Entanglement Characterization of Topological Phases of Kitaev's Honeycomb Model

University of Windsor

Matthew Alexander

27 April 2017

Abstract

The potential benefits of a quantum computer, while enticing, are limited by a number of obstacles. The standard design of a quantum computer relies on the interactions of fundamental particles. By manipulating these particles and interacting them in a controlled manner, calculations may be performed. One of the most daunting obstacles to the continued development of quantum computers is the fragility of these quantum systems. Any small perturbation of the systems can easily affect the state of the system, ruining the calculation. One proposed type of quantum computer, known as the topological quantum computer (TQC), has the potential to alleviate this concern. The fundamental components and operations of the TQC are based on a class of exotic particles known as “anyons”. By moving these anyons around each other in two dimensions, a type of interaction may be performed which is incredibly stable to small perturbations. As the brittle nature of quantum systems presents a steep challenge to the continued development of quantum computers, TQC have become an excellent candidate for quantum computation.

This work studies an anyonic system known as Kitaev’s honeycomb model. Various vortex sectors and topological phases of the model were studied in terms of their energy spectra and entanglement. In addition, the phenomena of edge states and vortex binding were studied.

Contents

Chapter 1

Introduction

1.0.1 Overview

In 1981 Richard Feynman gave a talk at the *First Conference on the Physics of Computation* at MIT. He spoke about how it appeared to be impossible to simulate the evolution of a general quantum system on a classical computer in an efficient way, and yet it might be possible to achieve this efficient simulation using a computer based on quantum mechanical elements [1]. Feynman's words (in their usual fashion) left an imprint on the masses, and over the next three decades the concept of quantum computers dazzled the minds of the public and inspired the research of countless scientists. For some, the construction of a fully functional quantum computer would mark the apex of scientific development. The true capabilities of a fully functioning quantum computer may still be the stuff of conjecture, but it is undeniable that the speed and efficiency promised by such a device, in particular with regard to its capability in simulating quantum mechanical systems, would undoubtedly be the catalyst to a new era of scientific discovery. Over the past 36 years the field of quantum information has developed significantly, with many valuable contributions made in the fields of quantum algorithms [2, 3, 4, 5], quantum cryptography [6, 7, 8] and quantum complexity theory [1, 9, 10]. Perhaps the most important field of quantum information, however, is quantum error correction. A quantum computer would be nothing more than a very intriguing paperweight without the ability to perform fault-tolerant calculations. This remains a significant challenge, as the standard designs of quantum computers suffer from their fragility.

The standard design of a quantum computer relies on manipulating and entangling quantum states in order to perform calculations. A necessary condition for these calculations to be computable is the maintainability of *quantum coherence* - a definite phase relation between the states of the system used in the calculation. The designs of high level quantum algorithms rely on the assumption that this coherence can be maintained on a time scale long enough for the desired computations to be performed. However, any small perturbation of the quantum system has the potential to cause quantum decoherence, destroying the phase relation between the quantum states in the system, and ruining the calculation [11]. As this is such an important issue within the field, a large degree of research has been done with the end goal of mitigating the effects of small perturbations of the standard systems proposed to be used in quantum computation [12, 13, 14, 15].

An alternative avenue has also been investigated, however, wherein a new type of quantum computing system, based on elements and interactions entirely different from those in the standard quantum computing model, is explored. This type of quantum computer is known as the Topological Quantum Computer, and its fundamental elements and interactions are based on a class of particle known as *anyons* [16]. Performing so-called “braiding” interactions between certain classes of anyons can produce a type of coherence which is incredibly stable to the small perturbations which would decoherence standard quantum computing systems.

The existence of anyons was first proposed by Jon Magne Leinaas and Jan Myrheim of the University of Oslo in 1977 [17]. For decades these particles were seen as no more than a mathematical curiosity. In 1997, Kitaev first proposed the use of anyons to build a topological quantum computer [18]. Recently Kitaev’s ideas have gained traction, as anyonic excitations have been created experimentally in fractional quantum Hall systems [19].

Despite potential benefits over the standard quantum computer design, the development of topological quantum computers is hindered by its own set of problems. In particular, the question of whether the specific type of anyons required for topological quantum computation exist remains unanswered. While “abelian” anyons have been discovered in fractional quantum Hall systems, the “non-abelian” anyons required for topological quantum computation have yet to be found or created experimentally. In addition, the statistical behaviour of anyons remains poorly understood to date. Due to this fact, the investigation of anyonic systems, to better understand the behaviour of these exotic particles, remains a pertinent area of research.

This work investigates an anyonic system known as the Kitaev Honeycomb Model. In this work we have reproduced results of standard papers in the field [20, 21, 22]. We have also performed a characterization of the model in terms of its entanglement. Entanglement is a powerful method of characterizing any condensed matter system, but to date a complete characterization of the Kitaev model in terms of its entanglement is lacking.

The remainder of this chapter will provide an introduction to the key concepts of topological quantum computation, as well as an explanation of the theory behind the method of entanglement characterization.

The second chapter is devoted to setting up the Kitaev model in terms of its anyonic excitations. In the first part of the chapter, we set up the model in terms of static spin-$\frac{1}{2}$ fermions, under spin couplings. By mapping the spins to Majorana fermions hopping within a $\mathbb{Z}_2$ gauge field, we are able to pick up impurities in the lattice system known as vortices. Non-abelian statistics are exhibited by Majorana modes associated with the vortices [21]. By studying various configurations of the vortices within the Kitaev model, we can gain a deeper understanding of the behaviour of these excitations.

In Chapter 3 we outline the methods used to calculate numerical results for the Kitaev model, and display the results we have obtained. The beginning section of the chapter describes the transformation of the honeycomb lattice to a square lattice in order to facilitate numerical calculations, and outlines the use of the correlation matrix to calculate measures of entanglement. The results section begins by examining the band structure of the model under various vortex configurations and hopping energies of the Majorana fermions. The next section will discuss the probability distributions of modes of the Kitaev model in the presence of vortices, and will display the tendency of certain modes to undergo binding to the vortices. The following section will discuss the effect of placing a "cut" into the system, breaking the periodicity in one direction and introducing "edge states". The chapter will conclude with a discussion of the entanglement characterization obtained for the model.

1.0.2 Anyons

A standard method to reduce the complexity of a given problem is to reduce the number of dimensions in which the problem lives. This is a technique familiar to all physicists who began their careers by studying the fascinating dynamics of boxes, ramps, and pulleys. However, in some cases, rather than simplifying the problem, this dimensional reduction happens to introduce a surprising richness to the system.

Consider the case of two identical particles situated in three dimensional space. The state of the total system is given by

$$|\psi(\mathbf{r}_1\mathbf{r}_2)\rangle \,. \tag{1.1}$$

We can define an operator $\hat{P}$ such that $\hat{P}$ acting on the state $|\psi(\mathbf{r}_1\mathbf{r}_2)\rangle$ swaps the positions of the two particles.

Intuitively, if we swap the particles twice, then we should arrive back at the initial state. Hence

$$\hat{P}^2\,|\psi(\mathbf{r}_1\mathbf{r_2})\rangle = |\psi(\mathbf{r}_1\mathbf{r_2})\rangle \,. \tag{1.2}$$

Since the eigenvalues of $\hat{P}^2$ are simply the squared eigenvalues of $\hat{P}$, we find that the eigenvalues of the permutation operator, $\hat{P}$, must be ± 1. Hopefully this result appears familiar - these eigenvalues correspond to the statistical behaviour of bosons and fermions: exchanging two identical bosons gives an overall phase factor of $+1$ to the wavefunction of the total system, while exchanging two identical fermions gives an overall phase factor of -1 to the wavefunction of the total system.

As the above argument does not depend on the dimensions of the wavefunction, intuitively our result should hold regardless of the dimension in which the problem lives. However, in two dimensions this is not the case.

In order to understand the differences between the problem when formulated in three dimensional space, when compared with two dimensional space, we will take a brief detour into the field of topology.

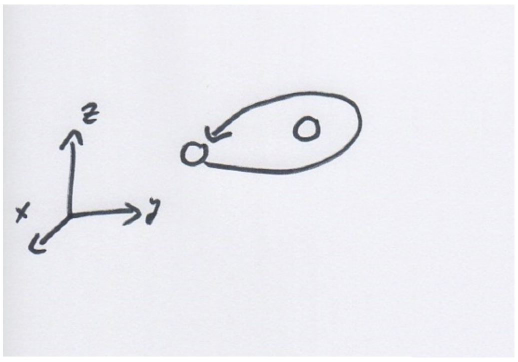
Topology concerns itself with the properties of different topological spaces. For our purposes we will not be required to delve deeply into what makes a space *topological* - we shall only concern ourselves with the usual Euclidean spaces of dimension 2 (denoted $\mathbb{R}^2$) and 3 (denoted $\mathbb{R}^3$).

Definition 1. *We shall call a space **simply connected** if it satisfies the following two properties:*

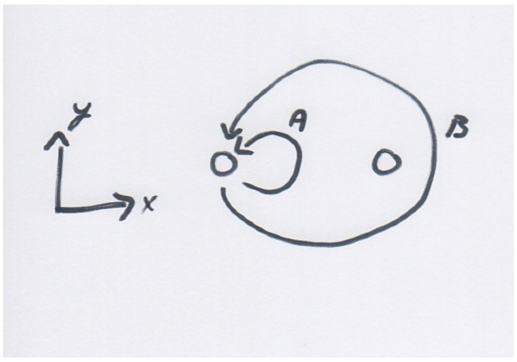
1. For any two points in the space we may draw a path between the two points, which lies completely within the space.

2. Given two points in the space, any path connecting those two points may be continuously deformed to any other path between those two points.

For ordinary Euclidean spaces, there is not much to be seen: every Euclidean space $\mathbb{R}^n$ is simply connected. However, if we begin to cut holes in our space, this result no longer holds for general $\mathbb{R}^n$.



(a) Moving one particle around another in 3D space. This path may be continuously deformed into the trivial loop.



(b) Moving one particle around another in 2D space. Path A may be continuously deformed into the trivial loop, but path B may not.

Figure 1.1: The difference between passing one particle around another in three dimensional and two dimensional space.

Let us consider the space $\mathbb{R}^3$ with a single hole cut into it. We will choose to draw a path whose starting and ending points are the same - ie. a closed loop. The simplest closed loop is the *trivial loop* - a path of 0 length. No matter which closed loop we choose to make, it is possible to continuously deform it back down to the trivial loop. It makes no difference whether the loop passes around the hole, or completely avoids it. Though this result alone is not enough to prove it, it is easy to see that this space is simply connected. In fact we can generalize this result:

Proposition 1. $\forall n \geq 3$, *the Euclidean space* $\mathbb{R}^n$ *with any finite number of holes in it is simply connected.*

So what happens in two dimensions? We now consider the space of $\mathbb{R}^2$ with a single hole cut into it. In this case, any closed loop which does not pass around the hole is able to be continuously deformed to the trivial loop, but any loop which encloses the hole cannot! If we tried to continuously deform path B in Fig 1.1b to the trivial loop, we would have to either pass through the hole, or jump out of the page - either way we would have to leave our topological space, which is not allowed. Since we have found two loops which share the same starting and ending points and yet are not continuously deformable into each other, we can see that this space is not simply connected.

We may now return to the case of identical particles. An equivalent operation to swapping the particles twice with the permutation operator, is moving one particle all the way around the other one. If we trace out the path of the moving particle, we see that this is simply the case of a closed loop in Euclidean space, where the stationary particle acts as a hole. Thus, using our results for the simply connectedness of Euclidean spaces with holes in them, we can see that moving one particle around the other in three spacial dimensions is topologically equivalent to not moving the particles at all (the trivial loop). This is what allowed us to make the argument that $\hat{P}^2 |\psi(\mathbf{r_1 r_2})\rangle = |\psi(\mathbf{r_1 r_2})\rangle$.

But as we have seen, moving one particle around the other is not topologically equivalent to keeping the particles stationary in *two* dimensional space. So we can no longer say that the state of the system in unchanged under the action of exchanging the identical particles twice. Hence we can no longer relate the state of the system before and after the particle exchanges:

$$\hat{P}^2 |\psi(\mathbf{r}_1\mathbf{r_2})\rangle \overset{?}{=} |\psi(\mathbf{r}_1\mathbf{r_2})\rangle \,. \tag{1.3}$$

It may seem that we are at an impasse. Luckily there are still restrictions we can force upon the wavefunction: If we normalize the wavefunction of the system before exchanging the particles, we can assume that the wavefunction remains normalized after the exchange. In this case the only thing we can pick up upon the exchange of the particles is a scalar phase factor or a unitary matrix:

$$\begin{aligned} \hat{P} |\psi(\mathbf{r}_1\mathbf{r_2})\rangle &= e^{i\theta} |\psi(\mathbf{r}_1\mathbf{r_2})\rangle \quad \text{or} \\ \hat{P} |\psi(\mathbf{r}_1\mathbf{r_2})\rangle &= U |\psi(\mathbf{r}_1\mathbf{r_2})\rangle \end{aligned} \tag{1.4}$$

where $UU^\dagger = U^\dagger U = I$.

This is a remarkable result! In the three dimensional case we still pick up a phase factor under the exchange of the identical particles, however the phase factor is restricted to be ± 1. Here there is no such restriction - the identical particles may pick up any phase under their exchange[1]. Because of this, particles which adhere to these types of statistics are known as "anyons". Furthermore, those particles which pick up a scalar phase factor under their exchange are known as abelian anyons, while those which pick up a unitary matrix are known as non-abelian anyons[2]. It should be noted that the argument made above does not explicitly prove the existence of anyons, it simply gives a reason to believe that they might exist.

The term "anyon" was first introduced by Frank Wilczek in 1982 [23], however at that time the utility of these particles was unknown. Wilczek, in particular, wrote in his 1982 paper that "practical applications of these [anyonic] phenomena seem remote". Within the following decades, however, applications of these particles became apparent. Kitaev showed that non-abelian anyons with appropriate properties can be used to efficiently create a quantum circuit [18].

The fundamental interaction which is used to bind these anyons together, and which is exploited in the design of topological quantum computers, is known as *braiding*. In order to intuitively understand these braids, we shall draw the world lines of the particles in Fig 1.1b in three dimensional spacetime.

By examining Fig 1.2c it can be seen that in the case where one anyon is moved all the way around the other in two spatial dimensions, the world lines of the two anyons intertwine. This binding of the world lines is the braiding interaction of the anyons. Clearly when one anyon avoids the other, as in Fig 1.2b, no braids are formed. As braids are formed by transporting anyons around each other, we can see from (1.4) that these braiding interactions correspond to the creation of phase factors for the anyons. In particular, for non-abelian anyons these braids correspond to unitary matrix phase factors. By braiding non-abelian anyons together in appropriate ways, the quantum logic gates of topological quantum computers can be created from the unitary matrices [24].

One may wonder why, considering its potential benefits, the field of topological quantum computation hasn't received more attention. Perhaps the largest problem

[1] θ must be a rational number

[2] The term non-abelian comes from the fact that matrices, in general, do not commute

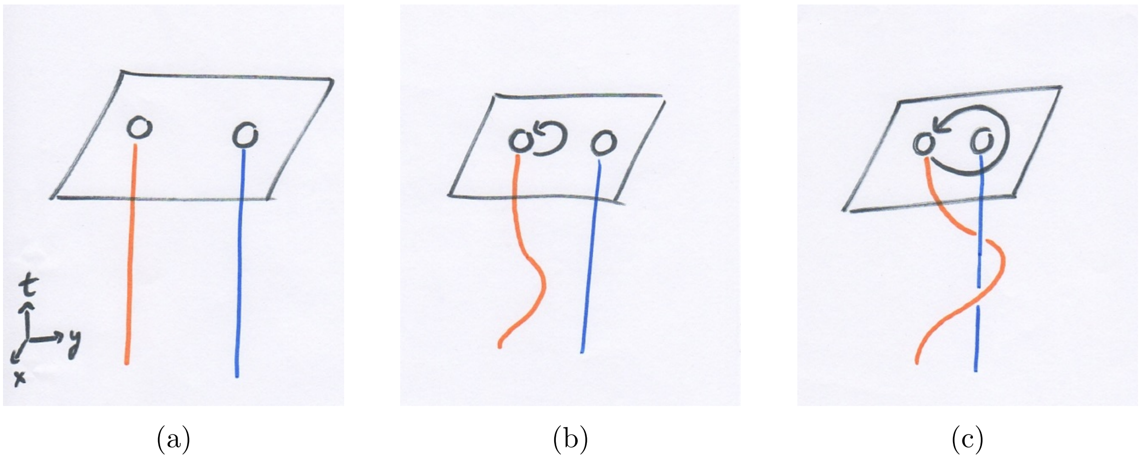


Figure 1.2: World lines of particles in 3 dimensional spacetime.

facing this field is the issue of *nucleated phases*. This problem may be thought of as the topological quantum computing analogue of decoherence. For any practical topological quantum computing device, a large number of anyons must be present in order to perform calculations. However, it has been shown that in the presence of a large number of anyons, the topological phase of an anyonic system may change - nucleated phases arise in which the excitations of the system no longer exhibit non-abelian statistics. Hence even if one were able to capture or create enough non-abelian anyons to create a topological quantum computer, without a way to combat the problem of nucleated phases, the anyons would no longer exhibit the required non-abelian behaviour to perform calculations, rendering the computer unusable.

In particular, Ludwig has shown that the problem of nucleated phases arises in the Kitaev model [25]. As this issue presents a serious obstacle to the construction of topological quantum computers, studying systems such as the Kitaev model, in which nucleated phases occur, is incredibly important.

1.0.3 Majorana Fermions

Paul Dirac's relativistic wave equation of 1928 was a remarkable success for the field of quantum theory - it was the first equation to fully account for special relativity in the context of quantum mechanics. In addition, this equation was the first to imply the existence of antimatter (which would be discovered only a few years later). In the following decade, Dirac's formulation led the physics community to rapid developments in their understanding of the quantum world: the positron was first observed by Carl Anderson in 1932 [26] and Dirac's formalism was crucial to the development of Enrico Fermi's theory of beta decay in 1934. Additionally, the Dirac equation explained the concept of spin as a consequence of relativity, correctly predicted the structure of the spectrum of hydrogen, and predicted a g-factor of 2 in the gyromagnetic ratio of the electron.

Despite its massive success, the form of the Dirac equation left at least one man unsatisfied. Ettore Majorana, a student of Fermi, had an idea for a modification of the Dirac equation which implied the existence of a new form of matter.

The Dirac equation is given by

$$(i\gamma^\mu \partial_\mu - m)\psi = 0 \tag{1.5}$$

where the $\{\gamma^k\}$ are are matrices which satisfy the following relation:

$$\{\gamma^\mu, \gamma^\nu\} = 2\eta^{\mu\nu} \tag{1.6}$$

where $\eta^{\mu\nu}$ is the metric tensor for flat space.

Dirac derived this form of a relativistic wave equation and went on to show that the 4×4 matrices

$$\gamma^0 = \begin{bmatrix} I & 0 \\ 0 & -I \end{bmatrix}, \quad \gamma^1 = \begin{bmatrix} 0 & \sigma_x \\ \sigma_x & 0 \end{bmatrix}, \quad \gamma^3 = \begin{bmatrix} 0 & \sigma_y \\ \sigma_y & 0 \end{bmatrix}, \quad \gamma^4 = \begin{bmatrix} 0 & \sigma_z \\ \sigma_z & 0 \end{bmatrix} \tag{1.7}$$

satisfied his equation (where the σ_i are the 2×2 Pauli matrices, I is the 2×2 identity matrix, and the 0 are 2×2 zero-matrices). However these matrices were not the only matrices which could satisfy the Dirac equation. A second choice of matrices is known as the *Weyl basis* and is achieved by replacing γ^0 with

$$\begin{bmatrix} 0 & I \\ I & 0 \end{bmatrix}. \tag{1.8}$$

One thing Majorana noticed was that in either choice of basis, the solutions of the Dirac equations were complex. Majorana wondered whether it was possible to reformulate the Dirac equation in such a way as to permit real solutions. To this end, Majorana put forward his own choice of gamma matrices in 1937, which satisfied the relation needed for the Dirac equation:

$$\gamma^0 = \begin{bmatrix} 0 & \sigma_y \\ \sigma_y & 0 \end{bmatrix}, \quad \gamma^1 = \begin{bmatrix} i\sigma_z & 0 \\ 0 & i\sigma_z \end{bmatrix}, \quad \gamma^3 = \begin{bmatrix} 0 & -\sigma_y \\ \sigma_y & 0 \end{bmatrix}, \quad \gamma^4 = -\begin{bmatrix} i\sigma_x & 0 \\ 0 & i\sigma_x \end{bmatrix}. \tag{1.9}$$

As each of these matrices are purely imaginary, the $(i\gamma^\mu \partial_\mu - m)$ term in the Dirac equation becomes purely real, permitting real solutions to the equation.

From the viewpoint of quantum field theory, the Dirac equation describes fields whose excitations are spin-$\frac{1}{2}$ particles. In this viewpoint, the solutions ψ and ψ^* of the Dirac equation describe fields whose excitations are particle-antiparticle pairs. Thus Majorana's reformulation of the Dirac equation permitted particles which were their own antiparticles.

Particles which are their own antiparticles have been found in nature. In fact, the photon is such a particle. However, the interesting implication of Majorana's formulation was that there may exist *fermions* which are their own antiparticles. To date, every particle which is known to be its own antiparticle is also a boson - the photon, neutral pions, and gravitons, for example. Fermions which have this property are known as *Majorana fermions*, and have yet to be discovered.

There are several candidates, however, which have been suggested to be Majorana fermions. Majorana himself speculated that neutrinos may be their own antiparticles. However, to date it is still unknown whether neutrinos truly have this property.

In addition, the theory of supersymmetry provides a plethora of Majorana fermion candidates. Supersymmetry is a proposed framework which extends the Standard

Model in order to attempt to solve problems which the Standard Model is unable to handle. Within the model of supersymmetry, for every fermionic particle there exists a corresponding boson with the same properties. These boson-fermion pairs are known as *superpartners*. If this were the case, there would exist a fermionic counterpart to the photon: the photino. As the photon is its own antiparticle, the photino would also be its own antiparticle under supersymmetry. The photino would thus be a Majorana fermion. Similarly, the superpartners of neutral pions and gravitons would also be Majorana fermions.

One of the most accessible areas of physics in which Majorana fermions may be found is the field of condensed matter physics. A number of theoretical condensed matter systems exist in which quasiparticle excitations, which behave as Majorana fermions, may be found. Indeed the Kitaev model is one such system which exhibits these excitations.

1.0.4 Entanglement

In 1935 the age of quantum mechanics was in full swing - fresh on the heels of Fermi's theory of beta decay, the field of quantum mechanics was continuing to develop rapidly. However, despite the developments of the field, some felt that the quantum picture of the universe was lacking. Albert Einstein, in particular, believed that it was impossible to provide a complete picture of reality using only quantum theory. In defense of this claim, Einstein, along with Boris Podolsky and Nathan Rosen published the "EPR paradox". This thought experiment introduced what Einstein later referred to as "spooky action at a distance", and what we know today as *entanglement*.

The thought experiment was as follows:

"Suppose that we have two [particles], I and II, which we permit to interact from the time t = 0 to t = T, after which time we suppose that there is no longer any interaction between the two parts. We suppose further that the states of the two systems before t = 0 were known. We can then calculate with the help of Schrödinger's equation the state of the combined system I+II at any subsequent time." [27]

Due to the Heisenberg uncertainty principle, it is impossible to know both the position and momentum of either particle exactly. However, one could measure the position of the first particle, and then, using their knowledge of the total system, deduce the position of the second particle. Similarly, a measurement of the momentum of the second particle could be used to deduce the momentum of the first particle. In this way, a measurement of only particle one immediately determines the state of the second particle, without disturbing the second particle with a physical measurement.

The fact that the measurement of one particle appeared to influence the second particle, regardless of the distance between the two particles, perturbed Einstein. Clearly, Einstein thought, this was nonsense - the fact that there appeared to be a persistent connection between the two particles, irrespective of distance, could not be a true physical phenomenon, and must point to a failure in the ability of quantum theory to completely describe reality.

It was Erwin Schrödinger who was able to shed light onto this problem. In a letter written to Einstein, he used referred to the strange phenomenon as "Verschränkung", and asserted that, as the two particles had interacted, one could no longer treat the complete system as two separate one-particle systems, but rather, could only examine the total two particle system. The fact that measuring one particle immediately determined the state of the other, was simply a consequence of the fact that one can only examine the total system, and thus a measurement of either particle is a measurement of the entire system, collapsing the entire system into one state (wherein the observables of both particle one and two are known).

Soon after, Schrödinger published two key papers discussing the EPR paradox, in which he defined the Verschränkung interaction as "entanglement" [28, 29].

To Schrödinger, entanglement was not something to be wary of, nor did it point to an inherent flaw in the description of reality provided by quantum theory. On the contrary, Schrödinger considered entanglement to be "*the* characteristic trait of quantum mechanics, the one that enforces its entire departure from classical lines of thought" [28].

Despite Schrödinger's praise of entanglement and assertions of its importance, the study of entanglement quickly died off and was left untouched for decades. To study entanglement in many-body systems proved to be incredibly challenging. In recent years however, entanglement has become a tool for studying and characterizing physical systems. In fact, some of the most complex and beautiful objects in the universe can be deeply understood in terms of their entanglement.

The early 1970s brought with them startling new discoveries about the behaviour of black holes. For many years it was assumed that black holes had zero entropy. However in 1971, Stephen Hawking showed that the total area of the event horizons of a set of black holes can never decrease [30]. If black holes truly did have zero entropy, then any matter falling into a black hole would reduce the entropy of the universe, and as the event horizon of the black hole could never decrease, this entropy could not be regained. This violates the second law of thermodynamics.

In 1972 Jacob Bekenstein proposed a solution: he argued that the entropy of a black hole was non-zero and scaled with the surface area of the event horizon. This result was puzzling to many, as in most systems, entropy naturally scales with volume, rather than surface area. However, in 1993 Mark Srednicki was able to show that by considering the entanglement between the systems inside and outside of the event horizon, and calculating the entanglement entropy of the total (bipartite) system, one finds that the entropy does in fact scale with the surface area of the black hole [31]. This consequence can be understood through the following argument: The entanglement entropy of a bipartite system is calculated by computing the density matrix of the total system, and then tracing out the elements of either of the two subsystems (essentially ignoring one of the two subsystems). This *reduced density matrix* is then used to compute the entanglement entropy. However, the entanglement entropy of the system must be the same, irrespective of which of the two subsystems is ignored. Since the state of the subsystem outside of the event horizon of the black hole is unaffected by a change in volume of the black hole, the entanglement entropy can thus not be affected by the black hole's volume. Moreover, the entanglement entropy of the black hole can only be affected by a parameter which is shared by both the subsystems inside and outside of the event horizon. Thus, the surface area of the event horizon is a perfectly acceptable parameter.

Clearly entanglement provides a powerful tool to characterize any physical system. In the field of condensed matter, entanglement remains one of the most powerful methods of characterization, and in fact entanglement characterization has been performed on the Kitaev model. However, to date a complete characterization of the model is lacking. As it will be essential in the following chapters, we shall devote this rest of this chapter to giving an overview of entanglement and the techniques involved in entanglement characterization. For more information on entanglement see the review articles by Horodecki et al [32], and Amico et al [33].

Consider two systems, A and B, with corresponding Hilbert spaces H_A and H_B, respectively. If we consider the total system to be made up of both of the subsystems, A and B, then the Hilbert space of the total system is given by the tensor product of the Hilbert spaces of the individual subsystems:

$$H_{tot} = H_A \otimes H_B. \tag{1.10}$$

We will begin our discussion of entanglement by assuming that we know the states of each of the subsystems: we'll say that system A is in state $|\psi_A\rangle$ and system B is in state $|\psi_B\rangle$. In this case the state of the total system is given by

$$|\psi_{AB}\rangle = |\psi_A\rangle \otimes |\psi_B\rangle . \tag{1.11}$$

This is a nice result: what the above equation tells us is that, given the states of the subsystems, we can easily construct the state of the total system. However, a more intriguing consequence is revealed when we look at this equation from the opposite direction: given the state of the total system, we are able to assign wavefunctions to the individual subsystems. This is a nice property, and we might expect that we will always be able to do this. However, this is not the case. In fact we give states of the form (1.11) a special name:

Definition 2. *Let $H = H_1 \otimes H_2 \otimes ... \otimes H_n$ be the Hilbert space of a system made up of n subsystems with individual Hilbert spaces $\{H_i\}_{i=1}^{n}$. States of the total system which can be expressed as the tensor product of pure states of the individual subsystems are referred to as* ***separable states****.*

A general state of the total system is of the form

$$|\psi_{AB}\rangle = \sum_{ij} c_{ij} |i_A\rangle \otimes |j_B\rangle \tag{1.12}$$

where $\{|i_A\rangle\}$ is a basis for subsystem A and $\{|j_B\rangle\}$ is a basis for subsystem B.

The problem of determining whether a general quantum state is separable is known as the *quantum separability problem* and has been proven to be NP-hard [34, 35].

If a state is not separable, it is called *entangled*. As our discussion above alluded to, given an entangled state of the total system, there is no way to assign a definite state to each of the subsystems A and B. In this way, the states of the subsystems can no longer be understood independently. Rather, we can only look at the state of the total system. This gives an idea of what is meant by saying that the system is entangled.

Often it is important to know the degree to which states are entangled. To this end, we introduce the following theorem:

Theorem 1 (Schmidt Decomposition)**.** *Let H_A and H_B be Hilbert spaces of dimensions m and n respectively. Assume $m \leq n$. Let $|\psi\rangle$ be a pure state of*

$H_{tot} = H_A \otimes H_B$. Then there exist orthonormal bases $\{|i_A\rangle\}_{i=1}^{m}$ for subsystem A and $\{|i_B\rangle\}_{i=1}^{n}$ for subsystem B, such that

$$|\psi\rangle = \sum_{i=1}^{m} \lambda_i \, |i_A\rangle \otimes |i_B\rangle$$

The λ_i are non-negative real numbers known as Schmidt coefficients, which satisfy $\sum_i \lambda_i^2 = 1$.

The *Schmidt number* of a state $|\psi\rangle$ is the number on non-zero Schmidt coefficients of its Schmidt decomposition. The Schmidt number of a state can be used as a measure of the state's entanglement:

Theorem 2. *A state ψ is entangled if and only if its Schmidt number is greater than one.*

Chapter 2

Kitaev's Honeycomb Model

2.0.1 The Model

The Kitaev model is a very important physical system. In the field of condensed matter the Kitaev model stands out as a particularly interesting model, as it is a spin model exhibiting a *spin liquid* phase in which the elementary excitations are Majorana fermions. Spin liquids are states of matter in which the spins of the materials exhibit strong fluctuations, even down to a temperature of absolute zero. Spin liquids are of particular interest to physicists, as they often exhibit fascinating properties, such as emergent gauge fields and fractional particle excitations [36]. We shall see shortly how a gauge field arises in the Kitaev model. In addition, as we have alluded to in the previous chapter, the fact that the Kitaev model exhibits non-abelian anyons and nucleated phases, makes it an important system for the field of quantum computation.

To create the model we place a spin-$\frac{1}{2}$ fermion on each site of a honeycomb lattice. The fermions are restricted from moving between lattice sites, but have spin interactions given by the following Hamiltonian:

$$H = -J_x \sum_{x-links} \sigma_i^x \sigma_j^x - J_y \sum_{y-links} \sigma_i^y \sigma_j^y - J_z \sum_{z-links} \sigma_i^z \sigma_j^z - K \sum_{i,j,k} \sigma_i^x \sigma_j^y \sigma_k^z \tag{2.1}$$

where J_x, J_y, and J_z give the spin interaction energies between the nearest neighbour fermions in the x, y, and z directions, respectively. The directions in the lattice are defined as in Fig 2.1.

The K term may be obtained from a perturbative expansion when a weak Zeeman magnetic field of the form $\mathbf{H} \cdot \boldsymbol{\sigma}$ is applied to the system. The third order term in this perturbative expansion is the only one which appears in the K term, as it is the smallest order term which breaks time reversal invariance [20].

Time reversal is described by an operator which is both anti-linear and unitary. ie.

$$\begin{aligned} &T^{-1} = T^{\dagger} \quad \text{and} \\ &T(c_1\psi_1(\mathbf{r}) + c_2\psi(\mathbf{r})) = c_1^* T(\psi_1(\mathbf{r})) + c_2^* T(\psi(\mathbf{r})). \end{aligned} \tag{2.2}$$

The time reversal operator acts on the Pauli spin operators as

$$T\sigma_i^{\alpha} T^{\dagger} = -\sigma_i^{\alpha}. \tag{2.3}$$

Using the Pauli algebra of $[\sigma^\alpha, \sigma^\beta] = i\epsilon^{\alpha\beta\gamma}\sigma^\gamma$ and $(\sigma_i^\alpha)^2 = I$, we find that any product of an even number of Pauli operators will respect time reversal symmetry. Thus in order to explicitly break time reversal invariance, we need to add a term with an odd number of spins coupling, such as our 3-spin coupling K term, to the Hamiltonian. This break in time reversal symmetry will provide us with non-trivial topological behaviour in our system, as in the fractional quantum Hall effect [37].

The summation involving the K term runs over spin triplets such that every plaquette contributes the six terms

$$K(\sigma_1^z\sigma_2^y\sigma_3^x + \sigma_2^x\sigma_3^z\sigma_4^y + \sigma_3^y\sigma_4^x\sigma_5^z + \sigma_4^z\sigma_5^y\sigma_6^x + \sigma_5^x\sigma_6^z\sigma_1^y + \sigma_6^y\sigma_1^x\sigma_2^z). \tag{2.4}$$

We can define operators $\hat{W}_p$ which are associated with each of the plaquettes (hexagons) in the lattice. The sites of each plaquette are numerated as given by Fig 2.1. Using this numeration, the plaquette operators are defined as:

$$\hat{W}_p = \sigma_1^x\sigma_2^y\sigma_3^z\sigma_4^x\sigma_5^y\sigma_6^z. \tag{2.5}$$

Remarkably, the plaquette operators commute with all of the operators $\sigma_i^\alpha\sigma_j^\alpha$ and $\sigma_i^x\sigma_j^y\sigma_k^z$, as well as with each other. This tells us that the Hamiltonian (2.1) has a set of "integrals of motion", $\{\hat{W}_p\}$. We can thus simplify the problem of diagonalizing the Hamiltonian, by diving the total Hilbert space into invariant subspaces given by the eigenspaces of the $\{\hat{W}_p\}$. Each subspace will be referred to as a *sector* and will be denoted as

$$H_{\lambda_1,\ldots,\lambda_n} \tag{2.6}$$

where λ_i are the chosen eigenvalues of each plaquette operator, $\hat{W}_i$, which characterize the eigenspace.

Then the total Hamiltonian can be divided into sectors as follows:

$$H_{tot} = \bigoplus_{\lambda_1,\ldots,\lambda_n} H_{\lambda_1,\ldots,\lambda_n} \tag{2.7}$$

where the direct sum goes over every combination of eigenvalues of the plaquette operators.

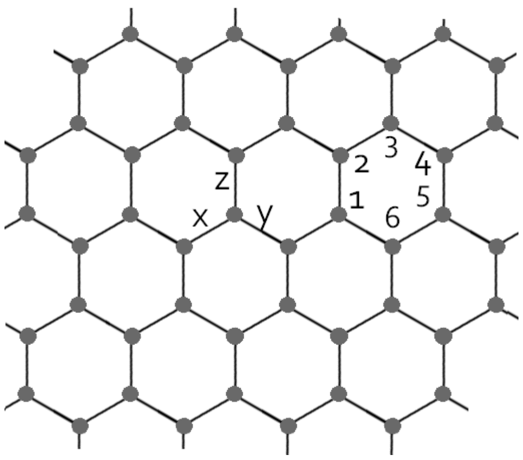


Figure 2.1: The honeycomb lattice.

Each plaquette operator has eigenvalues $+1$ and -1. We interpret $\hat{W}_p$ having an eigenvalue of -1 to mean that the plaquette p contains a vortex. The plaquette operators also satisfy the condition

$$\prod_p \hat{W}_p = I. \tag{2.8}$$

This tells us that the total number of -1 eigenvalues of all of the plaquette operators must be an even number. Hence the vortices in the lattice must always come in pairs.

The fact that we may restrict ourselves to diagonalizing the Hamiltonian in a fixed sector implies that the vortices exhibited by the system do not undergo dynamics - they are static within the plaquette they appear in. As we will see, the vortices within our lattice are central to our questions of topological phase, and hence, in order to stress their importance, we shall refer to these various sectors from this point on as *vortex sectors*.

We have thus reduced our original problem to solving our Hamiltonian (2.1) while restricted to a particular vortex sector, $H_{\lambda_1,\ldots,\lambda_n}$.

2.0.2 From a Spin Model to a Gauge Theory

We are now ready to perform the mapping we alluded to in the previous chapter. We shall first represent each spin-$\frac{1}{2}$ particle on the lattice by two fermionic modes, a_1 and a_2. We can then represent each of these fermionic modes by Majorana modes given by

$$c \equiv a_1 + a_1^\dagger, \quad b^x \equiv \frac{a_1 + a_1^\dagger}{i}, \quad b^y \equiv a_2 + a_2^\dagger, \quad b^z \equiv \frac{a_2 - a_2^\dagger}{i}. \tag{2.9}$$

It is easily checked that the each of the Majorana modes satisfy the following relations

$$\begin{aligned} &(b^\alpha)^2 = 1, \quad \{b^\alpha, b^\beta\} = 2\delta_{\alpha\beta}, \quad (b^\alpha)^\dagger = b^\alpha \\ &c^2 = 1, \quad \{b^\alpha, c\} = 2\delta_{\alpha\beta}, \quad c^\dagger = c. \end{aligned} \tag{2.10}$$

It is important to realize that these four Majorana operators act on a 4-dimensional Fock space, $\mathcal{F}_4$ while the two fermionic operators only act on a 2-dimensional subspace of this Fock space, $\mathcal{F}_2$. Thus in order to properly analyze our system in terms of these Majorana operators, we need to project down from the 4-dimensional Fock space to the 2-dimensional subspace. We achieve this with the projector $D = b^x b^y b^z c^z$.

Under the transformation of the projector, the following must hold

$$|\psi\rangle \in \mathcal{F}_2 \text{ if and only if } D\,|\psi\rangle = |\psi\rangle\,. \tag{2.11}$$

As the two dimensional subspace corresponds to the true physical picture, we share refer to $\mathcal{F}_4$ as the *extended space* and $\mathcal{F}_2$ as the *physical space*.

The next step is to represent the spin matrices in terms of Majorana fermions. We choose the representation $\sigma_i^\alpha = i b_i^\alpha c_i$. These operators satisfy the Pauli algebra when restricted to $\mathcal{F}_2$.

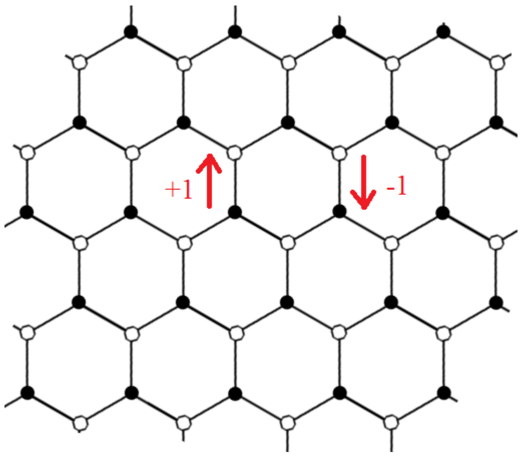


Figure 2.2: The bicoloured honeycomb lattice with an overall hopping orientation given by the arrows.

We find that the spin couplings are represented by the following products of operators:

$$\sigma_i^\alpha \sigma_j^\alpha = -i\hat{u}_{ij} c_i c_j, \quad \sigma_i^\alpha \sigma_j^\alpha \sigma_k^\alpha = -i\hat{u}_{ik}\hat{u}_{jk} c_i c_j \tag{2.12}$$

where we have defined the "link operators", $\hat{u}_{ij}$, as

$$\hat{u}_{ij} = i b_i^\alpha b_j^\alpha \tag{2.13}$$

where $\alpha = x, y, z$ depending on whether the sites i and j are connected by an x, y, or z link, respectively. Additionally, the link operators satisfy

$$\hat{u}_{ij} = -\hat{u}_{ji}, \quad \hat{u}_{ij}^2 = 1, \quad \hat{u}_{ij}^\dagger = \hat{u}_{ij}. \tag{2.14}$$

Using our new Majorana fermion / link operator representation, the Hamiltonian (2.1) is brought into the form

$$H = \frac{i}{4}\sum_{i,j} \hat{A}_{ij} c_i c_j, \quad \hat{A}_{ij} = 2J_{ij}\hat{u}_{ij} + 2K\sum_k \hat{u}_{ik}\hat{u}_{jk}. \tag{2.15}$$

The antisymmetry of the operators $\hat{u}_{ij}$ may be defined consistently by choosing an overall orientation of the links of the lattice such that an overall $+$ sign is assigned when our Majorana fermions hop from sites i to sites j and an overall $-$ sign is assigned when the fermions hop from j to i. To create this orientation, we colour the sites on the lattice white and black. We then choose our orientation as in Fig 2.2, where we will consider hopping from a black site to a white site to be positive hopping (where we pick up a factor $+1$) and hopping from a white site to a black site to be negative hopping (where we pick up a -1).

It turns out that the link operators commute with the Hamiltonian, as well as each other. As was the case with the plaquette operators, we can split the Hilbert space into eigenspaces of the $\hat{u}_{ij}$. The eigenvalues of the link operators, as with the plaquette operators, are $+1$ and -1. The total Hilbert space thus separates as:

$$H_{tot} = \bigoplus_u H_u \tag{2.16}$$

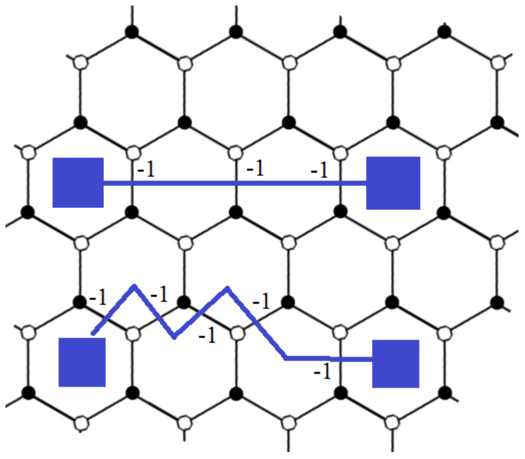


Figure 2.3: The appearance of vortices in the honeycomb lattice. The negative eigenvalues of the link operators are marked, along with two potential paths between paired vortices.

where H_u are the vortex sectors given by the link operators, and the direct sum goes over every combination of the eigenvalues of the link operators.

We can restrict ourselves to a particular vortex sector, characterized by a particular choice of eigenvalues for the link operators, by replacing the link operators in the Hamiltonian with their corresponding eigenvalues. We need to be mindful of a subtlety involving the link operators, however. When we apply the projector D, we change the values of the link operators. Hence the vortex sectors, H_u, are not invariant under the projection operator, and thus the states found in this subspace do not correspond to the physical subspace.

To correct this issue we construct an operator

$$\hat{W}_p = \prod_{(j,k)\in\text{boundary(p)}} \hat{u}_{jk} \tag{2.17}$$

where the product goes over link operators whose indices lie on the plaquette p. This operator is the corresponding plaquette operator for the extended space, and reduces to the plaquette operator for the physical space under the action of the projector, D. This operator commutes with the Hamiltonian and hence the Hamiltonian can be split into vortex sectors given by the operators $\hat{W}_p$.

As in the previous section, we identify the eigenvalue -1 of a plaquette operator with the appearance of a vortex on the corresponding plaquette. The eigenvalues of the link operators can be thought of as marking the path between two paired vortices, as in Fig 2.3. There are many choices of the eigenvalues of the link operators which will produce the same corresponding plaquette operator. In the remainder of this work we shall only consider eigenvalues $u_{ij} = -1$ which appear on the z-links between sites on the honeycomb lattice. Hence vortex pairing can only occur between vortices on the same horizontal line of plaquettes.

One can think of the link operators, $\hat{u}_{ij}$, as a $\mathbb{Z}_2$ gauge field. The fact that we may restrict ourselves to a particular vortex configuration corresponds to the gauge field being "static" - once the vortex sector is selected, the configuration of the vortices does not vary with time. It should be noted that the fact that the $\mathbb{Z}_2$ gauge field is static is not a common property of physical systems. Generally, gauge fields

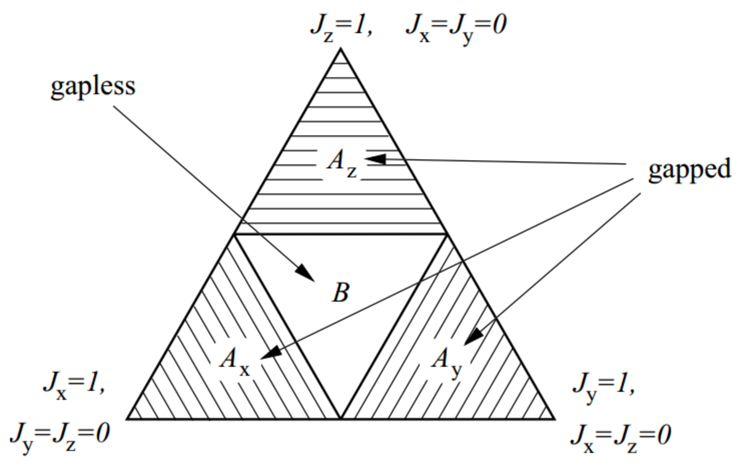


Figure 2.4: The topological phases of the Kitaev model in the vortex free sector. Retrieved from [21].

exhibit dynamics and, consequently, are difficult to deal with. The static gauge field of the Kitaev model is one of the reasons that we are interested in studying it.

We now have our Hamiltonian in quadratic form and have set up the operators required to analyze our system. From an analytic standpoint, the Hamiltonian (2.15) is in a form which may be used to derive analytic results. However, for our purposes it is better to put the Hamiltonian into a form which facilitates numerical calculations. To this end, in the next chapter we shall perform one more transformation of our system - moving from a honeycomb lattice to a square lattice.

2.0.3 Phases of the Kitaev Model

We shall now consider the topological phases exhibited by the Kitaev model. These phases may be swept through by tuning the hopping energies of the Majorana fermions: J_x, J_y, J_z and K. Fig 2.4 displays the phase diagram of the Kitaev model in the vortex free sector. It can be seen from the figure that the model exhibits four distinct topological phases, corresponding to a choice of the J terms. These phases have been labeled A_x, A_y, A_z, and B. The boundaries of these topological phases are given by the inequalities

$$|J_y| + |J_z| \leq |J_x|, \quad |J_z| + |J_x| \leq |J_y|, \quad |J_x| + |J_y| \leq |J_z|. \tag{2.18}$$

The system is considered to be in the B phase when all of the above inequalities are violated. The system is in the A_i phase ($i = x, y, z$) when the inequality $|J_a| + |J_b| \leq |J_i|$ holds, but the other two are violated. In order to facilitate this discussion, we shall only consider hopping energies with $1 \geq J_i \geq 0$ and $K \geq 0$. The topological phases given by negative hopping energies are similar.

We shall first consider the A phases. Each of these phases are distinct, but they are all phases which exhibit abelian anyons. We shall thus refer to these as *abelian phases*. In addition, each of these phases are gapped - a gap is present in the energy spectrum of the model while in these phases. As the eigenvalues of the

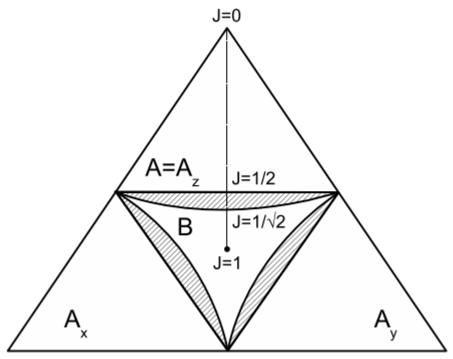


Figure 2.5: The topological phases of the Kitaev model in the vortex full sector. Retrieved from [20].

Hamiltonian come in positive-negative pairs, a gapped phase corresponds to the absence of Majorana zero modes (modes with zero energy). The behaviours of these phases are unchanged for any $K \geq 0$.

For the B phase, on the other hand, the K term does influence the properties of the phase. When $K = 0$, the B phase is gapless and exhibits no anyons. This B phase with $K = 0$ is a semi-metallic phase. By setting $K > 0$, we transform B into a gapped phase. In addition, with $K > 0$, the B phase becomes *non-abelian* - in this B phase the vortices of the system behave as non-abelian anyons.

We shall now consider how the topological phases of the Kitaev model change in the presence of vortices - ie. we shall study the nucleated phases of the system. First we shall consider the A phases of the model. Within the presence of vortices, the boundaries of these topological phases are enlarged. As can be seen in Fig 2.5, when a vortex is placed on each plaquette of the honeycomb lattice, the A phases of the model are enlarged by the shaded area. On the other hand, the non-abelian B phase of the Kitaev model transforms into a distinct abelian topological phase in the presence of vortices.

The nucleated phases of this system provide a steep challenge: within the non-abelian phase of the model, the vortices behave as non-abelian anyons, however as the number of vortices is increased, the topological phase becomes abelian.

Chapter 3

Calculations/Results

In this chapter we will go through the numerical and analytical results we have achieved relating to the Kitaev model. In this first part of this chapter we shall discuss the techniques used in performing numerical calculations for the model - specifically the transformation of the model to a square lattice, and the use of the correlation matrix to compute measures of entanglement in the system. We shall then explore the results obtained for the band structure of the model, the vortex binding of Majorana zero modes, the edges states, and the entanglement characterization.

3.0.1 Calculations

The Square Lattice with One Diagonal

In order to facilitate numerical calculations, we shall transform the honeycomb lattice into a square lattice. In making this transformation, we shall find a number of beneficial qualities, which we used to simplify the computations which needed to be performed in analyzing the model numerically:

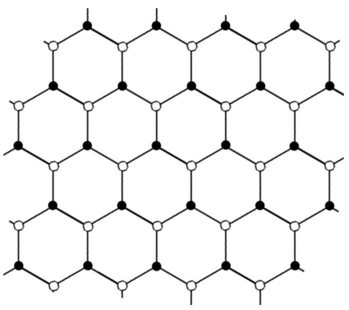
1. In making this transformation we reduce the number of explicit directions in which interactions must be programmed, from three to two.

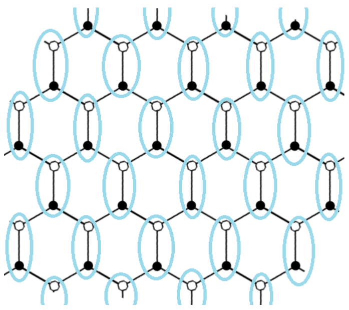
2. In going to the square lattice we lose next nearest neighbour interactions and will only have nearest neighbour interactions.

3. The square lattice lends itself naturally to the construction of *supercells* - smaller structures which contain the overall symmetry of the lattice, and can be periodically translated to reproduce the entire lattice.

The black and white colouration we have performed on the honeycomb lattice acts as an appropriate two site basis for our transformation to the square lattice. In Fig 3.1 we demonstrate how this two site basis can be used to transform to a square lattice with a single diagonal. Note that this new lattice is equivalent to a triangular lattice.

Looking at Fig 3.1c we can see that both nearest neighbour and next nearest neighbour hopping of the Majorana fermions between sites on the honeycomb lattice reduce to purely nearest neighbour hopping between the two site bases.


(a) The bicoloured honeycomb lattice.


(b) A 2 site basis of the lattice.

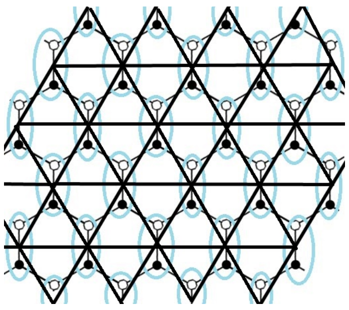

(c) Transformation to s square lattice with a single diagonal.

Figure 3.1: The steps in transforming the honeycomb lattice to a square lattice with one diagonal.

We shall now define creation and annihilation operators for each of the sites in the two site basis. Let the vector $\mathbf{r}$ point to a site on the square lattice. Let $a(\mathbf{r})$ be an operator which annihilates a Majorana fermion on the black site of the site $\mathbf{r}$ on the square lattice. Let $b(\mathbf{r})$ be the operator which annihilates a Majorana fermion on the white site of the site $\mathbf{r}$ on the square lattice. Since Majorana fermions are their own antiparticles, we have that $a(\mathbf{r}) = a^\dagger(\mathbf{r})$ and $b(\mathbf{r}) = b^\dagger(\mathbf{r})$ so the creation operators are equivalent to their annihilation operators.

Using these new operators, Hamiltonian (2.15) becomes:

$$\begin{aligned} H = \frac{-i}{4}\sum_{\mathbf{r}} & J_x a(\mathbf{r})b(\mathbf{r}-\hat{\mathbf{y}}) + J_y a(\mathbf{r})b(\mathbf{r}+\hat{\mathbf{x}}-\hat{\mathbf{y}}) + J_z a(\mathbf{r})b(\mathbf{r})\theta(\mathbf{r}) \\ & - K[a(\mathbf{r})a(\mathbf{r}+\hat{\mathbf{x}}) + a(\mathbf{r})a(\mathbf{r}-\hat{\mathbf{y}})\theta(\mathbf{r}-\hat{\mathbf{y}}) + a(\mathbf{r})a(\mathbf{r}+\hat{\mathbf{x}}-\hat{\mathbf{y}})\theta(\mathbf{r}+\hat{\mathbf{x}}-\hat{\mathbf{y}}) \\ & + b(\mathbf{r})b(\mathbf{r}+\hat{\mathbf{x}}) + b(\mathbf{r})b(\mathbf{r}-\hat{\mathbf{y}})\theta(\mathbf{r}) + b(\mathbf{r})b(\mathbf{r}+\hat{\mathbf{x}}-\hat{\mathbf{y}})\theta(\mathbf{r})] + \text{h.c.} \end{aligned} \tag{3.1}$$

where $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$ are unit vectors in the x and y directions, respectively. The coordinate axes of the lattice are given in Fig 3.2. The θ terms are defined such that $\theta = -1$ when a Majorana fermion hops through a link with corresponding link operator eigenvalue $u_{ij} = -1$, and $\theta = 1$ otherwise.

If our chosen vortex configuration displays translational symmetry, we can further break the total system into supercells, which can be periodically translated to give the entire system. We can break the Hamiltonian into an intracell term (describing interactions within the supercells) and an intercell term (describing interactions between cells):

$$H = H_{intra} + H_{inter}. \tag{3.2}$$

Now the Majorana hopping terms are given by $a(\mathbf{R},\mathbf{r})$ and $b(\mathbf{R},\mathbf{r})$, where the vector $\mathbf{R}$ points to a particular supercell, and the vector $\mathbf{r}$ points to a specific site within that supercell.

Assuming translational symmetry of the supercells, we can Fourier transform the Majorana hopping terms in the supercell label:

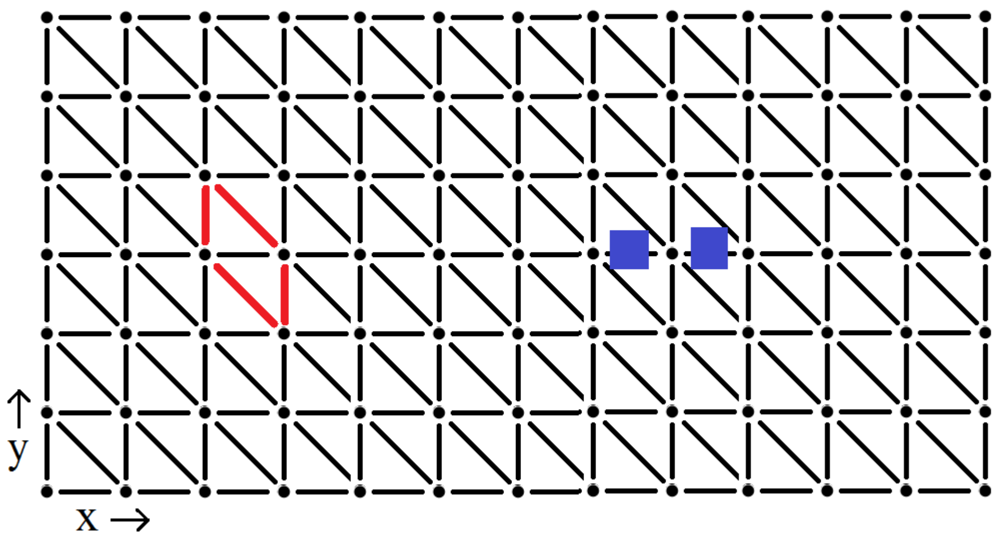


Figure 3.2: The square lattice with one diagonal. The red outline displays the appearance of the hexagons of the honeycomb lattice. Vortices are also shown, to illustrate how these impurities appear in this lattice structure.

$$\begin{aligned} a(\mathbf{R},\mathbf{r}) &= \sum_R e^{i\mathbf{R}\cdot\mathbf{p}} a(\mathbf{p},\mathbf{r}) \\ b(\mathbf{R},\mathbf{r}) &= \sum_R e^{i\mathbf{R}\cdot\mathbf{p}} b(\mathbf{p},\mathbf{r}). \end{aligned} \tag{3.3}$$

With this transformation performed, the total Hamiltonian becomes a sum over momentum $\mathbf{p}$, and position within a supercell $\mathbf{r}$. The Hamiltonian can be written in the form:

$$\sum_{\mathbf{p}} \begin{bmatrix} a_1(\mathbf{p}) \\ b_1(\mathbf{p}) \\ \vdots \\ a_n(\mathbf{p}) \\ b_n(\mathbf{p}) \end{bmatrix}^\dagger \begin{bmatrix} H_{1,1}(\mathbf{p}) & \dots & H_{1,2n}(\mathbf{p}) \\ & \vdots & \\ H_{2n,1}(\mathbf{p}) & \dots & H_{2n,2n}(\mathbf{p}) \end{bmatrix} \begin{bmatrix} a_1(\mathbf{p}) \\ b_1(\mathbf{p}) \\ \vdots \\ a_n(\mathbf{p}) \\ b_n(\mathbf{p}) \end{bmatrix} \tag{3.4}$$

where a_i is the Majorana hopping term a at the position $\mathbf{r} = i$ within the supercell, and n is the total number of sites in the supercell. Finding the eigenvalues of the Hamiltonian is then equivalent to diagonalizing the matrix (H) for each momentum value.

Choosing appropriate supercells within a given vortex configuration reduces the computation complexity significantly. Using this Hamiltonian we were able to perform the numerical calculations outlined in the following sections. In Appendix A we give the general supercell Hamiltonian, as well as the supercell Hamiltonians for the vortex free and vortex full sectors.

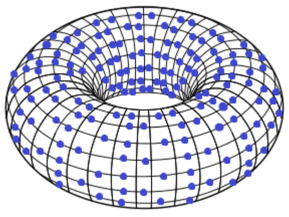


Figure 3.3: A vortex configuration wrapped around a torus.

The Correlation Matrix

The correlation matrix is an object which can be used to give several different measures of the entanglement of a system. Given a system, we may partition the total system into two subsystems, A and B. We can then compute the correlation matrix of the system through the following process:

The entries of the correlation matrix are given by

$$C_{ij} =< c_i c_j^\dagger > \tag{3.5}$$

where $c_i^\dagger$ is the creation operator for a particle at position i, and c_i is the corresponding annihilation operator.

One measure of entanglement which can be calculated using the correlation matrix is the *Renyi entropy* of the system. The n-th Renyi entropy can be related to the eigenvalues of the correlation matrix, λ_i, by

$$S_n = \frac{1}{n-1} \sum_i \ln[\lambda_i^n + (1-\lambda_i)^n]. \tag{3.6}$$

[See Appendix B for details.]

The *entanglement entropy* of a system is defined as

$$S = \lim_{n \to 1^+} S_n. \tag{3.7}$$

Taking this limit, we find that the entanglement entropy of a free fermion system in terms of the eigenvalues of the correlation matrix is given by

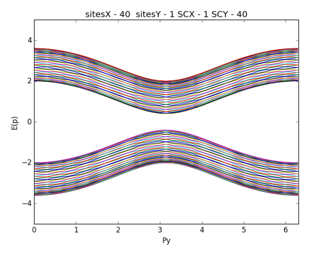
$$S = -\sum_i \lambda_i \ln \lambda_i + (1-\lambda_i) \ln(1-\lambda_i). \tag{3.8}$$

The entanglement entropy of a system is one way to measure its entanglement. A second measure of entanglement is given by the *entanglement spectrum* of the system. The entanglement spectrum is obtained by plotting the eigenvalues of the correlation matrix. Results from the use of both of these methods are presented later in this chapter.

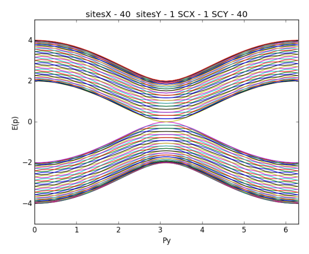
3.0.2 Results

Band Structure

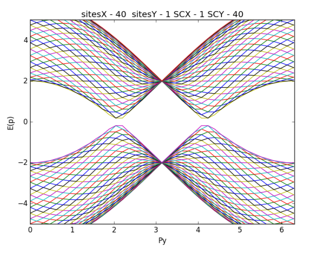
We shall begin the discussion of our results by considering the energy spectrum of the Kitaev model within different topological phases. We shall use a parameterization



(a) J = 0.4, K = 0

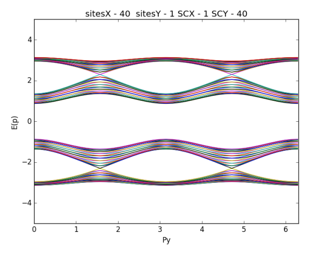


(b) J = 0.5, K = 0

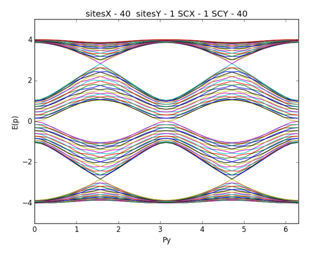


(c) J = 1, K = 0

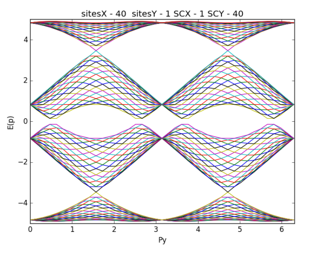
Figure 3.4: Energy spectra of the vortex free sector of the Kitaev model within various topological phases.



(a) J = 0.4, K = 0



(b) J = $\frac{1}{\sqrt{2}}$, K = 0



(c) J = 1, K = 0

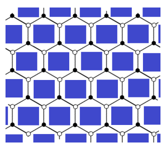
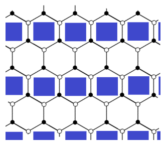
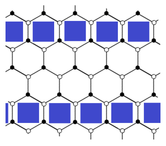
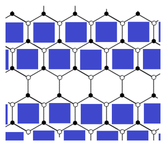
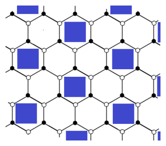
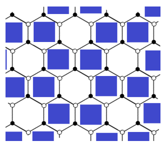
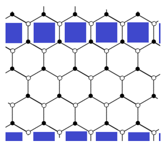
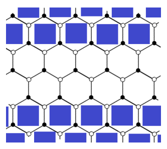
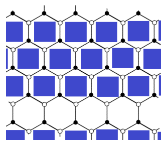
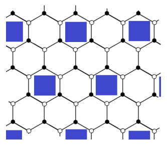
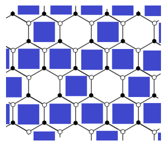
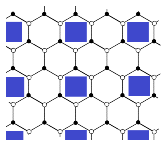
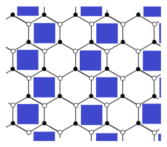
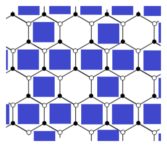
Figure 3.5: Energy spectra of the vortex full sector of the Kitaev model within various topological phases.

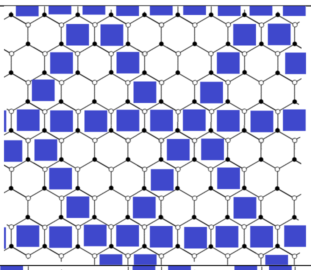
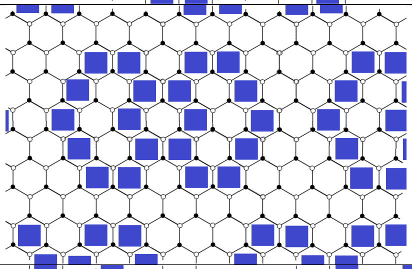
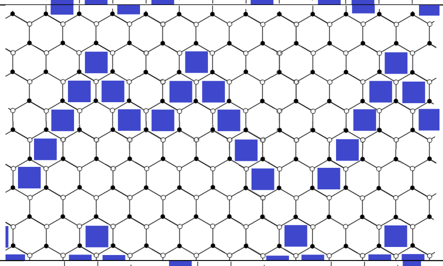
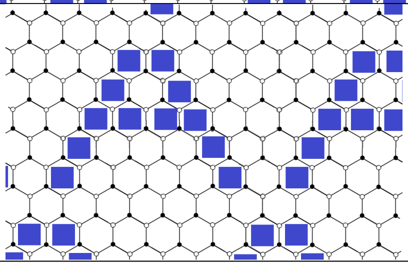
in which we define $J_x = J_y$ and restrict $J_z = 1$. We shall also assume periodic boundary conditions in both the $\hat{x}$ and $\hat{y}$ directions of the square lattice. This is equivalent to considering the Kitaev model wrapped around a torus.

Due to these periodic boundary conditions, we may choose to Fourier transform our Hamiltonian into momentum space, to simplify computations. We have chosen to Fourier transform only in the $\hat{y}$-direction. This choice will prove useful in comparing the results obtained in this section and those obtained when considering edge states. Due to the Fourier transform, the energy spectra shown below are plots of the eigenvalues of the Hamiltonian for various values of the momentum of the Majorana modes in the $\hat{y}$ direction. Transforming in one direction allows the model to be partitioned into separate chains. In the following figures the terms "sitesX" and "sitesY" refer to the dimensions of the individual chains, while the terms "SCX" and "SCY" refer to the number of chains in each direction.

In Fig 3.4 we show the band structure of the vortex free sector of the Kitaev model with the restriction $K = 0$. Under this restriction, the model sits within the abelian A phase in 3.4a, the phase on the border between the A and B phases in 3.4b, and the semi-metal B phase in 3.4c.

In Fig 3.5 we show the band structure for the vortex full sector of the model for similar topological phases: the A phase (3.5a), the border between the A and B phases (3.5b), and the B phase (3.5c). It can be seen that in the vortex full sector, four bands appear in the energy spectrum, whereas only two appear in the energy spectrum for the vortex free sector. The two central bands in the vortex full sector are vortex bands, while the outer bands are fermionic bands. Unsurprisingly, the vortex free sector exhibits only fermionic bands. It is possible to introduce a gap between the fermionic and vortex bands in the vortex full sector by setting $K > 0$.

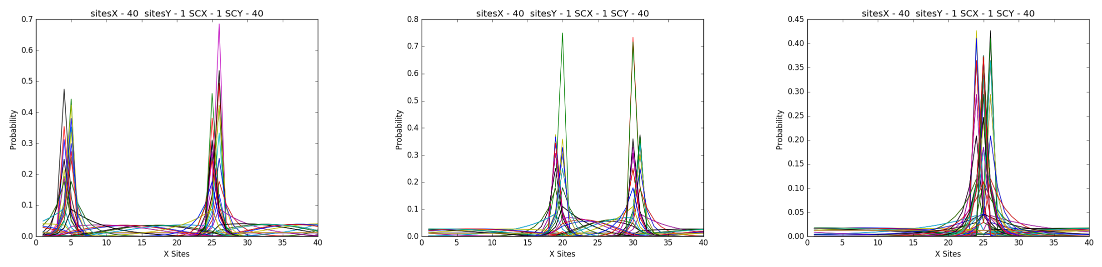
	Sector	Vortex Density	Energy per plaquette and per vortex
1		$\frac{1}{1}$	0.0669 0.0669
2		$\frac{1}{2}$	0.0519 0.1038
3		$\frac{1}{3}$	0.0414 0.1242
4		$\frac{2}{3}$	0.0538 0.0807
5		$\frac{1}{3}$	0.0259 0.0777
6		$\frac{2}{3}$	0.0600 0.0900
7		$\frac{1}{4}$	0.0340 0.1360
8		$\frac{2}{4}$	0.0424 0.0848
9		$\frac{3}{4}$	0.0587 0.0783
10		$\frac{1}{4}$	0.0416 0.1664
11		$\frac{3}{4}$	0.0741 0.0952
12		$\frac{1}{4}$	0.0253 0.1012
13		$\frac{2}{3}$	0.0461 0.0692
14		$\frac{3}{4}$	0.0718 0.0957

	Sector	Vortex Density	Energy per plaquette and per vortex
15		$\frac{11}{20}$	0.0548 0.0996
16		$\frac{20}{42}$	0.0438 0.0920
17		$\frac{14}{48}$	0.0332 0.1138
18		$\frac{12}{36}$	0.0339 0.1017

In Kitaev's original paper [21], he produced a table of numerical results containing the energy per vortex and energy per plaquette for various vortex configurations. We have reproduced Kitaev's results and tabulated them in the following tables. In these tables we have used the parameterization $J_x = J_y = J_z = 1$ and $K = 0$. The 14 results displayed in the first table are reproductions of the results published by Kitaev. The subsequent table contains four new vortex configurations which we analyzed.

We have found the energy of the vortex free sector to be $E_0 = -1.5746$, in agreement with Kitaev's results. By examining the following tables, it can be seen that the energy of every vortex configuration sits above the energy of the vortex free phase - the vortex free sector is the ground state of the Kitaev model.

Through examination of the tabulated results, we can see that the lowest energy per vortex is found in the vortex full sector. Indeed, in any configuration in which the vortices are pulled further away from each other, the energy of the vortices increases.



(a) Vortices at x = 5 and x = 25.

(b) Vortices at x = 20 and x = 30.

(c) Vortices at x = 24 and x = 26.

Figure 3.6: Vortex binding of Majorana zero modes.

Vortex Binding

As previously alluded to, it is possible for certain Majorana modes to bind to the vortices present in the Kitaev model. The only Majorana modes which undergo this binding are those modes with the smallest magnitude of energy. As the energy eigenvalues of the Kitaev model come in positive-negative pairs, there are always at least two modes which exhibit vortex binding. We have studied the behaviour of these lowest energy modes within the presence of a single vortex pair on each chain of the model.

Fig 3.6 displays the wavefunction of the lowest energy mode in the presence of a single vortex pair. Clear maxima can be seen around the locations of the vortices in the system. As the vortices are moved throughout the lattice, the mode remains bound.

We have thus found that vortex binding is indeed exhibited by the lowest energy Majorana modes. These Majorana bound states are known to exhibit non-abelian statistics [38]. As we have shown, these bound states remain stable, regardless of the proximity of the vortices in the vortex pair. Because of this, the controlled movement of vortices within the Kitaev model is a potential method for performing braiding operations between the non-abelian Majorana bound states. However, as we have only examined the Kitaev model within static vortex sectors, and have not studied the behaviour of the model in time, as we move between these vortex sectors, the true dynamical behaviour of the Majorana bound states is unknown.

Edge Modes

As the Kitaev Honeycomb Model is a topological system, the physics of the system depend strongly on the shape onto which it is embedded. While having periodicity in two directions is equivalent to wrapping the lattice around a torus, having periodicity in only one direction is equivalent to wrapping the system around a cylinder. Thus, we can consider breaking periodicity in one direction to be equivalent to cutting the torus into two cylinders.

We shall now consider the effects of breaking the periodicity in the $\hat{x}$ direction, while keeping the periodicity in $\hat{y}$ intact.

In Fig 3.7a we have plotted the energy spectrum for the vortex free phase wrapped around the cylinder. A distinct line passing through zero energy becomes present when we break periodicity in one direction. This line corresponds to a Majorana zero mode. Furthermore, the new modes exhibited when periodicity is broken are *edge states* of the system - they are localized at the edges of the system, where the torus was cut.

In Fig 3.7b we have plotted the wavefunction of one of the edge states. The various data points on the graph correspond to edge state wavefunctions with different momenta. It can be seen that regardless of the momentum of the edge mode, it is strongly localized at the edges of the system.

Entanglement Characterization

It has been shown by [39] that the entanglement entropy of the Kitaev model can be factored into a gauge field contribution, S_G and a fermionic contribution S_F:

$$S = S_G + S_F. \tag{3.9}$$

The gauge contribution to the entanglement entropy is known to be given by

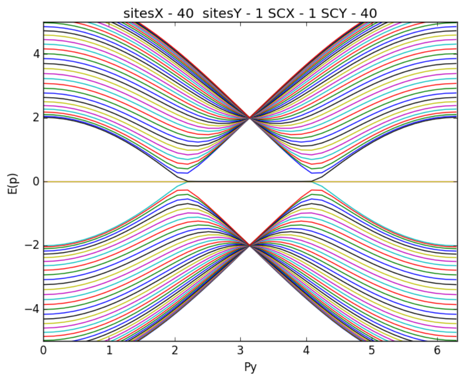
$$S_G = (L-1)\ln(2) \tag{3.10}$$

where L is the number of links of the honeycomb lattice which pass through the edge which defines the partition of the system into subsystems A and B.

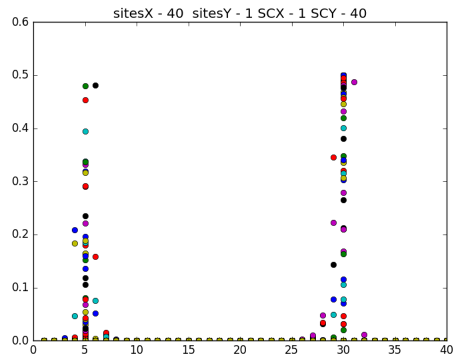
The fermionic contribution, on the other hand, is responsible for all of the non-trivial differences between the abelian and non-abelian phases of the honeycomb model. Thus, in the following discussion, we shall focus only on the fermionic sector.

As we have outlined in Section 3.0.1, measures of entanglement of the fermionic sector of the Kitaev model may be obtained using the correlation matrix. We shall begin by examining the entanglement spectra of various vortex sectors and topological phases of the model.

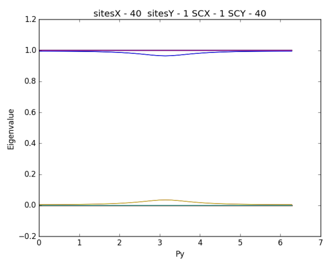
Fig 3.8 displays the entanglement spectra for three topological phases of the vortex free sector of the model - the A phase, (3.8a) the phase on the border between the A and B phases (3.8b), and the B phase (3.8c). Similarly, Fig 3.9 displays the entanglement spectra for the vortex full sector in the A phase (3.9a) the topological phase between the A and B phases (3.9b), and the B phase (3.9c).



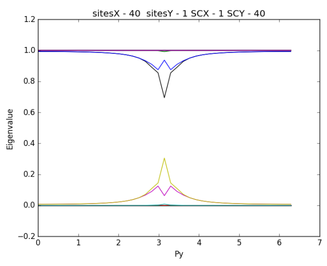
(a) The energy spectrum of the vortex free sector when wrapped around a cylinder. J = 1, K = 0.



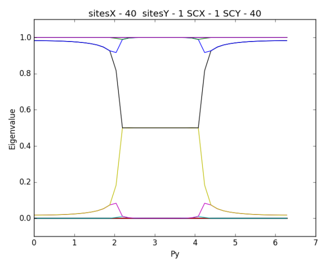
(b) Wavefunction of a Majorana mode of minimum energy of the vortex free sector, with cuts at x = 5 and x = 30.



(a) J = 0.4, K = 0

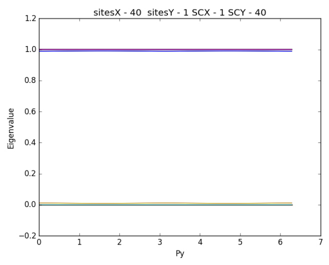


(b) J = 0.5, K = 0

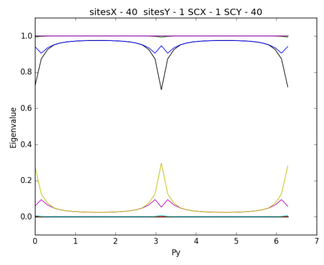


(c) J = 1, K = 0

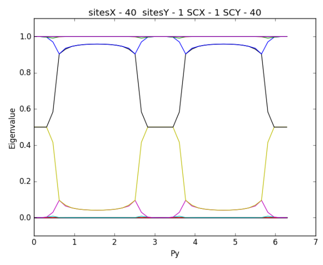
Figure 3.8: Entanglement spectra of the vortex free sector of the model.



(a) J = 0.4, K = 0



(b) J = $\frac{1}{\sqrt{2}}$, K = 0



(c) J = 1, K = 0

Figure 3.9: Entanglement spectra of the vortex full sector of the model.

It can be seen that even though the A phase of the vortex free sector exhibits slightly non-trivial behaviour, all of the interesting behaviour of the entanglement spectrum of the model occurs outside of the A phase.

It should be noted that in the B phase, the entanglement spectra of the vortex free and vortex full sectors exhibit distinctive lines at the eigenvalue of 0.5. These lines are reminiscent of the edge states which appeared when we cut the system. This is not a coincidence! The behaviour of the entanglement spectrum of a system is known to coincide with the behaviour of a system with broken periodicity. In fact, we have plotted the eigenvector corresponding to the entanglement eigenvalue 0.5, in Fig 3.10. It can be seen that this eigenvector behaves like an edge state in a system with cuts at $x = 1$ and $x = 20$.

However, it is important to understand that we have not physically altered the system when calculating its entanglement. Recall that when computing the entanglement of a system, the system is divided into two subsystems, one of which is ignored. In our computations we considered a square lattice of 40 sites in the $\hat{x}$ direction. We then only considered the subsection between $x = 1$ and $x = 20$. The result was a “edge state” localized at these boundaries, but the “edges” in this scenario correspond to the edges of our knowledge of the system, rather than its physical boundaries. The appearance of edge states in the entanglement spectrum is merely a consequence of our ignorance of the system. This is a remarkable result.

Finally, we shall examine the entanglement entropy of the system. In examining the entanglement entropy of the system, we can see that, again, the A phase of the vortex free sector exhibits more interesting behaviour than the A phase of the vortex full sector, but every other topological phase exhibits even more significant behaviour than the A phase.

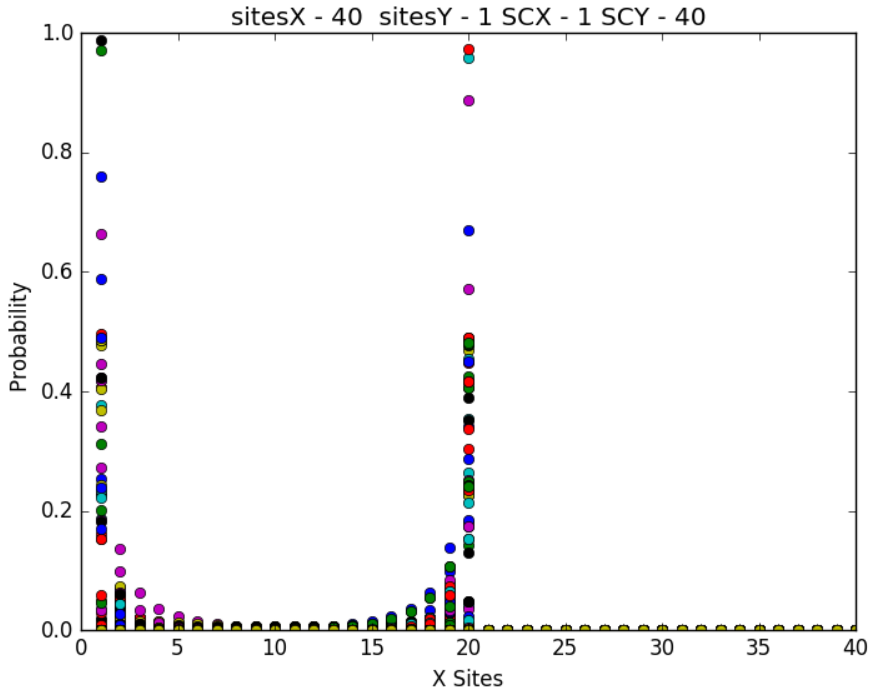
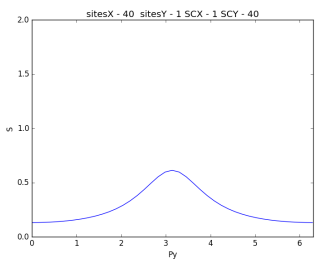
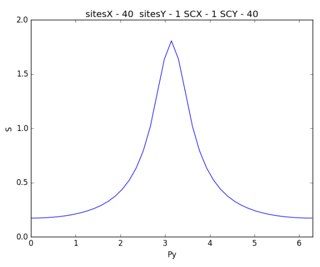


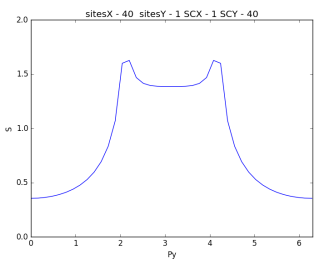
Figure 3.10: The "entanglement wavefunction" for the vortex free sector. Edges of the partition at x = 1 and x = 20.



(a) J = 0.4, K = 0

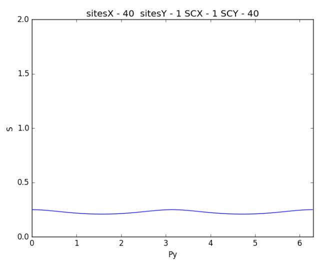


(b) J = 0.5, K = 0

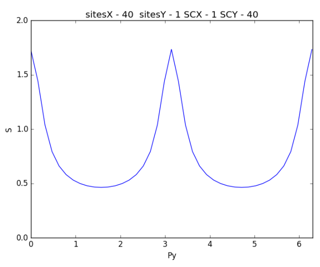


(c) J = 1, K = 0

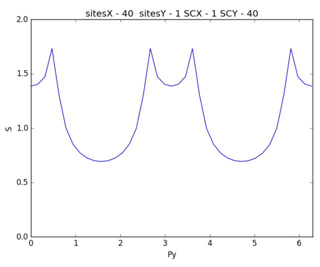
Figure 3.11: Entanglement entropy for the vortex free sector.



(a) J = 0.4, K = 0



(b) J = $\frac{1}{\sqrt{2}}$, K = 0



(c) J = 1, K = 0

Figure 3.12: Entanglement entropy for the vortex full sector.

Chapter 4

Conclusion

This thesis has provided an exploration into the Kitaev model. A overview of topics necessary to analyze the model, such as entanglement, Majorana fermions, and anyons, was provided. Results were obtained and discussed relating to the band structure and entanglement of various vortex sectors and topological phases of the model, as well as vortex binding and edge state analysis.

There are a number of directions in which future work may extend from this thesis. A complete characterization of the Kitaev model in terms of its entanglement is still lacking, and is something that should be pursued. In addition, the question of whether it is possible to determine information about a general vortex configuration from knowledge of the ground state may be given insight through continued study of the entanglement of the model under particular cuts.

A study of the Kitaev model under disorder may also provide fascinating insight: in two dimensional systems electrons undergo localization when disorder is induced. However, it has been shown that it is possible for Majorana fermions to enter a metallic phase under the introduction of disorder [40]. The behaviour of the Majorana fermions exhibited in the Kitaev model under disorder would be worthwhile to study.

It would also be of interest to study the Kitaev model outside of equilibrium. In particular, *quenches* may be performed, in which the model is initially placed into its ground state, and then the Hamiltonian is varied, and the state of the system is studied as it evolves in time under the new Hamiltonian. A study of the evolution of the entanglement of the model in time would be particularly interesting.

Appendix A

Supercell Hamiltonian

In this section we shall discuss how the Hamiltonians were obtained for general vortex configurations. We also give specific examples of the Hamiltonians for the vortex free and vortex full phases. We shall begin by discussing the general supercell Hamiltonian. All numerical results were obtained by considering appropriate supercells.

Let $\mathbf{r} = (x, y)$ be the vector pointing the site (x, y) on the square lattice supercell. Further, let N_x be the number of sites on the square lattice supercell in the $\hat{x}$ direction and let N_y be the number of sites on the square lattice supercell in the $\hat{y}$ direction.

In order present the equation as neatly as possible, we shall separate the Hamiltonian into four terms: The Hamiltonian treating interactions on the right edge of the supercell, the Hamiltonian treating the interactions on the bottom edge of the supercell, the Hamiltonian treating the interactions in the bottom right corner, and the Hamiltonian treating all of the other interactions (which we shall call the "central" Hamiltonian). We will factor out the term $\frac{-i}{4}$ from each of the four constituent Hamiltonians. Note that because of this, the +h.c. terms which normally appear in the Hamiltonians become −h.c.

Our form for the general supercell Hamiltonian then becomes:

$$H = \frac{-i}{4}[H_{central} + H_{rightedge} + H_{bottomedge} + H_{bottomright}] \tag{A.1}$$

where the above constituent Hamiltonians are given by:

$$\begin{aligned} H_{central} = \sum_{\substack{(x,y)\\ x\neq N_x\\ y\neq 1}} & J_x a(x,y)b(x,y-1) + J_y a(x,y)b(x+1,y-1) + J_z a(x,y)b(x,y)\theta(x,y) \\ & - K[a(x,y)a(x+1,y) + a(x,y)a(x,y-1)\theta(x,y-1) \\ & + a(x,y)a(x+1,y-1)\theta(x+1,y-1) + b(x,y)b(x+1,y) \\ & + b(x,y)b(x,y-1)\theta(x,y) + b(x,y)b(x+1,y-1)\theta(x,y)] - \text{h.c.} \end{aligned} \tag{A.2}$$

$$
\begin{aligned}
H_{rightedge} = \sum_{\substack{y \\ y\neq 1}} & J_x a(N_x,y)b(N_x,y-1) + e^{ip_x} J_y a(N_x,y)b(1,y-1) + J_z a(N_x,y)b(N_x,y)\theta(N_x,y) \\
& - K[e^{ip_x} a(N_x,y)a(1,y) + a(N_x,y)a(N_x,y-1)\theta(N_x,y-1) \\
& + e^{ip_x} a(N_x,y)a(1,y-1)\theta(1,y-1) + e^{ip_x} b(N_x,y)b(1,y) \\
& + b(N_x,y)b(N_x,y-1)\theta(N_x,y) + e^{ip_x} b(N_x,y)b(1,y-1)\theta(N_x,y)] - \text{h.c.}
\end{aligned} \tag{A.3}
$$

$$
\begin{aligned}
H_{bottomedge} = \sum_{\substack{x \\ x\neq N_x}} & e^{-ip_y} J_x a(x,1)b(x,N_y) + e^{-ip_y} J_y a(x,1)b(x+1,N_y) + J_z a(x,1)b(x,1)\theta(x,1) \\
& - K[a(x,1)a(x+1,1) + e^{-ip_y} a(x,1)a(x,N_y)\theta(x,N_y) \\
& + e^{-ip_y} a(x,1)a(x+1,N_y)\theta(x+1,N_y) + b(x,1)b(x+1,1) \\
& + e^{-ip_y} b(x,1)b(x,N_y)\theta(x,1) + e^{-ip_y} b(x,1)b(x+1,N_y)\theta(x,1)] - \text{h.c.}
\end{aligned} \tag{A.4}
$$

$$
\begin{aligned}
H_{bottomright} = {} & e^{-ip_y} J_x a(N_x,1)b(N_x,N_y) + e^{i(p_x-p_y)} J_y a(N_x,1)b(1,N_y) + J_z a(N_x,1)b(N_x,1)\theta(N_x,1) \\
& - K[e^{ip_x} a(N_x,1)a(1,1) + e^{-ip_y} a(N_x,1)a(N_x,N_y)\theta(N_x,N_y) \\
& + e^{i(p_x-p_y)} a(N_x,1)a(1,N_y)\theta(1,N_y) + e^{ip_x} b(N_x,1)b(1,1) \\
& + e^{-ip_y} b(N_x,1)b(N_x,N_y)\theta(N_x,1) + e^{i(p_x-p_y)} b(N_x,1)b(1,N_y)\theta(N_x,1)] - \text{h.c.}
\end{aligned} \tag{A.5}
$$

We shall now consider the form of this general Hamiltonian in two specific cases: the vortex free and vortex full sectors.

The Vortex Free Sector

We can compute the Hamiltonian of the vortex free sector by using a 1×1 unit cell. As there are no vortices in this sector, any single point on the square lattice can recreate the entire lattice under appropriate translations. Thus the Hamiltonian for the vortex free phase is

$$
\begin{aligned}
H_{VFree} = \frac{-i}{4} & [e^{-ip_y} J_x a(1,1)b(1,1) + e^{i(p_x-p_y)} J_y a(1,1)b(1,1) + J_z a(1,1)b(1,1) \\
& - K[e^{ip_x} a(1,1)a(1,1) + e^{-ip_y} a(1,1)a(1,1) + e^{i(p_x-p_y)} a(1,1)a(1,1) \\
& + e^{ip_x} b(1,1)b(1,1) + e^{-ip_y} b(1,1)b(1,1) + e^{i(p_x-p_y)} b(1,1)b(1,1)]] + \text{h.c.}
\end{aligned} \tag{A.6}
$$

Vortex Full

The Hamiltonian for the vortex full sector is computed by choosing a unit cell which is 2×1, in which either $\theta(1,1) = -1$ or $\theta(2,1) = -1$, but not both. We will choose $\theta(2,1) = -1$. The Hamiltonian for the vortex full phase is then

$$H_{VFull} = \frac{-i}{4}[H_{site1} + H_{site2}] \tag{A.7}$$

where the two constituent Hamiltonians are given by:

$$\begin{aligned} H_{site1} = {} & e^{-ip_y} J_x a(1,1)b(1,1) + e^{-ip_y} J_y a(1,1)b(2,1) + J_z a(1,1)b(1,1) \\ & - K[a(1,1)a(2,1) + e^{-ip_y} a(1,1)a(1,1) - e^{-ip_y} a(1,1)a(2,1) \\ & + b(1,1)b(2,1) + e^{-ip_y} b(1,1)b(1,1) + e^{-ip_y} b(1,1)b(2,1)] - \text{h.c.} \end{aligned} \tag{A.8}$$

$$\begin{aligned} H_{site2} = {} & e^{-ip_y} J_x a(2,1)b(2,1) + e^{i(p_x - p_y)} J_y a(2,1)b(1,1) - J_z a(2,1)b(2,1) \\ & - K[e^{ip_x} a(2,1)a(1,1) - e^{-ip_y} a(2,1)a(2,1) + e^{i(p_x - p_y)} a(2,1)a(1,1) \\ & + e^{ip_x} b(2,1)b(1,1) - e^{-ip_y} b(2,1)b(2,1) - e^{i(p_x - p_y)} b(2,1)b(1,1)] - \text{h.c.} \end{aligned} \tag{A.9}$$

General Vortex Sector

The Hamiltonians for general vortex configurations were obtained through the use of a computer. Code was created which contained all possible interaction directions. These interactions were then turned on and off, depending on the parameters given to the computer, such as supercell size and vortex locations. For more information, see Appendix C.

Appendix B

Entanglement from the Correlation Matrix

This chapter is devoted to showing how measures of entanglement appear from the correlation matrix. We begin by deriving the form of the correlation matrix in terms of eigenstates of the Hamiltonian. We then show how the eigenvalues of the correlation matrix can be used to compute the entanglement entropy of the system. We conclude by describing how the correlation matrix and entanglement entropy were computed in our numerical results.

B.0.1 Correlator of Two Ordinary Fermions

In this section we will derive the result for the correlator of two identical (non Majorana) fermions.

In particular we are interested in correlators of the form $< c_i c_j^\dagger >$ which we can use to construct a correlation matrix, like so:

$$C_{ij} =< c_i c_j^\dagger > . \tag{B.1}$$

The c_i and $c_j^\dagger$ terms are annihilation and creation operators for the fermions at sites i and j on the lattice, respectively. As the energy eigenfunctions of the Hamiltonian form a complete basis, we can transform these operators into our new basis:

$$c_i = \sum_n \phi_n(i)\gamma_n \tag{B.2}$$

where the operator γ_n represents the annihilation of a mode of momentum n in the system.

From here we can rewrite our correlator as

$$C_{ij} =< \sum_m \phi_m(i)\gamma_m \sum_n \phi_n^*(j)\gamma_n^\dagger > \tag{B.3}$$

$$C_{ij} = \sum_{m,n} \phi_m(i)\phi_n^*(j) < \gamma_m \gamma_n^\dagger >. \tag{B.4}$$

Now we can compute $< \gamma_m \gamma_n^\dagger >$ using the anticommutation relations of the fermionic creation and annihilation operators:

$$\{\gamma_m \gamma_n^\dagger\} = \delta_{m,n}. \tag{B.5}$$

Giving

$$< \gamma_m \gamma_n^\dagger >= \delta_{m,n} - < \gamma_m^\dagger \gamma_n > \tag{B.6}$$

$$< \gamma_m \gamma_n^\dagger >= \delta_{m,n} - \delta_{m,n} f(\epsilon_n) \tag{B.7}$$

$$< \gamma_m \gamma_n^\dagger >= \delta_{m,n}[1 - f(\epsilon_n)] \tag{B.8}$$

where $f(\epsilon_n)$ is the Fermi function. We will choose to analyze our system at absolute zero, thereby reducing the Fermi function to a step function:

$$f(\epsilon_n) = \begin{cases} 1 & \text{if } \epsilon_n < 0. \\ \frac{1}{2} & \text{if } \epsilon_n = 0. \\ 0 & \text{if } \epsilon_n > 0. \end{cases} \tag{B.9}$$

Substituting equation (B.8) into equation (B.4) gives

$$C_{ij} = \sum_{m,n} \phi_m(i)\phi_n^*(j)(\delta_{m,n}[1 - f(\epsilon_n)]) \tag{B.10}$$

$$C_{ij} = \sum_{n} \phi_n(i)\phi_n^*(j)[1 - f(\epsilon_n)]. \tag{B.11}$$

We will choose to keep only positive eigenvalues in the sum. This gives us the final result of

$$C_{ij} = \sum_{n} \phi_n(i)\phi_n^*(j) \tag{B.12}$$

where n sums over the positive eigenvalues.

B.0.2 Measures of Entanglement for Free Fermions

We shall now derive the form of the Renyi entropy and entanglement entropy for the case of free fermions.

It has been shown in [41] that the reduced density matrix of a free fermion (hopping) Hamiltonian must be the exponential of some free fermion operator:

$$\hat{\rho}_A = K \exp[-\hat{G}] \tag{B.13}$$

where K is a normalization constant and $\hat{G}$ is a free fermion operator.

Diagonalizing $\hat{G}$ obtains

$$\hat{\rho}_A = K \exp[-\sum_n \epsilon_n a_n^\dagger a_n] \tag{B.14}$$

where ϵ_n are the eigenvalues of $\hat{G}$ and a_n are fermionic operators.

Since we must have $\mathrm{Tr}(\hat{\rho}_A) = 1$, we find that the normalization constant is

$$K = \prod_i \frac{1}{1 + e^{-\epsilon_i}}. \tag{B.15}$$

We shall first relate the eigenvalues of the correlation matrix to the eigenvalues ϵ_i. The entries of the correlation matrix are given by

$$C_{ij} =< c_i c_j^\dagger >= \mathrm{Tr}(\hat{\rho}_A c_i c_j^\dagger). \tag{B.16}$$

We can write this as

$$\mathrm{Tr}(\hat{\rho}_A c_i c_j^\dagger) = \sum_\ell \langle \ell | \, K \exp(-\sum_n \epsilon_n a_n^\dagger a_n) c_i c_j^\dagger \, | \ell \rangle \, . \tag{B.17}$$

We can transform the c_i operators in terms of a basis of eigenfunctions for the eigenvalues, ϵ_i:

$$c_i = \sum_k \phi_k(i) a_k. \tag{B.18}$$

Using this, equation B.17 becomes:

$$\mathrm{Tr}(\hat{\rho}_A c_i c_j^\dagger) = \sum_{k,k',\ell} \langle \ell | \, K \exp(-\sum_n \epsilon_n a_n^\dagger a_n) \phi_{k'}(i) \phi_k^*(j) a_{k'} a_k^\dagger \, | \ell \rangle \, . \tag{B.19}$$

We can insert another identity, $\sum_L |L\rangle \langle L|$:

$$\begin{aligned}
\mathrm{Tr}(\hat{\rho}_A c_i c_j^\dagger) &= \sum_{k,k',\ell,L} \langle \ell | \, K \exp(-\sum_n \epsilon_n a_n^\dagger a_n) \, |L\rangle \langle L| \, a_{k'} a_k^\dagger \, |\ell\rangle \, \phi_{k'}(i) \phi_k^*(j) \\
&= \sum_{k,k',\ell,L} \langle \ell | \, K \exp(-\sum_n \epsilon_n a_n^\dagger a_n) \, |L\rangle \, \delta_{L,\ell} \delta_{k,k'} [1 - f(\epsilon_k)] \phi_{k'}(i) \phi_k^*(j) \\
&= \sum_k [\sum_\ell \langle \ell | \, K \exp(-\sum_n \epsilon_n a_n^\dagger a_n) \, |\ell\rangle] [1 - f(\epsilon_k)] \phi_k(i) \phi_k^*(j).
\end{aligned} \tag{B.20}$$

The first term in square brackets is simply the trace of the reduced density operator, and is thus 1. Also $f(\epsilon_k)$ is the Fermi-Dirac distribution. So

$$1 - f(\epsilon_k) = 1 - \frac{1}{e^{\epsilon_k} + 1} = \frac{1}{e^{-\epsilon_k} + 1} \tag{B.21}$$

and

$$C_{ij} = \mathrm{Tr}(\hat{\rho}_A c_i c_j^\dagger) = \sum_k \phi_k(i) \phi_k^*(j) \frac{1}{e^{-\epsilon_k} + 1}. \tag{B.22}$$

We can thus relate the eigenvalues of the correlation matrix, λ_i to the energy eigenvalues of the Hamiltonian by

$$\begin{aligned}
\lambda_i &= \frac{1}{e^{-\epsilon_i} + 1} \quad \text{or} \\
e^{\epsilon_i} &= \frac{\lambda_i}{1 - \lambda_i}.
\end{aligned} \tag{B.23}$$

We define the n-th Renyi entropy as

$$S_n = \frac{1}{n-1} \ln \mathrm{Tr}[\hat{\rho}_A^n]. \tag{B.24}$$

and the entanglement entropy as

$$S = \lim_{n \to 1^+} S_n. \tag{B.25}$$

Now consider the trace of $\hat{\rho}_A^n$:

$$\mathrm{Tr}\,\hat{\rho}_A^n = K^n \prod_i (1 + e^{-n\epsilon_i}). \tag{B.26}$$

So

$$\ln[\mathrm{Tr}\,\hat{\rho}_A^n] = n \ln K + \sum_i \ln(1 + e^{-n\epsilon_i}). \tag{B.27}$$

Using

$$1 + e^{-n\epsilon_i} = \frac{\lambda_i^n + (1-\lambda_i)^n}{\lambda_i^n} \tag{B.28}$$

we obtain

$$\ln[\mathrm{Tr}\,\hat{\rho}_A^n] = n \ln K + \sum_i [\ln \lambda_i^n + (1-\lambda_i)^n - n \ln \lambda_i]. \tag{B.29}$$

Using equation B.23, we find

$$\begin{aligned} -\sum_i n \ln \lambda_i &= n \sum_i \ln(\frac{1}{1+e^{-\epsilon_i}}) \\ &= -n \ln(\prod_i \frac{1}{1+e^{-\epsilon_i}}) \\ &= -n \ln K. \end{aligned} \tag{B.30}$$

Thus equation B.29 reduces to

$$\ln[\mathrm{Tr}\,\hat{\rho}_A^n] = \sum_i [\ln \lambda_i^n + (1-\lambda_i)^n]. \tag{B.31}$$

Hence the n-th Renyi entropy is given by:

$$S_n = \frac{1}{n-1} \sum_i \ln[\lambda_i^n + (1-\lambda_i)^n] \tag{B.32}$$

and taking the limit as $n \to 1^+$, we receive

$$S = -\sum_i \lambda_i \ln \lambda_i + (1-\lambda_i) \ln(1-\lambda_i). \tag{B.33}$$

B.0.3 Computing the Correlation Matrix

Constructing the correlation matrix for various vortex sectors was a necessity, due to the complexity of the Hamiltonians of the sectors. The entries of the correlation matrix were computed using equation B.12. The eigenstates used in this computation were the numerically computed eigenvectors of the Hamiltonian. Only those eigenvectors which corresponded to positive eigenvalues were used in computing the entries of the correlation matrix. The eigenvalues and eigenvectors of the correlation matrix were computed, and were used to calculate entanglement entropies, plot the entanglement spectra, and plot the "eigenvalue edge state" in Fig 3.10.

Appendix C

Codes

In this chapter we shall present and give a brief overview of each of the codes constructed for the computation of numerical results related to the Kitaev model. The following codes were written in Fortran.

C.0.1 Main Function

This section of code sets up all of the parameters required for the computation of the Hamiltonian. In the "Assignments" section, values can be changed which correspond to the positions of the vortices, hopping energies, cut locations, etc.

The code then calls the functions to compute the Hamiltonian and entanglement, and then prints the results for total energy and entanglement entropy. The eigenvalues obtained for the Hamiltonian and correlation matrix are written to a file, to be plotted.

```
program Honeycomb

implicit none

!——— Variables——————————————————

!———Parameters for setting up the lattice————

integer, parameter :: sitesX = 40, sitesY = 5
integer, parameter :: SCX = 2, SCY = 1 !Supercells
integer, parameter :: numPairs = 2 !Number of Vortex
   pairs on the lattice; −1 means vortex full
integer, parameter :: numCuts = 1 !Number of cuts
   through the system

integer, dimension(numPairs) :: VStartX, VStopX, VRow !
   Locations of each of the vortex pairs are stored in
   these arrays
integer :: keyCut, keyVConfig, TotNumPartinCut
integer :: Num0evinCorrelM, Num1evinCorrelM           !
   gives the num of bad ev in CorrelM in each pass
```

```
integer :: Tot0evinCorrelM, Tot1evinCorrelM          !tot
    number of 0 or 1 ev in CorrelM
integer, dimension(numCuts) :: cutStartX, cutStartY,
   cutStopX, cutStopY !Put these in cartesian coord ->
   they will converted later
integer, dimension(numCuts) :: numPartinCut !Number of
   particles in each of the cuts
integer, dimension(2) :: AcceptStart, AcceptStop

real, dimension(sitesX*sitesY, SCX*SCY) :: CorrelMEV
real, dimension(2*sitesX*sitesY, SCX*SCY) :: EnergyEV
complex, dimension(2*sitesX*sitesY*SCY*SCX) ::
   CompleteVector
real, dimension(2*sitesX*sitesY*SCY*SCX) ::
   CompleteVectorR

!--------LAPACK Parameters-------------------

integer, parameter :: LWMAX = 20000
integer, parameter :: LDHamilt = 2*sitesX*sitesY, LDVL =
    2*sitesX*sitesY, LDVR = 2*sitesX*sitesY

double precision, dimension(3*2*sitesX*sitesY) :: RWork
complex, dimension(LDHamilt, 2*sitesX*sitesY) :: Hamilt,
    VL, VR
real, dimension(2*sitesX*sitesY) :: eigens !    NOTE: If
    you make this double precision, it will not give
   correct eigenvalues!!!
complex, dimension(LWMAX) :: Work
integer :: info, LWork

!--------Interaction Energies / Energy Eigenvalues /
   Momentum / Entropy ------------------------------

double precision :: Jx, Jy, Jz, K, sumE
real, dimension(sitesX - 2) :: eigensums
real, dimension(2*sitesX*sitesY) :: firstev
double precision, dimension(2) :: momentum
real :: EntgEntropy, EntgEntTotal
!double precision :: momentum
!double precision, dimension(2*sitesX*sitesY) :: momVal
!real, dimension(2, 2*sitesX*sitesY) :: eigVal

!-------Counting Variables / Constants ----------

complex, parameter :: rootminus1 = (0,1.0)
double precision, parameter :: PI = acos(-1.0)
```

```
integer :: i,j, xx, ii, jj
integer :: cnt
integer :: gcd

!--------Assignments------------------------------------

LWork = -1
Jx = 1.0
Jy = 1.0
Jz = 1.0
K = 0.0

momentum = (/ dble(0.0), dble(0.0) /)
EntgEntropy = 0.0
EntgEntTotal = 0.0
VStartX = (/ 1, 40 /)                    !Right edge of first
    vortex
VStopX = (/ 1, 40 /)                     !Left edge of 2nd
   vortex
VRow = (/ 1, 1 /)                        !Row of vortex pair

cutStartX = (/ 30 /)
cutStartY = (/ 1 /)
cutStopX = (/ 30 /)
cutStopY = (/ sitesY /)

AcceptStart = (/ 3, 1 /)
AcceptStop = (/ 3, 4 /)

keyCut = 0                             ! -1 -> Just Accept.
   Cut
                                       ! 0 -> Nothing
                                       ! 1 -> Accept. Cut and
                                          other Cut
                                       ! 2 -> Just other Cut

cnt = 0
sumE = 0.0
TotNumPartinCut = 0
Num0evinCorrelM = 0
Num1evinCorrelM = 0
Tot0evinCorrelM = 0
Tot1evinCorrelM = 0

!---------Body of Program--------------------------------

if (keyCut .ne. 0) then
    do i = 1, numCuts
```

```
      !Find the number of particles in each cut
         numPartinCut(i) = 1 + gcd( (cutStopY(i) -
            cutStartY(i)) , (cutStopX(i) - cutStartX(i))
            )
      end do
   end if

   do i = 1, numCuts
      TotNumPartinCut = TotNumPartinCut + numPartinCut(i)
   end do

   do i = 1, sitesX-2
      eigensums(i) = 0.0
   end do

   call PositionRepSC(Hamilt, momentum, sitesX, sitesY, Jx,
      Jy, Jz, K, numPairs, VStartX, VStopX, VRow, keyCut,
     numCuts, cutStartX, cutStartY, cutStopX, cutStopY,
     numPartinCut, AcceptStart, AcceptStop)

   if (LWork == -1) then
      call CHEEV('V', 'U', 2*sitesX*sitesY, Hamilt,
         LDHamilt, eigens, Work, LWork, RWork, info)
      LWork = min(LWMAX, int(Work(1)))
   end if

   do i = 1, SCX
      do j = 1, SCY

         momentum = (/ 2.0*PI*dble(i)/dble(SCX) , 2.0*PI*
            dble(j)/dble(SCY) /)

         call PositionRepSC(Hamilt, momentum, sitesX,
            sitesY, Jx, Jy, Jz, K, numPairs, VStartX,
            VStopX, VRow, keyCut, numCuts, cutStartX,
            cutStartY, cutStopX, cutStopY, numPartinCut,
            AcceptStart, AcceptStop)

         call CHEEV('V', 'U', 2*sitesX*sitesY, Hamilt,
            LDHamilt, eigens, Work, LWork, RWork, info)

         do xx = 1, 2*sitesX*sitesY
            EnergyEV(xx, (SCY*(i - 1) + j) ) = eigens(xx
               )
         end do

         call Entanglement(Hamilt, sitesX, sitesY,
            EntgEntropy, Num0evinCorrelM, Num1evinCorrelM
```

```
                , CorrelMEV, SCX, SCY, (SCY*(i - 1) + j),
                eigens)

            Tot0evinCorrelM = Tot0evinCorrelM +
                Num0evinCorrelM
            Tot1evinCorrelM = Tot1evinCorrelM +
                Num1evinCorrelM
            EntgEntTotal = EntgEntTotal + EntgEntropy

            do xx = 1, 2*sitesX*sitesY
                sumE = sumE + abs(eigens(xx))
            end do

        end do
    end do

            sumE = sumE/(dble(4*(sitesX*sitesY -
                TotNumPartinCut)*SCX*SCY))
            sumE = (-1.0)*sumE + 1.5746

            EntgEntTotal = EntgEntTotal/(dble((4*(sitesX*
                sitesY - TotNumPartinCut)*SCX*SCY)))
            print *, ``Entanglement Entropy:'', EntgEntTotal
            print *, ""

            write(*, `` (A, f10.7) '') `` Energy per
                plaquette: '', sumE


        !Send Hamiltonian and Correlation matrix eigenvalues
            to file for printing

     open(2, file = ``correlmev.txt'', form = "formatted",
        status = "replace", action = "write")
     open(3, file = ``energyev.txt'', form = "formatted",
        status = "replace", action = "write")
     write(2, *) CorrelMEV
     write(3, *) EnergyEV

     close(2)
     close(3)


    end program Honeycomb
```

C.0.2 Entanglement

This subroutine constructs the correlation matrix from the energy eigenvectors of the Hamiltonian, using equation (B.12). Each time this subroutine is called, the eigenvectors of the Hamiltonian (which would have been computed for a specific momentum value) are passed into it, along with their corresponding eigenvalues. The correlation matrix is then constructed and diagonalized, and the eigenvalues are used to compute the entanglement entropy of the Hamiltonian. The eigenvalues of the correlation matrix are saved in CorrelMEV, which holds all of the eigenvalues of the correlation matrix for each momentum, and are passed back to the main function, so the entanglement spectrum can be plotted.

Additionally, if required, the eigenvectors of the correlation matrix can be passed back to the main function, for plotting (eg. in Fig 3.10).

```
subroutine Entanglement(Hamilt, sitesX, sitesY, EntgEntropy
    , num0, num1, CorrelMEV, SCX, SCY, whichPass, HamiltEV)

    implicit none

    integer :: sitesX, sitesY, count, whichPass, SCX, SCY
    complex, dimension(2*sitesX*sitesY, 2*sitesX*sitesY) ::
        Hamilt
    real, dimension(sitesX*sitesY, SCX*SCY) :: CorrelMEV
    real :: EntgEntropy, total
    integer ::i, j, xx
    real, dimension(2*sitesX*sitesY) :: HamiltEV

    !--------LAPACK Parameters-------------------------------

    integer, parameter :: LWMAX = 200000
    integer :: LDVL, LDVR, LDHamilt
    double precision, dimension(3*sitesX*sitesY) :: RWork
    complex, dimension(sitesX*sitesY, sitesX*sitesY) :: VL,
        VR
    complex, dimension(sitesX*sitesY, sitesX*sitesY) ::
        CorrelM
    real, dimension(sitesX*sitesY) :: eigens
    complex, dimension(LWMAX) :: Work
    integer :: info, LWork
    integer :: num1, num0

    !--------ASSIGNMENTS-------------------------------------

    LWork = -1
    LDVL = sitesX*sitesY
    LDVR = sitesX*sitesY
    LDHamilt = sitesX*sitesY
    count = 0
```

```
EntgEntropy = 0
num1 = 0
num0 = 0

!---------------------------------------------------------------

    !Zero the Correlation Matrix
do i = 1, sitesX*sitesY
    do j = 1, sitesX*sitesY
        CorrelM(i,j) = 0.0
    end do
end do

    !Run CHEEV once, to calibrate LWork
if (LWork == -1) then
    call CHEEV('V', 'U', sitesX*sitesY, CorrelM,
       LDHamilt, eigens, Work, LWork, RWork, info)
    LWork = min(LWMAX, int(Work(1)))
end if

    !Fill in entries in the  Correlation Matrix
do i = 1, sitesX*sitesY
    do j = 1, sitesX*sitesY
        do xx = (sitesX*sitesY + 1), 2*sitesX*sitesY
            CorrelM(i,j) = CorrelM(i,j) + Hamilt(i,xx)*
               conjg(Hamilt(j,xx))
        end do
    end do
end do

!Find eigenvalues of CorrelM

    call CHEEV('V', 'U', sitesX*sitesY, CorrelM,
       LDHamilt, eigens, Work, LWork, RWork, info)

!Find Entanglement entropy using ev of CorrelM

do i = 1, sitesX*sitesY
    CorrelMEV(i, whichPass) = eigens(i)
end do

do i = 1, sitesX*sitesY

    if ((abs(eigens(i)) <= 5E-7) .or. (abs(eigens(i) -
       1.0 ) < 3E-6)) then
    !Deal with small magnitudes to alleviate errors in
       rounding
        count = count + 1
```

```
                        !Keep a count of eigenvalues which
                           are 1 or 0
            if (abs(eigens(i)) <= 5E-7) then
                num0 = num0 + 1
            elseif (abs(eigens(i) - 1.0 ) < 3E-6) then
                num1 = num1 + 1
            else
                print *, ''ERROR: CORRELM ERR 2 ********''
                print *, ""
            end if

        elseif ( (eigens(i) < 1.0) .and. (eigens(i) > 0.0) )
           then
            EntgEntropy = EntgEntropy + (eigens(i)*log(
               eigens(i))) + ((1.0 - eigens(i))*log(1.0 -
               eigens(i)))
        else
            print *, ''ERROR: EV OF CORRELM >1 OR <0
               ********'', eigens(i)
            print *, ""
        end if

    end do

    EntgEntropy = EntgEntropy*(-1.0)

    end subroutine
```

C.0.3 Constructing the Hamiltonian

This subroutine constructs the supercell Hamiltonian, for a given vortex configuration, cut, and momentum values. This subroutine contains all possible interactions of lattice sites, organized by their position in the supercell in such a way as to speed up computation time. Before any of the interactions are computed, the subroutines CutSelection and VortexConfiguration are called, which have the potential to cause the program to ignore certain bonds (if they lie outside of the cut), or change the values of certain bonds (if they contain a negative $\theta(\mathbf{r})$ value).

```
subroutine PositionRepSC(Hamilt, momentum, sitesX, sitesY,
   Jx, Jy, Jz, K, numPairs, VStartX, VStopX, VRow, keyCut,
   numCuts, cutStartX, cutStartY, cutStopX, cutStopY,
   numPartinCut, AcceptStart, AcceptStop)
    !Two site basis

    implicit none

    double precision, dimension(2) :: momentum
```

```
integer :: sitesX , sitesY , numPairs, numCuts
complex, dimension(2*sitesX*sitesY , 2*sitesX*sitesY) ::
   Hamilt
double precision :: thetaR, thetaXY, thetaY
integer :: vortex , i , j
double precision :: Jx, Jy, Jz, K
complex, parameter :: rootminus1 = (0,1.0)
integer , dimension(numPairs) :: VStartX, VStopX, VRow
integer , dimension(numCuts) :: cutStartX , cutStartY ,
   cutStopX , cutStopY , numPartinCut
integer :: RightBond, DownBond, RightDownBond, ZBond
integer :: keyCut
integer :: Acceptable
integer , dimension(2) :: AcceptStart , AcceptStop

!————Fill in the matrix with zeros——————————

do i=1, 2*sitesX*sitesY
    do j =1, 2*sitesX*sitesY
         Hamilt(i,j) = 0.0
    end do
end do

!—————Fill in the bonds————————————————

do i=1, sitesX*sitesY
     !Make your bond basis: down, right , down and right
     !Then you only need to check if the position is on
         the right or bottom

     ! a = 2*i - 1
     ! b = 2*i

     call CutSelection(keycut , i , sitesX , sitesY ,
        RightBond, DownBond, RightDownBond, ZBond,
        Acceptable , AcceptStart , AcceptStop , numCuts,
        cutStartX , cutStartY , cutStopX , cutStopY ,
        numPartinCut)
     call VortexConfiguration(i , thetaR, thetaY , thetaXY,
         sitesX , sitesY , numPairs, VStartX, VStopX, VRow)


     !print *, "Pos - ", i
     !print *, "Accept?", Acceptable , " Right?",
        RightBond, " Down?", DownBond, " RD?",
        RightDownBond
     !print*, ""
```

```
if (Acceptable == 1) then
    !Only look at the particles/bonds within the cut

    !Z Bond
    Hamilt(2*i - 1, 2*i) = Jz*rootminus1*thetaR
    Hamilt(2*i, 2*i - 1) = conjg(Jz*rootminus1*
       thetaR)

    if (mod(i,sitesX) == 0) then !Right column

        !Right periodic bonds
        if (RightBond == 1) then

            Hamilt(2*i - 1, 2*(i-sitesX+1)-1) =
               -1.0*rootminus1*K*exp(rootminus1*
               momentum(1) ) + Hamilt(2*i - 1, 2*(i
               -sitesX+1)-1)
            Hamilt(2*(i-sitesX+1)-1, 2*i - 1) =
               conjg( -1.0*rootminus1*K*exp(
               rootminus1*momentum(1) ) ) + Hamilt
               (2*(i-sitesX+1)-1, 2*i - 1)

            Hamilt(2*i  , 2*(i-sitesX+1) ) = -1.0*
               rootminus1*K*exp(rootminus1*momentum
               (1) ) + Hamilt(2*i, 2*(i-sitesX+1) )
            Hamilt(2*(i-sitesX+1) , 2*i ) = conjg
               (-1.0*rootminus1*K*exp(rootminus1*
               momentum(1) ) ) + Hamilt(2*(i-sitesX
               +1) , 2*i )
        end if

        if (i <= sitesX) then !Bottom row
            !Periodic bottom right corner bonds
            if (RightDownBond == 1) then

                Hamilt(2*i - 1, 2*(sitesX*(sitesY
                   -1)+1) ) = Jy*rootminus1*exp(
                   rootminus1*(momentum(1) -
                   momentum(2))) + Hamilt(2*i - 1,
                   2*(sitesX*(sitesY-1)+1) )
                Hamilt(2*(sitesX*(sitesY-1)+1) , 2*
                   i - 1) = conjg(Jy*rootminus1*exp
                   (rootminus1*(momentum(1) -
                   momentum(2)))) + Hamilt(2*(
                   sitesX*(sitesY-1)+1) , 2*i - 1)

                Hamilt(2*i - 1, 2*(sitesX*(sitesY
                   -1)+1)-1 ) = -1.0*rootminus1*K*
                thetaXY*exp(rootminus1*(momentum
                (1) - momentum(2))) + Hamilt(2*i
                 - 1, 2*(sitesX*(sitesY-1)+1)-1
                )
            Hamilt(2*(sitesX*(sitesY-1)+1)-1 ,
                2*i - 1) = conjg(-1.0*rootminus1
                *K*thetaXY*exp(rootminus1*(
                momentum(1) - momentum(2)))) +
                Hamilt(2*(sitesX*(sitesY-1)+1)-1
                 , 2*i - 1)

            Hamilt(2*i , 2*(sitesX*(sitesY-1)
                +1) ) = -1.0*rootminus1*K*thetaR
                *exp(rootminus1*(momentum(1) -
                momentum(2))) + Hamilt(2*i, 2*(
                sitesX*(sitesY-1)+1) )
            Hamilt(2*(sitesX*(sitesY-1)+1) , 2*
                i ) = conjg(-1.0*rootminus1*K*
                thetaR*exp(rootminus1*(momentum
                (1) - momentum(2)))) + Hamilt
                (2*(sitesX*(sitesY-1)+1) , 2*i )
        end if

        !All the periodic bottom bonds
        if (DownBond == 1) then

            Hamilt(2*i - 1 , 2*(i + sitesX*(
                sitesY-1)) ) = Jx*rootminus1*exp
                (-1.0*rootminus1*momentum(2)) +
                Hamilt(2*i - 1 , 2*(i + sitesX*(
                sitesY-1)) )
            Hamilt(2*(i + sitesX*(sitesY-1)) ,
                2*i - 1) = conjg(Jx*rootminus1*
                exp(-1.0*rootminus1*momentum(2))
                ) +  Hamilt(2*(i + sitesX*(
                sitesY-1)) , 2*i - 1)

            Hamilt(2*i - 1 , 2*(i + sitesX*(
                sitesY-1)) - 1) = -1.0*
                rootminus1*K*thetaY*exp(-1.0*
                rootminus1*momentum(2)) + Hamilt
                (2*i - 1 , 2*(i + sitesX*(sitesY
                -1)) - 1)
            Hamilt(2*(i + sitesX*(sitesY-1)) -
                1 , 2*i - 1) = conjg(-1.0*
                rootminus1*K*thetaY*exp(-1.0*
                rootminus1*momentum(2))) +
                Hamilt(2*(i + sitesX*(sitesY-1))
                    - 1 , 2*i - 1)

                 Hamilt(2*i , 2*(i + sitesX*(sitesY
                    -1)) ) = -1.0*rootminus1*K*
                    thetaR*exp(-1.0*rootminus1*
                    momentum(2)) + Hamilt(2*i , 2*(i
                     + sitesX*(sitesY-1)) )
                 Hamilt(2*(i + sitesX*(sitesY-1)) ,
                    2*i ) = conjg(-1.0*rootminus1*K*
                    thetaR*exp(-1.0*rootminus1*
                    momentum(2))) + Hamilt(2*(i +
                    sitesX*(sitesY-1)) , 2*i )
              end if

           else !Right column, not bottom row

              !Partial periodic (right) right down
                 bonds
              if (RightDownBond == 1) then

                 !Will only fall off the edge on the
                     right, since it's not on the
                    bottom row
                 Hamilt(2*i - 1, 2*(i - (2*sitesX) +
                     1 )) = Jy*rootminus1*exp(
                    rootminus1*momentum(1)) + Hamilt
                    (2*i - 1, 2*(i - (2*sitesX) + 1
                    ))
                 Hamilt(2*(i - (2*sitesX) + 1 ) , 2*
                    i - 1) = conjg(Jy*rootminus1*exp
                    (rootminus1*momentum(1))) +
                    Hamilt(2*(i - (2*sitesX) + 1 ) ,
                     2*i - 1)

                 Hamilt(2*i - 1 , 2*(i - (2*sitesX)
                    + 1 )-1) = -1.0*rootminus1*K*
                    thetaXY*exp(rootminus1*momentum
                    (1)) + Hamilt(2*i - 1, 2*(i -
                    (2*sitesX) + 1 )-1)
                 Hamilt(2*(i - (2*sitesX) + 1 )-1 ,
                    2*i - 1) = conjg(-1.0*rootminus1
                    *K*thetaXY*exp(rootminus1*
                    momentum(1))) + Hamilt(2*(i -
                    (2*sitesX) + 1 )-1 , 2*i - 1)

                 Hamilt(2*i , 2*(i - (2*sitesX) + 1
                    ) ) = -1.0*rootminus1*K*thetaR*
                    exp(rootminus1*momentum(1)) +
                            Hamilt(2*i, 2*(i - (2*sitesX) +
                            1 ))
                        Hamilt(2*(i - (2*sitesX) + 1 ) , 2*
                            i ) = conjg(-1.0*rootminus1*K*
                            thetaR*exp(rootminus1*momentum
                            (1))) + Hamilt(2*(i - (2*sitesX)
                             + 1 ) , 2*i )
                    end if

                    !Non-periodic bottom bonds
                    if (DownBond == 1) then

                        Hamilt(2*i - 1, 2*(i-sitesX) ) = Jx
                            *rootminus1 + Hamilt(2*i - 1,
                            2*(i-sitesX) )
                        Hamilt(2*(i-sitesX), 2*i - 1 ) =
                            conjg(Jx*rootminus1) + Hamilt
                            (2*(i-sitesX), 2*i - 1 )

                        Hamilt(2*i - 1, 2*(i-sitesX) - 1 )
                            = -1.0*rootminus1*K*thetaY +
                            Hamilt(2*i - 1, 2*(i-sitesX) - 1
                             )
                        Hamilt(2*(i-sitesX) - 1, 2*i - 1 )
                            = conjg(-1.0*rootminus1*K*thetaY
                            ) + Hamilt(2*(i-sitesX) - 1, 2*i
                             - 1 )

                        Hamilt(2*i, 2*(i-sitesX) ) = -1.0*
                            rootminus1*K*thetaR + Hamilt(2*i
                            , 2*(i-sitesX) )
                        Hamilt(2*(i-sitesX), 2*i ) = conjg
                            (-1.0*rootminus1*K*thetaR) +
                            Hamilt(2*(i-sitesX), 2*i )

                    end if

                end if

            else !Not right column

                !All the non-periodic right bonds
                if (RightBond == 1) then
                    Hamilt(2*i -1 , 2*(i+1) - 1) = -1.0*
                        rootminus1*K + Hamilt(2*i -1 , 2*(i
                        +1) - 1)
                     Hamilt(2*(i+1) - 1 , 2*i -1) = conjg
                        (-1.0*rootminus1*K) + Hamilt(2*(i+1)
              - 1 , 2*i -1)

          Hamilt(2*i , 2*(i+1) ) = -1.0*
              rootminus1*K + Hamilt(2*i , 2*(i+1)
              )
          Hamilt(2*(i+1) , 2*i ) = conjg(-1.0*
              rootminus1*K) + Hamilt(2*(i+1) , 2*i
              )
      end if

       if (i < sitesX) then !Bottom row ->But not
           right corner

       !All the periodic bottom bonds
          if (DownBond == 1) then

              Hamilt(2*i - 1 , 2*(i + sitesX*(
                  sitesY-1)) ) = Jx*rootminus1*exp
                  (-1.0*rootminus1*momentum(2)) +
                  Hamilt(2*i - 1 , 2*(i + sitesX*(
                  sitesY-1)) )
              Hamilt(2*(i + sitesX*(sitesY-1)) ,
                  2*i - 1) = conjg(Jx*rootminus1*
                  exp(-1.0*rootminus1*momentum(2))
                  ) + Hamilt(2*(i + sitesX*(
                  sitesY-1)) , 2*i - 1)

              Hamilt(2*i - 1 , 2*(i + sitesX*(
                  sitesY-1)) - 1) = -1.0*
                  rootminus1*K*thetaY*exp(-1.0*
                  rootminus1*momentum(2)) + Hamilt
                  (2*i - 1 , 2*(i + sitesX*(sitesY
                  -1)) - 1)
              Hamilt(2*(i + sitesX*(sitesY-1)) -
                  1 , 2*i - 1) = conjg(-1.0*
                  rootminus1*K*thetaY*exp(-1.0*
                  rootminus1*momentum(2))) +
                  Hamilt(2*(i + sitesX*(sitesY-1))
                   - 1 , 2*i - 1)

              Hamilt(2*i , 2*(i + sitesX*(sitesY
                  -1)) ) = -1.0*rootminus1*K*
                  thetaR*exp(-1.0*rootminus1*
                  momentum(2)) + Hamilt(2*i , 2*(i
                   + sitesX*(sitesY-1)) )
              Hamilt(2*(i + sitesX*(sitesY-1)) ,
                  2*i ) = conjg(-1.0*rootminus1*K*
                  thetaR*exp(-1.0*rootminus1*
                momentum(2))) + Hamilt(2*(i +
                sitesX*(sitesY-1)) , 2*i )
            end if

            !Partial periodic (down) down right
               bond
            if (RightDownBond == 1) then
                !Will only fall off the bottom edge
                Hamilt(2*i - 1, 2*(i + sitesX*(
                   sitesY-1)+1) ) = Jy*rootminus1*
                   exp(-1.0*rootminus1*momentum(2))
                    + Hamilt(2*i - 1, 2*(i + sitesX
                   *(sitesY-1)+1) )
                Hamilt(2*(i + sitesX*(sitesY-1)+1)
                   , 2*i - 1) = conjg(Jy*rootminus1
                   *exp(-1.0*rootminus1*momentum(2)
                   )) + Hamilt(2*(i + sitesX*(
                   sitesY-1)+1) , 2*i - 1)

                Hamilt(2*i - 1, 2*(i + sitesX*(
                   sitesY-1)+1) - 1) = -1.0*
                   rootminus1*K*thetaXY*exp(-1.0*
                   rootminus1*momentum(2)) + Hamilt
                   (2*i - 1, 2*(i + sitesX*(sitesY
                   -1)+1) - 1)
                Hamilt(2*(i + sitesX*(sitesY-1)+1)
                   - 1, 2*i - 1) = conjg(-1.0*
                   rootminus1*K*thetaXY*exp(-1.0*
                   rootminus1*momentum(2))) +
                   Hamilt(2*(i + sitesX*(sitesY-1)
                   +1) - 1, 2*i - 1)

                Hamilt(2*i , 2*(i + sitesX*(sitesY
                   -1)+1) ) = -1.0*rootminus1*K*
                   thetaR*exp(-1.0*rootminus1*
                   momentum(2)) + Hamilt(2*i , 2*(i
                    + sitesX*(sitesY-1)+1) )
                Hamilt(2*(i + sitesX*(sitesY-1)+1)
                   , 2*i ) = conjg(-1.0*rootminus1*
                   K*thetaR*exp(-1.0*rootminus1*
                   momentum(2))) + Hamilt(2*(i +
                   sitesX*(sitesY-1)+1) , 2*i )
            end if

        else !Not bottom or right

           !Non-periodic bottom bonds
           if (DownBond == 1) then
        Hamilt(2*i - 1, 2*(i-sitesX) ) = Jx
           *rootminus1 + Hamilt(2*i - 1,
           2*(i-sitesX) )
        Hamilt(2*(i-sitesX), 2*i - 1 ) =
           conjg(Jx*rootminus1) + Hamilt
           (2*(i-sitesX), 2*i - 1 )

        Hamilt(2*i - 1, 2*(i-sitesX) - 1 )
           = -1.0*rootminus1*K*thetaY +
           Hamilt(2*i - 1, 2*(i-sitesX) - 1
            )
        Hamilt(2*(i-sitesX) - 1, 2*i - 1 )
           = conjg(-1.0*rootminus1*K*thetaY
           ) + Hamilt(2*(i-sitesX) - 1, 2*i
            - 1 )

        Hamilt(2*i, 2*(i-sitesX) ) = -1.0*
           rootminus1*K*thetaR + Hamilt(2*i
           , 2*(i-sitesX) )
        Hamilt(2*(i-sitesX), 2*i ) = conjg
           (-1.0*rootminus1*K*thetaR) +
           Hamilt(2*(i-sitesX), 2*i )
end if

 !Non periodic down right
 if (RightDownBond == 1) then

        Hamilt(2*i - 1, 2*(i + 1 - sitesX)
           ) = Jy*rootminus1 + Hamilt(2*i -
            1, 2*(i + 1 - sitesX) )
        Hamilt(2*(i + 1 - sitesX), 2*i - 1
           ) = conjg(Jy*rootminus1) +
           Hamilt(2*(i + 1 - sitesX) , 2*i
           - 1 )

        Hamilt(2*i - 1, 2*(i + 1 - sitesX)
           - 1) = -1.0*rootminus1*K*thetaXY
            + Hamilt(2*i - 1, 2*(i + 1 -
           sitesX) - 1)
        Hamilt(2*(i + 1 - sitesX) - 1, 2*i
           - 1) = conjg(-1.0*rootminus1*K*
           thetaXY) + Hamilt(2*(i + 1 -
           sitesX) - 1, 2*i - 1)

        Hamilt(2*i , 2*(i + 1 - sitesX) ) =
             -1.0*rootminus1*K*thetaR +
           Hamilt(2*i , 2*(i + 1 - sitesX))
```

```
                              Hamilt(2*(i + 1 - sitesX) , 2*i ) =
                                  conjg(-1.0*rootminus1*K*thetaR)
                                  + Hamilt(2*(i + 1 - sitesX) ,
                                 2*i)
                         end if

                    end if

               end if

          end if

     end do

     Hamilt = 2.0*Hamilt

     !---------------------------------------------------------------

     end subroutine
```

C.0.4 Vortex Configurations

This subroutine computes the values of thetaR, thetaY, and thetaXY, given the locations of the vortices in the lattice. These theta terms are passed into PositionRepSC, where their values determine the type of interactions that are placed into the Hamiltonian.

```
subroutine VortexConfiguration(position, thetaR, thetaY,
    thetaXY, sitesX, sitesY, numPairs, VStartX, VStopX, VRow)

     implicit none

     integer :: position, numPairs
     double precision :: thetaR, thetaY, thetaXY
     integer :: sitesX, sitesY
     integer, dimension(numPairs) :: VStartX, VStopX, VRow
     integer :: i

          !Set up x vortex pairs
          !VStartX, VStopX, VRow, are all arrays

          if (numPairs == 0) then
               thetaR = 1.0
               thetaXY = 1.0
               thetaY = 1.0

          elseif (numPairs < 0 ) then
               !Vortex full
```

```
        if (mod(mod(position,sitesX),2) == 1) then
        !Odd sites have vortex-links, EXCEPT if the last
            site in the x is odd
        !In that case we will end up having a one
           plaquette break before coming back with the
           pattern of vortices
        !This may be really useful
        !For vortex full use sitesX = even ; sitesX =
           odd will give this other interesting pattern
            thetaR = -1.0
            thetaY = -1.0
            thetaXY = 1.0
        else
            thetaR = 1.0
            thetaY = 1.0

            if (mod(position + 1,sitesX) == 0) then
                !if the second last column then thetaXY
                   must be 1.0, since the edges must
                   have no vortex links
                thetaXY = 1.0
            else
                !Otherwise, the next in the series has a
                    vortex-link
                thetaXY = -1.0

            endif
        end if

    else

        !First set it up so that there are no
           interactions through vortex bonds
        !Then add in the specific through-bond
           interactions
        !This way, multiple vortex pairs don't overwrite
            each other's interactions

        thetaR = 1.0
        thetaXY = 1.0
        thetaY = 1.0

        do i = 1, numPairs

            if (position >= (VStartX(i) + sitesX*(VRow(i
               ) - 1)) .and. position <= (VStopX(i) +
               sitesX*(VRow(i) - 1)) ) then
```

```
!Between the vortices - on the same row
    thetaR = -1.0
end if

if ( mod(position,sitesX) == 0) then
!If in the last column then thetaXY goes
   over the periodic boundary

    if ( ( (position + 1 - (2*sitesX) ) ) >=
        (VStartX(i) + sitesX*(VRow(i) - 1))
       .and. (position + 1 - (2*sitesX) ) <=
        (VStopX(i) + sitesX*(Vrow(i) - 1)) )
         then
        thetaXY = -1.0
    end if

end if

if (position <= sitesX) then
!First row needs to look at interactions
   through the periodic boundary

    if ( (position + sitesX*(sitesY - 1) )
       >= (VStartX(i) + sitesX*(VRow(i) - 1)
       ) .and. (position + sitesX*(sitesY -
       1) ) <= (VStopX(i) + sitesX*(Vrow(i)
       - 1)) ) then
        thetaY = -1.0
    end if

    if ( (position + sitesX*(sitesY - 1) + 1
        ) >= (VStartX(i) + sitesX*(VRow(i) -
        1)) .and. (position + sitesX*(sitesY
        - 1) + 1 ) <= (VStopX(i) + sitesX*(
       Vrow(i) - 1)) ) then

        thetaXY = -1.0
    end if

else

    if (position == (VStartX(i) + sitesX*
       VRow(i) - 1) .and. (VStartX(i) .ne.
       1) ) then
    !Above first vortex and to the left
        thetaXY = -1.0
    end if
```

```
                    if (position >= (VStartX(i) + sitesX*
                       VRow(i)) .and. position <= (VStopX(i)
                        + sitesX*Vrow(i)) ) then
                    !Row above vortices (and between them)

                        thetaY = -1.0

                        if (position .ne. (VStopX(i) +
                           sitesX*VRow(i)) ) then

                            thetaXY = -1.0

                        end if
                    end if
                end if
            end do
        end if

    end subroutine
```

C.0.5 System Cuts

This subroutine takes in the starting and ending points of any number of cuts and computes the points which lie in the cut and, based on the value of the key parameter, determines whether to tell the Hamiltonian to ignore these points (key = 2), ignore these points and partition the lattice using this cut (key = 1), to not tell the Hamiltonian anything (key = 0), or just tell the Hamiltonian to partition the lattice using the cut (key = −1).

This subroutine handles cuts which are vertical, horizontal, diagonal (positive and negative slope) and single points.

```
subroutine CutSelection(key, position, sitesX, sitesY,
   RightBond, DownBond, RightDownBond, ZBond, Acceptable,
   AcceptStart, AcceptStop, numCuts, cutStartX, cutStartY,
   cutStopX, cutStopY, numPartinCut)
    !This can't handle one col with a partial vertical cut
       or one row with partial horizontal cut

    implicit none

    integer :: key, sitesX, sitesY, position, RightBond,
       DownBond, RightDownBond, ZBond, Acceptable
    integer :: numCuts
    integer :: TypeofLine, LineSlope
    integer, dimension(numCuts) :: cutStartX, cutStartY,
       cutStopX, cutStopY, numPartinCut
    integer, dimension(2) :: AcceptStart, AcceptStop
```

```
real :: slopeCut , slopePos , postX , postY , postXRD ,
   postYRD
real :: XposOnLine, YposOnCol, XposOnRD, YposOnRD !The x
   -coord of the line as it passes through the row which
     the current position is on
integer :: i , j , checkDB , RightBreakOnThisCut ,
   DownBreakOnThisCut , WithinCutP1 , AboveCutPeriodic


RightBond = 1
DownBond = 1
RightDownBond = 1
ZBond = 1
Acceptable = 1

postX = real(mod((position - 1), sitesX) + 1)
postY = real( 1 + ((position - 1)/sitesX) )

!Position of right down
if ( postX == sitesX ) then
     postXRD = 1.0
else
     postXRD = postX + 1.0
end if

if ( postY == 1 ) then
     postYRD = sitesY
else
     postYRD = postY - 1.0
end if


if (abs(key) == 1) then
!Acceptibility Test (Method 1)

     if ( (AcceptStart(1) == AcceptStop(1)) .and. (
        AcceptStart(2) == AcceptStop(2)) ) then

          if (position == sitesX*sitesY) then
          !only print the error message at the end
               print *, ""
               print *, "Error:␣Only␣one␣point␣in␣
                  Acceptibility␣cut"
               print *, ""
          end if

          Acceptable = 0
          return
```

```
elseif (AcceptStart(1) == AcceptStop(1)) then
!Vertical line

    if (AcceptStart(1) > 1) then
        if ( postX < AcceptStart(1) ) then
        !Only take positions to the left of the cut
            Acceptable = 1
        else
            Acceptable = 0
            return
        end if

    else
    !If the vertical cut is through the 1st col
        if ( postX > AcceptStart(1) ) then
        !Only take positions to the right of the cut
            Acceptable = 1
        else
            Acceptable = 0
            return
        end if

    end if

elseif (AcceptStart(2) == AcceptStop(2)) then
!Horitonzal Line

    if (AcceptStart(2) < sitesY) then

        if ( postY > AcceptStart(2) ) then
        !Only take positions above the cut
            Acceptable = 1
        else
            Acceptable = 0
            return
        end if

    else
    !If horizontal cut is through top row

        if ( postY < AcceptStart(2) ) then
        !Only take positions below the cut
            Acceptable = 1
        else
            Acceptable = 0
            return
        end if
```

```
            end if

        else
            slopeCut = real(AcceptStop(2) - AcceptStart(2))/
               real(AcceptStop(1) - AcceptStart(1))
            XposOnLine = ( (postY - real(AcceptStart(2)) ) /
                slopeCut) + real(AcceptStart(1))

            if (slopeCut > 0) then
            !Positive line
                !Take only the points to the left of the cut
                if ( (XposOnLine - postX) > 0) then
                    Acceptable = 1
                else
                    Acceptable = 0
                    return
                end if

            elseif (slopeCut < 0) then
            !Negative line
                !Take only the points to the left of the cut
                if ( (XposOnLine - postX) > 0) then
                    Acceptable = 1
                else
                    Acceptable = 0
                    return
                end if
            else
                print *, "ERROR: EC1"
            end if

        endif
    end if

    if (key > 0) then

        do i = 1, numCuts
        !Go through each cut and see how position relates to
            the cuts

            RightBreakOnThisCut = 0
            DownBreakOnThisCut = 0
            WithinCutP1 = 0
            AboveCutPeriodic = 0

            if ( ( postY <= real(max(cutStartY(i),cutStopY(i
               )) + 1) ) .and. ( postY >= ( min(cutStartY(i)
            ,cutStopY(i)) ) ) ) then
             WithinCutP1 = 1
             if (postY == real( max(cutStartY(i),cutStopY
                (i)) + 1) ) then
                 AbovecutPeriodic = 1
             end if
        end if

        if (max(cutStartY(i),cutStopY(i)) == sitesY )
           then
             !This is something specifically for vertical
                 lines
             if ( postY == 1.0 ) then
                 if ( (postX == cutStartX(i)) .and. (
                    postX == cutStopX(i)) ) then
                     !This needs to be .and. otherwise
                        some lines may run incorrectly
                     if ( postY == min(cutStartY(i),
                        cutStopY(i)) ) then
                         !If you are in the cut
                         Acceptable = 0
                         return
                     end if
                 end if

                 WithinCutP1 = 1
                 AboveCutPeriodic = 1
             end if
        end if

        if (WithinCutP1 == 1) then
        !if ( ( 1 + ((position - 1)/sitesX) <= real(max(
           cutStartY(i),cutStopY(i)) + 1) ) .and. ( 1 +
           ((position - 1)/sitesX) >= ( min(cutStartY(i)
           ,cutStopY(i)) ) ) ) then
        !Within the top and bottom of cut (with +1 row
           on top) -> this is to account for the down
           and rightdown bonds

             if ( cutStopX(i) == cutStartX(i) ) then
!Vertical or point (infinite slope)

                 if ( cutStopY(i) == cutStartY(i) ) then
!Only one point in cut
                     if ( (postX == real(cutStartX(i)) )
                        .and. ( postY == real(cutStartY(i
                        )) ) ) then
                     !Position is the cut point
        Acceptable = 0
    end if

    if (postX == real(sitesX) ) then
    !On right edge

        if ( (cutStartX(i) == 1) .and. (
           real(cutStartY(i)) == postY)
           ) then
        !Periodic right bond
            Rightbond = 0
        end if

        if (postY == 1.0) then
        !In bottom right corner
            if ( (cutStartX(i) == 1) .
               and. (cutStartY(i) ==
               sitesY) ) then
                RightDownBond = 0
            end if

            if ( (cutStartX(i) == sitesX
               ) .and. (cutStartY(i) ==
               sitesY) ) then
                DownBond = 0
            end if
        else
        !On right edge, not in bottom
           corner

            if ( (cutStartX(i) == 1) .
               and. (real(cutStartY(i))
               == (postY - 1.0 )) ) then
                RightDownBond = 0
            end if

            if ( (cutStartX(i) == sitesX
               ) .and. (real(cutStartY(i
               )) == (postY - 1.0 )) )
               then
                DownBond = 0
            end if

        end if

    elseif (postY == 1.0) then

    !Bottom but not bottom right
```

```
        if ( (cutStartY(i) == 1) .and. (
            real(cutStartX(i)) == (postX
            + 1.0) ) ) then
        !Cut to the right on same line
            RightBond = 0
        end if

        if ( (cutStartY(i) == sitesY) .
           and. ( real(cutStartX(i)) ==
           postX ) ) then
        !Cut on periodic down
            DownBond = 0
        end if

        if ( (cutStartY(i) == sitesY) .
           and. ( real(cutStartX(i)) ==
           (postX + 1.0) ) ) then
        !Cut on periodic down right
            RightDownBond = 0
        end if

    else
    !Not bottom or right edge
        if ( ( real(cutStartY(i)) ==
           postY) .and. ( real(cutStartX
           (i)) == (postX + 1.0) ) )
           then
        !Cut to the right on same line
            RightBond = 0
        end if

        if ( (real(cutStartY(i)) == (
           postY - 1.0) ) .and. ( real(
           cutStartX(i)) == postX ) )
           then
        !Cut at down
            DownBond = 0
        end if

        if ( (real(cutStartY(i)) == (
           postY - 1.0) ) .and. ( real(
           cutStartX(i)) == (postX +
           1.0) ) ) then
        !Cut at down right
            RightDownBond = 0
        end if

    end if
```

```
                    else
!Vertical line
                        if (AboveCutPeriodic == 1 ) then
                        !One row above top of cut

                            if ( postX  == real(cutStartX(i)
                                ) ) then
                                DownBond = 0
                            end if

                            if (postX == sitesX) then
                            !Right edge
                                if ( 1.0 == real(cutStartX(i
                                   ) ) ) then
                                    RightDownBond = 0

                                    if (real(min(cutStartY(i
                                       ),cutStopY(i))) ==
                                       postY) then
                                        RightBond = 0
                                    end if
                                end if

                            else
                            !Not on right edge
                                if ( (postX + 1.0) == real(
                                   cutStartX(i) ) ) then
                                    RightDownBond = 0

                                    if (real(min(cutStartY(i
                                       ),cutStopY(i))) ==
                                       postY) then
                                        RightBond = 0
                                    end if
                                end if
                            end if

                        else
                        !Within the cut

                            if (postX == cutStartX(i) ) then
                            !On the line
                                Acceptable = 0
                                return
                            end if

                            if (postX == sitesX) then
                            !Right edge
```

```
          if (cutStartX(i) == 1) then
          !Cut is on left edge ->
             Right bond onto cut
              RightBond = 0

              if ( postY == min(
                 cutStartY(i),
                 cutStopY(i)) ) then
              !At the bottom of
                 current cut
                  if ( (postY == 1) .
                     and. (max(
                     cutStartY(i),
                     cutStopY(i)) ==
                     sitesY ) ) then
                  !If you're at the
                     bottom of the
                     lattice and on
                     the right edge
                     then down right
                     is top left ->
                     cut needs to
                     extend to top
                  !we know you're at
                     the bottom of the
                      lattice and
                     bottom of cut ->
                     bottom of cut =
                     bottom of lattice
                      RightDownBond =
                         0
                  end if
              else
              !Not at the bottom of
                 cut
                  RightDownBond = 0
              end if
          end if

      else
      !Not right edge
          if ((postX + 1.0) == real(
             cutStartX(i)) ) then
          !Cut to right of current
             position
              Rightbond = 0

              if ( postY == min(
                                                cutStartY(i),
                                                cutStopY(i)) ) then
                                             !At the bottom of
                                                current cut
                                                 if ( (postY == 1) .
                                                    and. (max(
                                                    cutStartY(i),
                                                    cutStopY(i)) ==
                                                    sitesY ) ) then
                                                !If you're at the
                                                    bottom of the
                                                    lattice -> cut
                                                    needs to extend
                                                    to top
                                                !we know you're at
                                                    the bottom of the
                                                     lattice and
                                                    bottom of cut ->
                                                    bottom of cut =
                                                    bottom of lattice
                                                     RightDownBond =
                                                        0
                                                end if
                                             else
                                             !Not at the bottom of
                                                cut
                                                RightDownBond = 0
                                             end if
                                         end if

                                     end if

                                 end if
                             end if

                     elseif ( cutStopY(i) == cutStartY(i) ) then
!Horizontal
                      !We need this because we can't find XposOnLine
                         with slope = 0

                         if ( AboveCutPeriodic == 1 ) then
                         !One row above top of cut

                             if (postX == real(sitesX) ) then
                             !Right col
                                 if (max(cutStartX(i),cutStopX(i)) ==
                                     sitesX ) then
                                     DownBond = 0
```

```
                    end if

                    if (min(cutStartX(i),cutStopX(i)) ==
                        1) then
                        RightDownBond = 0
                    end if

                else
                !Not Right Col
                    if ( (postX >= real(min(cutStartX(i)
                       ,cutStopX(i)))) .and. (postX <=
                       real(max(cutStartX(i),cutStopX(i)
                       ))) ) then
                    !Within the bounds of the cut in X
                        DownBond = 0
                    end if

                    if ( ((postX + 1.0) >= real(min(
                       cutStartX(i),cutStopX(i)))) .and.
                        ((postX + 1.0) <=  real(max(
                       cutStartX(i),cutStopX(i)) ) ))
                       then
                        RightDownBond = 0
                    end if

                end if


            else
            !Within cut -> on same row as cut

                if ( (postX <= real(max(cutStartX(i),
                   cutStopX(i))) ) .and. (postX >= real(
                   min(cutStartX(i),cutStopX(i))) ) )
                   then
                !Position is inside the cut
                    Acceptable = 0
                    return
                else
                !Not in cut, but on same line as cut
                    if (postX == real(sitesX) ) then
                    !Right col
                        if (min(cutStartX(i),cutStopX(i)
                           ) == 1) then
                        !Periodic right bond onto cut
                            RightBond = 0
                        end if
                    else
```

```
                        !Not right col
                            if ( (postX + 1.0) == real(min(
                               cutStartX(i),cutStopX(i)) ) )
                                then
                                Rightbond = 0
                            end if

                    end if

                end if

            end if

        else
!Not a point, vertical, or horizontal

                slopeCut = real( cutStopY(i) - cutStartY
                   (i) ) / real( cutStopX(i) - cutStartX
                   (i) )    !Slope of current cut

                if (postX == cutStartX(i) ) then
                    slopePos = real( postY - cutStopY(i)
                        ) / real( postX - cutStopX(i) )
                        !Slope from start point to
                        current position
                else
                    slopePos = real( postY - cutStartY(i
                       ) ) / real( postX - cutStartX(i)
                       ) !Slope from start point to
                       current position
                end if

                YposOnCol = ( (postX - real(cutStartX(i)
                   ) )*slopeCut ) + real(cutStartY(i))

                !if (postX == sitesX) then
                !Right edge
                !    YposOnRD = ( ( 1.0 - real(cutStartX
                   (i)) )*slopeCut ) + real(cutStartY(i)
                   )
                !else
                !Not on right edge
                !    YposOnRD = ( (postX + 1.0 - real(
                   cutStartX(i)) )*slopeCut ) + real(
                   cutStartY(i))

                !end if
```

```
if (postY == 1.0) then
!On bottom edge

    if (max(cutStartY(i),cutStopY(i)) ==
        sitesY ) then
    !RD is within the bounds of the cut
        XposOnRD = real( ( real(sitesY)
           - real(cutStartY(i) ) )/
           slopeCut ) + real(cutStartX(i
           ))

        if (postX == real(sitesX) ) then
        !Right edge
            YposOnRD = ( ( 1.0 - real(
               cutStartX(i)) )*slopeCut
               ) + real(cutStartY(i))
        else
        !Not on right edge
            YposOnRD = ( (postX + 1.0 -
               real(cutStartX(i)) )*
               slopeCut ) + real(
               cutStartY(i))

        end if

    else
    !RD is outside of the bounds of cut
        XposOnRD = sqrt(-1.0)         !
           Sets XposOnRD to "NaN" so all
            of the the if statements
           including it should
           automatically results in
           FALSE
        YposOnRD = sqrt(-1.0)         !I
           tried setting these to -1000,
            but the problem is some
           statements require 0<= x <= 1
            and some are the opposite (x
           >1 or x<0)
   !so making these really big or really
       small won't be able to skip all
     of the if statements that involve
     them
    end if

else
!Not on bottom edge
    if (min(cutStartY(i),cutStopY(i)) ==
            postY )  then
        !RD is outside of the bounds of cut
            XposOnRD = sqrt(-1.0)
            YposOnRD = sqrt(-1.0)
            !print *, "XPOSRD - ", XposOnRD
        else
        !RD is within the bounds of the cut
            XposOnRD = real( ((postY - 1.0)
               - real(cutStartY(i) ))/
               slopeCut ) + real(cutStartX(i
               ))

            if (postX == real(sitesX) ) then
            !Right edge
                YposOnRD = ( ( 1.0 - real(
                   cutStartX(i)) )*slopeCut
                   ) + real(cutStartY(i))
            else
            !Not on right edge
                YposOnRD = ( (postX + 1.0 -
                   real(cutStartX(i)) )*
                   slopeCut ) + real(
                   cutStartY(i))

            end if
        endif

    end if

    if ( (max(cutStartY(i),cutStopY(i)) ==
       sitesY) .and. (postY == 1) .and. (min
       (cutStartY(i),cutStopY(i)) .ne. 1) )
       then
    !If you're on the first row and the cut
       extends to the top of the lattice , we
        can still have  bond onto the cut
    !This is the periodic version
    !We need 3 conditions because if we're
       within the cut and the cut extends
       all the way from the top to the
       bottom,
    !then this case is taken care of in the
       "within the cut" case

        !Right bond will be ok, since the
           cut doesn't extend to the first
           row, and we're on the first row
```

```
          if ( YposOnCol == sitesY  ) then
               DownBond = 0
               DownBreakOnThisCut = 1
          end if

          if ( YposOnRD == sitesY ) then
               RightDownBond = 0
          end if


     elseif ( postY == ( max(cutStartY(i),
         cutStopY(i)) + 1) ) then
     ! one row above top of cut (not
         including the periodic version of
         this)

          if ( ( ( YposOnCol - postY ) < 0 ) .
              and. ( ( YposOnCol - postY ) >=
              -1 ) ) then      !setting this
              equal to 0 causes problems
          !Down bond onto cut

               !Be careful of down bonds that
                   aren't actually cut off
                   because the cut didn't extend
                    far enough
               if (max(cutStartX(i),cutStopX(i)
                   ) >= postX) then
               !The cut has to exist
                   underneathe the current
                   position

                    DownBond = 0
                    DownBreakOnThisCut = 1
               end if

          end if

          if ( postX == sitesX ) then
          !Right column (one row above top of
              cut)
               if (cutStartY(i) >= cutStopY(i)
                   ) then

                    if ( cutStartX(i) == 1 )
                        then
                         RightDownBond = 0
                    end if
```

```
                else
                    if ( cutStopX(i) == 1) then
                        RightDownBond = 0
                    end if
                end if

            else
            !Not right column (one row above top
                of cut)
                if (cutStartY(i) >= cutStopY(i)
                   ) then

                    if ( cutStartX(i) == postX +
                       1 ) then
                        RightDownBond = 0
                    end if
                else
                    if ( cutStopX(i) == postX +
                       1) then
                        RightDownBond = 0
                    end if
                end if
            end if


        else
        !Within the cut

            if (slopeCut == slopePos) then
            !current position is on the line
                Acceptable = 0
                return
            end if

            XposOnLine = ( (postY - real(
               cutStartY(i)) ) / slopeCut) +
               real(cutStartX(i))

            if (postX == sitesX) then
            !On right edge
                if (XposOnLine <= 1.0) then
                !Periodic Right Bond onto cut [
                   As long as it's less than 1
                   it will cut off that 1st col
                   point]
                !Also it's important to try not
                   to use == since we're dealing
                    with real numbers, and they
                could have error

                 RightBond = 0
                 RightBreakOnThisCut = 1
            end if

            if (( XposOnRD <= 1.0 ) .and. (
                XposOnRD > 0.0 ) ) then !I
                think this can be XposOnRD
                instead of XposOnLineBelow,
                since it shouldn't depend on
                the x-pos, only the y
                 RightDownBond = 0
            end if


        else
            !Not on the right edge

            if ( ( (XposOnLine - postX) >=
                0.0 ) .and. ( ( XposOnLine -
                postX) <= 1.0 ) ) then
            !Right Bond through cut
                 RightBond = 0
                 RightBreakOnThisCut = 1


                 if ( postY == 1.0 ) then
                 !On bottom edge; not bottom
                    right
                     if ( postYRD == YposOnRD
                          ) then
                          RightDownBond = 0
                     end if

                 else
                 !Not on bottom or right edge
                     if ( ( (YposOnRD -
                         postYRD) > 1 ) .or.
                         ( (YposOnRD - postYRD
                         ) <= 0 ) ) then
                     !Setting it >= 1 causes
                         a problem
                     !In the case that the
                         cut exists directly
                         to the right of the
                         current position we'
                         ll have YposOnRD -
```

```
                  postYRD = 1
                !Then if the slope is
                   negative, the RD bond
                    won't be blocked
                    RightDownBond = 0

                elseif ( ( (YposOnRD -
                   postYRD) == 1) .and.
                   (slopeCut > 0) ) then
                !This takes care of the
                   case mentioned above
                   - cut exists directly
                    to the right of
                   current position
                !In this case, any
                   positive slope will
                   cut off the RD bond
                    RightDownBond = 0

                end if
             end if

         end if

      end if

      if ( postY == min(cutStartY(i),
         cutStopY(i) ) ) then
      !If on the bottom row of the cut,
         down bonds are allowed except
         maybe periodic down ones

          if ( (postY == 1) .and. (max(
             cutStartY(i), cutStopY(i) )
            == sitesY ) ) then

             if ( cutStartY(i) >=
                cutStopY(i) ) then

                if ( cutStartX(i) ==
                   postX ) then
                    !Periodic Down Bond
                       onto cut
                    DownBond = 0
                    DownBreakOnThisCut =
                        1
                end if
```

```
                        if (postX == sitesX)
                           then
                             !Bottom right corner
                             if ( cutStartX(i) ==
                                 1 ) then
                                 RightDownBond =
                                     0
                           end if
                        else
                             !Bottom row not in
                                 right corner
                             if (cutStartX(i) ==
                                (postX  + 1 ) )
                                then
                                 RightDownBond =
                                     0
                           end if
                        end if

                  elseif ( cutStartY(i) <
                     cutStopY(i) ) then

                        if ( cutStopX(i) ==
                           postX) then
                             !Periodic Down Bond
                                 onto cut
                           DownBond = 0
                           DownBreakOnThisCut =
                                 1
                        end if

                        if (postX == sitesX)
                           then
                             !Bottom right corner
                             if ( cutStopX(i) ==
                                1 ) then
                                 RightDownBond =
                                     0
                           end if
                        else
                             !Bottom row not in
                                 right corner
                             if (cutStopX(i) == (
                                postX  + 1 ) )
                                then
                                 RightDownBond =
                                     0
                           end if
```

```
                end if

            end if

        end if

    else
    !Not on bottom row of cut; still
       within cut (not one row above)
    !Therefore not on bottom row of
       lattice (since it's within the
       cut but not at the bottom of the
       cut, so there must be something
       lower than it)
    !so don't have to worry about
       periodic down bonds

        if ( ( ( YposOnCol - postY ) <=
           0 ) .and. ( ( YposOnCol -
           postY ) >= -1 ) ) then
        !Down bond through cut

            DownBond = 0
            DownBreakOnThisCut = 1

            if (postX == real(sitesX))
               then
            !Right column
                !If on right column,
                   periodic RD bond will
                    only be cut off if
                   the RD point is on
                   the cut, since cuts
                   don't go through the
                   edge of the unit cell
                if (postXRD == XposOnRD
                   )then
                    RightDownBond = 0
                end if

            else
            !If not on right column
                if ( ( (XposOnRD -
                   postXRD) >=  0 ) .or.
                    ( (XposOnRD -
                   postXRD) < -1 ) )
                   then
                !Right Down bond
```

```
                                !Note: If we have <= -1
                                   then this may cause a
                                    problem -> if the
                                   cut exists directly
                                   below the cut then
                                !XposOnRD - postXRD = -1
                                    but if the slope of
                                   the cut is negative,
                                   then it won't block
                                   the RD bond

                                    RightDownBond = 0

                                elseif ( ((XposOnRD -
                                   postXRD) == -1) .and.
                                    (slopeCut > 0.0 ) )
                                   then
                                !This takes care of the
                                   case commented on
                                   above - the cut
                                   exists at the point
                                   directly below the
                                   current position.
                                !As long as the slope of
                                    the cut is positive
                                   then, it will cut off
                                    the RD bond
                                    RightDownBond = 0

                                end if

                            end if

                        end if

                    end if

                end if

                if ( (RightBreakOnThisCut == 0) .and. (
                   DownBreakOnThisCut == 0) ) then
                !If the current cut hasn't affected the
                   right of down bonds

                    if ( (XposOnRD - postXRD) <= 0 .and.
                        (XposOnRD - postXRD) >= -1 )
                       then
                        if ( (YposOnRD - postYRD) >= 0 .
                                            and. (YposOnRD - postYRD) <=
                                            1 ) then
                                             RightDownBond = 0
                                         end if

                                    end if
                                end if
                            end if
                        end if
                    end do
                end if

                end subroutine
```

Bibliography

[1] Richard P. Feynman. “Feynman and Computation”. In: ed. by Anthony J. G. Hey. Cambridge, MA, USA: Perseus Books, 1999. Chap. Simulating Physics with Computers, pp. 133–153. ISBN: 0-7382-0057-3. URL: `http://dl.acm.org/citation.cfm?id=304763.305688`.

[2] Peter W. Shor. “Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer”. In: *SIAM Review* 41.2 (1999), pp. 303–332. DOI: `10.1137/S0036144598347011`. eprint: `http://dx.doi.org/10.1137/S0036144598347011`. URL: `http://dx.doi.org/10.1137/S0036144598347011`.

[3] Miklos Santha. “Quantum Walk Based Search Algorithms”. In: *Theory and Applications of Models of Computation: 5th International Conference, TAMC 2008, Xi’an, China, April 25-29, 2008. Proceedings.* Ed. by Manindra Agrawal et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 31–46. ISBN: 978-3-540-79228-4. DOI: `10.1007/978-3-540-79228-4_3`. URL: `http://dx.doi.org/10.1007/978-3-540-79228-4_3`.

[4] Dave Bacon and Wim van Dam. “Recent Progress in Quantum Algorithms”. In: *Commun. ACM* 53.2 (Feb. 2010), pp. 84–93. ISSN: 0001-0782. DOI: `10.1145/1646353.1646375`. URL: `http://doi.acm.org/10.1145/1646353.1646375`.

[5] Andrew M. Childs and Wim van Dam. “Quantum algorithms for algebraic problems”. In: *Rev. Mod. Phys.* 82 (1 Jan. 2010), pp. 1–52. DOI: `10.1103/RevModPhys.82.1`. URL: `https://link.aps.org/doi/10.1103/RevModPhys.82.1`.

[6] M. Ardehali, H. F. Chau, and H.-K. Lo. “Efficient Quantum Key Distribution”. In: *eprint arXiv:quant-ph/9803007* (Mar. 1998). eprint: `quant-ph/9803007`.

[7] Eli Biham and Tal Mor. “Security of Quantum Cryptography against Collective Attacks”. In: *Phys. Rev. Lett.* 78 (11 Mar. 1997), pp. 2256–2259. DOI: `10.1103/PhysRevLett.78.2256`. URL: `https://link.aps.org/doi/10.1103/PhysRevLett.78.2256`.

[8] D. S. Bethune and W. P. Risk. “An autocompensating fiber-optic quantum cryptography system based on polarization splitting of light”. In: *IEEE Journal of Quantum Electronics* 36.3 (Mar. 2000), pp. 340–347. ISSN: 0018-9197. DOI: `10.1109/3.825881`.

[9] Ethan Bernstein and Umesh Vazirani. “Quantum Complexity Theory”. In: *SIAM Journal on Computing* 26.5 (1997), pp. 1411–1473. DOI: `10.1137/S0097539796300921`. eprint: `http://dx.doi.org/10.1137/S0097539796300921`. URL: `http://dx.doi.org/10.1137/S0097539796300921`.

[10] David Deutsch. "Quantum computational networks". In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences.* Vol. 425. 1868. The Royal Society. 1989, pp. 73–90.

[11] R Alicki. "Decoherence and the Appearance of a Classical World in Quantum Theory". In: *Journal of Physics A: Mathematical and General* 37.5 (2004), p. 1948. URL: `http://stacks.iop.org/0305-4470/37/i=5/a=B02`.

[12] Andrew M Steane. "Efficient fault-tolerant quantum computing". In: *Nature* 399.6732 (1999), pp. 124–126.

[13] Peter W Shor. "Scheme for reducing decoherence in quantum computer memory". In: *Physical review A* 52.4 (1995), R2493.

[14] Emanuel Knill and Raymond Laflamme. "Theory of quantum error-correcting codes". In: *Physical Review A* 55.2 (1997), p. 900.

[15] Charles H Bennett et al. "Mixed-state entanglement and quantum error correction". In: *Physical Review A* 54.5 (1996), p. 3824.

[16] C. Nayak et al. "Non-Abelian anyons and topological quantum computation". In: *Reviews of Modern Physics* 80 (July 2008), pp. 1083–1159. DOI: `10.1103/RevModPhys.80.1083`. arXiv: `0707.1889 [cond-mat.str-el]`.

[17] J. M. Leinaas and J. Myrheim. "On the theory of identical particles". In: *Nuovo Cimento B Serie* 37 (Jan. 1977), pp. 1–23. DOI: `10.1007/BF02727953`.

[18] A. Y. Kitaev. "Fault-tolerant quantum computation by anyons". In: *Annals of Physics* 303 (Jan. 2003), pp. 2–30. DOI: `10.1016/S0003-4916(02)00018-0`. eprint: `quant-ph/9707021`.

[19] F. E. Camino, Wei Zhou, and V. J. Goldman. "Realization of a Laughlin quasiparticle interferometer: Observation of fractional statistics". In: *Phys. Rev. B* 72 (7 Aug. 2005), p. 075342. DOI: `10.1103/PhysRevB.72.075342`. URL: `http://link.aps.org/doi/10.1103/PhysRevB.72.075342`.

[20] V. Lahtinen et al. "Spectrum of the non-abelian phase in Kitaev's honeycomb lattice model". In: *Annals of Physics* 323 (Sept. 2008), pp. 2286–2310. DOI: `10.1016/j.aop.2007.12.009`. arXiv: `0712.1164 [cond-mat.mes-hall]`.

[21] A. Kitaev. "Anyons in an exactly solved model and beyond". In: *Annals of Physics* 321 (Jan. 2006), pp. 2–111. DOI: `10.1016/j.aop.2005.10.005`. eprint: `cond-mat/0506438`.

[22] K. Meichanetzidis et al. "Anatomy of fermionic entanglement and criticality in Kitaev spin liquids". In: *Phys. Rev. B.* 94.11, 115158 (Sept. 2016), p. 115158. DOI: `10.1103/PhysRevB.94.115158`. arXiv: `1605.03629 [cond-mat.str-el]`.

[23] Frank Wilczek. "Quantum mechanics of fractional-spin particles". In: *Physical review letters* 49.14 (1982), p. 957.

[24] A. Roy and D. P. DiVincenzo. "Topological Quantum Computing". In: *ArXiv eprints* (Jan. 2017). arXiv: `1701.05052 [quant-ph]`.

[25] V. Lahtinen et al. "Topological liquid nucleation induced by vortex-vortex interactions in Kitaev's honeycomb model". In: *prb* 86.7, 075115 (Aug. 2012), p. 075115. DOI: `10.1103/PhysRevB.86.075115`. arXiv: `1111.3296 [cond-mat.mes-hall]`.

[26] Carl D Anderson. “The positive electron”. In: *Physical Review* 43.6 (1933), p. 491.

[27] Albert Einstein, Boris Podolsky, and Nathan Rosen. “Can quantum-mechanical description of physical reality be considered complete?” In: *Physical review* 47.10 (1935), p. 777.

[28] Erwin Schrödinger. “Discussion of probability relations between separated systems”. In: *Mathematical Proceedings of the Cambridge Philosophical Society*. Vol. 31. 04. Cambridge Univ Press. 1935, pp. 555–563.

[29] Erwin Schrödinger. “Naturwissenschaften 23 807 Schrödinger E 1935”. In: *Naturwissenschaften* 23.823 (1935), p. 152.

[30] Stephen W Hawking. “Gravitational radiation from colliding black holes”. In: *Physical Review Letters* 26.21 (1971), p. 1344.

[31] M. Srednicki. “Entropy and area”. In: *Physical Review Letters* 71 (Aug. 1993), pp. 666–669. DOI: `10.1103/PhysRevLett.71.666`. eprint: `hep-th/9303048`.

[32] R. Horodecki et al. “Quantum entanglement”. In: *Reviews of Modern Physics* 81 (Apr. 2009), pp. 865–942. DOI: `10.1103/RevModPhys.81.865`. eprint: `quant-ph/0702225`.

[33] L. Amico et al. “Entanglement in many-body systems”. In: *Reviews of Modern Physics* 80 (Apr. 2008), pp. 517–576. DOI: `10.1103/RevModPhys.80.517`. eprint: `quant-ph/0703044`.

[34] Leonid Gurvits. “Classical Complexity and Quantum Entanglement”. In: *J. Comput. Syst. Sci.* 69.3 (Nov. 2004), pp. 448–484. ISSN: 0022-0000. DOI: `10.1016/j.jcss.2004.06.003`. URL: `http://dx.doi.org/10.1016/j.jcss.2004.06.003`.

[35] S. Gharibian. “Strong NP-Hardness of the Quantum Separability Problem”. In: *ArXiv e-prints* (Oct. 2008). arXiv: `0810.4507 [quant-ph]`.

[36] Leon Balents. “Spin liquids in frustrated magnets”. In: *Nature* 464.7286 (2010), pp. 199–208.

[37] Jiannis K. Pachos. *Introduction to Topological Quantum Compuation*. 1st. New York, NY, USA: Cambridge University Press, 2012.

[38] Frank Wilczek. “Majorana returns”. In: *Nature Physics* 5.9 (2009), pp. 614–618.

[39] H. Yao and X.-L. Qi. “Entanglement Entropy and Entanglement Spectrum of the Kitaev Model”. In: *Physical Review Letters* 105.8, 080501 (Aug. 2010), p. 080501. DOI: `10.1103/PhysRevLett.105.080501`. arXiv: `1001.1165 [cond-mat.str-el]`.

[40] C. R. Laumann et al. “Disorder-induced Majorana metal in interacting non-Abelian anyon systems”. In: *prb* 85.16, 161301 (Apr. 2012), p. 161301. DOI: `10.1103/PhysRevB.85.161301`. arXiv: `1106.6265 [cond-mat.str-el]`.

[41] I. Peschel. “LETTER TO THE EDITOR: Calculation of reduced density matrices from correlation functions”. In: *Journal of Physics A Mathematical General* 36 (Apr. 2003), pp. L205–L208. DOI: `10.1088/0305-4470/36/14/101`. eprint: `cond-mat/0212631`.