here

$$H_{FH} = -t \sum_{\langle i,j \rangle, \sigma} \left(c_{i,\sigma}^\dagger c_{j,\sigma} + c_{j,\sigma}^\dagger c_{i,\sigma} \right) + U \sum_i n_{i,\uparrow} n_{i,\downarrow}. \tag{1.50}$$

The on-site interaction terms simply map to weight 2 operators as a number operator is a product of a creation and annihilation operator on the same site, which cancel each other's Z-string leaving only an operator on their corresponding qubit. Hopping terms are more difficult. For a 1D lattice one can simply order the modes such that mode i is neighboured by $i \pm 1$ and hopping terms between neighbouring sites will

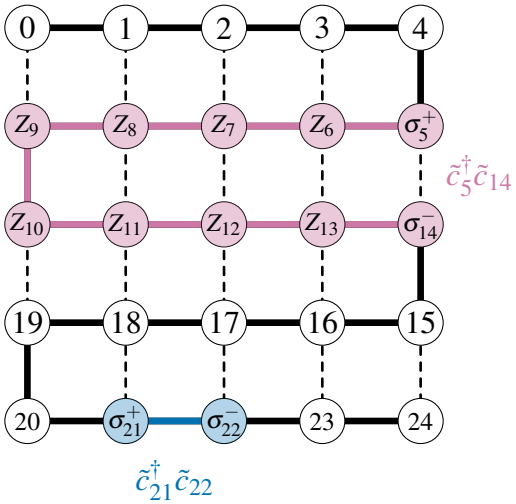


Figure 1.4: Qubits in a JW encoding of a 5×5 lattice of fermions. Qubits are numbered in "snake" ordering along the bold line, dashed lines indicate local connections between modes not adjacent in the ordering. JW encoded hopping terms between neighbouring modes adjacent (blue) and non-adjacent (purple) in the numbering are shown. Note that this qubit system encodes a single spin-layer of a Fermi-Hubbard system on a 5×5 grid.

map like

$$c_i^\dagger c_{i+1} \xrightarrow{JW} \sigma_i^+ \sigma_{i+1}^- \tag{1.51}$$

where the z-strings have cancelled out for all qubits $< i$. However, if the system is on a 2D square lattice then things get more complicated. Consider the "snake" ordering on a square lattice as shown in fig. 1.4. An encoded hopping term between modes adjacent on the lattice and in the ordering will map as in eq. (1.51) but neighbours who are not adjacent in the ordering will map like

$$c_i^\dagger c_j \xrightarrow{JW} \sigma_i^+ \left(\prod_{i<k<j} Z_k \right) \sigma_j^- \tag{1.52}$$

where the JW encoded operator has a Z-string on all qubits between i and j in the ordering (see fig. 1.4). In the worst case, a hopping term will have support on $2L$ qubits on an $L \times L$ square lattice, scaling as $O(N^{1/2})$ for N modes in big-O notation.

Other N-to-N encodings exist with better scaling for hopping terms, their exact workings are rather involved and not directly relevant to this work so they won't be explained in full detail. The first of these was proposed by Bravyi and Kitaev

in [12], with a somewhat clearer explanation in [51]. This so-called Bravyi-Kitaev (BK) encoding works by organising qubits in a binary-tree structure and uses this structure to choose mutually commuting Paulis with weights that scale with the depth of the tree, $\Theta(\log_2 N)$. These Paulis can be said to represent *Majorana operators*, an alternative basis for fermionic operators. For each mode there are two Majorana operators

$$\gamma_j = c_j + c_j^\dagger, \quad \overline{\gamma}_j = \frac{c_j + c_j^\dagger}{\mathrm{i}}. \tag{1.53}$$

They are Hermitian, traceless, self inverse and mutually anticommute for all modes so a set of mutually anticommuting Paulis provides a faithful representation. When used to represent terms of the Fermi-Hubbard Hamiltonian the encoded operators also scale as $\Theta(\log_2 N)$. More recently, Jiang et al developed an optimal N-to-N encoding [52] which uses a ternary tree structure instead to achieve the average operator scaling of $\Theta(\log_3 N)$, which they also prove to be optimal.

1.5.2 Local Encodings

A fundamental problem with N-to-N fermionic encodings is the fact that their encoded fermionic interactions are non-local and grow with the encoded system size. As discussed in section 1.3, this will cause further runtime scaling of Hamiltonian simulation with a growing system size, an important component of quantum approaches for understanding systems like the Fermi-Hubbard Model. These linear and logarithmic scalings may be "small" in the complexity theoretic sense but as current quantum devices are troubled by noise that accumulates over time, a problem that will likely persist in the near future, any savings on runtime allow for longer useful calculations (more on this in section 1.6). With this being the case, the field of local fermionic encodings with non-scaling encoded interactions is well motivated.

As alluded to earlier in this section, local encodings use more qubits than the number of modes of a fermionic system they encode, the Hilbert space of which will have a greater dimension than the Fock space it is to represent. The reason for this discrepancy is that local encodings work by representing fermionic states and operators as codewords and logical operators on a stabilizer correcting code which,

as discussed in section 1.4, use some $n > k$ qubits to represent a 2^k-dimensional system.

A number of local encodings exist, many having been developed within the past few years, although they all use one of two methods of construction. They will be reviewed in this section and organised according to these construction principles. After this their performance in simulating the Fermi-Hubbard model according to various measures will be discussed.

1.5.2.1 Jordan-Wigner String Cancellation Encodings

This method for constructing local encodings uses the predictable structure of Jordan-Wigner encoded operators to cancel out large amounts of Z operators while maintaining the encoded physics. It is easiest to understand when working with the Majorana operators discussed earlier

$$\gamma_j = c_j + c_j^\dagger, \quad \overline{\gamma}_j = \frac{c_j + c_j^\dagger}{\mathrm{i}}, \tag{1.54}$$

and with the knowledge of how JW encodes these operators

$$\begin{aligned} \gamma_j &\xrightarrow{JW} \left(\prod_{i<j} Z_i\right) X_j, \\ \overline{\gamma}_j &\xrightarrow{JW} \left(\prod_{i<j} Z_i\right) Y_j, \end{aligned} \tag{1.55}$$

which can be verified easily.

The first of these encodings was introduced by Verstraete and Cirac in [13] and re-examined in [14]. It will be referred to as the Verstraete-Cirac encoding or VC. It works by introducing auxiliary fermionic modes to the system being encoded, encodes this total system via Jordan-Wigner, creates a stabilizer group from the encoded Majorana operators on the auxiliary modes which then defines a codespace in which the long Z-strings of the JW encoded operators on the "primary" modes of the encoded system can be cancelled out by the auxiliary Majorana stabilizers. This allows local fermionic interactions in dimensions higher than 1 to be represented by

local qubit interactions on the encoding.

It is easiest to understand the above procedure by example so consider an $L \times L$ grid of N fermionic modes ordered as in fig. 1.4 with interactions between adjacent modes. When encoded via JW, a mode in the lattice bulk has 2 local interactions and 2 non-local. To begin constructing a VC encoding of the system, introduce for every mode i an auxiliary mode labelled i' and define a new ordering with primary and auxiliary modes interleaved i.e $1, 1', 2, 2', \ldots, N, N'$. Then for every non-local interaction between primary modes i and j where $i < j$ define

$$S_{ij} := -\mathrm{i}\overline{\gamma}_{i'}\gamma_{j'}, \tag{1.56}$$

these operators are self inverse, mutually commute with one another and also commute with any operator on the primary modes. Now encode this expanded system via Jordan-Wigner using the new ordering such that the encoded versions of these operators are

$$\tilde{S}_{ij} = -\mathrm{i}\tilde{\overline{\gamma}}_{i'}\tilde{\gamma}_{j'} = X_{i'}\left(\prod_{i<k<j} Z_k Z_{k'}\right) Z_j X_{j'}. \tag{1.57}$$

where a tilde denotes an encoded fermionic operator on qubits (see fig. 1.5). Define a stabilizer group $\mathcal{S}_{VC}$ generated by all $\tilde{S}_{ij}$ and define a stabilizer code where the codespace is the mutual $+1$ eigenspace of all stabilizers.

All JW encoded fermionic operators on the primary system commute with the S_{ij} so their encoded counterparts also commute and are therefore logical operators on the codespace. Furthermore since they still behave exactly like fermionic operators they can be taken to represent logical fermionic operators on an encoded Fock space. Fermi-Hubbard hopping operators between i and $i+1$ are weight 3 (having an extra Z from the i'th mode), however a hopping term $c_i^\dagger c_j$ between vertically connected, non-consecutive sites i and j still has a non-local form:

$$\tilde{c}_i^\dagger \tilde{c}_j = \sigma_i^+ Z_{i'}\left(\prod_{i<k<j} Z_k Z_{k'}\right)\sigma_j^- \tag{1.58}$$

This is no problem though because as the system is now in a stabilizer codespace,

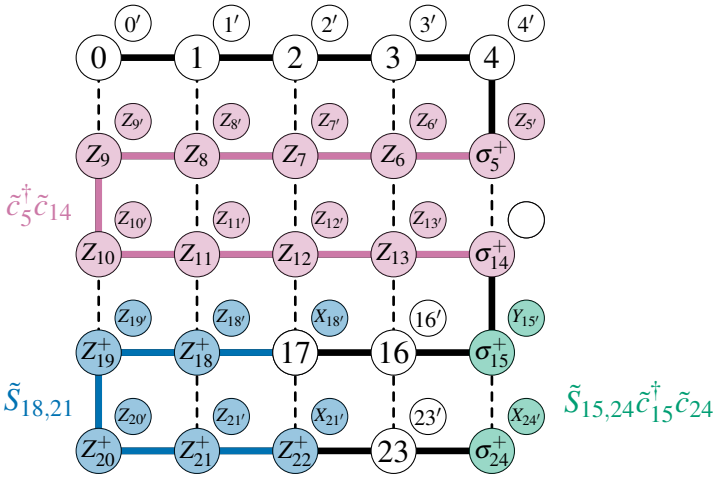


Figure 1.5: Qubits in the VC encoding for a 5×5 lattice of fermionic modes. Purple: a non-local JW encoded hopping operator, blue: a stabilizer used to cancel a Z string, green: an encoded hopping operator which has been localised by a stabilizer.

logical operators can be multiplied by stabilizers to change their form without altering their action, so one can write

$$\tilde{S}_{ij}\tilde{c}_i^\dagger \tilde{c}_j = -\mathrm{i}\sigma_i^+ Y_{i'} \sigma_j^- X_{j'}, \tag{1.59}$$

cancelling out most of the Z-string without changing the behaviour of the operator (see fig. 1.5). This brings the non-local hopping term down to a constant weight of 4 and also keeps the interactions geometrically local[8]. This can greatly improve the performance scaling of simulation algorithms as the locality of the encoded fermionic Hamiltonian can be kept low and constant regardless of encoded system size. Note that to encode the total FH model one needs to encode 2 grids of modes for each spin-layer. However, as the on-site interactions are already local under JW it suffices to apply VC to the two layers independently without having to define stabilizers for the on-site interactions.

For a grid of N fermions encoded with $2N$ qubits the stabilizer has rank $< N$ by the above procedure, meaning that the codespace has some extra gauge degrees of freedom in the auxiliary Majorana operators unused in S_{ij} operators. These can be fixed by simply pairing up the remaining Majoranas in a similar manner to the

[8] That is, spatially nearby when qubits are physically arranged in the lattice structure.

S_{ij} and including them in the stabilizer. This brings the dimension of the codespace down to $2^{2N-N} = 2^N$, thus encoding the full fermionic Fock space for N modes.

In the above example a single auxiliary mode allows for 2 non-local interactions involving the corresponding primary mode to be made local. This is because each of the two Majorana operators from the auxiliary mode can used as part of a S_{ij}. This can be extended to higher degree interaction graphs by adding more auxiliary modes and therefore more qubits, for example, on a 3D cubic lattice graph a mode in the bulk would require 2 auxiliary modes and therefore 3 qubits in the VC encoding.

Another protocol for a 2D square lattice was introduced by Steudtner et al. in [17]. It has a similar working principle so it will only be briefly summarised here. This encoding introduces extra rows of qubits between the primary rows and uses the extra degrees of freedom to define string operators that line up with non-local JW encoded interactions on the primary qubits and then slightly alters the JW operators to ensure that they commute with these strings. A stabilizer codespace is then defined in a similar way such that the altered JW interactions can be cancelled down. This encoding uses $2L$ fewer qubits than VC when encoding a Fermi-Hubbard system on a $L \times L$ grid but at the cost of weight 6 encoded on-site interactions (as opposed to 2).

1.5.2.2 Loop Stabilized Encodings

The remaining local encodings currently known all follow the same design philosophy first introduced in [12] and differ only in execution. The idea is to define qubit operators that replicate the behaviour of a certain basis of even fermionic operators (recall from section 1.1 that these generate all physical fermionic interactions). This basis is based on Majorana operators

$$\gamma_j = c_j + c_j^\dagger, \quad \overline{\gamma}_j = \frac{c_j + c_j^\dagger}{\mathrm{i}}. \tag{1.60}$$

It is worth remembering that these are hermitian, self-inverse and anticommute, put formally

$$\begin{aligned} \forall i: \quad & \gamma_i^2 = \overline{\gamma}_i^2 = I, \quad \gamma_i^\dagger = \gamma_i, \quad \overline{\gamma}_i^\dagger = \overline{\gamma}_i, \\ \forall i \neq j: \quad & \{\gamma_i, \gamma_j\} = \{\gamma_i, \overline{\gamma}_j\} = \{\overline{\gamma}_i, \overline{\gamma}_j\} = 0. \end{aligned} \tag{1.61}$$

Consider a fermionic system with modes and local 2-mode interactions indexed by the vertices $\mathbf{V}$ and edges $\mathbf{E}$ of a connected graph (a 2D grid, for instance), call this the *interaction graph*. For every vertex i and every (ordered) edge (i,j) define *vertex operator* V_i and *edge operator* E_{ij}

$$\begin{aligned} V_i &:= -\mathrm{i}\gamma_i\overline{\gamma}_i \\ E_{ij} &:= -\mathrm{i}\gamma_i\gamma_j. \end{aligned} \tag{1.62}$$

These operators form a complete basis for all even fermionic operators on the system and have the properties

$$\begin{aligned} & E_{ij}^2 = I, \quad V_j^2 = I, \quad E_{ij}^\dagger = E_{ij}, \quad V_j^\dagger = V_j, \quad E_{ij} = -E_{ji}, \\ i \neq j \neq k \neq l \neq m: \quad & [V_i, V_j] = 0, \quad [E_{ij}, V_k] = 0, \quad [E_{ij}, E_{kl}] = 0, \\ & \{E_{ij}, V_j\} = 0, \quad \{E_{ij}, E_{jk}\} = 0, \end{aligned} \tag{1.63}$$

that is, they are self inverse, hermitian, anticommute if they share one vertex index and edge operators differ by a sign if their order is switched. Another important property is that for any ordered set of modes $L = (i_1, i_2, \ldots, i_{|L|})$, with $i_1 = i_{|L|}$ so that L constitutes a cyclic path on the graph

$$\mathrm{i}^{|L|-1} \prod_{x=1}^{|L|-1} E_{i_x, i_{x+1}} = I, \tag{1.64}$$

i.e. closed loops of edge operators cancel to identity. The properties in eqs. (1.63) and (1.64) are equivalent to eq. (1.61) and thus completely define the operators in eq. (1.62). So, if one can define qubit operators $\tilde{V}_i$ and $\tilde{E}_{ij}$ that replicate these properties then they have defined an encoding for a fermionic system on the graph

restricted to one parity sector[9]. The parity sector can be defined by the action of the product of all $\tilde{V}_i$

$$\prod_i \tilde{V}_i = (-1)^P \tag{1.65}$$

where P is 0 for even parity and 1 for odd. For any such encoding, parity can be switched by simply changing the sign of a single $\tilde{V}_i$ without affecting any other properties. The interactions of the Fermi-Hubbard model can be written in terms of these operators

$$c_i^\dagger c_j + c_j^\dagger c_i = \frac{-\mathrm{i}}{2}(E_{ij}V_j + V_i E_{ij}), \quad n_i n_j = \left(\frac{1-V_i}{2}\right)\left(\frac{1-V_j}{2}\right) \tag{1.66}$$

so if local qubit representations of these operators can be defined then a local simulation of Fermi-Hubbard can be carried out. The encodings in this class aim to do just that.

These encodings successfully define Pauli operators that satisfy the conditions in eq. (1.63) but not eq. (1.64) so some of the loops of $\tilde{E}_{ij}$ operators form non-trivial Paulis. Fortunately these loops form an Abelian subgroup of the Paulis and (so long as it does not contain $-I$) can be chosen as a stabilizer group which defines a codespace on which the loop condition in eq. (1.64) is satisfied as well as all others, hence the name of this class of encodings.

The first encoding of this kind was defined by Bravyi and Kitaev in the same paper as the Bravyi-Kitaev encoding discussed earlier [12] and is known as the Bravyi-Kitaev Superfast encoding (BKSF) (the code was also produced independently in [53]). It is constructed by assigning a qubit to every *edge* of the interaction graph, such that for a square grid it is reminiscent of the toric code in section 1.4.2. Encoded vertex operators are Z on all edges around a vertex

$$\tilde{V}_i := \prod_{e:i\in e} Z_e. \tag{1.67}$$

Each edge (i,j) must be given an orientation $\varepsilon_{ij} = -\varepsilon_{ji} \in \{+1,-1\}$ and the edges

[9] This is sufficient for a physical simulation due to the fermion parity superselection rule discussed towards the end of section 1.1.

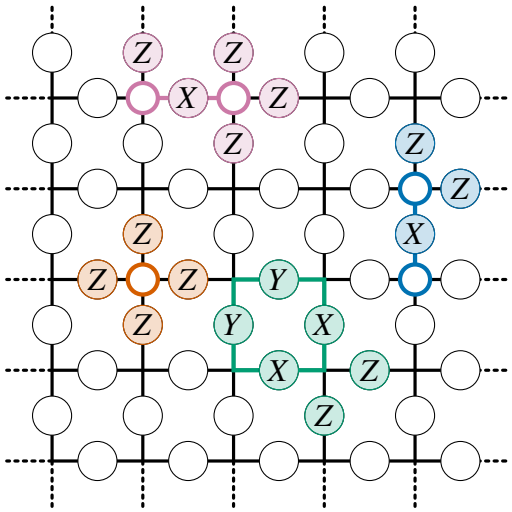


Figure 1.6: Operators on the BKSF encoding on a square lattice with edges around vertex ordered clockwise starting from north. Orange: A vertex operator, purple: a horizontal edge operator, blue: a vertical edge operator, green: a loop of edge operators around a lattice face, this is a stabilizer.

surrounding each vertex must be ordered so that edge operators may be defined

$$\tilde{E}_{ij} := \varepsilon_{ij} X_{(i,j)} \prod_{e:i\in e, e<(i,j)} Z_e \prod_{e':j\in e', e'<(i,j)} Z_{e'}, \tag{1.68}$$

that is, X on the edge (i,j), and Z on the edges surrounding vertices i and j which are before (i,j) in the ordering about that vertex. See fig. 1.6 for an illustration of these operators on a square lattice. These operators satisfy the relationships in eq. (1.63) and, as discussed, the group formed by loops of these edge operators is defined as the stabilizer such that eq. (1.64) is satisfied on the resulting codespace. The stabilizer rank is equal to the number of independent cycles on the interaction graph, which is $|\mathbf{E}| - |\mathbf{V}| + 1$, the dimension of the codespace is then $2^{|E|-(|\mathbf{E}|-|\mathbf{V}|+1)} = 2^{|\mathbf{V}|-1}$. This, paired with the fact that

$$\prod_i \tilde{V}_i = I \tag{1.69}$$

means that the BKSF encodes only the even parity fermionic subspace, as mentioned earlier this parity can be switched by adding a minus sign to the definition of one (or any odd number of) $\tilde{V}$.

In [15], Jiang et al show an encoding with a similar construction on a square

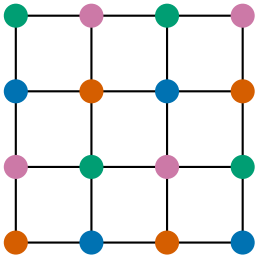


Figure 1.7: Grid site colouring of the MLSC. Alternating green-purple and blue-orange rows which shift horizontally.

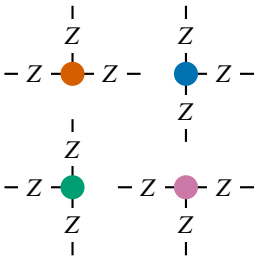


Figure 1.8: Vertex operators of the MLSC around all 4 possible coloured vertices.

lattice which they dub the “Majorana Loop Stabilized Code" (MLSC). This encoding has the novel property of being able to correct single qubit errors in the lattice bulk, that is when the stabilizers are measured, all single qubit errors in the bulk have a unique syndrome. The qubits are assigned to edges of the interaction graph as in BKSF but edge and vertex operators are dependent on a colouring of the lattice vertices and are best represented graphically. With the lattice coloured as in fig. 1.7, the vertex operators are as shown in fig. 1.8 and edge operators are as shown in fig. 1.9 with an arbitrary ± 1 orientation chosen as with the BKSF. Again, as with BKSF this encodes a single parity sector of a fermionic system. One may notice that the product of all vertex operators is not identity and leaves 4 Z operators around each face with an orange top-left corner. This is simply the operator corresponding to a loop of edges around this face and is therefore a stabilizer, so the product of all vertices is therefore identity up to stabilizers and the encoding therefore captures the even subspace if left unaltered by minus signs on the vertices.

Another loop stabilized encoding construction was proposed by Setia et al in [16] which can be defined on an arbitrary interaction graph of even degree. It is constructed by associating $d(i)/2$ qubits with every vertex i where $d(i)$ is the vertex

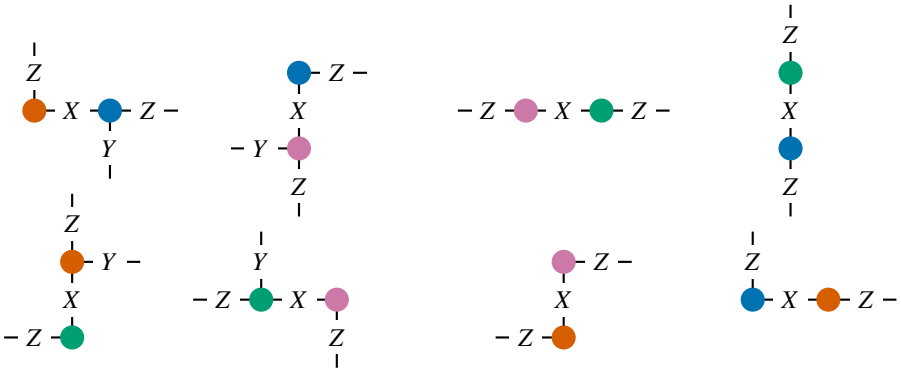


Figure 1.9: Edge operators of the MLSC between all possible pairs of coloured vertices.

degree. For each vertex i, the $d(i)/2$ qubits can be used to encode Majorana operators $\{\gamma^i_1,\ldots,\gamma^i_{d(i)}\}$ using some N-to-N encoding[10]. Around each vertex i an indexing of incident edges o_i is defined such that each incident edge (i,j) is assigned a unique index $o_i(j)\in\{1,\ldots,d(i)\}$, edge operators are then defined as

$$\tilde{E}_{ij} := \varepsilon_{ij}\,\tilde{\gamma}^i_{o_i(j)}\tilde{\gamma}^j_{o_j(i)}, \tag{1.70}$$

where ε_{ij} is the chosen $\pm$ orientation of the edge as in BKSF and MLSC and γ^i_k is a Majorana encoded on the qubits of vertex i under some N-to-N encoding (see fig. 1.10). Vertex operators are defined as

$$\tilde{V}_i := \prod_{k=1}^{d}\tilde{\gamma}^i_k. \tag{1.71}$$

The natural anticommutation relations of the encoded Majoranas at each site ensure that the relations in eq. (1.63) are satisfied while edge loops are used as stabilizers in the usual manner. This encoding has 1 qubit for every edge like BKSF and MLSC so its codespace also corresponds to a fixed parity fermionic subspace. As the interaction graph is even degree it has an Eulerian cycle ζ which contains every edge exactly once. The corresponding cycle of encoded edge operators $\tilde{\zeta}$ has the property

$$\tilde{\zeta} = \pm\prod_i V_i \tag{1.72}$$

[10] since the mode associations of these Majoranas is unimportant the overline notation has been dropped in favour of just numbering them 1 to $d(i)$.

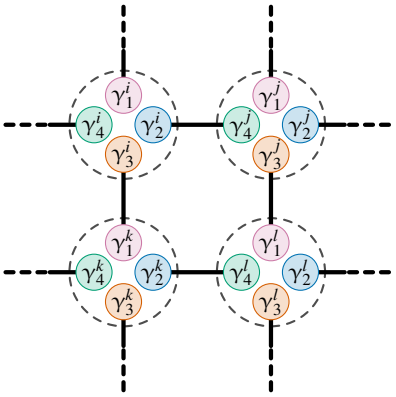


Figure 1.10: Edge operators on the Setia code for a square lattice between vertices i, j, k and l (denoted by dotted circles) in the Majorana picture. The bulk of a square lattice is degree 4 so each vertex has 4 Majoranas assigned to it (encoded by 2 qubits), with Majoranas around vertex i indexed by o_i. In this case, $o_i(j) = 2$ and $o_j(i) = 4$ so the edge operator for (i, j) is the product of the corresponding Majoranas encoded on vertex qubits.

where the sign and therefore the parity sector of the encoding depends on the choice of ε_{ij} for each edge. This encoding can correct single qubit errors if defined on a graph of minimum degree ≥ 6 and has the novel feature of encoded edge and vertex operators with weights that scale as $O(\log d)$ where d is the maximum degree of the graph, as opposed to the $O(d)$ scaling of BKSF. This can be achieved by choosing the Bravyi-Kitaev [12] encoding or the optimal encoding due to Jiang [52] to represent the Majoranas for each vertex. In the paper, the authors propose a single qubit error-correcting encoding of a Fermi-Hubbard Hamiltonian by including 2 "dummy" edges between spin layers at each site, ensuring the interaction graph is degree 6 (in the bulk) and requiring 3 qubits per mode. A construction can be defined with only 2 qubits per mode by independently encoding each spin layer as a square lattice graph but this will be unable to distinguish all single qubit errors.

1.5.2.3 Local Encodings On a Fermi-Hubbard Hamiltonian

Table 1.1 shows the performance of the encodings reviewed in this chapter when used to simulate a Fermi-Hubbard Hamiltonian on a square lattice. This is a useful comparison as this is one of the simplest fermionic models that will likely require a quantum solution as discussed in section 1.2.2. The metrics are relevant as they quantify the resource efficiency of the encodings, particularly in number of working

Mapping	BKSF [12]	VC [13, 14]	MLSC [15]	Steudtner [17]	Setia [16]
Qubit Number	$2L(L-1)$	$2L^2$	$2L(L-1)$	$2L^2-L$	$3L^2$
Qubit to Mode Ratio	$2-\frac{2}{L}$	2	$2-\frac{2}{L}$	$2-\frac{1}{L}$	3
Max Weight Hopping	6	4	4	5	4
Max Weight Coulomb	8	2	6	6	6
Encoded Fermionic Space	Even	Full	Even	Full	Even
Single Qubit Error Correction	No	No	Yes	No	Yes
Graph Geometry	General	General	Square Lattice	Square Lattice	Even Degree

Table 1.1: A comparison of local fermion encodings discussed in this chapter when encoding a Fermi-Hubbard system on an $L \times L$ square lattice. Max weight Coulomb and max weight hopping denote the maximum Pauli weights of the mapped Coulomb ($\tilde{n}_i\tilde{n}_j$) and nearest neighbour hopping ($\tilde{c}_i^\dagger\tilde{c}_j + \tilde{c}_j^\dagger\tilde{c}_i$) terms respectively. Encoded fermionic space denotes whether the full or even fermionic Fock space is represented. Graph geometry denotes the other interactions graphs which the mapping is tailored to.

qubits required and time needed for simulation (FH term weights). Hamiltonian term weights are quoted without accounting for connectivity on hardware which may increase their effective locality if SWAP gates are required for Trotter based simulation algorithms.

1.6 NISQ Hardware

Quantum hardware has come a long way since efforts began, however there is still some distance to go before fault-tolerant quantum computation is a real possibility. The current generation of quantum devices is known as the *NISQ* era, a term coined by Preskill in [11] standing for "Noisy Intermediate Scale Quantum". These devices are characterised by relatively low qubit counts of up to about 100, and noise levels too high to achieve fault tolerance with known error correction procedures compatible with their qubit counts.

Device	Rigetti Aspen-9 [54]	Google Weber [55]	IBM Peekskill [56]	IBM Lagos [56]
Qubit Count	32	54	27	7
2-qubit Gate Fidelity	92%	99.3%	98.98%	99.32%
Readout Fidelity	94%	99.5% / 97%	98.33%	98.89%
T_1 / µs	30	21	266.14	163.88
T_2 / µs	18	n/a	256.61	89.77
2-qubit Gate Time / ns (t_{2q})	148	32	413.27	314.074
T_1 / t_{2q}	203	656	643.99	521.79
T_2 / t_{2q}	122	n/a	620.93	285.82

Table 1.2: Several figures of merit for existing NISQ devices as reported by the companies (gate times for Rigetti were obtained from private correspondence). The 2-qubit gates used for the associated figures are CZ for Rigetti, $\sqrt{iSWAP}$ for Google and $CNOT$ for IBM. IBM has several devices available, Peekskill has the greatest T_1 to t_{2q} ratio and Lagos has the greatest 2-qubit gate fidelity. Google's documentation gives best, median and worst case values for each figure, best case is listed here. Separate readout fidelity figures are given for Google Weber corresponding to the error when measuring a prepared $|0\rangle$ or $|1\rangle$ state respectively.

A number of companies have produced NISQ hardware available for use by the public or businesses including Rigetti, Google and IBM. Notably, a Google made device was used to perform the first successful quantum supremacy experiment [57]. Figures of merit for some of the more recent devices are listed in table 1.2. The values of T_1 / t_{2q} and T_2 / t_{2q} are of particular interest as they give a rough figure of how many layers of 2-qubit gates can be applied before noise washes out any useful information from the system. The values in the table indicate that these NISQ devices can only manage a few hundred and that is without accounting for the error in 2-qubit gate application.

The limitations on circuit depth and lack of proper error correcting capabilities required to prolong this depth mean that for interesting calculations to be performed on NISQ devices quantum circuits need to be as shallow as possible. For this reason, quantum algorithms such as rapid factorisation of large numbers may be out of reach for the near future given the resources required [58]. However, simulation

of small scale instances of the 2D Fermi-Hubbard model which sit at the limits of classical quantum simulation [30] could be feasible on relatively small quantum devices and would represent a significant step towards solving the model and all research progress that may yield. As discussed, such a simulation would require a fermionic encoding and it would be beneficial for the encoding to have low weight FH interactions and also use as few qubits as possible. The main subject of chapters 2 and 3 is a novel encoding which outperforms previous works in these regards.

Chapter 2

A Compact Fermionic Encoding on a Square Lattice

Having covered the necessary background, the main topic of this thesis can now be introduced: a novel fermion to qubit mapping, the *Compact Encoding* (CE), which maps local fermionic interactions, Fermi-Hubbard Hamiltonian terms in particular, to local qubit operators while keeping operator weights and total qubit count low. Recalling the discussion from sections 1.3 and 1.6 in the introduction, low weight encoded operators are desirable as they reduce the runtime cost of Hamiltonian simulation algorithms, which will be essential if they are to be implemented on near term quantum hardware. This chapter will cover the encoding of a 2D square lattice fermionic interaction graph, which yields edge and vertex operators of weight no greater than 3 and has a qubit to mode ratio of no more than 1.5, outperforming all other known local encodings in these metrics simultaneously. The encoding does not correct nor detect all single qubit errors but it does have some error mitigating properties which will be further explored in chapter 4. See table 2.1 for a summary of these details. Constructions for further lattice geometries exist and will be discussed fully in chapter 3, this chapter will serve to familiarise the reader with general design principles of the encoding and also illustrate some interesting features.

The material from this chapter is largely based off [59] by the thesis author and Joel Klassen.

Lattice Type	$L \times L$ Square even face number	$L \times L$ Square majority even faces	$L \times L$ Square majority odd faces
Fermionic modes	$2L^2$	$2L^2$	$2L^2$
Qubit Number	$3L^2 - L$	$3L^2 - L - 1$	$3L^2 - L + 1$
Qubit to Mode Ratio	$1.5 - \frac{2}{L}$	$1.5 - \frac{2}{L} - \frac{1}{2L^2}$	$1.5 - \frac{2}{L} + \frac{1}{2L^2}$
Max Weight Hopping	3	3	3
Max Weight Coulomb	2	2	2
Encoded Fermionic Space	Full	Even	Full Plus Qubit
Corrects Single Qubit Errors?	No	No	No

Table 2.1: The qubit number and max Pauli weights, for the Fermi-Hubbard model, of the fermionic encodings presented in this chapter.

2.1 Preliminaries

The CE is constructed in a similar manner to the mappings discussed in section 1.5.2.2. For convenience some of the key concepts for this style of encoding will be covered again briefly.

Natural fermionic Hamiltonians are sums of products of even fermionic operators $c_k^\dagger c_k$, $c_j^\dagger c_k$, $c_j c_k$ and $c_j^\dagger c_k^\dagger$. Here $c_k^\dagger$ and c_k are the standard fermionic creation and annihilation operators. For modes on a connected graph, the algebra of even fermionic operators can be generated by the edge operators E_{jk} and vertex operators V_j associated with each edge (i, j) and vertex i

$$\begin{aligned} V_i &:= -\mathrm{i}\gamma_i\overline{\gamma}_i \\ E_{ij} &:= -\mathrm{i}\gamma_i\gamma_j. \end{aligned} \tag{2.1}$$

where γ_i and $\overline{\gamma}_i$ are the Majorana operators for mode i

$$\gamma_j = c_j + c_j^\dagger, \quad \overline{\gamma}_j = \frac{c_j + c_j^\dagger}{\mathrm{i}}, \tag{2.2}$$

which are hermitian, self-inverse and mutually anticommuting. The edge and vertex operators satisfy the conditions

$$\begin{gathered} E_{ij}^2 = I, \quad V_j^2 = I, \quad E_{ij}^\dagger = E_{ij}, \quad V_j^\dagger = V_j, \quad E_{ij} = -E_{ji}, \\ i \neq j \neq k \neq l \neq m: \quad [V_i, V_j] = 0, \quad [E_{ij}, V_k] = 0, \quad [E_{ij}, E_{kl}] = 0, \\ \{E_{ij}, V_j\} = 0, \quad \{E_{ij}, E_{jk}\} = 0, \end{gathered} \tag{2.3}$$

and

$$\mathrm{i}^{|L|-1} \prod_{x=1}^{|L|-1} E_{i_x, i_{x+1}} = I. \tag{2.4}$$

The terms in the Fermi-Hubbard Hamiltonian may be written in terms of these operators

$$c_i^\dagger c_j + c_j^\dagger c_i = \frac{-\mathrm{i}}{2}(E_{ij}V_j + V_i E_{ij}), \quad n_i n_j = \left(\frac{1-V_i}{2}\right)\left(\frac{1-V_j}{2}\right). \tag{2.5}$$

Note that Edge operators between spin layers are not required.

2.2 Construction

Consider fermions living on a square lattice of modes. For each vertex of the lattice, define a vertex qubit indexed by the fermionic site j. Now label the faces of the lattice even and odd in a checker-board pattern. For the sake of clarity, assume to begin with that there are in total an even number of faces, and so an equal number of even and odd faces. The case of an odd number of faces is examined in section 2.2.1. Note that the lattice may have unequal side lengths.

Associate a face qubit to the odd faces, as illustrated in fig. 2.1. Give an orientation to the edges of the lattice so that they circulate around the even faces clockwise or counter-clockwise, alternating on every row of faces, also illustrated in fig. 2.1.

Let $f(i,j)$ index the unique odd face adjacent to edge (i,j). For every edge

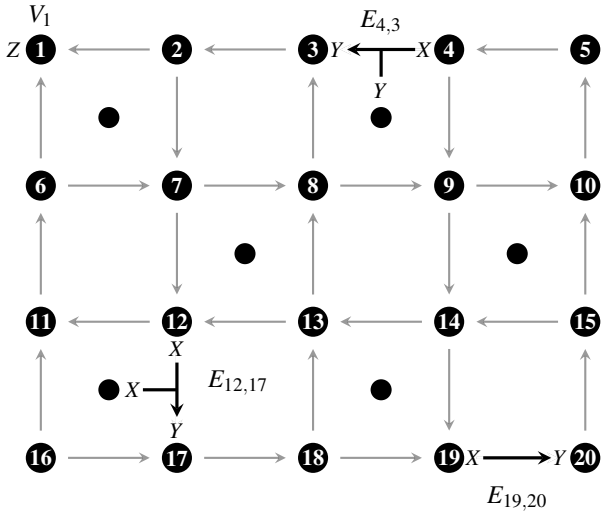


Figure 2.1: Qubit assignment, edge orientation, and examples of mapped edge and vertex operators for a 4×5 square lattice.

(i, j), with i pointing to j, define the following encoded edge operators[1].

$$\tilde{E}_{ij} := \begin{cases} X_i Y_j X_{f(i,j)} & \text{if } (i,j) \text{ is oriented downwards} \\ -X_i Y_j X_{f(i,j)} & \text{if } (i,j) \text{ is oriented upwards} \\ X_i Y_j Y_{f(i,j)} & \text{if } (i,j) \text{ is horizontal} \end{cases} \tag{2.6}$$

$$\tilde{E}_{ji} := -\tilde{E}_{ij}. \tag{2.7}$$

For those edges on the boundary which are not adjacent to an odd face, omit the third Pauli operator which is meant to be acting on the non-existent face qubit. For every vertex j define the encoded vertex operators

$$\tilde{V}_j := Z_j. \tag{2.8}$$

This specifies all encoded vertex and edge operators, they are illustrated in fig. 2.1.

It is not difficult to see that this encoding satisfies all the conditions in 2.3. The intuition is that one may think of a directed edge as having an X on the tail and a Y on the head. Whenever the head of one edge touches the tail of another, then those two edge operators anti-commute, while if two edges touch head to head or tail to tail,

[1]The difference in sign between the vertical up and down arrows ensures that closed loops around odd faces are equal to I and not $-I$.

then they commute. By adding a qubit at some faces, and choosing an appropriate orientation for the edges, one can enforce the additional necessary anti-commutation relations at the face qubits, as has been done here.

For M fermionic modes, this encoding uses fewer than $1.5M$ qubits. Most importantly this construction results in Fermi-Hubbard terms with Pauli weight at most 3. A critical feature which makes the Pauli weights and qubit numbers so low is that the face qubits are used extremely efficiently, each one enforcing anti-commutation relations at four bounding corners.

Just as in the encodings in section 1.5.2.2, the encoding also demands a restriction to a stabilizer code space, in order to satisfy eq. (2.4). The stabilizers $\tilde{S}_p$ are indexed by all of the closed loops p on the lattice, and are given by:

$$\tilde{S}_p = i^{(|p|-1)} \prod_{i=1}^{(|p|-1)} \tilde{E}_{p_i p_{i+1}} \tag{2.9}$$

However unlike these encodings, some of these stabilizers are equal to I. Take for instance the loop of edge operators going around vertices 7, 8, 12 and 13 in fig. 2.1, the product of those edges is I. This is true for every odd face. On the other hand the stabilizer loops around even faces are non-trivial. Both cases are illustrated in fig. 2.2. Therefore the number of independent stabilizer generators is half the number of faces, while the number of qubits is the number of fermionic modes plus half the number of faces. Thus the encoded Hilbert space is of the same dimension as the full fermionic Fock space.

Since the full fermionic Fock space is encoded, single fermions also admit a representation. It suffices to specify one Majorana operator, and all other fermions may be constructed using edge and vertex operators, and linear combinations of Majoranas. A logical Majorana operator γ_j must anti-commute with all edges adjacent to site j and the vertex operator V_j. Consider the corners of the lattice associated with an odd face. Such a corner j either has arrows pointing into it or pointing away from it. If the arrows point into the corner then choose the encoded Majorana operator to be $\tilde{\gamma}_j = X_j$, otherwise choose it to be $\tilde{\gamma}_j = Y_j$. The choice of

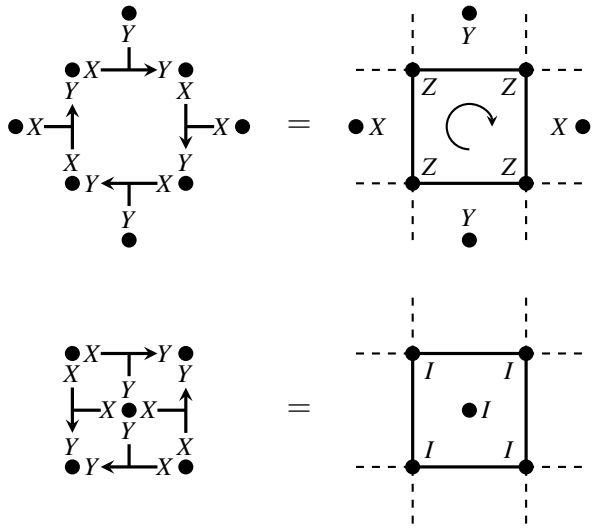


Figure 2.2: Loops of edge operators around faces on the square lattice. Note that loops around even faces are non-trivial Pauli operators and loops around odd faces cancel out to identity. Phases have been omitted.

corner is arbitrary. In the case of an even number of faces, there are two possible choices of corners, and once a corner is chosen, then the equivalent operator at the other corner corresponds to a *Majorana hole operator* $h_i := \gamma_i \prod_j V_j$, so called as it is the product of all Majorana operators *except* for γ_i at site i, where there is a "hole"[2]. Examples are illustrated in fig. 2.3.

2.2.1 Odd Number of Faces

If the lattice has an odd number of faces, then there are two possible checker-board patterns. In one case (case (a)) there is an extra even face and every corner is even. In the other case (case (b)) there is an extra odd face and every corner is odd. Once a choice has been made, then one may proceed with constructing the encoding as prescribed in section 2.2. This is illustrated in Figure 2.4.

In case (a) there is one more stabilizer than face qubit. Furthermore, it is not difficult to see that, up to stabilizers, $\prod_i \tilde{V}_i = I$, and so in this case the code space is restricted to the even fermion subspace, just as in the encodings in [12, 15, 16]. This is corroborated by the fact that, unlike for lattices with an even number of faces where a Majorana can be "injected" into an odd corner, here there are no odd corners at which to do this, and so odd fermionic operators do not admit a representation in

[2]The overlined $\overline{\gamma}_i$ also has a corresponding hole, $\overline{h}_i := \overline{\gamma}_i \prod_j V_j$

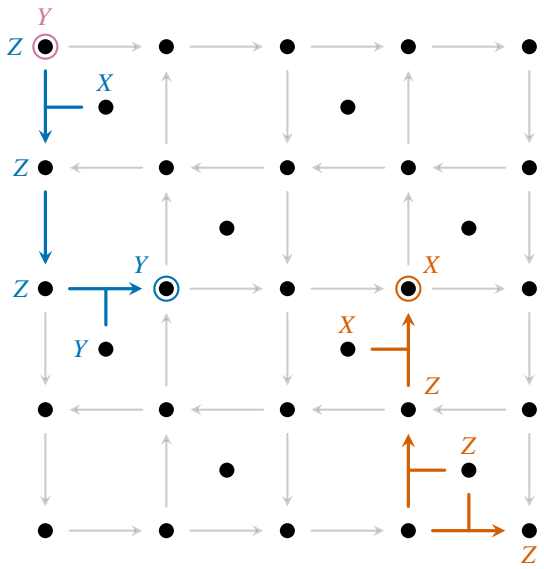


Figure 2.3: Single Majorana and hole operators on a lattice with an even number of faces. Purple: A single particle operator at an odd corner, we choose this to be the source of encoded single Majorana operators $\tilde{\gamma}_j$. Blue: A single Majorana operator in the bulk of the, transported from the top left odd corner by edge operators. Orange: A Majorana hole operator in the bulk of the lattice, transported from the other odd corner by edge operators.

this code.

Case (b) is more interesting. Here the number of face qubits is one more than the stabilizer rank, and so the encoded space is effectively the full fermionic tensored with one qubit: $\mathbb{C}^2 \otimes \mathbb{F}$. Furthermore, at each of the four odd corners, single qubit X or Y operators[3] inject Majorana-like operators that anticommute with the edge and vertex operators incident at the corner. They are described as "Majorana-like" as they behave like Majorana operators with respect to edge and vertex operators but they do not always anticommute with one another, suggesting some further details. Label the four "species" of Majorana A_i, B_i, C_i and D_i, injected at each of the four corners A, B, C, and D (as labelled in fig. 2.4) and then translated by edge operators to site i. These operators satisfy the following commutation and anti-commutation relations:

$$\begin{aligned} \{M_i, M_j\} &= 0 \quad \forall M \in \{A,B,C,D\},\ i \neq j \\ [M_i, M'_j] = \{M_i, M'_i\} &= 0 \quad \forall M \neq M' \in \{A,B,C,D\}. \end{aligned} \tag{2.10}$$

[3] X if arrows point to the corner, Y if they point away.

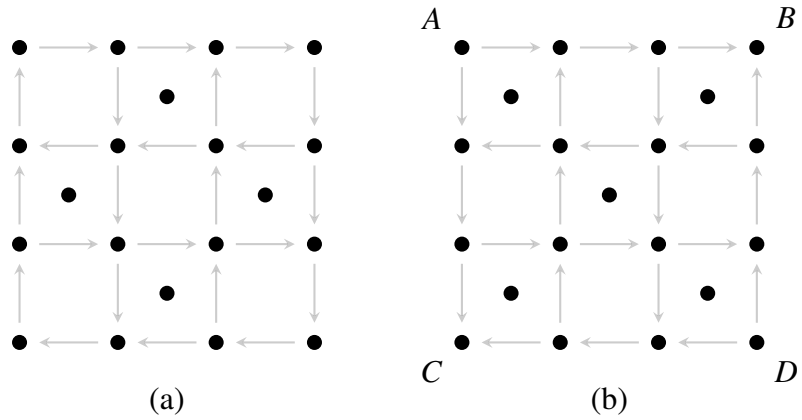


Figure 2.4: Two possible choices of encoding for a lattice with an odd number of faces. In case (b) with more odd faces, the corners have important properties and so are labelled.

That is, a pair of particles of the same species anticommute at different modes like normal Majoranas while a pair of different species will commute at different modes yet anticommute at the same mode.

Particles of the same species when translated to the same vertex will "fuse" to form identity up to a stabilizer, as is expected with self-inverse Majorana operators. One wonders what would happen if two different species were to be fused in the same manner. It turns out that pairs of different species fuse into non-trivial string defects:

$$\begin{aligned} A \times B \approx C \times D \approx \varepsilon_1 \\ A \times C \approx B \times D \approx \varepsilon_2 \\ A \times B \times C \times D \approx I, \end{aligned} \tag{2.11}$$

where this equivalence is modulo stabilizers and logical edge and vertex operations.

One may privilege one corner (for instance, A) as the Majorana operator on the fermionic system and the identity on the qubit system. The remaining corners may then be identified as hole operators on the fermionic system coupled to a Pauli operator on the qubit system

$$A_i = I \otimes \gamma_i, \quad B_i = Y \otimes h_i, \quad C_i = X \otimes h_i, \quad D_i = Z \otimes h_i \tag{2.12}$$

recalling that $h_i = \gamma_i \prod_j V_j$. It then immediately follows that the non-trivial string

defects correspond to Pauli operators on the logical qubit system,

$$\varepsilon_1 \approx Y, \quad \varepsilon_2 \approx X, \tag{2.13}$$

and that the "different species" of Majorana are in fact the same species of Majoranas and holes but dressed with Pauli operators acting on the extra encoded qubit. The encoded $\tilde{Y}$ and $\tilde{X}$ operators are illustrated in fig. 2.5. They are strings of Zs along the bottom and right edge respectively, and strings of Ys along the bottommost row of faces and Xs along the right most column of faces respectively.

Note that if one treats one of these operators as a stabilizer, then one restricts to the full fermionic code space without an extra logical qubit. In this case, as one should expect, there are only two corners in which to inject a Majorana, since injecting a Majorana at either of the other two corners would anti-commute with the chosen stabilizer. These two species are then the Majorana and its hole counterpart.

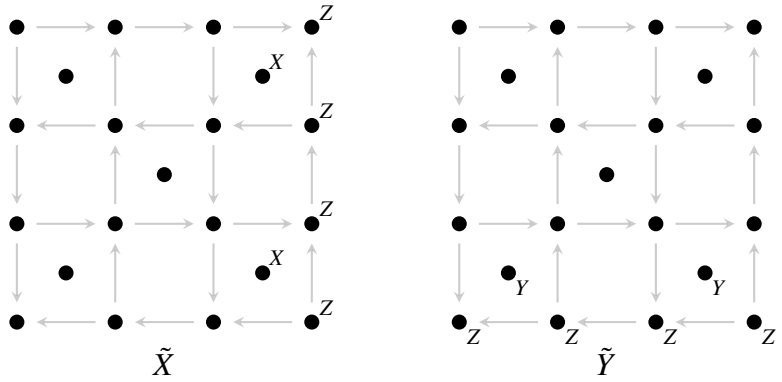


Figure 2.5: The encoded X and Y operators on the extra logical qubit in case (b). Here the X is formed by fusing a particle of species B with D and the Y by fusing C with D.

2.3 Connection to the Toric Code

The Pauli operators on the encoded qubit in case (b) of a lattice with an odd number of faces are string-like and span the length of the system. This structure is reminiscent of the *surface code* [60], an open boundary version of the toric code discussed in section 1.4.2, suggesting a connection between the error correcting code and this fermionic encoding. With this in mind, one quickly notices that the auxiliary face

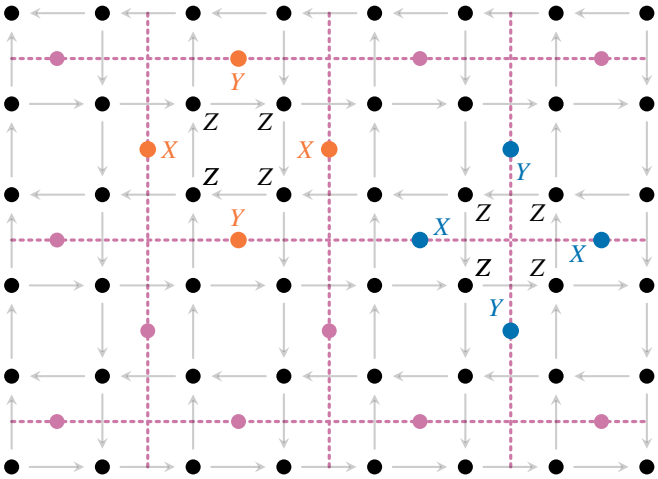


Figure 2.6: The toric code (dotted purple) embedded in the compact encoding. Each stabilizer is a tensor product of either a plaquette Π_p (red) or star Π_s (blue) operator, with a four qubit Z parity operator (black)

qubits are arranged exactly like the qubits of the toric code on a square lattice and that the non-trivial face loop stabilizers are simply the star (Π_S) and plaquette (Π_P) operators of the toric code (up to some local qubit rotations) tensored together with Z operators on 4 vertex qubits. See fig. 2.6 for illustration.

In the bulk, the codespace of the Compact Encoding is the degenerate ground state of the Hamiltonian

$$H_{CE} = -\sum \Pi_S \otimes Z_S - \sum \Pi_P \otimes Z_P \tag{2.14}$$

where Z_S and Z_P are the products of Z operators on vertex qubits surrounded by a star of plaquette. Similarly to the toric code Hamiltonian in eq. (1.44), H_{CE} admits e and m quasiparticles on the toric code lattice with an energy cost. However, when paired with the appropriate operators on the vertex qubits, the pair creation operators of the composite ε particles commute with H_{CE} allowing them to exist in the ground space with no energy penalty. In fact, a string of edge operators is exactly this as shown in fig. 2.7. In this sense, the Compact Encoding can be seen as leveraging the fermionic exchange statistics of the toric code ε particles to obtain the required anticommutation relations between edge operators. Similar connections exist in other fermionic encodings [13, 12] which will be discussed in greater detail in chapter 6.

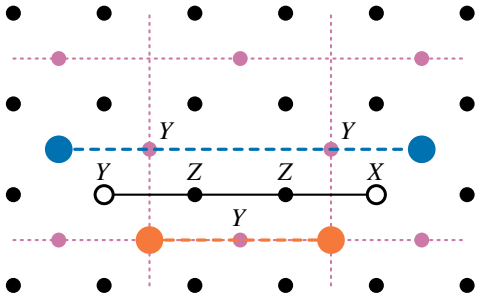


Figure 2.7: A string of edge operators (black) in the compact encoding corresponds to localized pairs of e (red) and m (blue) particles in the toric code, i.e. a pair of ε particles.

2.4 Discussion

The Compact Encoding constitutes a significant improvement on both the mode to qubit ratio and the Pauli weights of encoded local fermionic operators. This will give rise to low weight terms in the Fermi-Hubbard model and any fermionic Hamiltonian which is local on a square lattice graphs and whose terms are products of small numbers of vertex and edge operators. The author conjectures that no fermion to qubit encodings exist which have a smaller upper bound on the Pauli weights of vertex and edge operators.

Since the encoding has low weight edge and vertex operators, it is necessarily a low distance code. For example a Z error on any vertex qubit corresponds to a logical vertex operation. Naturally this means that it does not perform well at error correction or even detection but in chapter 4 it will be argued that for certain purposes this may not pose as serious a problem as one might expect.

From the perspective of implementations one downside to this encoding is that it is not clear that the low weight property can be preserved when restricting to hardware with a planar interaction graph and including all spins. This is not something widely considered in most other works, besides [17], and so it may be that this encoding still performs comparatively well under this restriction.

In [61] Clinton et al propose a novel pulse based scheme for simulating time evolution of quantum systems. One of the primary bottlenecks for the performance of this scheme is the Pauli weight of the encoded Hamiltonian terms. By employing

the Compact Encoding, simulations using such a pulse based scheme are expected to see an order of magnitude improvement in circuit depth over Verstraete-Cirac (see Table 1 of [61]), which has maximum weight 4 terms.

Chapter 3

The Compact Encoding on Further Lattice Geometries

Having explored the Compact Encoding on a square lattice, this chapter will show how the same design can be applied to various other lattice geometries. In 2D the encoding may be applied to all uniform tilings with degree less than 4. In these cases the generators of the even fermionic algebra are maximum weight 3, and the qubit to mode ratios range from < 1.25 to < 1.67. Of particular interest is the 4.8.8 uniform tiling, in which a spinful fermionic system on a square lattice may be embedded, such as the Fermi-Hubbard model. Additionally the Compact Encoding may be applied to a cubic lattice. In this case the generators of the even fermionic algebra are maximum weight 4, and the qubit to mode ratio is less than 2.5, contrasting with other local encodings which have maximum weights of at least 4 and qubit to mode ratios of at least 3. As previewed in the odd face numbered cases of the square lattice encoding, a surprising emergent property of some of the encodings presented here is the disparity between the size of the code space and the fermionic space to be encoded. In the cases where the code space is larger, multiple species of Majorana-like operators emerge in the encoded system. Additional stabilizers may be defined to remove these species.

To facilitate the analysis of these encodings, this chapter begins with a general group theoretic explanation of fermionic encodings in section 3.1. The encoding on a square lattice will then be re-examined through this formalism in section 3.1.2.

Section 3.1.3 discusses the emergent particle species in fermionic encodings and gives a bound on their number. Section 3.2 defines a restricted family of compact encodings on planar graphs and proves some useful properties, and in section 3.3 this restricted family is applied to all uniform planar graphs with degree less than 4. Finally the encoding on a cubic lattice is presented in section 3.4.

The content of this chapter is largely based on the work in the preprint [62] by the author and Joel Klassen.

3.1 Local Fermionic Encodings on Graphs

The aim of any fermionic mapping is to represent fermionic operators by qubit operators. All such mappings are necessarily restricted to discrete sets of fermionic modes, since the qubit system is finite dimensional. An important feature of natural fermionic systems is parity superselection, which forbids observables that do not commute with fermion number parity. Thus, insofar as one is concerned with representing natural observables, it suffices for fermionic mappings to represent operators which preserve parity, even fermionic operators.

The even fermionic operators form a group algebra $\mathbb{C}[M_E]$ where M_E is the group of even products of Majorana operators, with factors ± 1 and $\pm \mathrm{i}$:

$$M_E = \left\{ \pm \mathrm{i}^{0/1} \prod_j \gamma_j^{b_{2j}} \overline{\gamma}_j^{b_{2j+1}} \mid b \text{ is an even parity bit string} \right\} \tag{3.1}$$
$$\gamma_j = c_j + c_j^\dagger, \ \overline{\gamma}_j = (c_j - c_j^\dagger)/\mathrm{i}.$$

Thus a fermionic encoding must at minimum constitute a group representation τ of M_E:

$$\tau : M_E \to L(\mathcal{H}) \tag{3.2}$$

where $L(\mathcal{H})$ denotes the set of linear operators on the Hilbert space $\mathcal{H}$ of a system of qubits.

The group M_E, and by extension the group algebra $\mathbb{C}[M_E]$, are generated by the "edge" E_{ij} and "vertex" V_j operators defined for every mode i and every ordered pair

of modes (i, j)

$$E_{ij} := -\mathrm{i}\gamma_i\gamma_j,\ V_j := -\mathrm{i}\gamma_i\overline{\gamma}_j, \tag{3.3}$$

and phases $\{\pm 1, \pm \mathrm{i}\}$, which satisfy the established relations

$$\begin{gathered} E_{ij}^2 = I, \quad V_j^2 = I, \quad E_{ij}^\dagger = E_{ij}, \quad V_j^\dagger = V_j, \quad E_{ij} = -E_{ji}, \\ i \neq j \neq k \neq l \neq m: \quad [V_i, V_j] = 0, \quad [E_{ij}, V_k] = 0, \quad [E_{ij}, E_{kl}] = 0, \\ \{E_{ij}, V_j\} = 0, \quad \{E_{ij}, E_{jk}\} = 0 \end{gathered} \tag{3.4}$$

and for any ordered set of modes $L = (i_1, i_2, \ldots, i_{|L|})$, with $i_1 = i_{|L|}$ so that L constitutes a cyclic path, the relation

$$\mathrm{i}^{|L|-1} \prod_{x=1}^{|L|-1} E_{i_x, i_{x+1}} = I. \tag{3.5}$$

These relations completely fix the group structure of M_E and thus the group representation σ is completely specified by qubit operators satisfying these relations. The edge and vertex operators are self-inverse, hermitian, and only mutually commute or anticommute. This makes multi-qubit Pauli operators natural candidates for their representation. Indeed every existing fermionic mapping represents edge and vertex operators as Pauli operators – in some cases projected onto a subspace to ensure that all conditions are met.

There are two notions of locality which a design of a local fermionic encoding may wish to pursue: algebraic locality, wherein the *number* of qubits an encoded operator acts upon is bounded by some (monotonically increasing) function of the number of modes the fermionic operator acts on; and geometric locality, wherein the *maximum distance* (for some distance measure) between qubits an operator acts upon is bounded in a similar fashion. The compact encoding, and generalizations presented here, aims to preserve the geometric locality of operators, while minimizing the algebraic locality on qubits. The distance measure on a fermionic system used here is represented by a connected graph, dubbed the interaction-graph, whose vertices

correspond to fermionic modes, with distance given by the minimal path between modes. In the encodings considered the qubit systems are embedded in real space, and so proximity in real space is used as the distance measure on the qubit system. To this end it suffices to encode the vertex and edge operators associated with the edges and vertices of the graph to local edge and vertex operators on qubits, all other operators may be decomposed into these edge and vertex operators in a way that inherits this locality.

In general it is not feasible to assign local Pauli operators to the edges and vertices of the fermionic graph which satisfy all of relations in eqs. (3.4) and (3.5). This is because Relation (3.5) has a highly non-local character. Instead the strategy is to assign local Pauli operators which satisfy relations (3.4) and project via the stabilizer formalism into the subspace which respects eq. (3.5). In this case the stabilizers are the products of closed loops of edges in the fermionic graph. In this way, one is constructing a representation of a group structure on the graph, defined by relations (3.4), and quotienting out the subgroup corresponding to the cycle space of the graph.

Formally, given an undirected, connected graph $G = (\mathbf{V} = \{v_i\}, \mathbf{E} = \{\{v_i, v_j\}\})$ define the finitely presented group M_G whose presentation comprises the vertices $\mathbf{V}$, the directed versions of the edges $\mathbf{E}_D := \{e_{ij} = (v_i, v_j), e_{ji} = (v_j, v_i) \mid \forall \{v_i, v_j\} \in \mathbf{E}\}$ and $\{\pm 1, \pm \mathrm{i}\}$, along with the relations

$$\begin{gathered} e_{ij}^2 = I, \quad v_j^2 = I, \quad e_{ij} = -e_{ji}, \\ i \neq j \neq k \neq l \neq m: \quad [v_i, v_j] = 0, \quad [e_{ij}, v_k] = 0, \quad [e_{ij}, e_{kl}] = 0, \\ \{e_{ij}, v_j\} = 0, \quad \{e_{ij}, e_{jk}\} = 0 \end{gathered} \tag{3.6}$$

In the notation of group presentations:

$$M_G := \langle \mathbf{V} \cup \mathbf{E}_D \cup \{\pm 1, \pm i\} | \text{Eqs. 3.6} \rangle \tag{3.7}$$

Then define the abelian normal subgroup of directed cycles $\mathcal{C}_G \triangleleft M_G$ (see ap-

pendix A.1 for details)

$$\mathcal{C}_G = \left\{ \mathrm{i}^{|L|-1} \prod_{x=1}^{|L|-1} e_{i_x,i_{x+1}} \mid L = (i_1, i_2, \ldots, i_{|L|}), i_1 = i_{|L|}, e_{i_x,i_{x+1}} \in \mathbf{E}_D \right\} \tag{3.8}$$

Note that $\mathcal{C}_G$ is isomorphic to the cycle space of G (see appendix A.1), and its elements are invariant under a choice of first and last element, or total orientation.

Appendix A.1 shows that given a fermionic system, and a corresponding connected fermionic graph G, the group of even Majorana operators M_E is isomorphic to the quotient group $M_G/\mathcal{C}_G$ via the invertible mapping

$$f : M_G/\mathcal{C}_G \to M_E. \tag{3.9}$$

Thus if one can identify a representation

$$\sigma : M_G \to L(\mathcal{H}) \tag{3.10}$$

such that all elements of $\sigma(\mathcal{C}_G)$ have a common $+1$ eigenspace $\mathcal{U} \subseteq \mathcal{H}$, then one can construct a representation τ of M_E by considering the projection of σ into $\mathcal{U}$ i.e.

$$\sigma_{\mathcal{U}} := \mathrm{Proj}_{\mathcal{U}} \circ \sigma. \tag{3.11}$$

Noting that $\mathcal{C}_G \subseteq \ker(\sigma_{\mathcal{U}})$ the action of τ may be defined

$$\begin{gathered} \tau(\mu) = (\tau \circ f^{-1})(m\mathcal{C}_G) = \sigma_{\mathcal{U}}(m), \\ \forall \mu \in M_E,\, m \in M_G \;\; : \;\; \mu = f(m\mathcal{C}_G) \end{gathered} \tag{3.12}$$

which constitutes a faithful representation of M_E. Choosing a representation σ which maps into the n qubit Pauli group

$$\mathcal{P}_n = \{\pm 1, \pm \mathrm{i}\} \times \{I, X, Y, Z\}^{\otimes n}, \tag{3.13}$$

then since $\mathcal{C}_G$ is abelian, and provided that $-I \notin \sigma(\mathcal{C}_G)$, $\mathcal{U}$ automatically exists [63]

and corresponds to a stabilizer code space of the stabilizer group $\mathcal{S} := \sigma(\mathcal{C}_G)$.

To summarize, given a connected fermionic graph G, corresponding to some fermionic system, to construct a local fermionic encoding it suffices to specify a local mapping σ of the edges and vertices of the graph to multi-qubit Pauli operators, satisfying the relations 3.6, such that no element of $\mathcal{C}_G$ is mapped to $-I$. An example is the Jordan Wigner encoding, where the fermionic graph is a line, there are no cycles and so $\mathcal{C}_G$ is trivial, and $\sigma(e_{ij}) = \sigma_{\mathcal{U}}(e_{ij}) = X_i Y_j$, $\sigma(v_i) = \sigma_{\mathcal{U}}(v_i) = Z_i$.

As in the previous section, a tilde denotes the representation of an operator, ie $\tilde{e}_{ij} := \sigma(e_{ij})$ and $\tilde{E}_{ij} := \tau(E_{ij})$. In cases where $\tilde{E}_{ij}$ or $\tilde{V}_{ij}$ is set as equal to a Pauli operator, it is implicit that this Pauli operator is projected into the subspace $\mathcal{U}$. Thus in the case where e_{ij} is defined for some edge (i, j), $\tilde{E}_{ij}$ and $\tilde{e}_{ij}$ may be used interchangeably (similarly for $\tilde{V}_i$ and $\tilde{v}_i$), as was the case in chapter 2. Nevertheless it is worth emphasising the conceptual difference between e_{ij}, v_i and E_{ij}, V_i. The former are elements of an abstract group M_G corresponding only to edges and vertices of a particular graph, while the latter are elements of the group of even Majorana monomials M_E, which has no particular graph structure.

3.1.1 Counting Stabilizers

Using the construction described in the previous section, one specifies a representation σ of M_G which prescribes how the even Majorana monomials M_E of a fermionic system are encoded into a stabilizer code space $\mathcal{U}$. However the group algebra generated by the logical operators that act on $\mathcal{U}$ – the operators which commute with the stabilizer group $\mathcal{S} := \sigma(\mathcal{C}_G)$ – may be larger than the group algebra $\mathbb{C}[M_E]$. In other words the codespace may encode more than just parity preserving fermionic states, there may be additional structure.

This additional structure will be indicated by the dimension of the code space. The dimension of a fermionic system with M modes is 2^M. Fixing parity reduces the dimension by half, ie 2^{M-1}. The dimension of the code space $\dim(\mathcal{U})$ will depend on the dimension of the original Hilbert space and the size of the minimal set of generators of S and may diverge from this value.

Recalling that for a group X, the minimum size of a set of generators is the rank

of X, denote this by $D(X)$. For an encoding employing N qubits, the dimension of the code space is $\dim(\mathcal{U}) = 2^{N-D(S)}$. When working with fermionic encodings it is most useful to consider how the degrees of freedom in the encoded space differ from the usual degrees of freedom of the fermionic space:

$$\dim(\mathcal{U}) = 2^{M+\Delta}, \tag{3.14}$$

where Δ is the *disparity*

$$\Delta := N - M - D(S). \tag{3.15}$$

When the disparity is -1 the code space encodes only the even fermionic states. When the disparity is 0 the code space encodes the full fermionic Hilbert space. When the disparity is positive, the code space encodes Δ additional qubit degrees of freedom.

Here and throughout an important subgroup of M_G is the group of all cycles that are mapped to the identity under σ, ie $\ker(\sigma|_{\mathcal{C}_G})$, where $\sigma|_{\mathcal{C}_G}$ is the restriction of the representation σ to the subgroup $\mathcal{C}_G$. For notational ease define

$$K := \ker(\sigma|_{\mathcal{C}_G}) \tag{3.16}$$

Theorem 1. *Given a fermionic encoding σ for a connected fermionic graph G, the rank of the stabilizer group S is $D(S) = D(\mathcal{C}_G) - D(K)$.*

Proof. S corresponds to a faithful representation of the quotient group $\mathcal{C}_G/K$, and so $|S| = |\mathcal{C}_G/K|$. Furthermore, by Lagrange's theorem

$$|\mathcal{C}_G/K| = |\mathcal{C}_G|/|K|.$$

Because $a^2 = I$, $\forall a \in \mathcal{C}_G$, the elements of $\mathcal{C}_G$ correspond to the elements of the vector space $\mathbb{Z}_2^{D(\mathcal{C}_G)}$ so that $|\mathcal{C}_G| = |\mathbb{Z}_2^{D(\mathcal{C}_G)}| = 2^{D(\mathcal{C}_G)}$. Similarly for S and K: $|S| = 2^{D(S)}$ and $|K| = 2^{D(K)}$. Thus

$$D(S) = D(\mathcal{C}_G) - D(K) \tag{3.17}$$

□

Corollary 2.

$$\Delta = (N - M) - (D(\mathcal{C}_G) - D(K)) \tag{3.18}$$

Proposition 3. $D(\mathcal{C}_G) = |\boldsymbol{E}| - |\boldsymbol{V}| + 1$

Proof. Recalling that $\mathcal{C}_G$ is isomorphic to the cycle space of G, $D(\mathcal{C}_G)$ is equal to the circuit rank of G, which satisfies:

$$D(\mathcal{C}_G) = |\mathbf{E}| - |\mathbf{V}| + \beta_0 \tag{3.19}$$

β_0 is the 0th Betti number, ie the number of connected components of the graph, in this case 1. □

3.1.2 The Compact Encoding on a Square Lattice

The Compact encoding introduced in chapter 2 can be understood through the concepts explained in the previous sections. This encoding uses the fermionic graph formalism from section 3.1, where the graph is a square lattice, with each vertex corresponding to a fermionic mode. It defines qubit representations of the edge and vertex operators from eq. (3.3) such that any local interaction term has a Pauli weight no greater than 3 and does this with a qubit to mode ratio of < 1.5.

The definitions of the edge and vertex operators given in eqs. (2.6) and (2.8) strictly only specify the action of the mapping $\sigma(e_{ij}) = \tilde{e}_{ij}$ and $\sigma(v_i) = \tilde{v}_i$ as they only satisfy the relations in eq. (3.6). The definitions of $\tilde{E}_{ij}$ and $\tilde{V}_i$ are obtained by projecting into the $+1$ eigenspace of $\sigma\mathcal{C}_G$ on the square lattice graph.

The cycle group $\mathcal{C}_G$ of a planar graph is minimally generated by the cycles around faces. However the cycles around odd faces are mapped to the identity under σ. Thus K is minimally generated by the cycles around odd faces, and so

$$D(K) = OF \tag{3.20}$$

where OF denotes the number of odd faces on the lattice. The stabilizer group is then generated by the even face cycles (see Theorem 8 in Section 3.2).

From eqs. (3.18) and (3.20) and the fact that $N - M = OF$ and $D(\mathcal{C}_G) = EF + OF$, where EF is the number of even faces, one has

$$\Delta = OF - EF \tag{3.21}$$

for the encoding on a square lattice.

3.1.3 Particle Species on Fermionic Encodings

Different values of the disparity Δ cause an encoding to have different properties. In cases where $\Delta > 0$ one can define *distinct particle species*. These find use in defining stabilizers to restrict excess space in these encodings and to detect errors. To illustrate this we will first recall the effects of different disparities on the square lattice encoding and then present a result on particle species for general values of Δ.

On a square lattice with a checkerboard pattern there are either equal numbers of even and odd faces or one more even/odd face so the disparity Δ may take the values -1, 0, 1. In the $\Delta = -1$ case the codespace has dimension 2^{M-1}, the even fermionic operators on M modes provide a faithful representation of the set of linear operators on a space of this size so no other operators may be defined. The fact that only even fermionic operators are represented implies that this is a fixed fermion parity sector. The operator $\prod_j V_j$ is equal to identity up to stabilizers in this case, identifying the parity sector as even. The odd parity sector may also be simulated by flipping the sign of one vertex operator (or indeed any odd number).

In the $\Delta = 0$ case the codespace has dimension 2^M. The even fermionic operators do not fully represent all linear operators over this space, the representation is only complete with the inclusion of odd (parity violating) fermionic operators. It suffices to define a representation of a single Majorana operator, since all other subsequent odd operators may be generated by applying edge and vertex operators. A single

Majorana operator γ_j satisfies the relations

$$\begin{gathered}\{\gamma_i,\gamma_j\}=0\,,\quad \{V_i,\gamma_i\}=0\,,\quad [V_i,\gamma_j]=0\quad j\neq i\\ \{E_{ij},\gamma_i\}=0\,,\quad [E_{ij},\gamma_k]=0\quad k\neq j\quad k\neq i\\ \gamma_i^2=I\,,\quad E_{ij}\gamma_j=\gamma_i\,,\quad V_i\gamma_i=\mathrm{i}\overline{\gamma}_i.\end{gathered}\tag{3.22}$$

Consider the corners of the lattice associated with an odd face. Such a corner j either has arrows pointing into it or pointing away from it. If the arrows point into the corner then define the encoded Majorana operator $\tilde{\gamma}_j := X_j$, otherwise define it to be $\tilde{\gamma}_j := Y_j$ (the encoded Majorana of the second kind, $\tilde{\overline{\gamma}}_j$, is given, up to a phase, by multiplying by a vertex operator). These single Majorana operators may be moved around the lattice by multiplication with strings of edge operators as shown in fig. 2.3. From this it is clear that encoded single Majorana operators $\{\tilde{\gamma}_i,\tilde{\overline{\gamma}}_i\}$ take the form of strings of Paulis anchored at one end to their corner of origin. Denote this corner the *injection point* of the Majorana operators. On the $\Delta=0$ square lattice, there are two equally suitable injection points from which single Majorana operators may be defined. If one is chosen to be the injection point for single Majorana operators then the single particle operators injected at the other corner will correspond to encoded Majorana hole operators $\{\tilde{h}_i,\tilde{\overline{h}}_i\}$, where $h_i := \gamma_i\prod_j V_j$ and $\overline{h}_i := \overline{\gamma}_i\prod_j V_j$.

The encoded single Majorana operators $\{\tilde{\gamma}_i,\tilde{\overline{\gamma}}_i\}$ and $\{\tilde{h}_i,\tilde{\overline{h}}_i\}$ are examples of *distinct particle species*.

Definition 4 (Particle Species)**.** *Given a fermionic encoding, a set of Pauli operators $\mathcal{M}=\{\mathcal{M}_i,\overline{\mathcal{M}_i}\}$, indexed over vertices is called a Particle Species if it satisfies the algebraic relations of single Majorana operators with respect to the encoded fermionic operators, i.e.*

$$\begin{gathered}\{\mathcal{M}_i,\mathcal{M}_j\}=0\,,\quad \{\tilde{V}_i,\mathcal{M}_i\}=0\,,\quad [\tilde{V}_i,\mathcal{M}_j]=0\quad j\neq i\\ \{\tilde{E}_{ij},\mathcal{M}_i\}=0\,,\quad [\tilde{E}_{ij},\mathcal{M}_k]=0\quad k\neq j\quad k\neq i\\ \mathcal{M}_i^2=I\,,\quad \tilde{E}_{ij}\mathcal{M}_j=\mathcal{M}_i\,,\quad \tilde{V}_i\mathcal{M}_i=\mathrm{i}\overline{\mathcal{M}_i},\end{gathered}\tag{3.23}$$

with equalities up to stabilizers where relevant.

Definition 5 (Distinct Particle Species)**.** *Particle Species $\mathcal{M}$ and $\mathcal{M}'$ are said to be distinct if*

$$\{\mathcal{M}_i, \mathcal{M}'_i\} = 0\,, \quad [\mathcal{M}_i, \mathcal{M}'_j] = 0\,, \quad j \neq i. \tag{3.24}$$

In the $\Delta = 1$ case the codespace is larger than the fermionic system with dimension 2^{M+1}, this is interpreted as simulating the full fermionic system and an extra logical qubit degree of freedom, i.e. $\mathcal{F} \otimes \mathbb{C}^2$. As with $\Delta = 0$, single particle operators may be "injected" at the odd corners but instead there are four choices, meaning that four distinct particle species may be simultaneously defined. One may always choose one of these species to be the encoded single Majorana operators of the fermionic system $\{\tilde{\gamma}_i, \tilde{\bar{\gamma}}_i\}$. The other three Majorana species correspond to $h_i \otimes X$, $h_i \otimes Y$, $h_i \otimes Z$ where the Paulis act on the extra logical qubit degree of freedom. The Paulis may be assigned to species arbitrarily so long as they form an anticommuting set. These Pauli operators can be isolated by fusing pairs of these Majorana species on the same vertex such that the fermionic part cancels to identity. As shown in fig. 2.5, the Paulis have a non-local string like form across the lattice. Having isolated these non-local Pauli operators, one of them may be taken as a stabilizer such that only the fermionic space is encoded.

It may be desirable to restrict this excess Hilbert space via this stabilizer as it reduces the number of logical operators that act nontrivially on the fermionic space and therefore reduces the number of possible errors. In particular, restricting via one of these stabilizers on the $\Delta = 1$ square lattice means that single X and Y errors may only occur on two of the corners rather than 4.

Other encodings introduced in this chapter have $\Delta > 1$. This greater disparity precipitates more particle species.

Theorem 6. *For any fermionic encoding, the number m of distinct Majorana species is bounded from above by*

$$m \leq 2\Delta + 2, \tag{3.25}$$

Proof. The earlier arguments for the $\Delta = -1, 0$ cases on the square lattice apply generally and the number of particles in each case is consistent with eq. (3.25).

Consider the $\Delta \geq 1$ case, in which the encoding represents a Hilbert space of dimension $2^{M+\Delta}$ which may be taken to be a fermionic system of M modes composed with Δ qubits, $\mathcal{F} \otimes (\mathbb{C}^2)^{\otimes \Delta}$.

Consider a set of m different simultaneously defined species $\{\mathcal{M}_i^{(k)}\}_{k=1}^m$. Any one of these may be chosen to be the encoded single Majoranas, choose $\mathcal{M}^{(1)} = \{\tilde{\gamma}_i, \tilde{\bar{\gamma}}_i\}$. For any given vertex, consider the set of operators

$$\{P_k^i\}_{k=2}^m = \{\mathrm{i}\, \mathcal{M}_i^{(1)} \mathcal{M}_i^{(k)} \prod_j \tilde{V}_j\}_{k=2}^m = \{-\mathrm{i}\, \tilde{h}_i\, \mathcal{M}_i^{(k)}\}_{k=2}^m. \tag{3.26}$$

Note that for any two vertices i and j, P_k^i and P_k^j are related by a loop of edge operators and are therefore in fact the same operator, accordingly the vertex index is now redundant and will be dropped.

By the relations in definitions 4 and 5, each operator P_k commutes with every fermionic operator, including our chosen single Majoranas $\mathcal{M}^{(1)}$ and so each element only acts non-trivially on the excess $(\mathbb{C}^2)^{\otimes \Delta}$ space. All P_k are Pauli operators as they are the product of Pauli operators (all single particle operators must be Paulis otherwise they could not be mapped to Majorana operators). Finally all P_k are mutually anticommuting.

Combining the above properties we see that $\{P_k\}_{k=2}^m$ is a set of mutually anti-commuting Paulis acting only on the $(\mathbb{C}^2)^{\otimes \Delta}$ with an element for each defined single particle species except $\mathcal{M}^{(1)}$. The maximum size of such a set is $2\Delta + 1$ by Lemma 4.5 from [64] and by Corollary 4.6 from the same paper, a set of this size exists so there can therefore be a maximum of $2\Delta + 1$ P_k operators and the maximum number of single particle species is $2\Delta + 2$.

□

From the above proof one can see that particle species not chosen to be encoded Majoranas are in fact the same Majoranas/holes but dressed with Pauli operators on the excess space which can be isolated by fusion of differently dressed species on the same vertex. Thus in the case where a maximal set of these particle species can be identified they may be used to define stabilizers which restrict the code space to

one with $\Delta = 0$.

3.2 Generalizing the Compact Encoding in 2D

The essential form of the compact encoding is as follows. A graph is supplied, and every edge of the graph is assigned an orientation. A vertex qubit is associated with every vertex of the graph. The vertex operators are defined to be Z operators on their associated vertex qubits, and the edge operators are tentatively defined to act with an X operator on the vertex which the edge is pointing away from, and a Y operator on the vertex which the edge is pointing towards. Finally, auxiliary qubits are introduced, and the edge operators are made to act additionally on these auxiliary qubits in order to resolve any instances where pairs of edge operators sharing a vertex do not yet commute. The essential feature which makes the compact encoding compact is that the vertex operators are weight 1, and the tentative edge operators are weight 2, with an increase in weight bounded by the number of remaining anti-commutation relations needing to be resolved. An important additional feature which appears in the particular implementation of this encoding procedure is that the auxiliary qubits may be used to resolve many anti-commutation relations, thus significantly reducing the number of auxiliary qubits required.

In the square lattice encoding, the auxiliary qubits are confined to faces adjacent to the edges, and edges are only permitted to act on those adjacent auxiliary qubits. However in principle there is no reason why this must be the case. auxiliary qubits could be used to resolve anti-commutation relations between edges not sharing a face. Furthermore, in the square lattice encoding, each face is associated with at most one auxiliary qubit. This is also not essential. However, one valuable consequence of imposing these kinds of constraints is that it makes the design and analysis of the encoding simpler. This motivates introducing a particular subclass of the compact encoding which lends itself well to analysis, and to which the square case belongs. This subclass aims to preserve the geometric locality and extremely low Pauli weight of the edge and vertex operators.

Definition 7 (Weight 3 Planar Encoding)**.** *A fermionic encoding on a planar*

fermionic graph where:

- *Every vertex has one vertex qubit assigned;*
- *Every auxiliary qubit is associated with a unique face;*
- *Every face is associated with at most one auxiliary qubit;*
- *Every vertex operator is a single Pauli Z on the assigned vertex qubit;*
- *Every edge operator is composed of a Pauli X or Y on each incident vertex qubit and a Pauli X, Y or Z on at most one face qubit from an adjacent face.*

The following theorem allows the disparity of codes that fit definition 7 to be easily determined by counting faces. Note here that for conceptual simplicity the "unbounded" face surrounding the graph is excluded when considering faces, but it may be included with minor modifications.

Theorem 8. *Given a Weight 3 planar encoding, $K = ker(\sigma|_{\mathcal{C}_G})$ is minimally generated by face cycles.*

Proof. The full set of face cycles on a planar graph form an independent basis of the cycle space so any subset of these will also be independent. It suffices now to prove that any element of K can be reduced to identity by application of face cycles in the kernel.

For a planar graph, every cycle has a unique decomposition into a product of face cycles. It is first necessary to show that for a cycle in K, at least one of the face cycles in its unique decomposition must be in K. Consider a cycle $c \in K$, for any vertex in the cycle, the edge operators in $\sigma(c)$ must cancel to identity on the corresponding vertex qubit. Let one of the edge operators act on this qubit with X, label this operator $\tilde{E}_1$. Since edge operators may not apply Z to vertex qubits the only way to cancel this is with another edge operator that applies X, label this $\tilde{E}_2$. These edge operators must anticommute so they must share a face and each act on the qubit with different Paulis (X and Y). The product E_1E_2 then applies a Z to this face qubit which can only be cancelled by other edge operators around the associated face.

The Z may be cancelled by a single edge operator acting with a Z or two acting with X and Y. Let E_Z be an edge operator applying Z to the face qubit. E_Z must anticommute with E_1 and E_2 so it must share vertex qubits with both, forming a 3-edge loop (to commute with either of them it would also need to share vertex qubits but this would require it to anticommute anyway), furthermore it would need to act with the same Paulis as E_1 and E_2 on their respective shared vertex qubits. The product $\mathrm{i}^3 E_1 E_2 E_Z$ (with the appropriate orientation) must be identity, meaning the associated face cycle is in K (see fig. 3.1).

For the other case, let E_X and E_Y be edge operators around the face that act with X and Y on the face qubit respectively. For similar reasons to above, E_Y (E_X) must anticommute with E_1 (E_2) meaning they will share a vertex qubit, as well as that, E_X and E_Y must anticommute and share a vertex themselves with the edges forming a 4-edged loop. As above, the edges that share vertex qubits will act on them with the same Paulis and the product $\mathrm{i}^4 E_1 E_2 E_X E_Z$ (with appropriate orientation) will be identity and the associated face cycle is in K (see fig. 3.1 for illustration).

Consider a cycle $c \in K$, which admits a decomposition into a set of face cycles $F \not\subseteq K$

$$c = \prod_{f \in F} f. \tag{3.27}$$

One can construct a new cycle c' by removing all cycles in $F \cap K$

$$c' = \left(\prod_{f \in F \cap K} f \right) c \tag{3.28}$$

However c' must be in K, which is a contradiction since it includes no face cycles in K.

□

The fact that face qubits may only be cancelled to identity in two ways yields the following corollary.

Corollary 9. *The face cycles in K of a Weight 3 planar encoding may only be around 3 sided or 4 sided faces.*

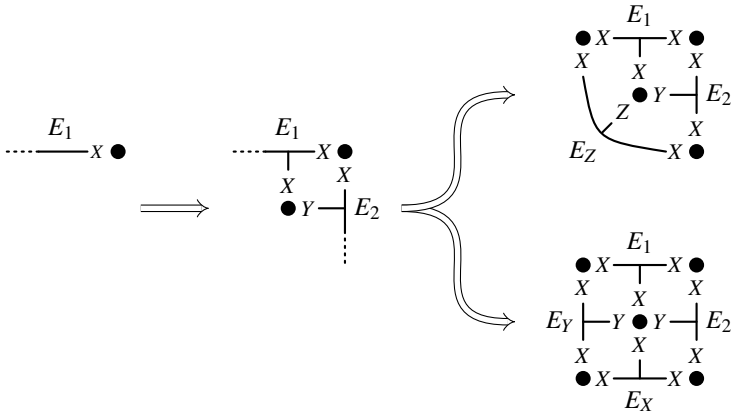


Figure 3.1: Graphical representation of the proof of theorem 8.

Corollary 10. *The disparity of a Weight-3 Planar Encoding is given by:*

$$\Delta = F_K - EF, \tag{3.29}$$

F_K denoting the number of face cycles in K, and EF denoting the faces without a qubit.

Proof. Recalling that $\Delta = N - M - (D(\mathcal{C}_G) - D(K))$ we note that $N - M$ is the number of faces with a qubit, and $D(\mathcal{C}_G) - D(K) = F - F_K$ where F is the number of faces. Thus $D(\mathcal{C}_G) - D(K) = EF + N - M - F_K$, and so $\Delta = F_K - EF$. □

3.3 Examples of Weight-3 Planar Encodings

This section presents weight 3 planar encodings for every possible uniform tiling of degree ≤ 4, except for the square tiling which has already been thoroughly investigated. Listed with each encoding is the qubit to mode ratio and how their disparity Δ scales with lattice size and shape.

3.3.1 The Hexagonal Lattice (6.6.6 Uniform Tiling)

For the hexagonal lattice, orient every edge except for the bottom edge of every face so that they circulate clockwise on even columns of faces and counterclockwise on odd columns, as illustrated in fig. 3.2. This ensures that heads touch tails for all edges except for the bottom edge of every hexagon. Add an auxiliary qubit for every face. As in the square lattice case, encoded vertex operators are Z operators on the

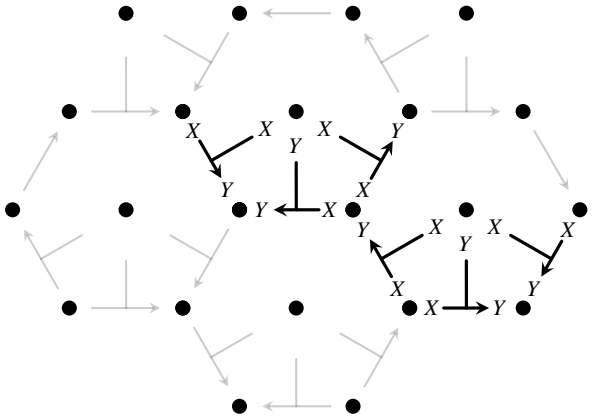


Figure 3.2: Edge orientation, qubit placement and edge operators for the hexagonal lattice encoding.

corresponding vertex qubit. Encoded edge operators are as illustrated in fig. 3.2 with the sign orientation chosen arbitrarily for each edge. Note that along the top of the lattice, "hanging" auxiliary qubits must be included to maintain anticommutation relations between edges. The qubit to mode ratio is ≤ 1.5.

The stabilizer $\mathcal{S}$ is generated by the loops of encoded edge operators around the hexagonal faces. As none of these are identity, the kernel must be trivial by theorem 8 and the stabilizer rank is the number of faces. As every face has an auxiliary qubit associated, the disparity Δ is simply the number of extra qubits along the top of the lattice which, depending on the exact lattice, is between $C-1$ and $C-2$ where C is the number of columns.

Different species of Majorana may be injected into the code as X or Y operators on vertices along the bottom of the lattice from which edges are either uniformly pointing towards or away. As well as this, in the case where a hanging auxiliary qubit along the top is used by only 2 edge operators, a single particle operator can be injected as a Z on the hanging qubit, this operator corresponds to a single particle vertex shared by the two edge operators (see fig. 3.3 for illustration).

3.3.2 Diagram Notation

The encodings for the remaining lattice structures are more consistent in their structure so the following shorthand is used to simplify their illustration. An edge incidence on a vertex will either be an arrow or a blank line. As has been standard

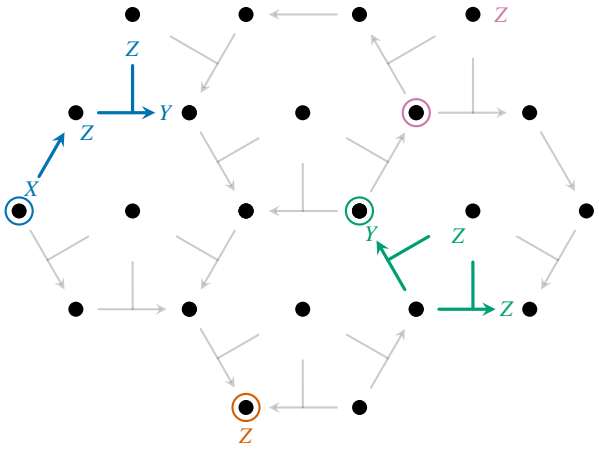
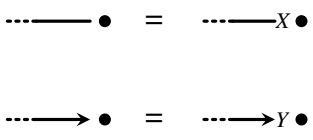


Figure 3.3: The encoding on the above hexagonal lattice has 2 extra qubits and so its disparity is $\Delta = 2$. Distinct particle species can be injected at the corners with only 2 edges pointing to/from them and at the hanging auxiliary qubits shared by only 2 edges. Orange: A single particle operator at an injection site along the bottom of the lattice, green: a single particle operator injected at a corner injection site and transported by edge operators into the lattice, purple: a single particle operator injected via an operator on a hanging qubit, note that this operator corresponds to a particle operator acting on the circled vertex in the fermionic system, blue: a single particle operator injected at via a hanging qubit and transported by edge operators into the lattice.

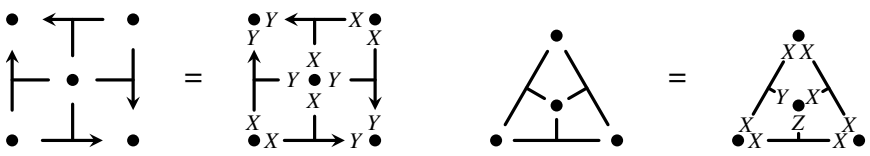
thus far, an arrow denotes that the corresponding edge operator will act on that qubit with a Y and a blank line denotes an X.



(3.30)

With these choices of Pauli acting on the vertex qubits, the vertex operators V_j on these codes are represented by Z on the corresponding qubit as in previous cases.

All the encodings in the following sections have odd faces (faces with a qubit assigned) of only two forms. The following shorthands denote how their surrounding edge operators act on the face qubit



(3.31)

Edge incidences may be switched arbitrarily provided that incidences on the same vertex qubit commute and anticommute in the same manner. The Paulis acting on face qubits may also be changed provided that opposite edges act with the same Pauli in the square case and that all three are different in the triangular case.

3.3.3 The 4.8.8 Uniform Tiling

The 4.8.8 uniform tiling and the associated encoding are illustrated in Figure 3.4. This tiling has the notable property that it may be readily used to represent a spinful fermionic system on a square lattice, as illustrated in Figure 3.5. Using the compact encoding on a 4.8.8 uniform tiling, a spinful fermi-hubbard model on a square lattice may be represented on a planar hardware interaction architecture, with weight-2 spin-spin interactions, and weight-4 hopping terms.

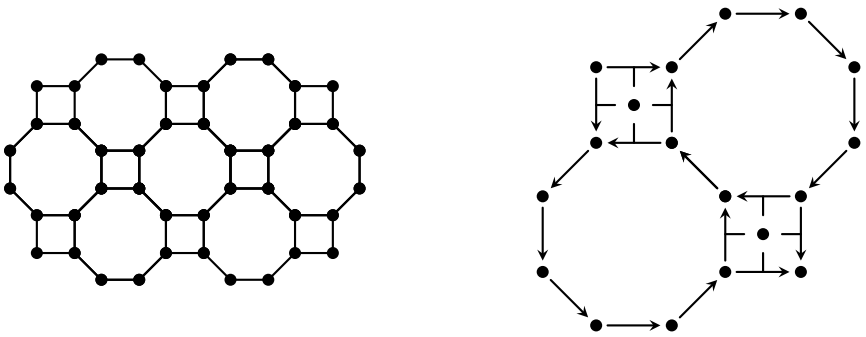


Figure 3.4: The 4.8.8 Uniform Tiling and the unit cell of its encoding.

Loops around octagonal faces are non-trivial Paulis and generate the stabilizer, loops around square faces are identity. The qubit to mode ratio is < 1.25.

This tiling pattern has the same even/odd face pattern as the square lattice with octagons in place of the even square faces and so the disparity follows the same rules.

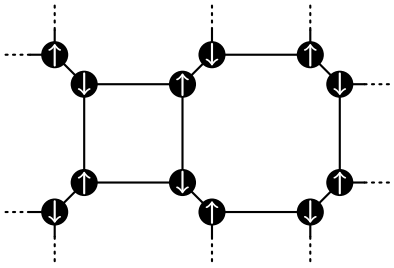


Figure 3.5: Layout of a spinful fermionic system on a square lattice, embedded in the 4.8.8 uniform tiling.

That is, for a rectangular shaped lattice such as the one shown in fig. 3.4 the disparity is given by

$$\Delta = OF - EF \tag{3.32}$$

and may take values of -1, 0 or 1.

3.3.4 The 6.4.3.4 Uniform Tiling

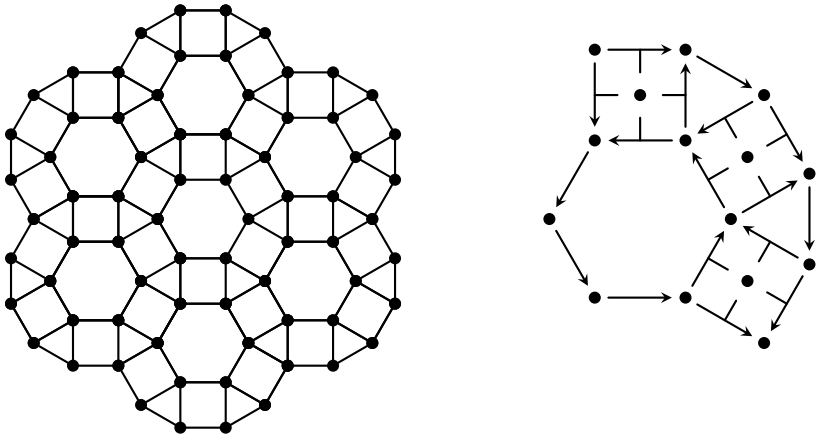


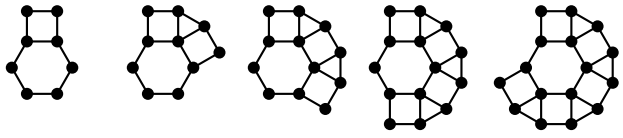
Figure 3.6: The 6.4.3.4 Uniform Tiling and the unit cell of its encoding.

See fig. 3.6 for the lattice structure and the unit cell of the encoding. Loops around triangular and hexagonal faces are non-trivial Paulis and generate the stabilizer, loops around square faces are identity. The qubit to mode ratio is < 1.5.

Proposition 11. *For a connected 6.4.3.4 lattice without holes and with all hexagonal faces fully surrounded by square and triangular faces (e.g. fig. 3.6), the disparity is*

$$\Delta = -1.$$

Proof. Consider a 6.4.3.4 lattice with a single fully surrounded hexagonal face. Clearly the disparity is -1 in this case as there are seven even faces and six odd faces. Consider constructing a lattice by adding fully surrounded hexagonal faces. Each hexagonal face added will amount to adding one of the following combinations of faces:



These combinations all have the same number of even and odd faces and will not change the disparity from -1. □

The lattice can be made to simulate the full fermionic algebra by adding a single square face to the outer edge where single particle operators may be injected at its corners.

3.3.5 The 4.6.12 Uniform Tiling

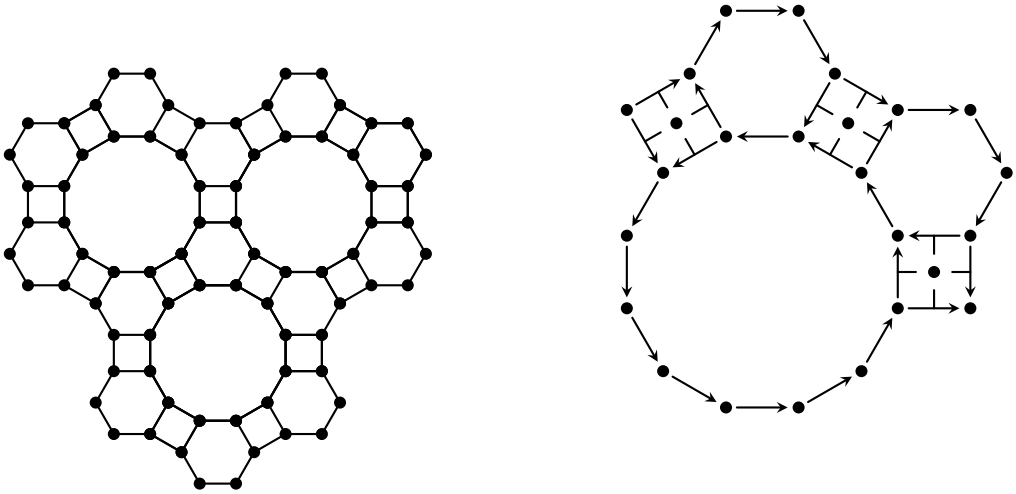


Figure 3.7: The 4.6.12 Uniform Tiling and the unit cell of its encoding.

See fig. 3.7 for the lattice structure and the unit cell of the encoding. Loops around hexagonal and dodecagonal faces are non-trivial Paulis and generate the stabilizer, loops around square faces are identity. The qubit to mode ratio is < 1.25.

This tiling has the same odd/even face pattern as the 6.4.3.4 tiling and so its disparity follows a similar rule in that a 4.6.12 lattice with no holes and fully surrounded dodecagonal faces will have disparity

$$\Delta = -1.$$

3.3.6 The Kagome Lattice (3.6.3.6 Uniform Tiling)

The Weight-3 Planar encodings shown so far do not have a disparity which grows with the lattice bulk, this is because the unit cells have as many faces in K as they do faces with no auxiliary qubits. However the Weight-3 planar encoding for the

Kagome lattice does not have this property.

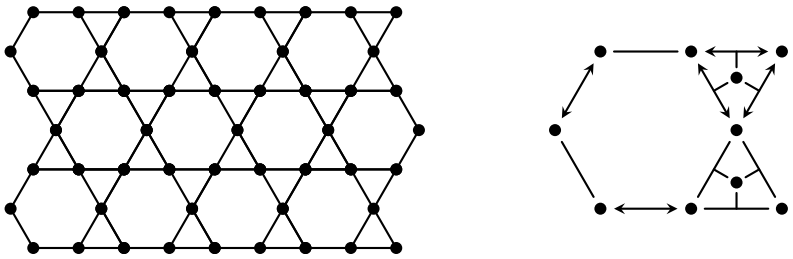


Figure 3.8: (Left) A Kagome lattice with two triangular corners. (Right) The unit cell of the encoding showing all possible edge operators and faces.

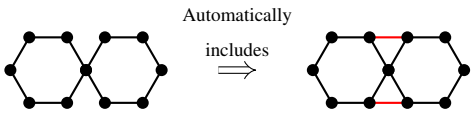
See fig. 3.8 for the lattice structure and the unit cell of the encoding. Loops around hexagonal faces are non-trivial Paulis and generate the stabilizer, loops around triangular faces are identity. The qubit to mode ratio is < 1.67.

Proposition 12. *The disparity of the encoding for a connected Kagome lattice of any shape without holes is given by*

$$\Delta = HF + TC - 2 = \frac{1}{2}(TF + TC - 2) \tag{3.33}$$

where HF is the number of hexagonal faces, TF is the number of triangular faces and TC is the number of triangular corners, that is, the number of vertices on the lattice boundary which belong only to a triangular face.

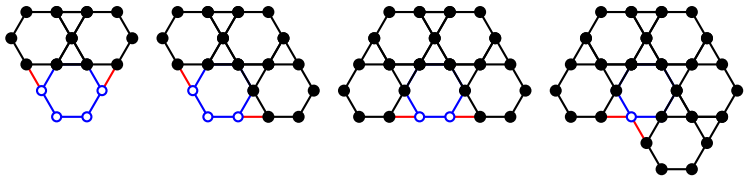
Proof. Consider a triangular face, the product of any two encoded edge operators around the face gives the third one up to some phase, therefore if an encoding on a lattice includes two edge operators around such a face (e.g. on the lattice boundary), then the third is included in the lattice automatically (illustrated below).



From this we can see that for a lattice of any shape, the triangular faces between hexagonal faces are automatically included in the encoding.

Now consider adding a Hexagonal face to the boundary of an existing lattice with no triangular corners. No matter where it is added, it will create two “slots”

between itself and other hexagonal faces which induce a triangular face as shown above. All contexts in which a Hexagonal face can be added are shown below to illustrate this (the fifth is captured in the previous diagram), the extra vertices and edges forming the new hexagon are shown in blue and the third edge of the induced triangle face is shown in red.



Any Kagome lattice without triangular corners may be constructed this way, beginning with a single hexagonal face and adding further hexagons one at a time.

From Corollary 10 the disparity of this encoding is

$$\Delta = TF - HF \tag{3.34}$$

where TF is the number of triangular faces and HF is the number of hexagonal faces. A single hexagonal face has $\Delta = -1$. A hexagonal face added to it will change the Δ by +1 as it will bring two triangular faces with it as shown earlier. From this it is clear that $\Delta = HF - 2$ for a lattice with no triangular corners.

A lattice with triangular corners may be constructed by simply adding triangular faces to the boundary of a lattice with no such corners. Each added triangular corner will increase Δ by 1 so the general formula is therefore $\Delta = HF + TC - 2$.

For the equivalent expression in terms of TF rather than HF consider the lattice with no triangular corners once more. When constructed as before it is clear that every Hexagonal face added also adds two triangular faces except for the first one so in this case:

$$TF = 2HF - 2. \tag{3.35}$$

If triangular corners are added to such a lattice then they will each necessarily add another triangular face so in general the relationship is

$$TF = 2HF + TC - 2. \tag{3.36}$$

Solving for HF and substituting into the first expression in eq. (3.33) yields the second. □

From the above one can see that the disparity of the encoding on a Kagome lattice actually grows with the lattice size and so, by theorem 6, does the number of distinct single particle species one can simultaneously define. As discussed in section 3.1.3 the appropriate fusions of these species can produce any Pauli operator on the extra $(\mathbb{C}^2)^{\otimes\Delta}$ space attached to the encoding. To reduce errors, this space can be restricted to a single state by finding a set of commuting Paulis to act as a stabilizer, all that remains is to find a maximal set of particle species which can be fused to form stabilizer generators.

As with the square lattice encoding, vertices on which edge operators act with only one type of Pauli operator can serve as injection points for distinct species. These are only found on triangular corners on the lattice boundary and some lattices may not even have these so there must be other ways to define single particle species.

These other injection points take a more involved form. Consider a triangular face, the three edge operators around a triangular face will act on their vertex qubits with Pauli, X and each of them will act on the face qubit with a different Pauli (see (3.31)). This argument also applies to triangular faces where edge operators act with Y on vertex qubits, simply substitute X with Y where appropriate. Label these edge operators E_X, E_Y and E_Z according to their face qubit support. Also label the vertex qubits q_X, q_Y and q_Z according to the edge operator opposite, so q_X is opposite E_X etc. Now define an operator which acts with an X on q_Z and with a Z on the face qubit. This operator will anticommute with the vertex operator on q_Z, all edge operators incident on q_Z and will commute with all other edge and vertex operators. By definition 4 this is a member of a single particle species. Consider a single operator defined similarly on q_X (X on q_X and X on the face qubit), it is also a single particle operator but it anticommutes with the particle on q_Z, by definition 5 this makes them members of the same species. This is confirmed by seeing that the two are related by the edge operator E_Y. See fig. 3.9 for illustration.

From this one can see that, as well as triangular corners, distinct single particle

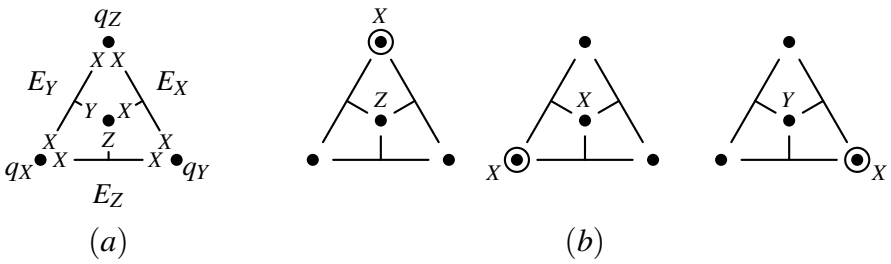


Figure 3.9: (a) Labelling of edge operators and qubits around a triangular face on the Kagome Lattice encoding. (b) Single particle operators on the circled vertices. These are all of the same species and may be transformed into each other by edge operators. On an equivalent face where the edge operators act on vertices with Y, the particle operators also act on vertices with Y.

species may be associated with any triangular face and injected at any of its vertices (including triangular faces that contain a triangular corner). Particles injected at different triangular faces and corners satisfy definition 5 as distinct species so a set of distinct species for every triangular face and triangular corner can be simultaneously defined. Substituting the second expression in eq. (3.33) into eq. (3.25) reveals that this set is maximal and so the species may be used to define any operator on the excess Hilbert space. See fig. 3.10 for examples of single particle operators on a Kagome lattice encoding.

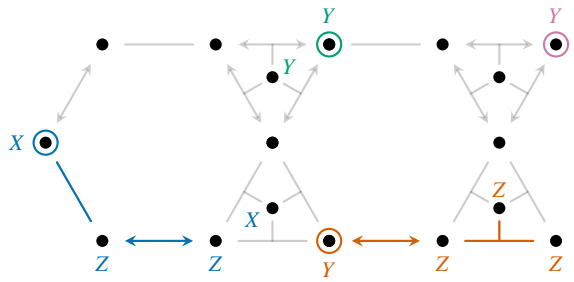


Figure 3.10: Single particle operators on a Kagome lattice at the circled vertices. Purple: A single particle operator at a triangular corner. Orange: A single particle operator transported from a triangular corner by edge operators. Green: A single particle operator defined at a triangular face as in fig. 3.9. Blue: A single particle operator transported from a triangular face via edge operators.

With a maximal set of single particle species, stabilizers may be defined to restrict the excess Hilbert space of the encoding. To do this, simply group 2Δ of the particle species into Δ disjoint pairs and fuse each pair on any vertex. The resulting Δ operators will commute and act trivially on the encoded fermions so they can be

added to the stabilizer. It is preferable to pair up particles with injection sites that are close to each other, that way the stabilizer formed by their fusion will have a low Pauli weight, making its measurement less costly (see fig. 3.11 for illustration). Pairing up 2Δ of the species will cut the Hilbert space such that the full fermionic Fock space is encoded. The remaining pair of particle species then represent the single Majorana operators and the hole operators. It may also be desirable to pair these up to form a stabilizer equivalent to the product of all vertex operators, fixing the parity sector of the encoded space.

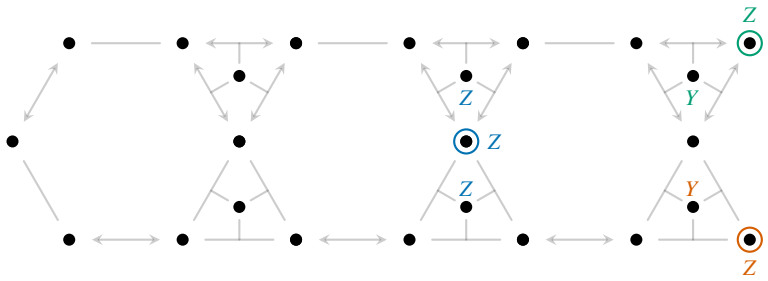


Figure 3.11: Possible stabilizers to restrict the excess Hilert space on a Kagome lattice encoding, formed by fusing particle species on the circled vertices. Blue: Fusion of two species injected at adjacent triangular faces. Green: Fusion of one species injected at a triangular corner and one injected at the same triangular face. Orange: Same as Green. The particle species injected at the remaining triangular faces are then the Majorana and hole operators.

3.3.7 The 3.12.12 Uniform Tiling

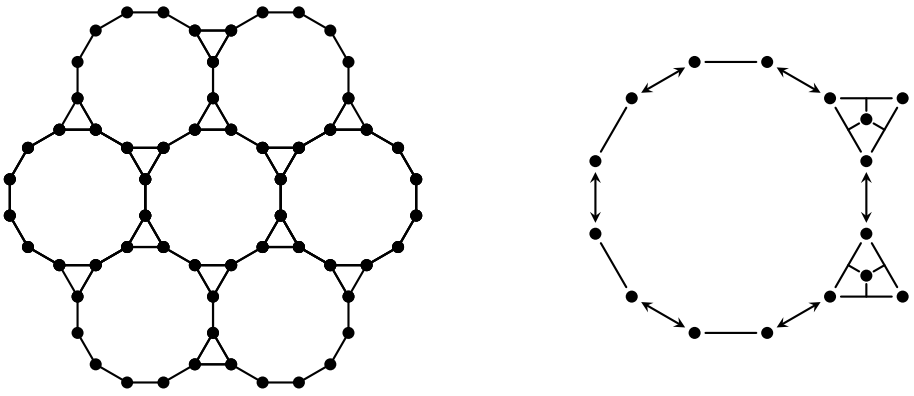


Figure 3.12: The 3.12.12 Uniform Tiling and the unit cell of its encoding.

See fig. 3.12 for the lattice structure and the unit cell of the encoding. Loops around dodecagonal faces are non-trivial Paulis and generate the stabilizer, loops

around triangular faces are identity. The qubit to mode ratio is < 1.34.

This tiling has the same even/odd face pattern as the Kagome lattice, with dodecagons instead of hexagons, accordingly the disparity of this encoding is

$$\Delta = DF + TC - 2 \tag{3.37}$$

where DF is the number of dodecagonal faces and TC is the number of triangular corners. The argument in the proof for proposition 12 is easily modified for this case.

3.4 A Cubic Encoding

3.4.1 Construction

Having covered a range of 2D geometries, a version of the encoding on a cubic lattice will be introduced. This encoding is a generalization of the 2D square encoding reviewed in chapter 2 in the following sense. Construction begins by defining an orientation for each edge in the lattice as in the 2D case, with the condition that any 2D slice of the lattice has an edge orientation identical to a 2D encoding. As with the 2D case, this ensures that edge operators that fail to anticommute at a vertex are clustered around the same faces, allowing them to share auxiliary qubits and resolve this. There are a number of ways to do this. Consider the $(0,0,0)$ corner of the cubic lattice. This corner has three edges extending away from it in the $\hat{x}$, $\hat{y}$ and $\hat{z}$ direction. Specifying the orientation of those three edges completely specifies the orientation of all other edges[1], given the condition that any 2D slice looks identical to the edge orientation of a 2D encoding. Since we do not care about the case where all orientations are inverted, there are four different ways to orientate these three edges. In one case, all arrows point away from the corner, and in the three other cases, two arrows point away from the corner, and a third arrow points into the corner. This is illustrated in fig. 3.13. Thus there are four possible ways to give an orientation to the

[1] More explicitly, if an orientation is decided for an edge aligned with the $\hat{a}$ direction, then all edges running along the same straight line as it must be oriented the same way, and every edge an odd (even) number of lattice translations in the $\hat{y}$ or $\hat{z}$ directions must be oriented the other (same) way. Thus picking an orientation for a single edge parallel to the $\hat{x}$ direction fixes all others parallel to that direction. The same logic applies in the other cardinal directions.

edges of the lattice.

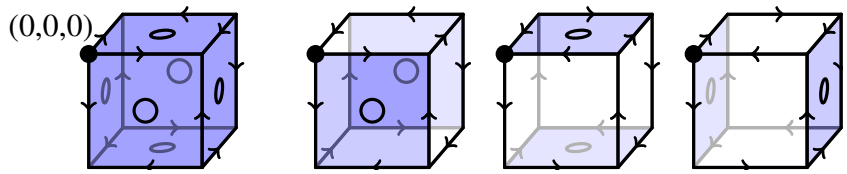


Figure 3.13: Four possible orientations to the edges of the cubic lattice. Odd faces are coloured blue and have a circle in the center to denote the extra qubit. The leftmost cell is odd and the remaining three are even.

In the case of the 2D encoding, the orientations of the edges are in accordance with a checkerboard labelling of the faces, so that an even face has oriented edges circulating around it, and an odd face has all edges around it touching head to head or tail to tail. A similar checkerboard labelling is induced by one's choice of edge orientation for the corner $(0,0,0)$, with all three faces surrounding that corner being odd in the case where all edges point away from the corner, and with only one face being odd in the other three choices of edge orientation.

Regardless of one's choice of orientation, a given cubic cell in the lattice will either have all six faces odd, referred to as an *odd cell*, or exactly two opposite faces odd, referred to as an *even cell* (see fig. 3.13). There are no other possibilities. This is most easily seen by checking the four possible edge orientations of the $(0,0,0)$ corner in the case of a $2\times 2\times 2$ cubic lattice. The cells will be oriented in the cubic lattice such that every odd cell shares its faces only with even cells, and every even cell shares its odd faces only with odd cells, as illustrated in fig. 3.14. Thus every odd face can either be associated with a unique odd cell, or else it is a face on the boundary of the cubic lattice. These odd faces on the lattice boundary that do not belong to an odd cell are referred to as *isolated odd faces*.

Having specified an orientation of the lattice, and an even/odd labelling of all of the faces and cells, the encoding may now be described. In fact the encoding is almost exactly the same as in the 2D case. Associate a qubit with every vertex of the lattice, and a qubit with every odd face of the lattice. The vertex operator at vertex i is given by $\tilde{V}_i = Z_i$, and the edge operator for edge (i,j), with i pointing to j is given by $\tilde{E}_{ij} := X_i Y_j P(i,j)$ ($\tilde{E}_{ji} := -\tilde{E}_{ij}$) where $P(i,j)$ is the same Pauli operator P

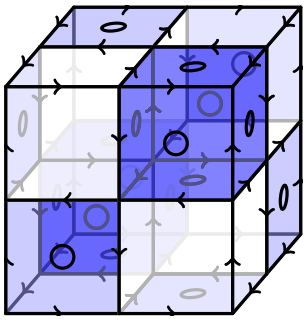


Figure 3.14: The unit cell of the encoding which includes two odd cells (front top right, back bottom left) and six even cells in different orientations.

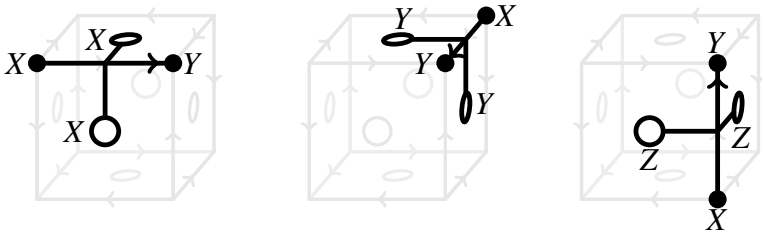


Figure 3.15: Edge operators of the cubic encoding along edges aligned in the x, y and z directions (shown from left to right). If an edge is part of an isolated odd face then it will only act on one face qubit, if it is only part of two even faces then it will only act on vertex qubits.

on every face qubit adjacent to (i, j), where if the edge (i, j) is aligned to the $\hat{x}$ ($\hat{y}$, $\hat{z}$) direction, then $P = X$ (Y, Z) respectively. This ensures that any two edges, going in orthogonal directions, which share an adjacent odd face, will act on that face qubit with different Pauli operators. Finally, for every *isolated odd face*, choose a single adjacent edge (i, j), with i pointing to j, and modify the definition of the operator for that edge to be $\tilde{E}_{ij} := -X_i Y_j P(i, j)$. This ensures that the product of the edges around that face is equal to I and not $-I$. Fig. 3.15 illustrates the operators of this encoding.

As every edge is adjacent to at most 2 odd faces, the maximum weight of the edge operators in this encoding is 4. In the bulk there are $6/4 = 1.5$ odd faces for every vertex so the number of qubits per mode is ≤ 2.5.

3.4.2 Disparity

The disparity of the encoding, Δ, is more complicated to compute than for the planar cases. In particular, $D(K)$ is not simply equal to the number of odd faces, as in the

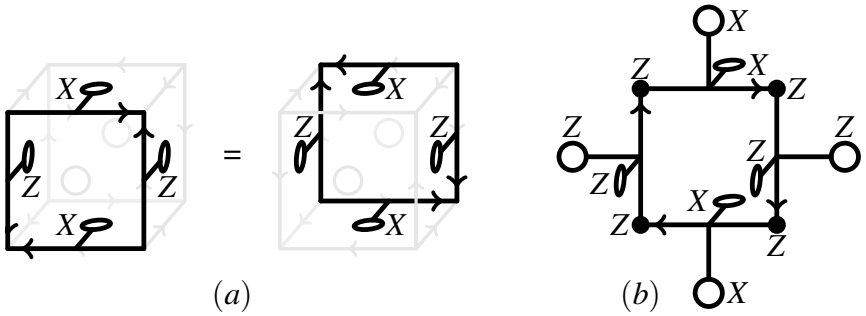


Figure 3.16: Loop operators around odd and even faces in the cubic encoding. (a) shows the identical loop operators around odd faces opposite to each other on an odd cell. Loop operators around odd cells aligned in different directions will have a similar form only with different Paulis. Loop operators around isolated odd faces are identity. (b) shows a loop operator around an even face. Similarly, operators oriented in different directions will have the same shape but different Paulis. Loop operators around even faces on the lattice boundary will have "hanging" Paulis omitted.

square lattice encoding.

Lemma 13.

$$D(K) = IOF + 2OC. \tag{3.38}$$

Where IOF denotes the number of isolated odd faces, and OC denote the number of odd cells.

Proof. Since K is an abelian finitely generated group, its rank $D(K)$ is the size of a maximal independent subset of elements.

Note that for cycles a around isolated odd faces $\sigma(a) = 1$, and so $a \in K$. Furthermore, for a cycle a around an odd face that is bounding an odd cell, the cycle b on the opposite face of that cell satisfies $\sigma(a) = \sigma(b)$ (see Figure 3.16), and so the cycle ab around that pair of faces is in K.

Consider a subset $g \subseteq K$ consisting of the cycles around:

- each isolated odd face.
- one pair of opposite faces for every odd cell.
- a second different pair of opposite faces for every odd cell

We claim that $K = \langle g \rangle$, and that no element of g can be generated by any other elements of g. Thus $D(K) = |g| = IOF + 2OC$.

First we argue that every element of g is independent. This can be seen straightforwardly by noting that every odd face only shares edges with odd faces bounding a common odd cell. Thus all the isolated odd faces are independent. Furthermore each cycle ab around a pair of opposite odd faces in g only shares edges with the other pair of opposite odd faces $cd \in g$, and $ab \neq cd$. Thus all elements are independent.

Secondly we argue that K is generated by g, via proof by contradiction. Assume there exists an element $a \in K$ that is not in $\langle g \rangle$. We first note that there exists a set of odd face cycles F such that

$$a = \prod_{b \in F} b \tag{3.39}$$

This can be seen by noting that if a cycle a is in K then for every edge e_1 in a pointing into (away from) a vertex v there exists another edge e_2 in a which also points into (away from resp.) vertex v. Furthermore by inspection e_1 and e_2 must bound a unique common odd cell. Thus a may be decomposed into a product of cycles, where each of these cycles are confined to the edges bounding a unique odd cell. Such cycles may be generated by the odd face cycles bounding the odd cell.

For any odd cell c, we may define $F_c = \{b | b \in F \text{ and } b \text{ bounds } c\}$, and the corresponding product

$$a_c := \prod_{b \in F_c} b.$$

Without loss of generality we may assume to have chosen an a s.t. no $b \in F$ is an isolated odd face. Thus $a = \prod_c a_c$. We note that any two odd face cycles b and b' that bound different odd cells have representations $\sigma(b)$ and $\sigma(b')$ that act on disjoint qubits. It must follow that $a_c \in K$. It can be seen by enumerating cases that all possible forms of a_c, ie all possible combinations of products of face cycles bounding a given odd cell whose product yields an element of K, may be generated by elements in g. Thus $a \in \langle g \rangle$, which is a contradiction.

□

Given this expression for $D(K)$, the disparity Δ may now be determined from counting arguments.

Theorem 14. *Let an odd corner vertex of a cubic lattice be a corner vertex of the lattice whose associated corner cell is odd. Given a cubic lattice encoding as defined above, the disparity is given by*

$$\Delta = \frac{OCV}{2} - 1 \in \{-1, 0, 1, 3\}$$

Where OCV denotes the number of odd corner vertices.

Proof. Let V denote the number of vertices, E the number of edges, F the number of faces, C the number of cells, OF/EF the number of odd/even faces, OC/EC the number of odd/even cells, IOF the number of isolated odd faces, N the number of qubits, and M the number of fermionic modes.

Note that for this encoding

$$N - M = OF. \tag{3.40}$$

Note also that

$$IOF = OF - 6OC. \tag{3.41}$$

Note also that the Euler characteristic of a cubic lattice is given by

$$V - E + F - C = 1. \tag{3.42}$$

This can be seen most easily by counting the edges vertices faces and cells of a cubic lattice with a single cell, and noting that the Euler characteristic is an invariant of the lattice size, since it is a topological invariant. Thus, using eq. (3.19) retrieve

$$D(\mathcal{C}_G) = F - C. \tag{3.43}$$

Substituting these expressions, and eq. (3.38), into eq. (3.18) gives

$$\Delta = OF - EF - 3OC + EC \tag{3.44}$$

We need only count odd/even faces and cells. For a given vertex in the lattice, we define:

$$\Delta_v = OF_v/4 - EF_v/4 - 3OC_v/8 + EC_v/8 \tag{3.45}$$

where OF_v, EF_v, OC_v, EC_v are the number of odd faces, even faces, odd cells and even cells, respectively, containing the vertex v. Δ_v counts the number of odd/even faces and cells per vertex, with the factors of $1/4$ or $1/8$ corresponding to the number of other vertices sharing respectively a face or a cell. Thus

$$\Delta = \sum_v \Delta_v \tag{3.46}$$

The values of Δ_v for vertices in the bulk ($deg(v) = 6$), on the face ($deg(v) = 5$), on an edge ($deg(v) = 4$), and on a corner ($deg(v) = 3$) of the cubic lattice, may be computed by inspection of the vertex neighbourhoods illustrated in fig. 3.17. For a vertex v in the bulk of the lattice

$$\Delta_v = 6/4 - 6/4 - 3*2/8 + 6/8 = 0. \tag{3.47}$$

For a vertex v on the face of the lattice

$$\Delta_v = 4/4 - 4/4 - 3*1/8 + 3/8 = 0. \tag{3.48}$$

For a vertex v on the edge of the lattice, either

$$\Delta_v = 2/4 - 3/4 - 3*0/8 + 2/8 = 0, \tag{3.49}$$

or

$$\Delta_v = 3/4 - 2/4 - 3*1/8 + 1/8 = 0. \tag{3.50}$$

Finally, for a vertex v on the corner of the lattice, either

$$\Delta_v = 3/4 - 0/4 - 3*1/8 + 0/8 = 3/8 \tag{3.51}$$

if v is adjacent to an odd cell (an odd corner vertex), or

$$\Delta_v = 1/4 - 2/4 - 3*0/8 + 1/8 = -1/8 \tag{3.52}$$

if v is adjacent to an even cell (an even corner vertex). Thus, since there are 8 corner

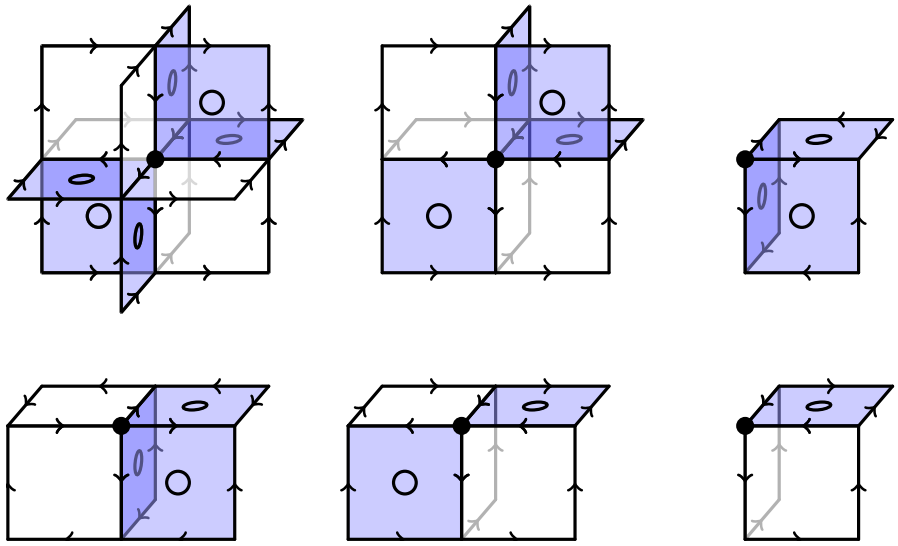


Figure 3.17: The possible neighbourhoods of vertices in the bulk, on a face, on a edge, and on a corner of the cubic lattice.

vertices, which are either even corner vertices or odd corner vertices

$$\Delta = \frac{3}{8}OCV - \frac{1}{8}(8 - OCV) = \frac{1}{2}OCV - 1. \tag{3.53}$$

All that remains is to show that $\Delta \in \{-1, 0, 1, 3\}$.

First note that an odd cell only shares faces with even cells, an even cell only shares odd faces with odd cells, and the face opposite an odd face on an even cell is also an odd face. Therefore, considering the three columns of cells extending away from a given odd cell in the three cardinal directions, each column will consist of alternating even and odd cells. Thus, *given an odd corner cell, a corner opposite to*

it in one of the three cardinal directions will be odd if the side length of the lattice in that direction is even, and even if the side length is odd.

Second note that any cell sharing an even face with an even cell must also be an even cell. Furthermore, the face opposite an even face on an even cell is also an even face. Therefore, if we consider the three columns of cells extending away from a given even cell in the three cardinal directions, only the cells in one of those columns contain odd cells, and the other two columns must only contain even cells. Thus, *for a given even corner cell, of the three corner cells opposite this corner cell along the cardinal directions, at least two of them must be even corner cells, and the other is odd if and only if the side length of the lattice in that direction is odd.*

If a lattice has no odd corners, then the disparity is $\Delta = -1$. If on the other hand a lattice has at least one odd corner vertex, then the observations above completely fix the even or oddness of the remaining corner vertices based solely on the even/oddness of the side lengths of the lattice. This is illustrated in fig. 3.18. In these cases the number of corner vertices is 8, 4 or 2, which yields disparities of 3, 1 and 0.

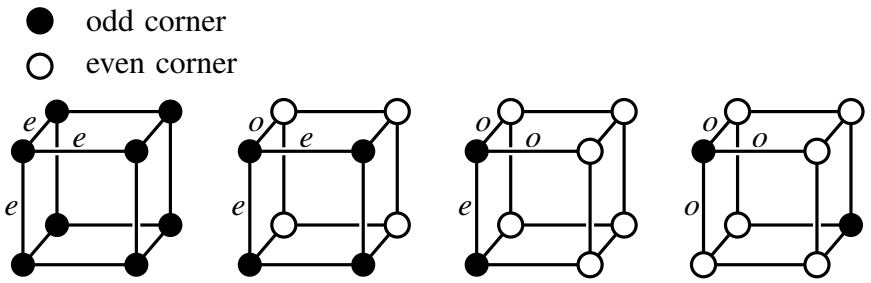


Figure 3.18: The possible configurations of even and odd corners on the cubic encoding, assuming there is at least one odd corner. The letters denote whether a lattice edge has an even (e) or odd (o) number of vertices.

□

For a lattice with infinite extent in any direction, there are no corner vertices, and so the disparity is -1, corresponding to an encoding representing the even fermionic Hilbert space. The 2D square encoding is formally a subcase of the cubic encoding outlined here, and so this also applies to the 2D square encoding.

3.4.3 Particle Species

Particle species are straightforward to identify on this encoding. The odd corners, whose number dictates the disparity of the encoding, serve as injection sites for single particle operators just as in the 2D case. For $\Delta = 0$, 1 and 3 cases the 2, 4 and 8 odd corners beget distinct particle species, two of which can be assigned as the encoded single Majorana $\tilde{\gamma}_i$ and hole $\tilde{h}_i$ operators. Any remaining species are taken to be holes tensored with an element of some maximal anticommuting set of Paulis on the excess encoded qubit space.

3.5 Discussion

This chapter has illustrated how the Compact encoding may be applied to a wide range of lattices and importantly has shown how to analyse the codespace of these encodings. Hopefully these examples will be useful to others in tailoring the Compact encoding to their own needs.

It would be valuable to have a general procedure for constructing these encodings. In particular a procedure for choosing the orientations of the edges of the graphs, and assignment of auxiliary qubits, so that the weights of the edge operators are minimal and act locally. The constructions presented here were not too diffcult to find, but ultimately emerged from a process of trial and error. It is possible that the problem may be straightforwardly framed as a combinatoric optimization problem, however this suggests that a generic set of instructions for optimal constructions is unlikely, especially given that ultimately the optimization problem will be dependent on the particular notion of locality furnished by the quantum computing device.

It was discussed in chapter 2, section 2.3 that the Compact Encoding on a square lattice may be interpreted as condensing fermion-like excitations of the toric code into the codespace. This is revealed by the fact that removing the vertex qubits yields the toric code on a square lattice. Similar connections can be found with some of the uniform tiling constructions in this chapter. The encoding on the 4.8.8 tiling utilises the square lattice toric code in a similar fashion to the square lattice encoding while the 6.4.3.4 and 4.6.12 encodings are connected to the toric code

on a hexagonal lattice in the same manner (or equivalently a triangular lattice). As well as this, stabilizers of the encodings on the Kagome lattice and the 3.12.12 tiling resemble the stabilizers of the color code [65] when the vertex qubits are removed, indicating a possible connection with the fermion-like excitations present in that construction [66]. The encoding on a cubic lattice also reveals a topological code when its vertex qubits are removed, however this underlying code does not appear to resemble anything in the existing literature [46, 60, 67, 68, 69, 70] so it will be explored in more detail in chapter 5.

Chapter 4

Mitigating Errors on the Compact Encoding

As discussed in previous sections, it is desirable for fermionic encodings to encode local fermionic interaction operators as low weight, local qubit operators. This yields lower circuit depths for Hamiltonian simulation algorithms, an important component of quantum procedures for solving classically intractable fermionic systems such as the Fermi-Hubbard model. Recall also, that local fermionic encodings which achieve the former, are in fact stabilizer codes where the encoded fermionic terms are the logical operators and the encoded Fock space forms the codespace, any deviation from which is flagged by measuring the stabilizers, be they loops of edge operators or Jordan-Wigner strings. The error detecting and correcting abilities of stabilizer codes are also desirable for simulation of fermionic systems as the suppression of errors will allow for longer and more detailed computations.

Herein lies a trade-off inherent to fermionic encodings. A good error correcting code has a large code distance which reduces the likelihood of undetectable errors but a low weight fermionic encoding aims to achieve the opposite of this. This chapter argues that, despite the apparent conflict between low-weight fermionic operators and error correction and detection, there can exist valuable error mitigating properties of fermionic encodings that do *not* need to be sacrificed in the pursuit of low-weight fermionic operators. In particular, in the context of fermionic simulation of natural systems, one might tolerate—or even desire—some noise in the physical qubits,

provided that this noise translates into "natural" fermionic noise in the simulated fermionic system. This will depend crucially on the choice of encoding, and on what fermionic operators the low-weight undetectable errors correspond to in the logical fermionic space.

This point has been made in a more general context in [71]. There, the authors prove that for any local Hamiltonian simulations, as defined rigorously in that paper, local physical noise in the simulator system corresponds to local noise in the system being simulated. They also show that such local Hamiltonian simulations can indeed be constructed; in fact, they show there exist simple, universal quantum Hamiltonians that are able to simulate *any* target Hamiltonian, to arbitrary precision. The general theoretical results of [71] do not address specific natural noise models, nor do they address fermionic systems.

This chapter demonstrates how local noise maps to local noise on the encoded system – for specific families of local noise models – on the square and cubic lattice versions of the Compact encoding from this work. The analysis considers independent, identically distributed (iid) noise on the quantum system and studies the effect on the virtual fermionic system. Although this is a simplistic (but widely used) model for noise on quantum devices, the salient features of the results depend only on the locality of the noise model rather than the specific form. Furthermore, this model is a surprisingly good match for the noise observed in current hardware [57].

In summary, section 4.1 shows that the natural noise experienced by a fermionic system coupled to a bosonic bath experiences *fermionic phase noise*, section 4.2 shows that the vast majority of undetectable single qubit errors map to this noise on the encodings considered and section 4.3 shows how the noise for which this is not the case can be avoided. Some discussion follows in section 4.5.

The work in this section is based off material from the paper [59] by the author and Joel Klassen.

4.1 Natural Noise on Fermionic Lattice Models

A common fermionic lattice model is one in which lattice sites correspond to atomic positions. These atomic positions are often considered to be fixed. However, one may consider the possibility of phonons in the lattice of atoms, and how these phonons couple to the electrons as a source of noise. To first order, this coupling is dominated by low energy acoustical modes [72, 73], with interaction Hamiltonian known as the *Fröhlich Hamiltonian*

$$H_{\text{int}} = \frac{1}{V}\sum_{\mathbf{k},\sigma}\sum_{\mathbf{q}} g_{\mathbf{q}} c^{\dagger}_{\mathbf{k}+\mathbf{q},\sigma} c_{\mathbf{k},\sigma}\left(b_{\mathbf{q}} + b^{\dagger}_{-\mathbf{q}}\right), \tag{4.1}$$

where $c^{(\dagger)}_{\mathbf{k},\sigma}$ and $b^{(\dagger)}_{\mathbf{k}}$ are, respectively, the annihilation and creation operators for fermions and phonons with momentum $\mathbf{k}$ and spin σ in the case of fermions.

For a thermal bosonic bath, the effective noise model of this interaction on the fermionic system is spontaneous hopping of fermions into different momentum modes. In the position basis this translates into dephasing noise [74], since motion in momentum space corresponds to phase shifts in position space. The fermionic dephasing operator is:

$$\left(1-2n_j\right) = -\mathrm{i}\gamma_j\overline{\gamma}_j. \tag{4.2}$$

A more thorough derivation of this noise model follows.

4.1.1 Derivation of Fermionic Phase Noise

Consider the Fröhlich Hamiltonian [75, 76], which reads

$$H_{\text{int}} = \sum_{\mathbf{k},\sigma}\sum_{\mathbf{q}} g_{\mathbf{q}} c^{\dagger}_{\mathbf{k}+\mathbf{q},\sigma} c_{\mathbf{k},\sigma}\left(b_{\mathbf{q}} + b^{\dagger}_{-\mathbf{q}}\right). \tag{4.3}$$

Assume for simplicity a momentum-independent coupling $g_q = g := 1$.[1] A Fourier transform of the fermionic and bosonic operators yields the position space expression

$$H_{\text{int}} = \sum_{\mathbf{x},\sigma} n_{\mathbf{x}\sigma}(b_{\mathbf{x}} + b^{\dagger}_{\mathbf{x}}), \tag{4.4}$$

[1] A more sophisticated and realistic analysis could be performed for a momentum-dependent coupling, this will not be attempted here.

where $n_{\mathbf{x}\sigma} = c^{\dagger}_{\mathbf{x},\sigma} c_{\mathbf{x},\sigma}$ is the number operator for spin $\sigma \in \{\uparrow,\downarrow\}$ on site $\mathbf{x}$. Given an interaction strength γ, the interaction unitary is $U_{\text{int}} = \mathrm{e}^{-\mathrm{i}\gamma H_{\text{int}}}$.

A system prepared in a state $\rho = \rho_{\uparrow} \otimes \rho_{\downarrow} \otimes \rho_{\mathcal{B}}$, in the spin-up, spin-down fermionic spaces and bosonic space respectively, will evolve under the channel

$$\mathcal{U}_{\text{int}}(\rho) = U_{\text{int}} \rho_{\uparrow} \otimes \rho_{\downarrow} \otimes \rho_{\mathcal{B}} U^{\dagger}_{\text{int}} \tag{4.5}$$

and the effective channel on the spin-up sector will be

$$\Lambda(\rho_{\uparrow}) = \mathrm{Tr}_{\downarrow,\mathcal{B}} \left[U_{\text{int}} \rho_{\uparrow} \otimes \rho_{\downarrow} \otimes \rho_{\mathcal{B}} U^{\dagger}_{\text{int}} \right]. \tag{4.6}$$

Using the property of the partial trace,

$$\mathrm{Tr}_{\mathcal{W}}[T(I_{\mathcal{V}} \otimes S)] = \mathrm{Tr}_{\mathcal{W}}[(I_{\mathcal{V}} \otimes S)T] \quad \forall S \in L(\mathcal{W}) \;\; \forall T \in L(\mathcal{V} \otimes \mathcal{W}) \tag{4.7}$$

and writing

$$\begin{aligned} U_{\text{int}} &= e^{-\mathrm{i}\gamma(H_{\uparrow}+H_{\downarrow})t} \\ &= e^{-\mathrm{i}\gamma H_{\uparrow} t} e^{-\mathrm{i}\gamma H_{\downarrow} t} \\ &= U_{\uparrow} U_{\downarrow} \end{aligned} \tag{4.8}$$

where $H_{\sigma} = \sum_{\mathbf{x}} n_{\mathbf{x}\sigma}(b_{\mathbf{x}} + b^{\dagger}_{\mathbf{x}})$ with $[U_{\uparrow}, U_{\downarrow}] = 0$, we have

$$\begin{aligned} \Lambda(\rho_{\uparrow}) &= \mathrm{Tr}_{\downarrow,\mathcal{B}} \left[U_{\downarrow} U_{\uparrow} \rho_{\uparrow} \otimes \rho_{\downarrow} \otimes \rho_{\mathcal{B}} U^{\dagger}_{\uparrow} U^{\dagger}_{\downarrow} \right] \\ &= \mathrm{Tr}_{\downarrow,\mathcal{B}} \left[\left(U_{\downarrow} I_{\uparrow} \otimes \rho_{\downarrow} \otimes I_{\mathcal{B}} \right) \cdot \left(U_{\uparrow} \rho_{\uparrow} \otimes I_{\downarrow} \otimes \rho_{\mathcal{B}} U^{\dagger}_{\uparrow} U^{\dagger}_{\downarrow} \right) \right] \\ &= \mathrm{Tr}_{\downarrow,\mathcal{B}} \left[\left(U_{\uparrow} \rho_{\uparrow} \otimes I_{\downarrow} \otimes \rho_{\mathcal{B}} U^{\dagger}_{\uparrow} U^{\dagger}_{\downarrow} \right) \cdot \left(U_{\downarrow} I_{\uparrow} \otimes \rho_{\downarrow} \otimes I_{\mathcal{B}} \right) \right] \\ &= \mathrm{Tr}_{\downarrow,\mathcal{B}} \left[\left(U_{\uparrow} \rho_{\uparrow} \otimes \rho_{\downarrow} \otimes \rho_{\mathcal{B}} U^{\dagger}_{\uparrow} \right) \right] \\ &= \mathrm{Tr}_{\mathcal{B}} \left[U_{\uparrow} \rho_{\uparrow} \otimes \rho_{\mathcal{B}} U^{\dagger}_{\uparrow} \right]. \end{aligned} \tag{4.9}$$

Assuming that the bosonic bath is in a finite temperature Gibbs state in a completely thermalised bath configuration, with inverse temperature β and partition function Z,

the effective channel can be rewritten in the Fock basis $|\vec{N}_{\mathcal{B}}\rangle$ of $\mathcal{B}$ as

$$\Lambda(\rho_\uparrow) = \sum_{\vec{N}_{\mathcal{B}},\vec{N}'_{\mathcal{B}}} \frac{e^{-\beta|\vec{N}'_{\mathcal{B}}|}}{Z} \langle \vec{N}_{\mathcal{B}}| U_\uparrow |\vec{N}'_{\mathcal{B}}\rangle \rho_\uparrow \langle \vec{N}'_{\mathcal{B}}| U_\uparrow^\dagger |\vec{N}_{\mathcal{B}}\rangle . \tag{4.10}$$

Noting that $\langle \vec{N}_{\mathcal{B}}|\vec{N}'_{\mathcal{B}}\rangle = \delta_{\vec{N}_{\mathcal{B}},\vec{N}'_{\mathcal{B}}}$ and $\langle \vec{N}_{\mathcal{B}}| H_\uparrow |\vec{N}_{\mathcal{B}}\rangle = 0$, a second order expansion of $U_\uparrow$ in $\tau = \gamma t$ yields

$$\begin{aligned}\Lambda(\rho_\uparrow) =& O(\tau^3) + \rho_\uparrow \\ &+ \tau^2 \sum_{\mathbf{x},\mathbf{x}'} \Gamma_{\mathbf{x}\mathbf{x}'}(\beta) n_{\mathbf{x}\uparrow}\rho_\uparrow n_{\mathbf{x}'\uparrow} \\ &- \frac{\tau^2}{2} \sum_{\mathbf{x},\mathbf{x}'} \Omega_{\mathbf{x}\mathbf{x}'}(\beta) \left(n_{\mathbf{x}\uparrow} n_{\mathbf{x}'\uparrow}\rho_\uparrow + \rho_\uparrow n_{\mathbf{x}\uparrow} n_{\mathbf{x}'\uparrow} \right)\end{aligned} \tag{4.11}$$

where

$$\Gamma_{\mathbf{x}\mathbf{x}'}(\beta) = \sum_{\vec{N}_{\mathcal{B}},\vec{N}'_{\mathcal{B}}} \langle \vec{N}_{\mathcal{B}}| (b_{\mathbf{x}} + b_{\mathbf{x}}^\dagger) |\vec{N}'_{\mathcal{B}}\rangle \langle \vec{N}'_{\mathcal{B}}| (b_{\mathbf{x}'} + b_{\mathbf{x}'}^\dagger) |\vec{N}_{\mathcal{B}}\rangle \frac{e^{-\beta|N'_{\mathcal{B}}|}}{Z} \tag{4.12}$$

and

$$\begin{aligned}\Omega_{\mathbf{x}\mathbf{x}'}(\beta) &= \sum_{\vec{N}_{\mathcal{B}}} \langle \vec{N}_{\mathcal{B}}| (b_{\mathbf{x}} + b_{\mathbf{x}}^\dagger)(b_{\mathbf{x}'} + b_{\mathbf{x}'}^\dagger) |\vec{N}_{\mathcal{B}}\rangle \frac{e^{-\beta|N_{\mathcal{B}}|}}{Z} \\ &= \sum_{\vec{N}_{\mathcal{B}},\vec{N}'_{\mathcal{B}}} \langle \vec{N}_{\mathcal{B}}| (b_{\mathbf{x}'} + b_{\mathbf{x}'}^\dagger) |\vec{N}'_{\mathcal{B}}\rangle \langle \vec{N}'_{\mathcal{B}}| (b_{\mathbf{x}} + b_{\mathbf{x}}^\dagger) |\vec{N}_{\mathcal{B}}\rangle \frac{e^{-\beta|N_{\mathcal{B}}|}}{Z} \\ &= \Gamma_{\mathbf{x}\mathbf{x}'}(\beta).\end{aligned} \tag{4.13}$$

Noting that $\Gamma_{\mathbf{x}\mathbf{x}'}(\beta) = \delta_{\mathbf{x}\mathbf{x}'}\Gamma_{\mathbf{x}\mathbf{x}}(\beta) = \delta_{\mathbf{x}\mathbf{x}'}\Gamma(\beta)$ where $\Gamma_{\mathbf{x}\mathbf{x}} = \Gamma(\beta)$ does not depend on $\mathbf{x}$, we have

$$\begin{aligned}\Lambda(\rho_\uparrow) =& O(\tau^3) + \rho_\uparrow \\ &+ \tau^2\Gamma(\beta) \sum_{\mathbf{x}\uparrow} n_{\mathbf{x}\uparrow}\rho_\uparrow n_{\mathbf{x}\uparrow} \\ &- \frac{\tau^2}{2}\Gamma(\beta) \sum_{\mathbf{x}\uparrow} \left(n_{\mathbf{x}\uparrow}^2\rho_\uparrow + \rho_\uparrow n_{\mathbf{x}\uparrow}^2 \right).\end{aligned} \tag{4.14}$$

Using the fact that $n_{\mathbf{x}}^2 = n_{\mathbf{x}}$ this can be written

$$\Lambda(\rho_\uparrow) = O(\tau^3) + \rho_\uparrow + \tau^2\Gamma(\beta)\sum_{\mathbf{x}}\left(n_{\mathbf{x}\uparrow}\rho_\uparrow n_{\mathbf{x}\uparrow} + \frac{1}{2}(n_{\mathbf{x}\uparrow}^2\rho_\uparrow + \rho_\uparrow n_{\mathbf{x}\uparrow}^2)\right). \tag{4.15}$$

Considering the fermionic phase operator defined as $\phi_{\mathbf{x}\uparrow} = (1 - 2n_{\mathbf{x}\uparrow})$, then

$$\sum_{\mathbf{x}}\left(n_{\mathbf{x}\uparrow}\rho_\uparrow n_{\mathbf{x}\uparrow} + \frac{1}{2}(n_{\mathbf{x}\uparrow}^2\rho_\uparrow + \rho_\uparrow n_{\mathbf{x}\uparrow}^2)\right) = \frac{1}{4}\sum_{\mathbf{x}}(\phi_{\mathbf{x}\uparrow}\rho_\uparrow\phi_{\mathbf{x}\uparrow} - \rho_\uparrow) \tag{4.16}$$

and thus

$$\Lambda(\rho_\uparrow) = \left(1 - \frac{\tau^2\Gamma(\beta)M}{4}\right)\rho_\uparrow + \frac{\tau^2\Gamma(\beta)}{4}\sum_{\mathbf{x}}\phi_{\mathbf{x}\uparrow}\rho_\uparrow\phi_{\mathbf{x}\uparrow} + O(\tau^3) \tag{4.17}$$

to second order, where M is the number of modes. The above argument applies entirely to the spin-down sector as well.

The above derivation has shown that the reduced evolution channel of a system of spin-$\frac{1}{2}$ fermions coupled to a bosonic bath (e.g. vibrational modes in a material) is approximately equivalent to a random phase flips occurring on position modes after every small time step with some small probability quadratic in the step's duration. From this one sees that if a simulated fermionic system were to experience similar noise then it's arguable that the simulation would still be "true to life". In fact, the following sections will show that for the Compact Encoding, the primary form of any undetectable errors on a simulation will be of this form.

4.2 Mapping Physical Errors to Logical Errors

This section shows that for the square and cubic lattice instances of the Compact encoding shown in chapter 3, all weight-1 qubit errors fall into one of three categories: detectable errors, errors that correspond to mode-weight-1 phase noise, and errors that correspond to mode-weight-1 Majorana operators. The *mode-weight* of a fermionic operator counts the number of fermionic modes acted on non-trivially (not to be confused Pauli weight). Thus, aside from the Majorana errors, all undetectable weight-1 errors correspond to low-mode-weight, local, and arguably natural

fermionic noise as per the derivation in section 4.1. Although the Majorana errors can only occur on very few sites, they can take a fermionic state into one which violates parity superselection. Section 4.3 presents some techniques for avoiding or detecting these errors on the VC and Compact encodings.

The analysis here is motivated by the assumption that weight-1 Pauli noise dominates in the given noise model. This is consistent with the most commonly studied qubit noise model, iid depolarising noise, amongst others.

Proposition 15. *On the Compact encoding for square and cubic lattices, the only non-trivial, undetectable, Pauli weight-1 errors are:*

- *Z operators on primary qubits, which map to mode-weight-1 fermionic phase errors $Z_j \mapsto -\mathrm{i}\gamma_j\overline{\gamma}_j$.*
- *In the case where the full fermionic space is encoded, X or Y errors on those corners adjacent only to an odd face, which map to single Majorana or Majorana hole operators, depending on the choice of convention.*[2]

Proof. There can be X, Y or Z errors, either on the lattice face qubits (auxiliary qubits), or the vertices (primary qubits). They will be addressed separately.

Auxiliary Qubits. By inspection, every face qubit on the square and cubic lattice encodings is acted upon by at least two even face stabilizers with different Pauli operators. Therefore every single Pauli error on a face qubit will flip at least one stabilizer generator and thus is detectable.

Primary Qubits. Stabilizers only act with Z on vertex qubits, stabilizers have no support on vertex qubits that are not contained in even faces (odd corner vertices). Any Z errors on vertex qubits are then undetectable as they always commute with stabilizers; they correspond to fermionic phase errors. X and Y errors on vertex qubits not on odd corners are detectable as they anticommute with stabilizers. X and Y errors on odd corner vertices are undetectable as no stabilizers touch them, these

[2] For example $X_i \mapsto \gamma_i$, $Y_i \mapsto \overline{\gamma}_i$ or $X_i = \gamma_i \prod_j (-i\gamma_j\overline{\gamma}_j)$, $Y_i \mapsto \overline{\gamma}_i \prod_j (-i\gamma_j\overline{\gamma}_j)$. For a fixed fermionic parity, the operator $\prod_j (-i\gamma_j\overline{\gamma}_j)$ is a good quantum number equal to ± 1, and so fermionic hole operators can be thought of as mode-weight-1 fermionic operators.

correspond to single Majorana or Majorana hole operators on the encoded fermionic system.

□

The above shows that, under low-weight biased noise, the primary source of undetectable noise on a fermionic lattice simulation using the Compact Encoding will be weight-1 Z errors on vertex qubits. In the simulated fermionic picture, these correspond to position mode phase flips, exactly the form of the noise channel derived in section 4.1.1. This suggests that even if such a simulation was subject to noise then it would still produce physically meaningful results. The following sections will discuss the mitigation of unphysical, undetectable parity switching errors and a strategy for the correction of detectable low weight errors.

4.3 Mitigating Parity Switching Errors on the Square Lattice

Parity switching errors, such as single Majoranas, can lead to violations of parity superselection in the encoded fermionic system. One strategy to mitigate these errors, up to first order, is to measure the parity of the fermionic system as a stabilizer. This parity stabilizer is given by the product of vertex operators at every fermionic site, which in our case corresponds to the product of Z operators on every primary vertex qubit. However if one is performing non-destructive and coherent stabilizer measurements, or is interested in measuring observables that do not commute with the vertex operators, then such a stabilizer can be very costly to measure coherently. This section shows how for the square case of the Compact encoding, these weight-1 parity switching errors can be avoided through a minor modification to the construction.

A lattice with an odd number of faces can avoid parity switching errors altogether by choosing the appropriate chequerboard pattern of the auxiliary qubits such that none are placed on corner faces, permitting only the representation of parity-preserving fermionic operators. Other lattice shapes always have auxiliary qubits in at least two odd corner vertex qubits and so will always permit parity switching errors.

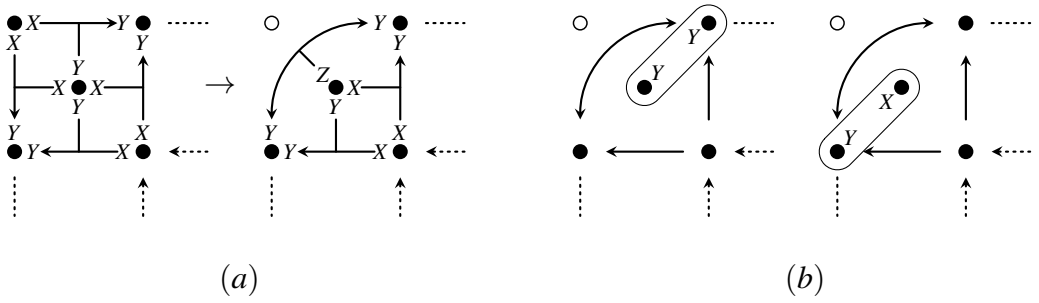


Figure 4.1: Lattice modification to create weight-2 single Majorana/hole operators. (a) The change in edge operators (b) the Majorana/hole operators on the new corner sites. For a corner face where the arrows are all pointing in the other direction, the action of the new edge operator and the new Majorana operators on the vertex qubits will be X.

If the odd corner vertex qubits are removed as shown in fig. 4.1, then the parity switching errors correspond to Pauli weight-2 errors—i.e. the single Majorana or hole operator—has a weight-2 qubit representation. One can preserve most of the structure of the corner, by introducing a new diagonal edge operator connecting those vertex qubits which had previously been connected to the removed site. To ensure the correct anti-commutation relations, this new edge operator acts with a Z on the face qubit and will act with the same Pauli operator on its incident vertex qubits as the two edge operators bounding the odd face (see fig. 4.1). The cycle operator formed by these three edge operators is the identity, provided the correct sign convention is chosen for the diagonal edge.

Single Majorana (or Majorana hole) operators may be added on either of the sites which were previously adjacent to the removed corner site, by applying a weight-2 Pauli operator on the corresponding vertex qubit and on the face qubit. Note that these operators anti-commute with all incident edge operators and the vertex operator on that site.

4.4 Partial Correction of Detectable X and Y Errors

On the compact encoding, X and Y errors on primary sites are detectable, but they are not distinguishable as they differ only by Z, which is itself an undetectable logical error. This means that neither error is correctable since it is impossible to know

which correction to apply and the application of the wrong correction leads to a Z error. Normally this would mean that the codes do not lend themselves to active error correction throughout a circuit run.

However, one can disregard the distinction between X or Y errors and apply a random correction, i.e. an X or Y. This yields a 50% chance that the error is corrected; and otherwise the error will be mapped to an undetectable Z error which maps onto natural fermionic phase noise as per the contents of section 4.1. Together with the fact that all single qubit Pauli errors on auxiliary qubits are distinguishable in both the VC and Compact encodings, this means that active error correction *can* be used for all single qubit errors, at the expense of introducing additional phase noise on the simulated fermionic system.

4.5 Discussion

The features and techniques described in this chapter are relevant for near term quantum algorithms with no active error correction or fault tolerance. Since fermionic encodings are indispensable for fermionic simulation, they constitute a significant fixed overhead in representing any fermionic systems on NISQ devices. One may not be able to afford any additional overhead for supplementary error detection. In this context the time scale of any coherent quantum evolution would have to be upper bounded to ensure that the probability of an error is $\ll 1$. Individual runs might then be post-selected based on whether errors are detected.

Take for example a quantum simulation via the Suzuki-Trotter expansion described in section 1.3, i.e. approximating the Hamiltonian as

$$e^{-\mathrm{i}Ht} = \left(e^{-\mathrm{i}h_1 t/m} e^{-\mathrm{i}h_2 t/m} \cdots e^{-\mathrm{i}h_N t/m}\right)^m + \varepsilon, \tag{4.18}$$

with some error $\varepsilon \sim O(t^2/m)$ and where h_i are the local Hamiltonian terms. Suppose for the sake of argument that the error model is such that errors only occur between Trotter steps, and not in the circuit decomposition of these steps. Given that syndrome measurements are done by measuring stabilizers, which commute with all terms in the Hamiltonian, any sufficiently spatially distant weight-1 Pauli errors in the

volume of the computation can be detected, and those computations may be post-selected away. However if two errors occur within the volume of the computation which cancel their respective syndromes, then these runs can not be post-selected, and will contribute to the overall expected accuracy of the computation. Naturally, the question arises how one might address these higher order errors within this framework.

A natural extension of the work in this chapter is to fully explore how higher Pauli-weight errors map under these encodings. In particular, if higher-weight errors also map to natural local fermionic noise, then it may be possible to similarly mitigate qubit noise to an even higher order. For example, a number of undetectable weight-2 operators on the Compact encoding for a square lattice correspond to edge operators along the boundary which expand into sums of hopping terms along the lattice boundary and pair creation and annihilation operators on the corresponding sites. This suggests a physical error model in which the simulated system is a subsystem of a superconductor-like material where pairs of fermions may enter and exit at the boundary. It will also be worth extending the analysis in this chapter to the other 2D encodings presented in this work and the other local encoding construction in the literature.

One interesting feature of these results is that there is an inbuilt preference for a particular choice of weight-1 Pauli errors. Thus the work presented here may be especially applicable in cases were the hardware is already biased towards certain Pauli errors.

It is worth noting that if stabilizers are only measured at the end of a run, then, depending on the observables one is interested in measuring, the cost of some stabilizer measurements may be significantly reduced, since they can be performed destructively and non-coherently. For example, if one is purely interested in measuring fermion density in the system, then the highly non-local parity operator can be measured simply by measuring every qubit in the Z basis. Thus allowing one to detect the Majorana errors described in this work, without having to resort to any modifications of the encodings.

This material remains relevant even if one does have fault tolerance. Mitigating a large fraction of errors already one level above the error-correcting code would allow a reduction in overhead.

Chapter 5

Code Underlying the Cubic Compact Encoding

In chapter 2, section 2.3 it was shown how the compact encoding may be thought of as a method for condensing the particle excitations of the toric code into the low energy subspace. In that case removing the vertex qubits from the stabilizer generators reveals toric code stabilizers on the auxiliary qubits. This feature translates to the cubic encoding, where the removal of the vertex qubits also reveals a topological code structure. This code bears some resemblance to a 3D generalization of the toric code presented in [67, 77], which has qubits living on the faces of cubes, and employs weight 6 stabilizers on each cube. The code underlying the cubic encoding also has qubits living on cube faces however they are more sparsely arranged with the simplest stabilizers being weight 4 and 8 and associated with faces. Having investigated the existing literature the author has been unable to find a code with a structure obviously resembling this [60, 46, 68, 69] so this section will describe the code's structure and features on a finite cubic lattice with open boundary conditions (referred to as an *open lattice*) and a lattice on a 3-torus.

The code described in this chapter will be referred to as the *underlying code*. This chapter will also contain numerous references to specific cardinal directions. Unless otherwise stated, diagrams will consistently be oriented using the convention shown in fig. 5.1, with the x axis oriented left-to-right, z axis top-to-bottom and the y axis running in and out of the page.

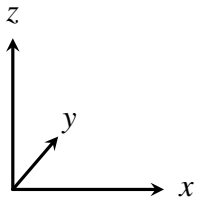


Figure 5.1: The convention used for cardinal directions in this section. All subsequent diagrams of 3D structures are oriented this way unless otherwise stated.

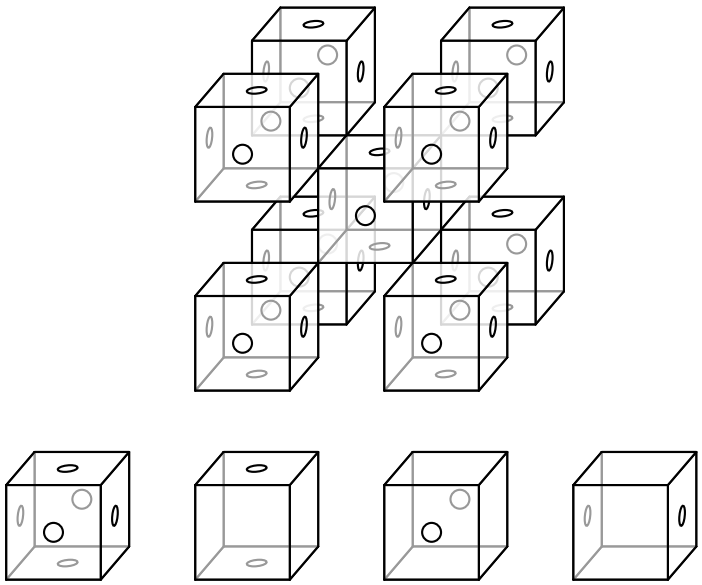


Figure 5.2: (Upper) Odd cells arranged touching corner-to-corner, even cells lie in the gaps between odd cells. (Lower) An odd cell and all orientations of even cells. Qubits live on all faces of odd cells, denoted by circles.

5.1 Structure and Codespace

As with the cubic encoding, the underlying code is defined on a cubic lattice with faces and cells labelled even or odd. Odd faces form all 6 sides of odd cells which are arranged touching corner-to-corner with the remaining cells being labelled even and, consequently having 4 even faces and 2 odd faces opposite one another (see fig. 5.2). Qubits live only on odd faces and the stabilizer group is generated by operators identical to the loop operators of the cubic encoding, only without the vertex qubits (see fig. 5.3). In the case of the 3-torus, the stabilizer of the cubic encoding includes loop operators corresponding to non-contractible cycles (see fig. 5.4) as the loop condition on edge operators must be satisfied for every loop. Equivalent operators are not included in the stabilizer of this code.

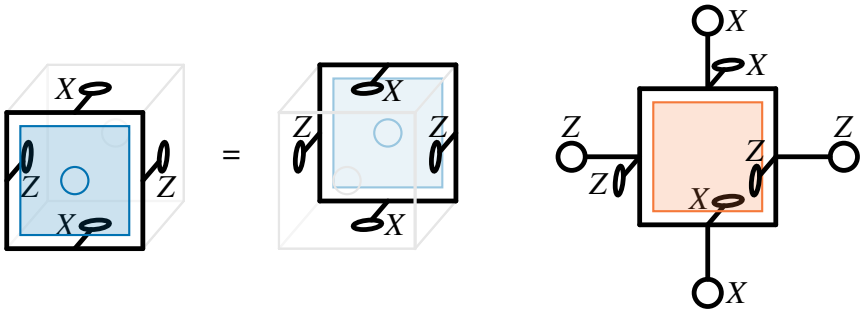


Figure 5.3: Stabilizer generators of the underlying code associated with lattice faces. An odd face stabilizer operator is identical to the stabilizer operator associated with the opposite face on the same odd cell. In later diagrams odd face stabilizers will be denoted by blue squares with circles in the centre and even face stabilizers by red squares, as indicated in this figure.

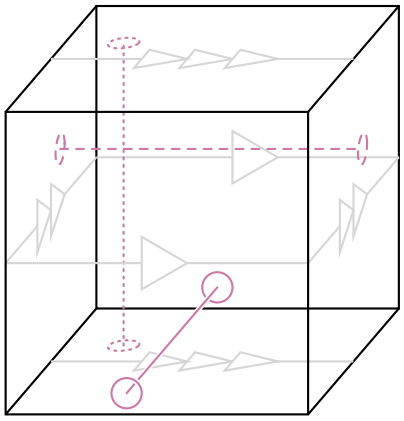


Figure 5.4: Non-contractible cycles on a 3-torus. Boundaries with matching arrow labels are joined such that the directions of the arrows match. There are 3 such cycles that cannot be reduced to one another by continuous deformation. These are denoted by the solid, dashed and dotted purple lines between the boundaries.

The relationship between the cubic fermionic encoding and the underlying code can be formalised by a mapping between the linear operators on their respective Hilbert spaces:

$$\mu : L(\mathcal{H}_F) \to L(\mathcal{H}_C), \tag{5.1}$$

where $\mathcal{H}_F$, and $\mathcal{H}_C$ are the Hilbert spaces of the fermionic encoding and the underlying code respectively. The mapping μ is simply amounts to the removal of all vertex qubit support from Pauli operators in $L(\mathcal{H}_F)$. Furthermore, as discussed in section 3.1, the cubic encoding can be related to the edge and vertex group M_G (see

eq. (3.6) and section 3.1) by the mapping

$$\sigma : M_G \to L(\mathcal{H}_F). \tag{5.2}$$

The composition of these mappings, $\mu \circ \sigma$, yields

$$\nu : M_G \to L(\mathcal{H}_C). \tag{5.3}$$

Applying the mapping σ to the cycle group $\mathcal{C}_G \triangleleft M_G$ produces the cubic encoding stabilizer $\mathcal{S}_F$ and in the case of open boundary conditions, applying μ to $\mathcal{S}_F$ produces the stabilizer $\mathcal{S}_C$ of the underlying code. In the case of the cubic lattice on a 3-torus however the underlying code stabilizer is

$$\nu(\mathcal{C}'_G) = \mu(\sigma(\mathcal{C}'_G)) = \mu(\mathcal{S}'_F) = \mathcal{S}_C \tag{5.4}$$

where $\mathcal{C}'_G$ is the subgroup of $\mathcal{C}_G$ containing only contractible cycles and $\mathcal{S}'_F$ is the corresponding subgroup of $\mathcal{S}_F$. On open lattices $\mathcal{C}'_G = \mathcal{C}_G$ and $\mathcal{S}'_G$ so for the rest of this section, discussion may be limited to these subgroups.

Similarly to the discussion in section 3.1.1, important groups are the subgroups of $\mathcal{C}'_G$ mapped to identity via σ and ν and the subgroup of $\mathcal{S}'_F$ mapped to identity by μ, i.e. $\ker(\sigma|_{\mathcal{C}'_G})$, $\ker(\nu|_{\mathcal{C}'_G})$ and $\ker(\mu|_{\mathcal{S}'_F})$, respectively. To avoid overcrowded notation, define

$$\begin{aligned} K_\sigma &:= \ker(\sigma|_{\mathcal{C}'_G}) \\ K_\nu &:= \ker(\nu|_{\mathcal{C}'_G}) \\ K_\mu &:= \ker(\mu|_{\mathcal{S}'_F}). \end{aligned} \tag{5.5}$$

Lemma 16. *The rank of stabilizer $\mathcal{S}_C$ is $D(\mathcal{S}_C) = D(\mathcal{C}'_G) - D(K_\nu) = D(\mathcal{C}'_G) - D(K_\sigma) - D(K_\mu)$.*

Proof. By identical argument to theorem 1 we have that $|\mathcal{S}_C| = |\mathcal{C}'_G/K_\nu|$, $|\mathcal{S}'_F| =$

$|\mathcal{C}'_G/K_\sigma|$ and $|\mathcal{S}_C| = |\mathcal{S}'_F/K_\mu|$. By Lagrange's theorem we have that

$$\begin{aligned} |\mathcal{S}_C| = |\mathcal{S}'_F|/|K_\mu| &= |\mathcal{C}'_G/K_\sigma|/|K_\mu| \\ &= (|\mathcal{C}'_G|/|K_\sigma|)/|K_\mu| = |\mathcal{C}'_G|/|K_\nu| \end{aligned} \tag{5.6}$$

All elements of $\mathcal{C}'_G$, $\mathcal{S}'_F$ and $\mathcal{S}_C$ commute and square to identity so we have that $|\mathcal{S}_C| = 2^{D(\mathcal{S}_C)}$, $|\mathcal{C}'_G| = 2^{D(\mathcal{C}'_G)}$ and $|K_i| = 2^{D(K_i)}$ for $i = \sigma, \mu, \nu$. Therefore

$$D(\mathcal{S}_C) = D(\mathcal{C}'_G) - D(K_\nu) = D(\mathcal{C}'_G) - D(K_\sigma) - D(K_\mu) \tag{5.7}$$

□

Lemma 17.

$$D(K_\mu) = \begin{cases} 1, & \text{if } OCV = 0, \\ 0, & \text{if } OCV > 0 \end{cases} \tag{5.8}$$

where OCV is the number of odd corner vertices of the lattice (i.e. number of lattice corners that are part of odd cells, as in theorem 14).

Proof. As the mapping μ simply removes vertex qubits from operators in $\mathcal{S}_F$, the non-trivial elements of K_μ are operators with support only on vertex qubits.

Consider a product of cubic encoding edge operators (fig. 3.15). Any given face qubit is only acted upon by edge operators with the Paulis corresponding to the plane its face is aligned in. For example, a qubit on a face in the xy-plane is only acted on with X and Y by edge operators. This means that the only way to cancel any edge operator's action on a face qubit with another edge operator is by applying the operator on the opposite edge of the qubit's associated face. Furthermore, this implies that for any product of edge operators with no support on face qubits, the inclusion of any edge operator with face qubit support implies the inclusion of every parallel edge operator with which it shares an odd face. Consequently, the only products of edge operators with no support on face qubits are combinations of:

- Sets of four parallel edges belonging to the same odd cell
- Sets of two parallel edges belonging to the same isolated odd face

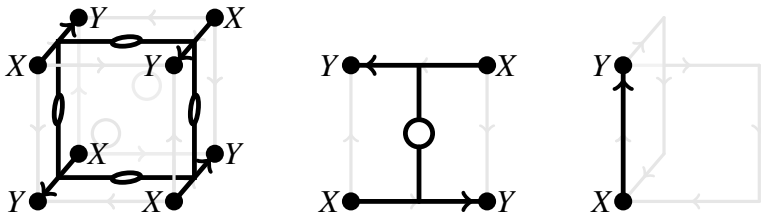


Figure 5.5: (Left, centre) Products of parallel edge sets which have no support on face qubits on an odd cell and an isolated odd face. (Right) An edge operator that belongs to no odd cells along the edge of the lattice. All elements of K_μ must be a product of operators of these types.

- Edges that belong to no odd faces.

Sets of parallel edges belonging to the same odd cell / isolated odd face will be referred to as *parallel sets*. Each odd cell has 3 such sets and each isolated odd face has 2, see fig. 5.5 for illustration. As the elements of $\mathcal{S}'_F$ are products of edge operators, this constraint applies to members of K_μ. A further constraint on the elements of K_μ is that they must be products of edge operators that form a closed cycle, that is every vertex must have an even number of incident edges in the product.

Consider an element of K_μ with an even number of parallel sets on an odd cell as part of its edge product. Every vertex on this cell will then have an even number these edges incident. To maintain even edge incidence, all odd cells / isolated odd faces (IOF) that share a vertex with this odd cell must also have an even number of their parallel sets as part of the edge product. Every odd cell / IOF shares a vertex with another odd cell / IOF so this extends to the entire lattice. This element of K_μ will then have an even number of edge operators from each odd cell / IOF acting on each vertex as every vertex belongs to an odd cell / IOF. Recalling that edge operators on the same odd cell / IOF act with the same Pauli on shared vertices the resulting operator will have no support on the vertex qubits or face qubits, making it a trivial element of K_μ.

It then follows that any non-trivial element of K_μ must be a product of edge operators consisting of an odd number of parallel sets on each odd cell and IOF. They must still form a cycle however, with every vertex having an even number of incident edges in the product. No such operator exists for a lattice with $OCV > 0$ as

the corner vertices at odd corners are not shared by any other odd cells, odd faces or edges belonging to neither. This means they will have an odd number of incident edges, hence $D(K_\mu) = 0$ in this case.

If $OCV = 0$ then there may exist vertices along the edges of the lattice and at the corners that only belong to one odd cell or IOF. In this case, they will share an edge which belongs to no odd cells or faces which must then be present in the product to ensure it is a cycle. So then for a lattice with $OCV = 0$, a non-trivial element of K_μ must be a product of an odd number of parallel edge sets for every odd cell and IOF and every edge that belongs to no odd cells. Every vertex will be acted on by two of these objects, one with an X and one with a Y meaning that the only non-trivial K_μ element is Z on every vertex qubit, hence $D(K_\mu) = 1$.

In the 3-torus case every vertex is shared by two odd cells so the cycle condition is satisfied automatically by a product of odd numbers of parallel sets on every odd cell. As in the open boundary case, this will result in the only element of K_μ being Z on every vertex qubit. The lattice on the 3-torus has no boundary and thus, no corners so the lemma holds. □

Theorem 18. *The codespace of the underlying code has dimension 2^{N_C}, i.e. it encodes N_C logical qubits, where, for an open cubic lattice:*

$$N_C = \begin{cases} 0, & \text{if } OCV = 0 \\ \frac{OCV}{2} - 1, & \text{if } OCV > 0 \end{cases} \tag{5.9}$$

with $\frac{OCV}{2} - 1 \in \{0, 1, 3\}$ and for a cubic lattice on a 3-torus:

$$N_C = 3. \tag{5.10}$$

Proof. Let N be the number of qubits used in an instance of the code. The dimension of the codespace is then $2^{N - D(\mathcal{S}_C)}$, equivalent to $N - D(\mathcal{S}_C)$ qubits. As N is the number of odd faces, OF, and $D(\mathcal{S}_C) = D(\mathcal{S}'_F) - D(K_\mu)$ then, by lemma 17 for the

case of open boundary conditions we have

$$N_C = \begin{cases} \Delta + 1, & \text{if } OCV = 0 \\ \Delta, & \text{if } OCV > 0. \end{cases} \tag{5.11}$$

Here $\Delta = OF - D(\mathcal{S}'_F)$ is the disparity for the cubic encoding on the same lattice. This is equivalent to eq. (5.9) by theorem 14.

For the 3-torus case we have by lemmas 13 and 16

$$\begin{aligned} N_C = N - D(\mathcal{S}_C) =& N - D(\mathcal{C}'_G) + K_\sigma + K_\nu \\ =& N - D(\mathcal{C}'_G) + 2OC + 1. \end{aligned} \tag{5.12}$$

The Euler characteristic of a lattice on a 3-torus is given by

$$V - E + F - C = 0, \tag{5.13}$$

where V, E, F, C are the numbers of vertices, edges. This can be seen by considering the fact that the Euler characteristic is multiplicative for Cartesian products of graphs. That is

$$\chi(G \times H) = \chi(G)\chi(H) \tag{5.14}$$

where $\chi(\cdot)$ denotes the Euler characteristic of a graph. The Euler characteristic of a circle graph is 0 and the graph of a cubic lattice on a 3-torus is simply Cartesian product of 3 of these. Using eqs. (3.19) and (5.13) we have the rank of the total cycle group on a 3-torus

$$D(\mathcal{C}_G) = F - C + 1. \tag{5.15}$$

A minimal generator for $\mathcal{C}_G$ on a 3-torus must contain exactly 3 non-contractible cycles. A generator for $\mathcal{C}'_G$ may then be obtained by removing these 3 cycles from the set. The rank of $\mathcal{C}'_G$ is then

$$D(\mathcal{C}'_G) = D(\mathcal{C}_G) - 3 = F - C - 2. \tag{5.16}$$

Substituting into eq. (5.12) we have

$$\begin{aligned} N_C &= N - F + C + 2 + 2OC + 1 \\ &= OF - (EF + OF) + (EC + OC) + 2OC + 3 \\ &= 3OC - EF + EC + 3. \end{aligned} \tag{5.17}$$

Inspection the neighbourhoods of each vertex (illustrated in fig. 3.17) reveals that for every vertex there are $\frac{2}{8}$ odd cells, $\frac{6}{8}$ even cells and $\frac{6}{4}$ even faces. From this it is clear that $3OC - EF + EC = 0$ and thus

$$N_C = 3 \tag{5.18}$$

for the code on a 3-torus. □

5.2 Excitations, Logical Operators and Code Distance

5.2.1 Geometrical Pictures

With the codespaces found, the natural next step is to identify the logical operators with which to navigate them. It will be helpful to use some geometric interpretations to describe operators and syndromes on the code in this section. These interpretations should be explored and justified before continuing.

5.2.1.1 Building Operators from Strings

Consider a cubic lattice, either on a 3-torus (with even side lengths) or with open boundaries and label the faces odd and even in the pattern required of the underlying code. Define the groups $\mathbb{C}$, $\mathbb{F}$, $\mathbb{E}$ and $\mathbb{V}$ as the sets of unordered subsets of cells, faces, edges and vertices with the group action as the disjoint union of two elements.

Consider now, the Pauli group $\mathcal{P}_{\mathcal{H}_F}$ on the Hilbert space of the cubic fermionic encoding $\mathcal{H}_F$ and define $\mathcal{P}'_{\mathcal{H}_F} = \mathcal{P}_{\mathcal{H}_F}/\{I, -I, \mathrm{i}I, -\mathrm{i}I\}$, i.e. the Pauli group with

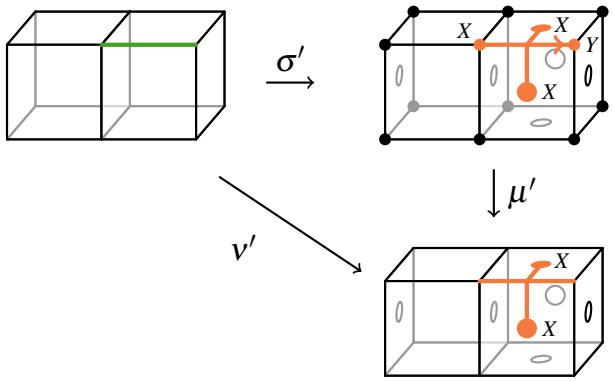


Figure 5.6: Graphical representation of the mappings σ', μ', and $\nu' = \mu' \circ \sigma'$. An edge on the lattice highlighted in green is transformed by σ' into an edge operator on the cubic compact encoding which is transformed by μ' into a code edge operator on the underlying code, each highlighted in red.

phases ignored. Define the mapping[1]

$$\sigma' : \mathbb{E} \to \mathcal{P}'_{\mathcal{H}_F} \tag{5.19}$$

which maps elements of $\mathbb{E}$ to the product of corresponding fermionic edge operators in the cubic compact fermionic encoding on the lattice up to a sign (orientation irrelevant). Then define the mapping

$$\mu' : \mathcal{P}'_{\mathcal{H}_F} \to \mathcal{P}'_{\mathcal{H}_C} \tag{5.20}$$

where $\mathcal{H}_C$ is the Hilbert space of the code underlying the cubic encoding, i.e. $\mathcal{H}_F$ with all vertex qubits removed. The mapping μ' simply takes elements of $\mathcal{P}'_{\mathcal{H}_F}$ and removes their vertex qubit support, leaving them otherwise unchanged.

The composition $\nu' = \mu' \circ \sigma'$ maps elements of $\mathbb{E}$ to the corresponding product of *code edge operators*, where the code edge operator for a given edge is the corresponding edge operator from the cubic encoding with the vertex support removed, see fig. 5.6 for an illustration. This formalises a relationship between edges on the lattice and the subgroup of the Pauli operators on the code formed by the code edge operators.

[1] The following mappings are analogous to the mappings σ,μ and ν defined earlier but since they act on different spaces they are primed in the notation to distinguish them.

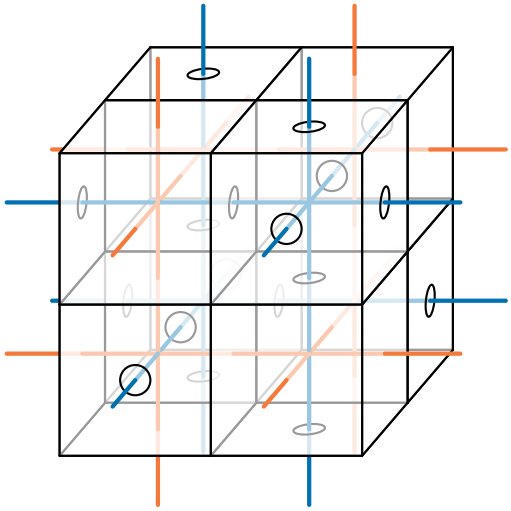


Figure 5.7: The lattice of the underlying code (black) and its dual lattice (blue/red). Dual edges of odd faces are coloured blue and dual edges of even faces are coloured red. Note that not every dual face is fully surrounded by dual edges, namely the faces dual to edges at the boundary of the original lattice.

Upon inspection of the generators one can see that the stabilizer is obtained by applying the mapping ν' to the subgroup of $\mathbb{E}$ corresponding to contractible cycles on the lattice. It will be useful at times to think of stabilizers and some operators as strings on the lattice mapped under ν'. It is worth noting that this mapping is not bijective and can map different elements of $\mathbb{E}$ to the same Pauli operator, meaning that strings of code edge operators are not associated with a unique set of edges on the lattice. However this picture still provides a useful intuition for building operators.

5.2.1.2 Operators as Surfaces

Consider the dual lattice of the lattice upon which an instance of the underlying code is defined. Cells, faces, edges and vertices on the dual lattice are respectively dual to vertices, edges, faces and cells on the original lattice. Label the edges of the dual lattice odd and even according to the face on the original lattice that they are dual to. Define also the groups $\mathbb{C}^*$, $\mathbb{E}^*$, $\mathbb{F}^*$ and $\mathbb{V}^*$ analogously to $\mathbb{C}$ etc. As will be shown, it is possible to define a mapping between Pauli operators on the code and the face group $\mathbb{F}^*$ on the dual lattice, thereby formalising an interpretation of Pauli operators as surfaces.

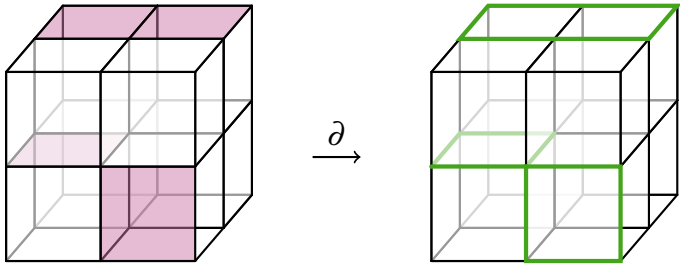


Figure 5.8: Illustration of the mapping ∂. A set of faces highlighted in purple is mapped to the set of edges highlighted in green.

It is first helpful to formalise a relationship between Pauli errors – more precisely, their syndromes – on the code and the edges of the dual lattice. Consider the mapping on the original lattice

$$\partial : \mathbb{F} \to \mathbb{E} \tag{5.21}$$

which maps an element of $\mathbb{F}$ to the set of edges which bound it. Each element of $\mathbb{F}$ is then mapped to a contractible cycle of edges, see fig. 5.8 for an illustration. One can then see that under the composed map $\Gamma = \nu' \circ \partial$, faces map to stabilizer operators.

Define now a generating set g of the stabilizer $\mathcal{S}_C$ as the set of all stabilizers that single faces map to under Γ, formally

$$g = \bigcup_{f \in \mathbb{F}} \Gamma(f) \;\; s.t. \;\; |f| = 1. \tag{5.22}$$

Note that some f will map to the same stabilizer, these operators appear only once in g. Subsets of this generating set correspond to syndromes of errors on the code, the set of syndromes is then equivalent to the power set of the generator $P(g)$. A mapping from syndromes to faces on the lattice

$$\Sigma : P(g) \to \mathbb{F} \tag{5.23}$$

can then be defined as

$$\forall s \in P(g) \;\; : \;\; \Sigma(s) = \bigcup_{f \in \mathbb{F}} f \;\; \text{s.t.} \;\; |f| = 1, \, \Gamma(f) \in s \tag{5.24}$$

i.e. a syndrome maps to the set of all single faces which map to elements of the syndrome. This may include multiple faces which each map to the same stabilizer. From here on, the syndrome of an error and the faces that the syndrome maps to will be used interchangeably.

Now consider the syndrome of single Pauli error. For example an X error on the qubit associated with an odd face parallel to the xz plane (i.e. facing the y direction). The X will anticommute with the stabilizers associated with every face that shares the odd face's edges oriented in the z direction, that is, every generator that each of those faces maps to under Γ. Via Σ, this syndrome then maps to those 6 faces. The syndrome of a Z error on the same qubit analogously maps to the 6 faces which share the odd face's x oriented edges. Similar rules apply for errors on faces parallel to the xy and yz planes (see fig. 5.9 for illustration). By mapping each Pauli operator to its syndrome and then applying the mapping Σ the mapping

$$\theta : \mathcal{P}'_{\mathcal{H}_C} \to \mathbb{F} \tag{5.25}$$

is defined which maps Paulis to sets of faces.

Define the mapping

$$\delta : \mathbb{F} \to \mathbb{E}^* \tag{5.26}$$

which maps faces on the original lattice to their dual edges. One sees that the composition $\delta \circ \theta$ maps Paulis on the code to contractible cycles of edges on the dual lattice. This is reminiscent of the 3D toric code on which the syndromes of X errors also map to contractible loops on the dual lattice. These errors can then be mapped onto surfaces on the dual lattice whose boundaries are formed by these loops [78].

Indeed a similar picture can be justified for errors on the underlying code, in fact all errors can be mapped onto surfaces rather than just certain types. Consider a single Pauli error on the underlying code that maps to a closed loop of 6 dual edges under $\delta \circ \theta$, e.g. an X error on an odd face parallel to the xz plane. This loop encloses a 2×1 surfaces parallel to the xy plane, centred on the odd edge dual to the qubit's odd face and with its long side running along the x direction. Call this surface

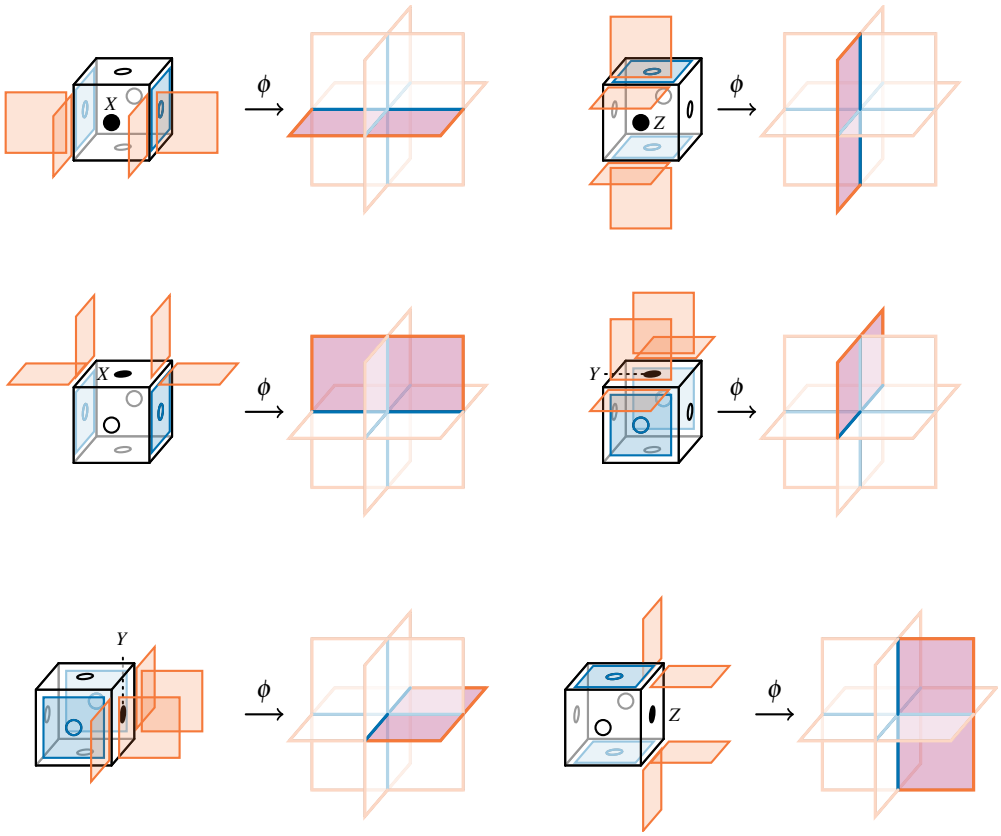


Figure 5.9: Single Pauli errors on odd cells in the bulk of the underlying code and their associated domino surface elements on the dual lattice under the mapping ϕ. Filled circles denote the qubit the error acts on, even and odd faces whose stabilizers are flipped by an error are highlighted in red and blue. Purple filled faces denote the associated surface on the dual lattice with its boundary highlighted. Errors on the opposite odd faces to those pictured have mirrored behaviour.

a *domino* surface element due to its shape and associate it with the Pauli error (see fig. 5.9). A Z error on the same qubit is then associated with a domino centred on the same (dual) edge but with its long side running along the z direction. For each qubit on the code lattice, 2 Pauli operators can be chosen that correspond to the 2 domino elements centred on its face's dual odd edge. These pairs of Paulis form a generating set for $\mathcal{P}'_{\mathcal{H}_C}$ and are shown in fig. 5.9.

Consider now the set of all domino elements centred on odd dual edges. These surface elements form a generating set for the subgroup of surfaces $G \triangleleft \mathbb{F}^*$ such that

$\partial^*(G) = \mathrm{Im}(\delta' \circ \theta)$. Here the mapping

$$\partial^* : \mathbb{F}^* \to \mathbb{E}^* \tag{5.27}$$

maps faces on the dual lattice to the product of their surrounding edges, analogous to ∂ on the original lattice. As discussed earlier, a one-to-one mapping can be defined between a generating set for $\mathcal{P}'_{\mathcal{H}_C}$ and the domino elements that generate G. This then defines a mapping

$$\phi : \mathcal{P}'_{\mathcal{H}_C} \to \mathbb{F}^* \tag{5.28}$$

whereby any Pauli error on the underlying code is mapped to a surface on the dual lattice whose boundary corresponds to the syndrome over the stabilizer generator g. From this one can find operators that commute with the stabilizer by finding closed surfaces on the dual lattice that can be decomposed into the domino elements in fig. 5.9. A Pauli operator P will be said to *induce* the surface $\phi(P)$ later in this work.

This interpretation of Pauli operators on the code yields an alternative interpretation of the stabilizer generators. While they can be interpreted as contractible loops of code edge operators, they can now also be viewed as surfaces enclosing a volume on the dual lattice. The surface due to an even face stabilizer encloses a $2 \times 2 \times 1$ volume and the surface due to an odd face stabilizer cancels out to nothing (see fig. 5.10). The surfaces induced by an even stabilizer capture deformations which can transform between elements of G without changing their boundary under ∂^*.

The restricted set of surface primitives (the dominos) mean that Pauli operators map to only a limited set of surfaces which may not be freely deformed. Compare this to, say, the surface operators of the 3D toric code where products of X errors correspond to arbitrary tilings of lattice faces and products of vertex stabilizers correspond to arbitrary enclosing surfaces. However, this rigidity present in the underlying code allows full sets of logical operators to be constructed from the surface-like Pauli operators, unlike the 3D toric code which requires string-like Z products to fully navigate the codespace.

In the case of open boundary conditions, the edges at the lattice boundaries

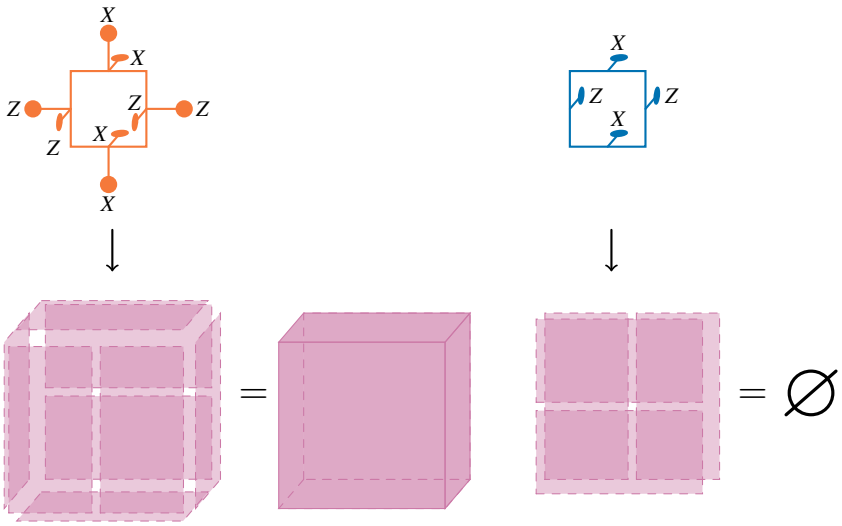


Figure 5.10: The surface picture admits surface interpretations of stabilizers as closed surfaces. Stabilizers formed from loops around even faces induce a surface enclosing a $2 \times 2 \times 1$ volume. Stabilizers formed from loops around odd faces induce no surface as their constituent parts cancel out.

are dual to faces which are not fully surrounded by edges (see fig. 5.7). These are included in the surfaces induced by Pauli operators on some instance of the underlying code on these lattices and their syndrome only corresponds to their boundary on existing edges in the dual lattice. Here "existing" edges refers to edges on the dual lattice which have a dual face on the original lattice. Faces in the bulk of the dual lattice are fully surrounded by existing edges but faces corresponding to edges at the boundary of the original lattice are not and are partially surrounded by implicitly defined "absent" edges. If a Pauli operator induces a surface whose boundary is only on the absent edges then it must be a stabilizer or a logical operator (see fig. 5.11).

The following subsections will define full sets of logical operators for the underlying code on a 3-torus and on the 2 cases of open cubic lattices for which it has non-trivial codespace.

5.2.2 3-Torus

The logical $\tilde{Z}$ operators on the 3 qubits in the 3-torus case can most easily be found by considering the non-contractible cycles of the 3-torus (see fig. 5.4). Recall that the Pauli operators which these map to under ν' are chosen to be absent from the

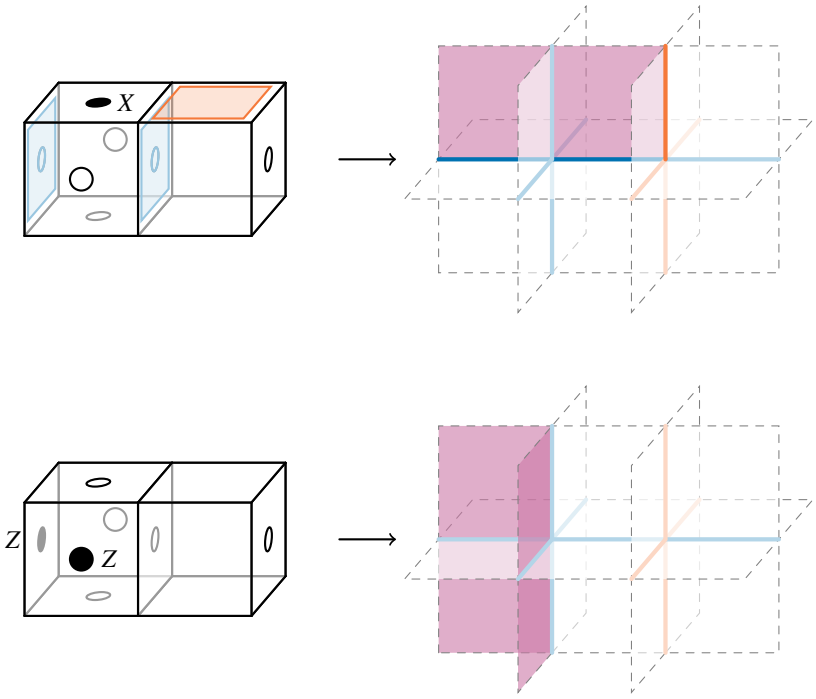


Figure 5.11: Pauli errors on the Underlying Code on a $2 \times 1 \times 1$ lattice with their syndromes (left) and the surfaces they map to on the dual lattice (right). The dashed lines on the dual lattices denote "absent edges". (Upper) A single qubit error has a non-trivial syndrome, it maps to a surface with boundary on non-absent edges. (Lower) A two qubit error with no syndrome, the boundary of its corresponding surface on the dual lattice is purely on absent edges.

code stabilizer. Considering these Pauli operators one finds 3 commuting logical operators on the code space which can be chosen as the 3 logical $\tilde{Z}$ operators. Each unique non-contractible cycle runs along a cardinal direction of the lattice so define their operators accordingly as $\tilde{Z}_x$, $\tilde{Z}_y$ and $\tilde{Z}_z$.

To see that these operators have no syndrome consider an open ended straight line of code edge operators and switch to the surface picture discussed earlier. Each edge operator induces a folded surface on the dual lattice, due to the alternating positions of the qubits along the string of edge operators these surfaces alternate in orientation down the line and form an open ended tube whose boundaries run along the edges corresponding to the dual edges of all even faces which share the vertices at each end of the string on the original lattice (see fig. 5.12). These boundaries will change orientation depending on their position in the lattice but for a tube induced by string of even length, the boundaries at each end will be identical. The 3-torus

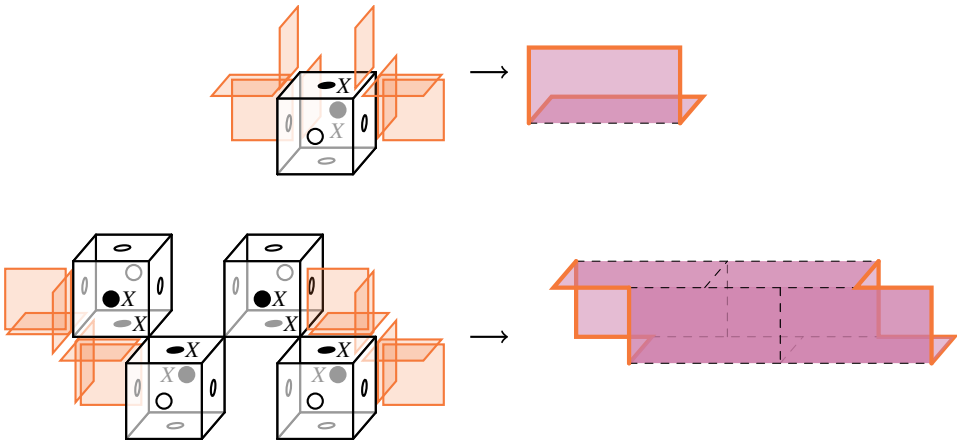


Figure 5.12: (Upper) A single code edge operator along the x direction and maps to a folded surface. (Lower) An open string of edge operators maps to an open ended tube-like surface. The syndromes of the Paulis and the boundaries of the surfaces are highlighted in red.

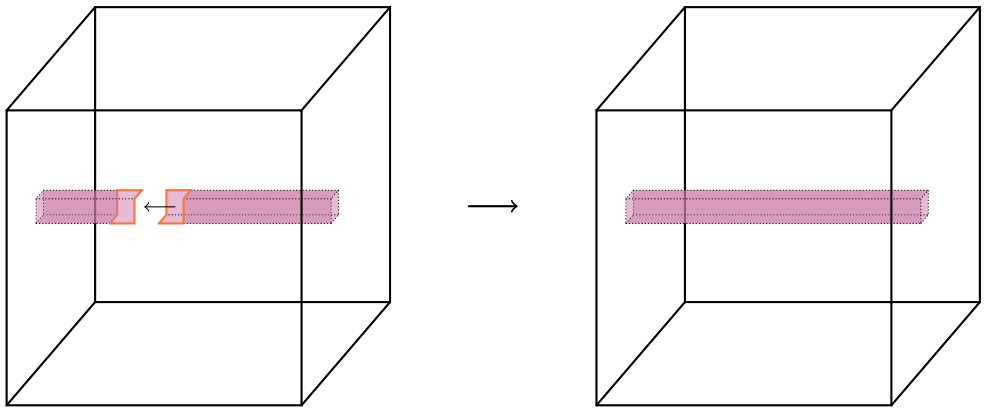


Figure 5.13: A logical $\tilde{Z}_x$ operator is formed by connecting a tube which loops round the torus.

housing the code must be of even length in each cardinal direction. This means that if a straight tube loops back into itself the boundaries at each end will line up, closing the surface and so the Pauli operator which induced it has no syndrome (see fig. 5.13).

Their absence from the stabilizer can also be understood in the surface picture. Products of stabilizers are also closed surfaces but they must be built out of $2 \times 2 \times 1$ blocks implying that their total cross sectional area in every cell layer of the lattice must be even. The cross sectional area of a logical $\tilde{Z}_x$ corresponding to a straight tube spanning the x direction has a cross sectional area of 1 in every cell layer oriented in the x direction meaning that it cannot be constructed by stabilizers. By contrast, a

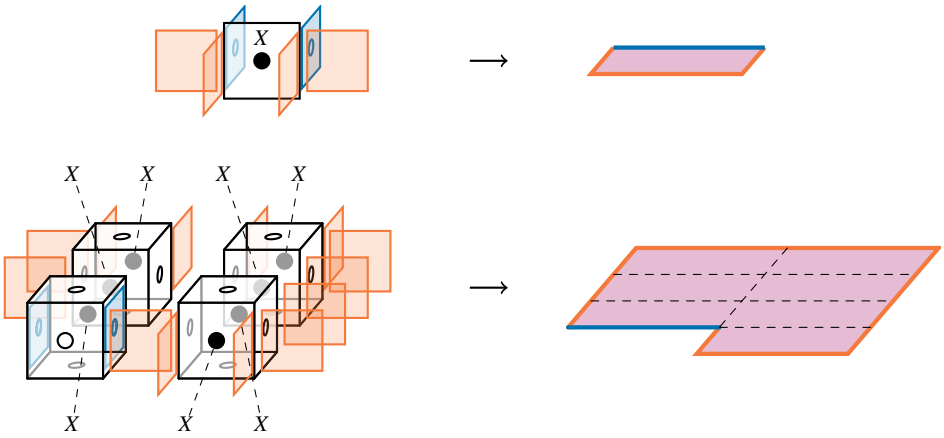


Figure 5.14: (Upper) A single Pauli error which induces a domino surface element. (Lower) Multiple Pauli errors which induce a tiled surface of dominos. The syndromes of the errors and the corresponding boundaries of the surfaces are highlighted in red for even faces/edges and blue for odd faces/edges.

straight, looping tube of cross section 2 at each of these layers can be constructed via stabilizers but simply corresponds to $\tilde{Z}_x^2 = 1$.

To find logical $\tilde{X}$ operators, one needs to find another surface which cannot be contracted to a point via the deformations induced by the stabilizers. Consider an odd cell. An X operator on one of its 2 faces oriented in the y direction induces a domino surface on the dual lattice oriented in the z direction. These dominos can be tiled to form larger flat surfaces (see fig. 5.14). If an X acts on every on face facing the y direction in a layer of cells across the xy-plane then the induced surface will completely span the corresponding layer of faces on the dual lattice (see fig. 5.15). This operator has no syndrome because the surface has no boundary and it cannot be contracted to a point even by arbitrary continuous deformations so it clearly is not in the stabilizer. This also implies that it is distinct from $\tilde{Z}_x$, $\tilde{Z}_y$, $\tilde{Z}_z$, since it cannot be deformed into the tubes which form these operators as they would be contractible under such unrestricted action. Similar surfaces can be defined parallel to all 3 cardinal planes of the lattice. Define the logical operator $\tilde{X}_i$ as the Pauli operator which induces the, flat, unbounded surface oriented in the direction of the $\tilde{Z}_i$'s induced tube (up to action by stabilizers).

To find the code distance of the underlying code on the 3-torus one must consider the minimum weights of the logical operators

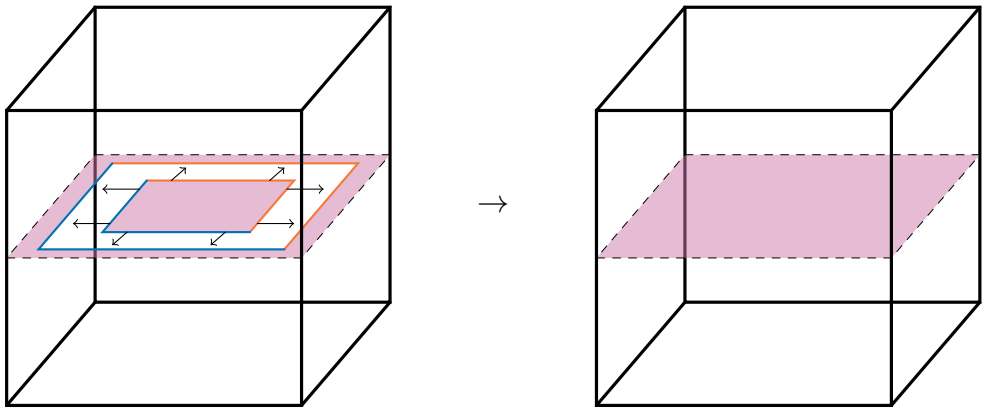


Figure 5.15: A logical $\tilde{X}$ operator is formed by spreading a surface out across a whole plane of the torus.

Lemma 19. *Given an instance of the underlying code on a 3-torus of lengths L_x, L_y, L_z in each cardinal direction, the minimum weight $w(\cdot)$ of logical operator $\tilde{Z}_i$ is given by*

$$w(\tilde{Z}_i) = 2L_i, \tag{5.29}$$

where $i \in \{x, y, z\}$.

Proof. Logical $\tilde{Z}$ operators are non-contractible loops of code edge operators. Consider the operator $\tilde{Z}_z$, a cycle of code edge operators forming this operator will pass through every layer of cells in the lattice parallel to the xy plane an odd number of times, this means it will have an odd number of z-oriented edges in each of these layers. Therefore in each layer, there will be at least one odd cell with an odd number of its z-oriented edges present in the cycle. In the Pauli picture, the product of an odd number of edge operators on a single odd cell results in a weight-2 operator (see fig. 5.16) whose weight cannot be reduced by edge operators in different directions so therefore, any instance of $\tilde{Z}_z$ must be at least weight $2L_z$. This lower bound is saturated by the operator corresponding to a straight line of edge operators looping through the 3-torus in the z direction.

The above argument also applies to $\tilde{Z}_x$ and $\tilde{Z}_y$. □

Lemma 20. *Given an instance of the underlying code on a 3-torus of lengths*

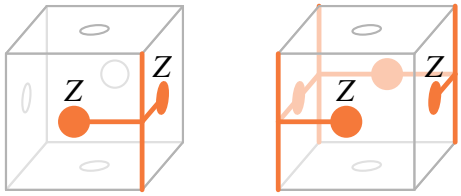


Figure 5.16: For any odd cell, a product of an odd number of its associated z-oriented code edge operators will have weight 2 on the cell's qubits. This weight cannot be reduced by multiplying by x or y-oriented edge operators.

L_x, L_y, L_z in each cardinal direction, the minimum weight $w(\cdot)$ of logical operator $\tilde{X}_i$ is given by

$$w(\tilde{X}_i) = \frac{L_j L_k}{2}, \tag{5.30}$$

where $i, j, k \in \{x, y, z\}$ and $i \neq j \neq k$.

Proof. A logical $\tilde{X}_z$ operator must map to a surface such that any straight line drawn through the 3-torus in the z direction must pass through it. The smallest surface to satisfy this is the completely flat plane perpendicular to the z direction which has area $L_x L_y$. Any deformation of this surface will have at least this surface area so therefore any encoding of a logical $\tilde{X}_z$ operator must correspond to a surface with at least this area, furthermore it must have at least this area of surface perpendicular to the z direction. Every single qubit Pauli operator induces a surface of area at most 2 on cells oriented in the z direction so a surface with area $L_x L_y$ oriented that way must be induced by a Pauli operator of at least weight $L_x L_y/2$. This lower bound is saturated by the $\tilde{X}_z$ which induces a flat surface.

The above argument applies equally to $\tilde{X}_x$ and $\tilde{X}_y$. □

Theorem 21. *Given an instance of the underlying code on a 3-torus of size $L_z \leq L_y \leq L_x$ in each cardinal direction, its code distance is*

$$d_C = \begin{cases} 2, & \text{if } L_z = L_y = 2, \\ 2L_z, & \text{otherwise.} \end{cases} \tag{5.31}$$

Proof. The code distance is the weight of the lowest weight logical Pauli on the code,

i.e. the lowest weight of a product of $\tilde{Z}_x$, $\tilde{Z}_y$, $\tilde{Z}_z$, $\tilde{X}_x$, $\tilde{X}_y$, $\tilde{X}_z$. Any product involving an $\tilde{X}_i$ is subject to the lower bound from lemma 20 as the area of the i-oriented component of the induced surface cannot be made lower than $L_j L_k$. Any product of $\tilde{Z}$ operators is subject to the lower bound in lemma 19 for any involved $\tilde{Z}_i$ as the Pauli must be a product of code edge operators with at least one i-oriented edge in every layer of the lattice perpendicular to the i direction. The lowest weight of a logical Pauli is then the lowest weight of a single logical $\tilde{X}_i$ or $\tilde{Z}_i$.

For a lattice of lengths $L_z \leq L_y \leq L_x$, the lowest weights of $\tilde{X}_i$ and $\tilde{Z}_i$ operators are

$$w(\tilde{Z}_z) = 2L_z, \quad w(\tilde{X}_x) = \frac{L_y L_z}{2}. \tag{5.32}$$

In the case where $L_z = L_y = 2$, these weights are $w(\tilde{X}_x) = 2 < w(\tilde{Z}_z) = 4$ giving a code distance of 2. For lattices larger in these dimensions $w(\tilde{Z}_z) < w(\tilde{X}_x)$ giving a code distance of $2L_z$. □

5.2.3 Open Boundary Conditions

By theorem 18 the only cases of the code on an open cubic lattice with finite codespace are those with 4 and 8 odd corners, encoding 1 and 3 logical qubits respectively. One may expect the logical operators of these versions of the code to have similar forms to those in the 3-torus version. In fact one finds that all logical operators in the open lattice cases can be constructed from strings of code edge operators like the logical $\tilde{Z}_i$ operators on a 3-torus.

Consider an open string of edge operators as illustrated in fig. 5.12. Such an operator only excites the stabilizers associated with each even face that contains the end vertices. An odd corner vertex is part of no even faces so it then follows that a string operator between 2 odd corner vertices will have no syndrome meaning they are either stabilizers or logical operators. As all stabilizers are closed loops within the lattice, a path of edge operators between two lattice corners cannot be constructed from only stabilizers implying that such a path is a logical operator.

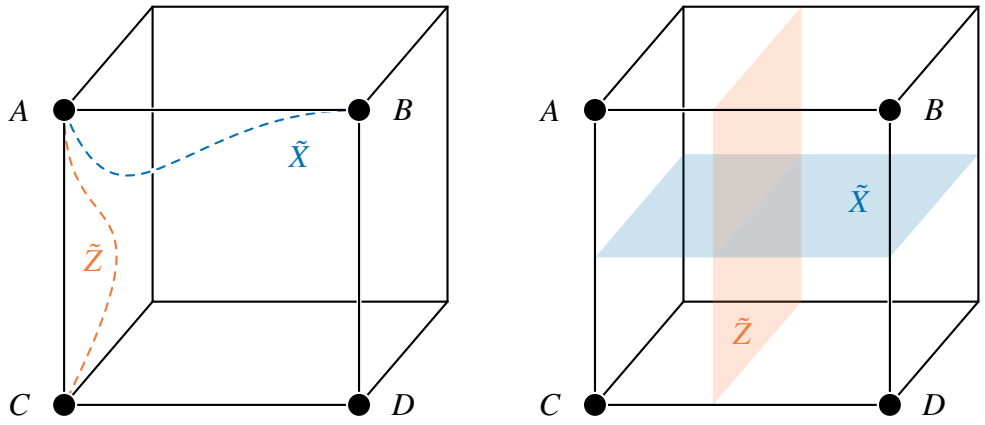


Figure 5.17: Illustrations of logical operators on the underlying code on an open lattice with 4 odd corners. (Left) Strings of edge operators between the odd corners form logical operators. (Right) Pauli operators that induce surfaces across the whole lattice form equivalent logical operators.

5.2.3.1 4 Odd Corners

Consider an open lattice instance of the code with 4 odd corners (they must all be on one side of the lattice as shown in fig. 3.18) and label the odd corners A, B, C, D as in fig. 5.17. Consider strings of edge operators S_{ij} that connect corners i and j. The strings S_{AB} and S_{AC} commute with the stabilizer and anticommute with one another, meaning that they must be non-trivial logical operators. Furthermore, because the 4 odd corner case only encodes a single qubit then one may identify

$$\begin{aligned} S_{AB} &:= \tilde{X} \\ S_{AC} &:= \tilde{Z} \\ S_{BC} &:= \tilde{Y}. \end{aligned} \tag{5.33}$$

Other strings connecting odd corners are equivalent to the above logical Paulis under stabilizers with $S_{BD} = \tilde{Z}$, $S_{CD} = \tilde{X}$ and $S_{AD} = \tilde{Y}$.

One might wonder whether flat surface operators like the 3-torus logical $\tilde{X}$ operators exist in this case. In fact, they do and are also logical Paulis on the encoded qubit. Say that the lattice edges AB and AC are oriented in the x and z directions then the flat surface operator oriented in the x direction (with boundaries on the lattice faces oriented in the y and z directions) encodes a logical $\tilde{X}$ with a similar surface representation for $\tilde{Z}$. See fig. 5.17 for illustration. These equivalences are

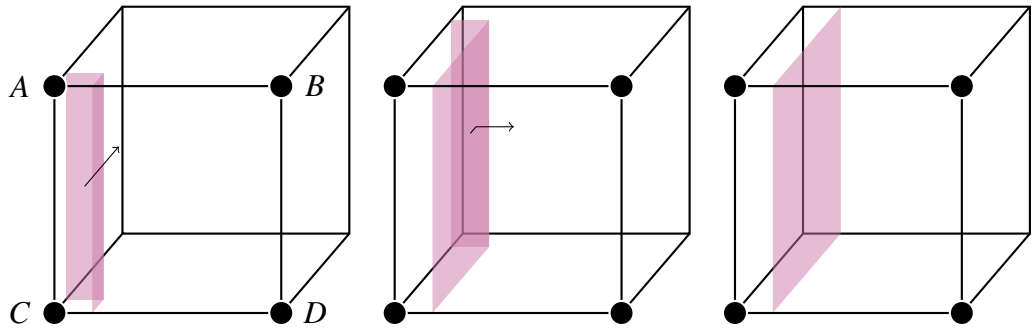


Figure 5.18: Illustration of the equivalence between logical operators in string and surface form. A string of edge operators from corner A to C is a logical $\tilde{Z}$ operator and in the surface picture, corresponds to the folded surface shown on the left. Through multiplication by stabilizer operators this surface can be extruded out into a flat surface which represents the same logical operator on the codespace.

clearer when the operators are viewed in the surface picture. A logical $\tilde{Z}$ operator encoded by a string of edge operators down the boundary edge between corners A and C induces an open surface rather than a tube due to the truncation of the lattice and the dual lattice, this surface can then be "stretched out" by stabilizers to form a flat surface. As this deformation is achieved only via stabilizers, the two surfaces correspond to the same logical operator (see fig. 5.18).

Theorem 22. *For an instance of the code underlying the cubic fermionic encoding on an open lattice with 4 odd corners, side lengths $L_z \leq L_x$ and L_y in the cardinal directions and where L_x, L_z are odd and L_y is even, the code distance is*

$$d_C = L_z + 1 \tag{5.34}$$

Proof. Consider L_z layers of cells orthogonal to the z direction. Denote these layers as odd or even in an alternating manner such that the first layer is odd, there are then $(L_z+1)/2$ odd layers (see fig. 5.19).

Define a logical $\tilde{X}$ on the code to be an operator that induces a flat surface orthogonal to the z direction. These surfaces can only be induced by operators acting only on qubits within a single odd layer of the lattice. On any given odd cell in this layer, the $\tilde{X}$ acts with X on 2 faces oriented in the y direction *or* with Y on the 2 faces oriented in the x direction (these are the two ways to ways to make the same

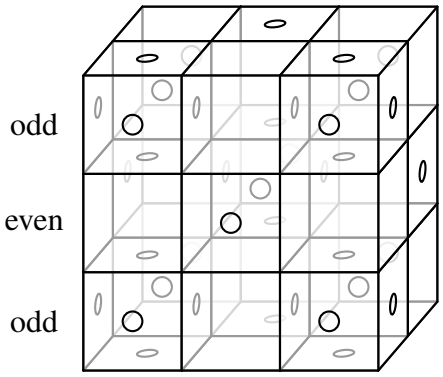


Figure 5.19: Odd and even cell layers perpendicular to the z direction on an open lattice with 4 odd corners. In this case $L_x = L_z = 3$ and $L_y = 2$.

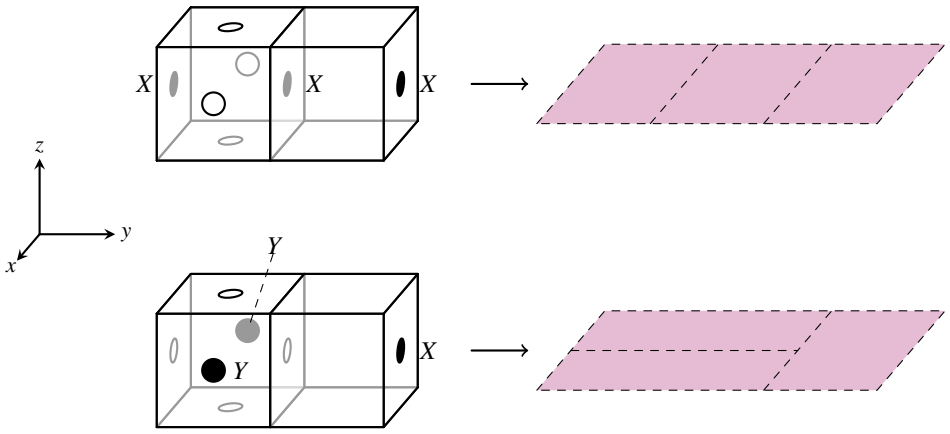


Figure 5.20: Two operators that induce the same flat surface in a $1 \times 2 \times 1$ lattice with 4 odd corners, note that the action on odd cells can be changed while inducing the same surface. Both these operators are the same logical operator on this instance of the code as they are equivalent up to a stabilizer. Orientation has been changed for this diagram such that it is consistent with the text while still being easy to interpret.

flat square surface) and on any given isolated odd face on the boundary it acts with a X (as all isolated odd faces in the layer will be y-oriented, see fig. 5.20 at the back of the lattice).

To anticommute with the flat surface $\tilde{X}$ operator in any odd layer, a logical $\tilde{Z}$ operator must, in each odd layer, include an X or Z on an x-oriented face *or* a Y or Z on a y-oriented face lower bounding its weight by $(L_z+1)/2$. These Pauli operators induce a total surface comprising one face parallel to the z direction in every cell layer of the dual lattice. This surface is bounded by edges in the bulk of the dual lattice and therefore an operator composed only of these Paulis has a syndrome,

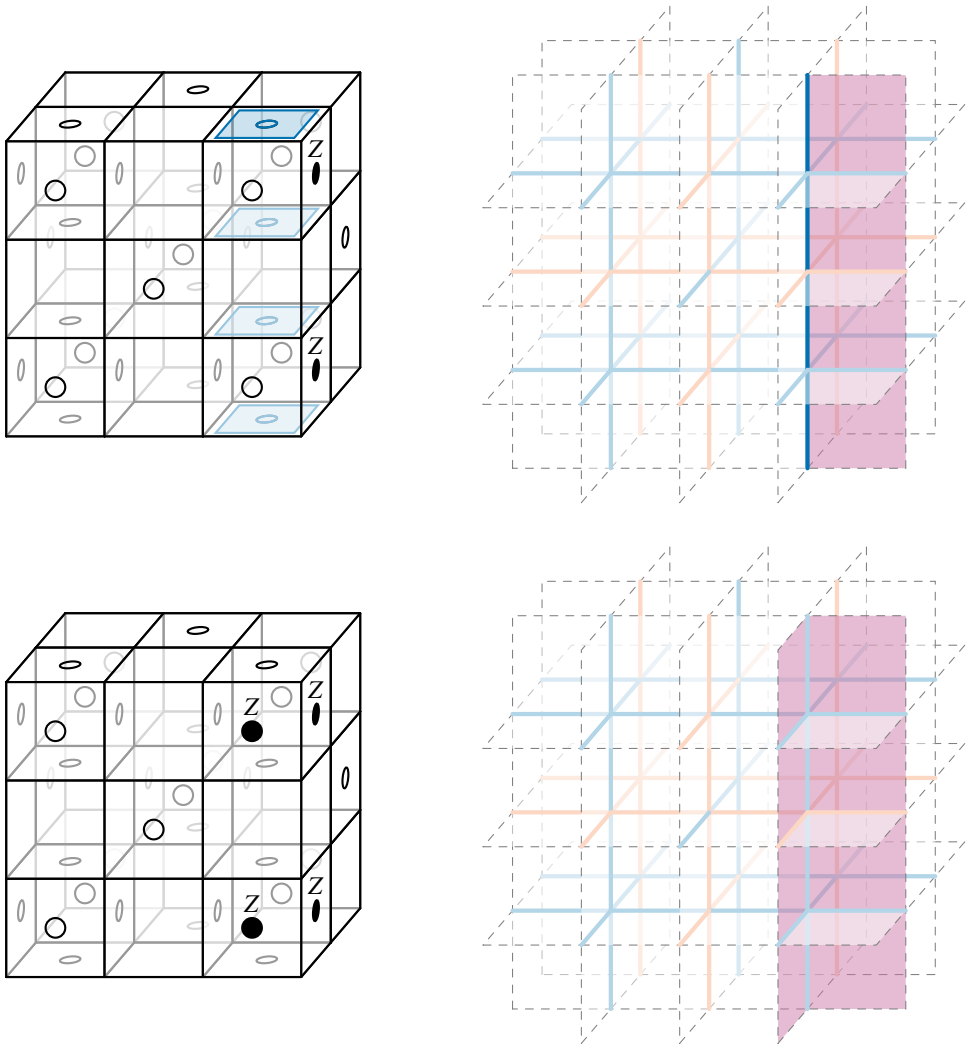


Figure 5.21: (Upper) A Pauli inducing a surface consisting of one face parallel to the z direction in every layer of cells in the dual lattice, it has a syndrome corresponding to an open boundary parallel to z of length $(2L_z+1)/2$ which must be cancelled out by an operator of at least weight $(2L_z+1)/2$ (lower).

meaning that a logical $\tilde{Z}$ must include more Paulis to cancel this syndrome. To this end, the remaining part of $\tilde{Z}$ must also include Paulis (on a disjoint set of qubits) which induce a surface including at least one face parallel to z in every cell. The most efficient way to form such a surface from domino surface elements is with $(L_z+1)/2$ dominos with long sides along the z direction meaning that the remaining component of $\tilde{Z}$ must include at least $(L_z+1)/2$ single Paulis (see fig. 5.21 for an illustrated example). This tightens the lower bound on the weight of $\tilde{Z}$ to L_z+1.

The above argument applies equally to the weight of $\tilde{Y}$ as it must also anticom-

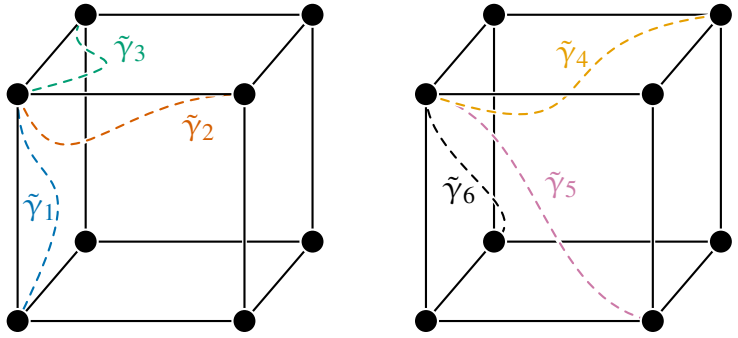


Figure 5.22: For the underlying code on an open lattice with 8 odd corners, strings of code edge operators connecting corners as shown correspond to 6 mutually anticommuting logical operators. These can be interpreted as anticommuting Paulis or as a Majorana basis on the codespace.

mute with any $\tilde{X}$ and the same argument up to a reshuffle in Paulis and cardinal directions lower bounds the weight of $\tilde{X}$ to $L_x + 1 \geq L_z + 1$.

The lower bound of $L_z + 1$ is saturated by the operator formed from a product of edge operators running directly from one odd corner to its neighbour in the z direction so therefore the code distance is $L_z + 1$. □

5.2.3.2 8 Odd Corners

In the 8 odd corner case, 3 qubits are encoded and again, a full set of logical operators can be encoded with only strings of code edge operators. Choosing a privileged corner, the 7 string operators that connect it to each other corner correspond to a set of 7 anticommuting Paulis which form a complete generating set for the Pauli group on 3 qubits [64]. One may also consider these strings to represent 6 Majorana operators with the 7th representing the product of all 6, see fig. 5.22 for illustration.

Surface operators also exist in the 8 odd corner case but they make the most sense when seeking out logical $\tilde{X}$ and $\tilde{Z}$ operators for the 3 encoded qubits. One can define 3 commuting logical Paulis with strings connecting 3 disjoint pairs of corners, each pair sharing a lattice boundary edge in the x, y and z direction and label these $\tilde{Z}_x$, $\tilde{Z}_y$ and $\tilde{Z}_z$. Logical $\tilde{X}$ operators can then be defined as the surfaces oriented in the same direction as the corresponding $\tilde{Z}$ strings, see fig. 5.23 for illustration.

Theorem 23. *For an instance of the code underlying the cubic fermionic encoding on an open lattice with 8 odd corners, with side lengths $L_z \leq L_y \leq L_x$ in the cardinal*

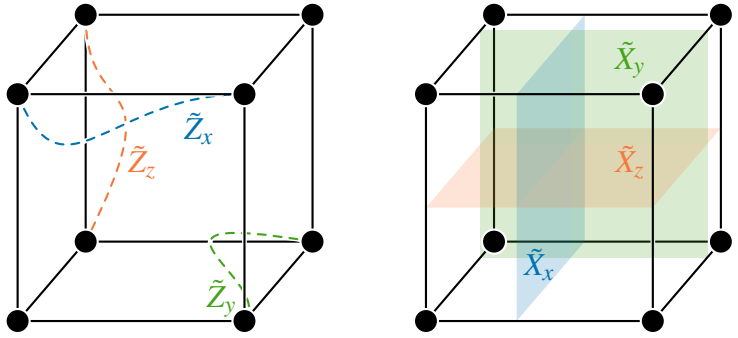


Figure 5.23: Other logical operator representations on an open lattice with 8 odd corners. (Left) Strings of code edge operators connecting odd corners as shown correspond to commuting logical operators, these can be chosen as logical $\tilde{Z}$'s on the 3 encoded qubits. (Right) Operators which induce these surfaces spanning the lattice also correspond to commuting logical Paulis which can be chosen as the logical $\tilde{X}$'s on each encoded qubit. Strings and surfaces of the same colour denote operators on the same encoded qubit.

directions, the code distance is

$$d_C = L_z + 1 \tag{5.35}$$

Proof. Any logical Pauli is a product of code edge operators forming closed loops and at least one string connecting 2 odd corners which must pass through at least L_z layers of cells in some direction. Each odd corner can be considered to be connected to, at most, one other corner without loss of generality (the product of a pair of paths connecting corners A to B and A to C is equivalent to one path connecting B to C). A logical Pauli anticommutes with another if its path components share an odd number of odd corners. A product of paths incident on all odd corners an odd number of times encodes the identity operator on the code space, this can be checked by considering a "Majorana basis" for logical operators as illustrated in fig. 5.22.

Label the four odd corners A, B, C, D, E, F, G and H. Consider a logical Pauli operator P of fixed form whose edge product includes (w.l.o.g.) a single path of edge operators between odd corner pairs AB denoted S_{AB}, up to 2 other paths between 2 other disjoint pairs (fix these to be CD and EF) and any closed loops of edge operators on a disjoint set of edges to the paths, denoted L, so $P = S_{AB} \cdot S_{CD} \cdot S_{EF} \cdot L$. Consider a logical operator $Q = S_{AG}$, i.e. a path of edge operators between corners

A and G. This operator will commute with all components of P except the path between corner pair AB with which it must anticommute.

Via multiplication by stabilizers, Q may be deformed into an arbitrary path between odd corners A and G. It may be deformed such that it is identical to the S_{AB} component of P from odd corner A up to an arbitrary vertex v along the length of S_{AB} where the strings diverge. The anticommutation between this deformed Q and S_{AB} must occur on the edge operators of S_{AB} incident at v. All edges before this pair are shared exactly with the deformed Q and will commute and all edges after share no vertices with Q and will commute (edge operators anticommute only if they share a single vertex).

The vertex v at which divergence occurs can be at any point along the length of S_{AB} so P must have support on every odd cell / isolated odd face that S_{AB} passes through (v can be chosen such that anticommutation occurs between edge operators on any odd cell / isolated odd face). As P is a product of code edge operators, the minimum support it may have on an odd cell is 2, since S_{AB} must pass through at least L_z layers of cells, it must pass through at least $(L_z+1)/2$ odd cells (no isolated odd faces exist in $\geq (L_z+1)/2$ cell layers of the lattice in any direction), this lower bounds the weight of any logical Pauli by (L_z+1).

This bound is saturated by the logical operator composed of a product of edge operators forming a straight line from one odd corner to its neighbour in the z direction. □

5.3 Discussion

The underlying code encodes 3 qubits when defined on a 3-torus and 1 or 3 qubits when defined on an open lattice. Its code distance scales linearly with its lattice's shortest side length and therefore scales as $\Theta(N^{1/3})$ with its total number of qubits N. This distance is outperformed by other LDPC codes, in particular an instance of the "fiber bundle codes" detailed in [79] have minimum distance $\Theta(N^{3/5})$ up to polylogarithmic factors, while the underlying code's performance is more in line with the 3D toric code in this regard. That said, the 3D toric code on an open lattice

(3D surface code) only encodes a single qubit while the underlying code encodes up to 3.

While its basic performance as an error correcting code is not the best the underlying code is not entirely uninteresting. The 3 dimensional dependence of its stabilizer operators is suggestive of the XYZ product code construction given in [70] (which generalises the structure detailed by Chamon in [67]), but it does not immediately fit into the formalism. If the underlying code cannot be described by this then perhaps another general construction or even a more general version of the XYZ construction can be found which captures it.

The fact that the excitations of this code are all string-like may remind some readers of the 4D toric code [60], whose excitations are also entirely string-like. The 4D toric code behaves as a self correcting quantum memory because its string like excitations scale with the weight of the minimal associated Pauli error (i.e. the smallest surface the string bounds). This is not the case in the underlying code due to the rigidity of the surface operators that form the string excitations. For instance, consider a pair of excitations associated with the end of a tube operator (as illustrated in fig. 5.12). This pair can be separated arbitrarily while their total size remains constant, however the correction required to resolve their syndrome scales linearly with the separation. This rigidity may still yield benefits in decoding, as it would reduce the number of possible ways to resolve syndromes.

The potential advantages in decoding do not end there. Pauli errors of a single type give rise to surfaces and tubes of specific orientation. For example, tube operator on the underlying code running along the z direction is composed entirely of Z errors. This directional bias of Pauli errors is reminiscent of the behaviour of the $XZZX$ surface code, the high performance of which against biased Pauli noise channels was shown in [80].

Also worth noting is that the excitations at each end of a tube operator are fermionic in character by the criterion established by Levin and Wen in [46]. Looking back to the cubic fermionic encoding from which the underlying code emerges, one sees that, much like the 2D case, a string of encoded fermionic edge operators creates

a pair of fermions on the auxiliary qubit system with the energy cost removed by the stabilizer support on the vertex qubits. This is yet another fermionic encoding which appears to work by utilising fermionic excitations on an underlying topologically ordered system.

Chapter 6

Concluding Remarks and Outlook

This thesis has presented a novel fermion to qubit mapping with a number of interesting properties. It is shown to outperform all existing fermionic encodings in both qubit to mode ratio and in the weight of encoded local 2-mode interactions when simulating fermionic systems on various lattice types. These are two important metrics for the near term quantum simulation of fermionic systems such as the 2D Fermi-Hubbard Model, a system which is classically intractable for even small system sizes outside of the half-filled regime. As well as this, low weight undetectable errors on the square and cubic lattice encodings were shown to map onto natural noise for a fermionic system, a feature which may be useful if they are to be used on NISQ devices with limited error correcting facilities.

Also presented were some theoretical results that apply more generally to fermionic encodings, in particular those relating to the size of the codespace of fermionic encodings. These proved useful when analysing the variants of the compact encoding introduced in this work. They may also find use in the analysis of any new encodings discovered in the future.

Inspection of the compact encoding on a square lattice revealed an unexpected direct link to the toric code. The loop stabilizers of the encoding are in fact square lattice toric code stabilizers tensored with local parity check operators which allow the fermionic ε pair excitations of the toric code to exist in the ground state. The compact encoding on other 2D lattice structures also display similar links to other toric code variations and in some cases the colour code. The cubic lattice case

however was found to contain a 3D topological code which appears to be novel. The codespace and code distance of this code were found exactly for both periodic and open boundary conditions. While its code distance is similar to that of the 3D toric code its unusual structure may yield benefits for decoding algorithms.

There are two avenues for future work motivated by the content of this thesis. One would be to test instances of the compact encoding on some of the many NISQ devices available such as IBM or Rigetti's hardware. Such experiments would gauge how well the results of this thesis play out in real life and perhaps inform any modifications one may need to make to the encoding to improve real world performance.

Another interesting line of enquiry would be to further explore the connection between local fermionic encodings and topological models with fermion-like excitations. As well as the compact encoding from this work, numerous other fermionic encodings display similar links. The BKSF encoding [12] on a square lattice is effectively a toric code system with the reduced stabilizer generated by only pairs of adjacent plaquette and star operators. Similarly to the square lattice compact encoding, this allows the fermionic ε excitation of the toric code to exist in the ground state and strings of encoded fermionic edge operators are equivalent to pairs of these. The VC encoding [13, 14] is more similar still to the compact encoding; an appropriately chosen generator shows that its stabilizers are also in fact toric code stabilizers combined with local parity checks. The difference here is that the VC encoding uses a toric code rotated by $\pi/4$. A square lattice instance of the code due to Setia et al [16] can be found such that it is equivalent to a colour code system on a hexagonal lattice with a reduced stabilizer, similar to the BKSF with the toric code. It can then be shown that the excitations allowed in the ground space of this system are also fermionic and are in fact equivalent to the ε excitations of one of the toric codes underlying the color code [81, 66]. The author conjectures that similar links to topological models can be found in the other local fermionic encodings and that these underlying structures may be the thing that links the seemingly separate approaches for representing fermionic systems on qubits. A unified picture

of fermionic encodings may prove useful in finding new encodings, for instance, protocols tailored to specific fermionic systems or hardware.

Contribution Statement

The research direction of studying fermionic encodings came from my supervisor Toby Cubitt. The introduction and conclusion are written by me. A statement of contribution to the main body chapters is given below.

Chapter 2: The initial construction of the encoding came from Joel Klassen after discussions together. Further details including the edge sign fixing and behaviour under different lattice side lengths are my work. The connection to toric code was first noticed by Joel Klassen, technical details of this are joint work between him and myself.

Chapter 3: The discussion of local encodings on graphs in the first section is joint work with contributions shared between myself and Joel Klassen, except for the results on particle species which are my own work. The results on generalising the 2D compact encoding are my own work. The examples of encodings on different lattice geometries came from discussions with Joel but the proofs of their more detailed properties are my work. The construction of the 3D encoding is joint work from me and Joel Klassen, the lemma on stabilizer generator is my work, the theorem on the disparity which follows from it is from Joel.

Chapter 4: The initial idea of investigating how errors map through fermionic encodings came from my supervisor Toby Cubitt. The initial derivation of fermionic phase noise came from Joel but was refined by me into the form in this document. The results on error mapping in the compact encoding are my own work.

Chapter 5: The idea for this chapter was inspired by Joel's observation of the toric code in the compact encoding on a square lattice but all content is my own work.

Bibliography

[1] Richard P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21(6), 1982.

[2] Seth Lloyd. Universal quantum simulators. *Science*, 273(5278):1073–1078, 1996.

[3] Dorit Aharonov and Amnon Ta-Shma. Adiabatic quantum state generation and statistical zero knowledge. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 20–29, 2003.

[4] Andrew M. Childs. On the relationship between continuous-and discrete-time quantum walk. *Communications in Mathematical Physics*, 294(2):581–603, 2010.

[5] Dominic W. Berry, Andrew M. Childs, and Robin Kothari. Hamiltonian simulation with nearly optimal dependence on all parameters. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 792–809. IEEE, 2015.

[6] Guang Hao Low and Isaac L. Chuang. Optimal hamiltonian simulation by quantum signal processing. *Phys. Rev. Lett.*, 118:010501, 2017.

[7] Guang Hao Low and Isaac L. Chuang. Hamiltonian simulation by qubitization. *Quantum*, 3:163, 2019.

[8] Andrew M. Childs, Dmitri Maslov, Yunseong Nam, Neil J. Ross, and Yuan Su. Toward the first quantum simulation with quantum speedup. *Proceedings of the National Academy of Sciences*, 115(38):9456–9461, 2018.

[9] Eugene P. Wigner and Pascual Jordan. Über das paulische äquivale-nzverbot. *Z. Phys.*, 47:631, 1928.

[10] Daniel S. Abrams and Seth Lloyd. Simulation of many-body fermi systems on a universal quantum computer. *Physical Review Letters*, 79(13):2586, 1997.

[11] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

[12] Sergey B. Bravyi and Alexei Kitaev. Fermionic quantum computation. *Ann. Phys.*, 298(1):210–226, 2002.

[13] Frank Verstraete and J. Ignacio Cirac. Mapping local hamiltonians of fermions to local hamiltonians of spins. *J. Stat. Mech. Theory E.*, 2005(09):P09012, 2005.

[14] James D. Whitfield, Vojtěch Havlíček, and Matthias Troyer. Local spin operators for fermion simulations. *Phys. Rev. A*, 94:030301, 2016.

[15] Zhang Jiang, Jarrod McClean, Ryan Babbush, and Hartmut Neven. Majorana loop stabilizer codes for error mitigation in fermionic quantum simulations. *Phys. Rev. Applied*, 12:064041, 2019.

[16] Kanav Setia, Sergey Bravyi, Antonio Mezzacapo, and James D. Whitfield. Superfast encodings for fermionic quantum simulation. *Phys. Rev. Research*, 1(3):033033, 2019.

[17] Mark Steudtner and Stephanie Wehner. Quantum codes for quantum simulation of fermions on a square lattice of qubits. *Phys. Rev. A*, 99:022308, 2019.

[18] Alexei Kitaev. Fault-tolerant quantum computation by anyons. *Ann. Phys.*, 303(1):2 – 30, 2003.

[19] Tom Banks. *Modern quantum field theory: a concise introduction*. Cambridge University Press, Cambridge, United Kingdom, 2008.

[20] Eva Pavarini, Erik Koch, and (eds.). *Simulating Correlations on Computers*, volume 11 of *Modelling and Simulation*. Verlag des Forschungszentrum Jülich, 2021.

[21] Gian-Carlo Wick, Arthur S. Wightman, and Eugene P. Wigner. The intrinsic parity of elementary particles. *Phys. Rev.*, 88:101–105, 1952.

[22] Gerhard C. Hegerfeldt, Kamil Kraus, and Eugene P. Wigner. Proof of the fermion superselection rule without the assumption of time-reversal invariance. *Journal of Mathematical Physics*, 9(12):2029–2031, 1968.

[23] John Hubbard. Electron correlations in narrow energy bands. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 276(1365):238–257, 1963.

[24] Elbio Dagotto. Correlated electrons in high-temperature superconductors. *Rev. Mod. Phys.*, 66:763–840, 1994.

[25] Zhang Jiang, Kevin J. Sung, Kostyantyn Kechedzhi, Vadim N. Smelyanskiy, and Sergio Boixo. Quantum algorithms to simulate many-body physics of correlated fermions. *Physical Review Applied*, 9(4):044036, 2018.

[26] John Bardeen, Leon N. Cooper, and John R. Schrieffer. Microscopic theory of superconductivity. *Phys. Rev.*, 106:162–164, 1957.

[27] Leticia Tarruell and Laurent Sanchez-Palencia. Quantum simulation of the hubbard model with ultracold fermions in optical lattices. *Comptes Rendus Physique*, 19(6):365–393, 2018.

[28] Elliott H. Lieb and Fa-Yueh Wu. Absence of mott transition in an exact solution of the short-range, one-band model in one dimension. *Phys. Rev. Lett.*, 20:1445–1448, 1968.

[29] Chunjing Jia, Brian Moritz, Cheng-Chien Chen, B. Sriram Shastry, and Thomas P. Devereaux. Fidelity study of the superconducting phase diagram

in the two-dimensional single-band hubbard model. *Phys. Rev. B*, 84:125113, 2011.

[30] Susumu Yamada, Toshiyuki Imamura, and Masahiko Machida. 16.447 tflops and 159-billion-dimensional exact-diagonalization for trapped fermion-hubbard model on the earth simulator. In *SC'05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, pages 44–44. IEEE, 2005.

[31] Richard Blankenbecler, Douglas J. Scalapino, and Robert L. Sugar. Monte carlo calculations of coupled boson-fermion systems. i. *Phys. Rev. D*, 24:2278–2286, 1981.

[32] E. Y. Loh, James E. Gubernatis, Richard T. Scalettar, S. R. White, Douglas J. Scalapino, and Robert L. Sugar. Sign problem in the numerical simulation of many-electron systems. *Phys. Rev. B*, 41:9301–9307, 1990.

[33] Raimundo R. dos Santos. Introduction to quantum monte carlo simulations for fermionic systems. *Brazilian Journal of Physics*, 33:36–54, 2003.

[34] Matthias Troyer and Uwe-Jens Wiese. Computational complexity and fundamental limitations to fermionic quantum monte carlo simulations. *Phys. Rev. Lett.*, 94:170201, 2005.

[35] Zi-Xiang Li and Hong Yao. Sign-problem-free fermionic quantum monte carlo: Developments and applications. *Annual Review of Condensed Matter Physics*, 10:337–356, 2019.

[36] Dave Wecker, Matthew B. Hastings, Nathan Wiebe, Bryan K. Clark, Chetan Nayak, and Matthias Troyer. Solving strongly correlated electron models on a quantum computer. *Phys. Rev. A*, 92:062318, 2015.

[37] Daniel S. Abrams and Seth Lloyd. Quantum algorithm providing exponential speed increase for finding eigenvalues and eigenvectors. *Phys. Rev. Lett.*, 83:5162–5165, 1999.

[38] Alexei Kitaev. Quantum measurements and the abelian stabilizer problem. *arXiv preprint quant-ph/9511026*, 1995.

[39] Euan Allen. Efficient quantum simulation. In *Advanced Quantum Information Theory Student Essays*. Bristol University, 2015.

[40] Dominic W. Berry, Graeme Ahokas, Richard Cleve, and Barry C. Sanders. Efficient quantum algorithms for simulating sparse hamiltonians. *Communications in Mathematical Physics*, 270(2):359–371, 2007.

[41] Andrew M. Childs and Nathan Wiebe. Hamiltonian simulation using linear combinations of unitary operations. *arXiv preprint arXiv:1202.5822*, 2012.

[42] Guang Hao Low, Theodore J. Yoder, and Isaac L. Chuang. Methodology of resonant equiangular composite quantum gates. *Physical Review X*, 6(4):041067, 2016.

[43] Daniel Gottesman. Class of quantum error-correcting codes saturating the quantum hamming bound. *Phys. Rev. A*, 54:1862–1868, 1996.

[44] Emanuel Knill, Raymond Laflamme, and Wojciech H. Zurek. Resilient quantum computation. *Science*, 279(5349):342–345, 1998.

[45] Dorit Aharonov and Michael Ben-Or. Fault-tolerant quantum computation with constant error rate. *SIAM Journal on Computing*, 2008.

[46] Michael Levin and Xiao-Gang Wen. Fermions, strings, and gauge fields in lattice spin models. *Phys. Rev. B*, 67(24):245316, 2003.

[47] Sergey Bravyi, Jay M. Gambetta, Antonio Mezzacapo, and Kristan Temme. Tapering off qubits to simulate fermionic hamiltonians. *arXiv preprint arXiv:1701.08213*, 2017.

[48] Mark Steudtner and Stephanie Wehner. Fermion-to-qubit mappings with varying resource requirements for quantum simulation. *New Journal of Physics*, 20(6):063010, 2018.

[49] William Kirby, Bryce Fuller, Charles Hadfield, and Antonio Mezzacapo. Second-quantized fermionic hamiltonians for quantum simulation with poly-logarithmic qubit and gate complexity, 2021.

[50] Chris Cade, Lana Mineh, Ashley Montanaro, and Stasja Stanisic. Strategies for solving the fermi-hubbard model on near-term quantum computers. *Physical Review B*, 102(23):235122, 2020.

[51] Vojtěch Havlíček, Matthias Troyer, and James D. Whitfield. Operator locality in the quantum simulation of fermionic models. *Phys. Rev. A*, 95:032332, 2017.

[52] Zhang Jiang, Amir Kalev, Wojciech Mruczkiewicz, and Hartmut Neven. Optimal fermion-to-qubit mapping via ternary trees with applications to reduced quantum states learning. *arXiv preprint arXiv:1910.10746*, 2019.

[53] Yu-An Chen, Anton Kapustin, and Đorđe Radičević. Exact bosonization in two spatial dimensions and a new class of lattice gauge theories. *Annals of Physics*, 393:234–253, 2018.

[54] Rigetti Aspen-9 QPU Specifications. https://qcs.rigetti.com/qpus/. Accessed 2021-10-13, limited information available in above link, further info available with account at https://qcs.rigetti.com/lattices with account access.

[55] Google Quantum AI Quantum Computer Datasheet. https://quantumai.google/hardware/datasheet/weber.pdf. Accessed: 2021-10-13, archive link: https://web.archive.org/web/20211010024253/https://quantumai.google/hardware/datasheet/weber.pdf.

[56] IBM Quantum Services. https://quantum-computing.ibm.com/services?services=systems. Accessed: 2021-10-13, requires (free) account to view.

[57] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C. Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando Brandao, David A. Buell,

et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[58] Craig Gidney and Martin Ekerå. How to factor 2048 bit rsa integers in 8 hours using 20 million noisy qubits. *Quantum*, 5:433, 2021.

[59] Charles Derby, Joel Klassen, Johannes Bausch, and Toby Cubitt. Compact fermion to qubit mappings. *Phys. Rev. B*, 104:035118, 2021.

[60] Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. Topological quantum memory. *J. Math. Phys.*, 43(9):4452–4505, 2002.

[61] Laura Clinton, Johannes Bausch, and Toby Cubitt. Hamiltonian simulation algorithms for near-term quantum hardware. *Nature communications*, 12(1):1–10, 2021.

[62] Charles Derby and Joel Klassen. A compact fermion to qubit mapping part 2: Alternative lattice geometries. *arXiv preprint arXiv:2101.10735*, 2021.

[63] Daniel Gottesman. *Stabilizer Codes and Quantum Error Correction*. PhD thesis, California Institute of Technology, (1997).

[64] Rahul Sarkar and Ewout van den Berg. On sets of commuting and anticommuting paulis. *arXiv preprint arXiv:1909.08123*, 2019.

[65] Hector Bombin and Miguel A. Martin-Delgado. Topological quantum distillation. *Phys. Rev. Lett.*, 97:180501, 2006.

[66] Markus S. Kesselring, Fernando Pastawski, Jens Eisert, and Benjamin J. Brown. The boundaries and twist defects of the color code and their applications to topological quantum computation. *Quantum*, 2:101, 2018.

[67] Claudio Chamon. Quantum glassiness in strongly correlated clean systems: An example of topological overprotection. *Physical review letters*, 94(4):040402, 2005.

[68] Jeongwan Haah. Local stabilizer codes in three dimensions without string logical operators. *Phys. Rev. A*, 83(4):042330, 2011.

[69] Paul Webster and Stephen D. Bartlett. Fault-tolerant quantum gates with defects in topological stabilizer codes. *Phys. Rev. A*, 102(2):022403, 2020.

[70] Anthony Leverrier, Simon Apers, and Christophe Vuillot. Quantum xyz product codes. *arXiv preprint arXiv:2011.09746*, 2020.

[71] Toby S. Cubitt, Ashley Montanaro, and Stephen Piddock. Universal quantum hamiltonians. *P. Natl. A. Sci.*, 115(38):9497–9502, 2018.

[72] Arkady Fedorov, Leonid Fedichkin, and Vladimir Privman. Decoherence rate of semiconductor charge qubit coupled to acoustic phonon reservoir. *Phys. Rev. A*, 69:032311, 2004.

[73] Henrik Bruus and Karsten Flensberg. *Many-Body Quantum Theory in Condensed Matter Physics: An Introduction*. Oxford University Press, 2004.

[74] Alexey A. Melnikov and Leonid E. Fedichkin. Quantum walks of interacting fermions on a cycle graph. *Sci. Rep. UK*, 6:34226, 2016.

[75] Herbert Fröhlich. Electrons in lattice fields. *Adv. Phys.*, 3(11):325–361, 1954.

[76] Eva Pavarini, Erik Koch, Richard Martin, and Richard Scalettar. *The physics of correlated insulators, metals, and superconductors*, volume 7 of *Modelling and Simulation*. Theoretische Nanoelektronik, 2017.

[77] Sergey Bravyi, Bernhard Leemhuis, and Barbara M. Terhal. Topological order in an exactly solvable 3d spin model. *Ann. Phys.*, 326(4):839 – 866, 2011.

[78] Maria F. Araujo de Resende. A pedagogical overview on 2d and 3d toric codes and the origin of their topological orders. *Reviews in Mathematical Physics*, 32(02):2030002, 2020.

[79] Matthew B. Hastings, Jeongwan Haah, and Ryan O'Donnell. Fiber bundle codes: breaking the n 1/2 polylog (n) barrier for quantum ldpc codes. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1276–1288, 2021.

[80] J. Pablo Bonilla Ataides, David K. Tuckett, Stephen D. Bartlett, Steven T. Flammia, and Benjamin J. Brown. The xzzx surface code. *Nature communications*, 12(1):1–12, 2021.

[81] Aleksander Kubica, Beni Yoshida, and Fernando Pastawski. Unfolding the color code. *New Journal of Physics*, 17(8):083026, 2015.

Appendix A

Supplemental Material to Chapter 3

A.1 Properties of the cycle group $\mathcal{C}_G$

The cycle group $\mathcal{C}_G$ is defined by eq. (3.8). Here we prove some properties of this group. We say a cycle $c \in \mathcal{C}_G$ contains an edge if its expression as a product of edges contains that edge. We say a cycle c contains a vertex if it contains an edge incident on that vertex.

Proposition 24. *$\mathcal{C}_G$ is in the centralizer of M_G.*

Proof. Consider an element $c \in \mathcal{C}_G$. For every vertex contained in c, c contains an even number of edges incident on that vertex. M_G is generated by edge and vertex operators. First we note that c commutes with all vertex operators since it contains an even number of edge operators incident on that vertex. Next we note that c commutes with all edge operators, since for every edge operator e, c contains an even number of edge operators not equal to e and incident on common vertices with e (this is true even if c contains e). Thus $\mathcal{C}_G$ commutes with M_G. □

Corollary 25. *$\mathcal{C}_G$ is an Abelian normal subgroup of M_G.*

Definition 26 (Eulerian Graph)**.** *A graph is Eulerian if each of its vertices has an even number of incident edges.*

Definition 27 (Simple Cycle)**.** *A connected Eulerian subgraph in which all vertices have degree two.*

Definition 28 (Cycle Space γ of G)**.** *Given a graph $G = (\mathbf{E}, \mathbf{V})$, let $\mathcal{E} = P(\mathbf{E})$ be the power set of edges in the graph, also known as the edge space. The edge space forms an Abelian group, with the product operation being the disjunctive union $ab = a \cup b - a \cap b$. The cycle space γ is the set of edge sets of Eulerian subgraphs of G, and it is a subgroup under this group operation. Every element of the cycle space can be expressed as the product of simple cycles, thus there exists a basis – called a cycle basis – consisting of independent simple cycles which generates γ.*

Proposition 29. *$\mathcal{C}_G$ is isomorphic to the cycle space γ of G.*

Proof. Every simple cycle $a \in \gamma$ is a set of undirected edges which may be given an ordering and directedness such that one may traverse the cycle, passing over each vertex in a exactly once. The choice of ordering is unique up to the choice of starting point and the direction of travel. For every simple cycle $a \in \gamma$ define the function $f : \gamma \to \mathcal{C}_G$ in accordance with an arbitrary choice of starting point and direction of travel

$$f(a) = i^{|a|} \prod_{e_{i,i+1} \in a} e_{i,i+1}. \tag{A.1}$$

We may note that $f(a)$ is invariant under any choice of starting point and direction of travel by noting that a simple cycle never goes over the same vertex twice and so

$$e_{1,2}e_{2,3}...e_{|a|,1} = e_{2,3}...e_{|a|,1}e_{1,2} \tag{A.2}$$

and furthermore inverting direction of travel yields

$$\begin{aligned}
e_{1,|a|}...e_{3,2}e_{2,1} &= (-1)^{|a|}e_{|a|,1}...e_{2,3}e_{1,2} && \text{(A.3)}\\
&= (-1)^{|a|}e_{1,2}(e_{|a|,1}...e_{2,3}) && \text{(A.4)}\\
&= (-1)^{|a|}e_{1,2}e_{2,3}(-1)^{1}(e_{|a|,1}...e_{3,4}) && \text{(A.5)}\\
&= (-1)^{|a|}e_{1,2}e_{2,3}...e_{|a|-1,|a|}(-1)^{|a|-2}e_{|a|,1} && \text{(A.6)}\\
&= e_{1,2}e_{2,3}...e_{|a|,1} && \text{(A.7)}
\end{aligned}$$

Thus one may uniquely retrieve a from $f(a)$, and so f is invertible on the simple

cycles. Choose a cycle basis B of γ. Define f on all of γ as the product of its action on B, ie if $c = \prod b_i$, $b_i \in B$, then $f(c) = \prod_i f(b_i)$. We can see by construction that f is a homomorphism. Furthermore, since f is invertible on all simple cycles, it is invertible on the cycle basis, and is thus an isomorphism. □

Proposition 30. *M_E is isomorphic to $M_G/\mathcal{C}_G$*

Proof. Define the mapping

$$f : M_G/\mathcal{C}_G \to M_E \tag{A.8}$$

with action on each vertex v_i and each directed edge e_{ij}

$$f(v_i\mathcal{C}_G) = V_i, \quad f(e_{ij}\mathcal{C}_G) = E_{ij} \tag{A.9}$$

and

$$\begin{aligned} f(v_i\mathcal{C}_G \cdot a) &= f(v_i\mathcal{C}_G)f(a), \\ f(e_{ij}\mathcal{C}_G \cdot a) &= f(e_{ij}\mathcal{C}_G)f(a) \end{aligned} \tag{A.10}$$

for some $a \in M_G/\mathcal{C}_G$.

The mapping is clearly invertible for all products of V_i. For E_{ij} corresponding to edges in G, for some path $\{k_1, \ldots, k_L\}$ with $k_1 = i$ and $k_L = j$, the identification

$$f\left(\mathrm{i}^{L-1}\prod_{x=1}^{L-1} e_{k_x,k_{x+1}}\mathcal{C}_G\right) = E_{ij} \tag{A.11}$$

can also be made, however the left hand side is equivalent to $f(e_{ij}\mathcal{C}_G)$. For E_{ik} corresponding to edges not in G, for some path between i and k, a similar identification to eq. (A.11) can be found where the left hand side will also be unique for the same reason. The mapping is then invertible for all products of V_i and E_{ij}.

The properties of V_i and E_{ij} shown in eq. (3.4) are satisfied by $f^{-1}(V_i)$ and $f^{-1}(E_{ij})$ due to eqs. (3.6) and (A.10) and proposition 24. The cycle condition in eq. (3.5) is satisfied by inverse mappings of cycles because for sequence of vertices $s = (i_1, \ldots, i_L)$ with $i_1 = i_L$ and path $p = (i'_1, \ldots, i'_{L'})$ with $i'_1 = i'_{L'}$ where p passes

through all the vertices in s in order at least once

$$\begin{aligned} f^{-1}\left(\mathrm{i}^{L-1}\prod_{x=1}^{L-1}E_{i_x,i_{x+1}}\right) &= \mathrm{i}^{L-1}\prod_{x=1}^{L-1}f^{-1}(E_{i_x,i_{x+1}}) \\ &= \mathrm{i}^{L'-1}\prod_{x=1}^{L'-1}e_{i'_x,i'_{x+1}}\mathcal{C}_G \\ &= \mathbb{I}\mathcal{C}_G. \end{aligned} \tag{A.12}$$

The mapping is then an invertible homomorphism and therefore an isomorphism.

□