Tensor Networks and The Ising Model

Dylan Griffith
(420122140/dgri9968)
School of Physics,
University of Sydney,
New South Wales,
Australia.
dyl.griffith@gmail.com

October 26, 2012

Abstract

Quantum many-body physics is important in understanding a range of physical phenomena. Lattice spin models, such as the Ising model, can successfully capture much of the complex behaviour of strongly interacting many-body systems. Recently, there has been considerable interest in such models in the context of quantum computation, in particular, experiments in ion traps have motivated the study of the Ising model on a triangular lattice.

Recently numerical approximation techniques have been developed to allow for polynomial time simulation of certain spin lattice systems. We implemented the infinite Time Evolution Block Decimation algorithm and numerical results relating to the ground state behaviour of the frustrated Ising model on triangular ladders are presented.

We studied spin ladders in quasi-1D structures to gain insight into the 2D Ising model on a triangular lattice. Our results indicate that we have competing terms in the Hamiltonian of our frustrated triangular Ising systems which leads to a rich phase diagram, consisting of three distinct phases. We present some accurate quantitative results for spin ladders as well as qualitative predictions relating to the 2D Ising model on a triangular lattice. These results have the potential to give us some understanding of this as yet unsolved problem.

Acknowledgements

I would like to thank my supervisors Prof. Stephen D. Bartlett and Dr Aroon O'Brien for all their support and patience throughout my research.

Statement of Student Contribution

- I personally have written code in C to perform exact diagonalisation of 2D triangular lattice systems of any size under the Quantum Ising Model
- I implemented the $\mathbb{Z}_2$ symmetry for exact diagonalisation by creating another exact diagonalisation code in C
- I have written MATLAB code to tensor operations
- I have written MATLAB code to implement the iTEBD algorithm for 1D infinite spin chains, as well as the appropriate code for extending this to infinite ladder chains of various widths
- I have performed benchmarking of both the iTEBD and exact diagonalisation tools
- I have written scripts to run simulations with varying parameters and have created all the plots in this thesis using only my own code
- I was responsible for analysis of results (all of the analysis of results in chapter 4 and 5 was my own work)

I am most grateful to my supervisors Prof. Stephen D. Bartlett and Dr Aroon O'Brien for their guidance in formulating the particular problems studied in this thesis as well their assistance with finding suitable literature for my research. I was also provided with some direction regarding the possible causes of bugs throughout the implementation process by my supervisors.

I certify that this report contains work carried out by myself except where otherwise acknowledged.

Signed .....

Date: October 26, 2012

Contents

Chapter 1

Introduction

Quantum many-body systems underlie a great deal of real world physics. Given that such systems are very complex if all the microscopic physics is to be considered it is necessary to develop suitable models to predict physical phenomena. Lattice spin models, including the Ising model, allow simplified representation of strongly interacting physical systems[1]. There is a great deal of interest in these models in the field of condensed matter physics and more recently quantum information theory and quantum computation[2, 3, 4, 5].

These systems are inherently difficult to study, however, due to the exponentially increasing size of the Hilbert space as the number of quantum particles increases. Solving problems of this nature using exact numerical methods are considered intractable and efforts to simulate systems of this kind have been limited to tens of particles. Exact numerical methods such as exact diagonalisation can be made more efficient by using the symmetries of these systems, however, the problem is still inherently intractable.

More recently approximate methods such as the infinite Time Evolving Block Decimation (iTEBD)[6, 7, 8], Density Matrix Renormalisation Group (DMRG)[9, 10, 11, 12] and Multi-scale Entanglement Renormalisation Ansatz[13, 14, 15] have been developed in order to gain insight into the low lying energy states of quantum many-body systems.

In this research project we are interested in studying the ground state behaviour of the frustrated Ising model on triangular lattices, including triangular ladder chains. We will be concerned with understanding the role of frustration in the phase diagram of such systems. In order to model these complex systems we will utilise numerical techniques including exact diagonalisation with finite size scaling as well as new numerical approximation algorithms such as the infinite Time Evolution Block Decimation. We have discovered competition between terms in our frustrated Hamiltonians which lead to a rich phase diagram for triangular ladder systems.

In this thesis several quantitative results relating to the frustrated Ising model on triangular ladder chains, as well as qualitative predictions about the phase diagram of the frustrated Ising model on two-dimensional triangular lattices, will be presented.

This chapter of the report will be concerned with introducing this field of research as well as key concepts to understanding quantum many-body physical systems. In the following chapter we will begin to look at an exact numerical method, known as exact diagonalisation, for studying the systems we will be interested in and we will explore some of the limitations of this technique. In this chapter we will also describe how symmetries may allow us to improve the efficiency of this numerical method. In chapter 3 we will look at the main numerical tools of our research, namely the Matrix Product State representation and the infinite Time Evolution Block Decimation algorithm, which allows us to efficiently find and examine the ground states of one-dimensional spin lattice systems. Chapter 4 will be concerned with benchmarking these algorithms on known results for one-dimensional Ising chains, and chapter 5 will present some new results relating to triangular ladder chains. Finally in chapter 6 we will present the conclusions of our research as well as describe how what we have learned may be used in the future for understanding more complex systems.

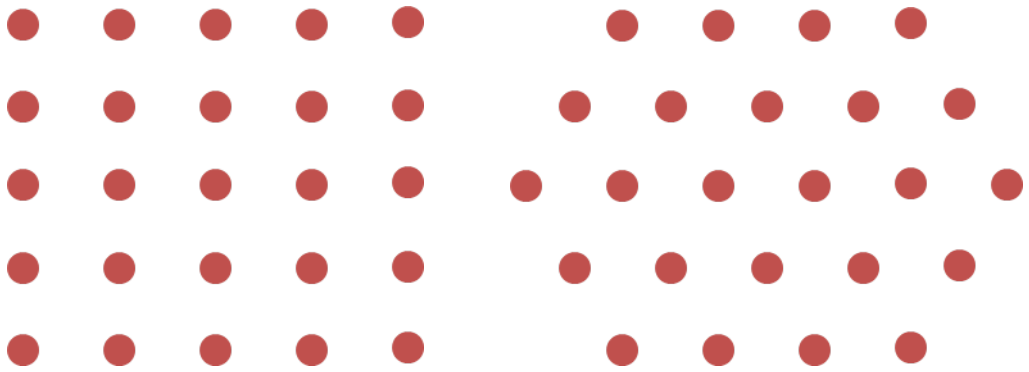
A very common method for simplifying strongly interacting many-body systems is to describe them as spin lattices.

1.1 Spin Lattices

Let us define a spin lattice as a system of particles (and from here on we will only talk about spin half particles) each in a quantum state defined by some superposition of the basis states $|\uparrow\rangle$ and $|\downarrow\rangle$. Essentially we can consider $|\uparrow\rangle$ as being a spin pointing along z axis in the positive direction and $|\downarrow\rangle$ as being a spin pointing along z axis in the negative direction.

Now we can give a representation of a single spin however we would like to represent a lattice of multiple spins. For this we use some standard notations. If for example we had a lattice of two spins with one in the state $|1\rangle$ and the other in the state $|2\rangle$ we denote the state of the system as being in the state $|1\rangle \otimes |2\rangle = |1\rangle\,|2\rangle = |1, 2\rangle$ or more compactly we denote the system defined by $|1\rangle = |\uparrow\rangle$ and $|2\rangle = |\downarrow\rangle$ as simply $|\uparrow\downarrow\rangle$ (ie. we often drop the comma). More generally a system of N particles with the i^{th} particle being in the state $|i\rangle$ can be written as $\prod_i^N |i\rangle = |1, 2, ..., N\rangle$.

Spin lattice models can be single or multi-dimensional and they become far more complex as the dimension increases. For example an infinitely long 1 dimensional chain of spins under Ising Model interactions is exactly solvable using analytic methods[16]. In 2 dimensions many different lattice models have been studied including a square lattice and triangular lattice. Analytic solutions exist only for special cases in two dimensions[17, 18].



(a) Square lattice in two dimensions. Note that each lattice point has four nearest neighbours (ignoring the boundaries).

(b) Triangular lattice in two dimensions. Note that each lattice point has six nearest neighbours (ignoring the boundaries).

Figure 1.1: Lattice models.

There have been several efforts made recently to experimentally realise lattice models[2, 19]. So far up to the order of hundreds of particles on a controlled lattice have been experimentally simulated[2].

1.2 Quantum Ising Model

The Quantum Ising Model is a model that can be applied to understand quantum phase transitions, magnetism and long range order in matter. It is a simple model but its behaviour is rich enough to display basic physical phenomena[4]. The Ising model is of great interest in the field of quantum information theory and quantum computation[2, 20, 21].

The Ising model can be defined in the following way,

$$\hat{H} = J \sum_{\langle i,j \rangle} \hat{S}_z^{[i]} \hat{S}_z^{[j]} + h \sum_i \hat{S}_x^{[i]}. \tag{1.1}$$

The J term in the Hamiltonian corresponds to the exchange constant (interaction strength) of nearest neighbour spins, the h term corresponds to the external field strength and the $\hat{S}^{[i]}$ are the Pauli operators acting on the i^{th} spin. The $\langle i,j\rangle$ notation refers to a summation over all nearest neighbours i and j in a lattice of spins.

Now we must introduce the idea of the "Ising Spin". We say that i^{th} the spin can be oriented in the up or down direction denoted in the Dirac notation as $|\uparrow\rangle_i$ or $|\downarrow\rangle_i$, respectively. The following properties hold for the Pauli operators,

$$\hat{S}_x^{[i]}|\uparrow\rangle_i = |\downarrow\rangle_i, \hat{S}_z^{[i]}|\uparrow\rangle_i = +|\uparrow\rangle_i$$
$$\hat{S}_x^{[i]}|\downarrow\rangle_i = |\uparrow\rangle_i, \hat{S}_z^{[i]}|\downarrow\rangle_i = -|\downarrow\rangle_i.$$

From these we can see that the S_i^z operators are diagonal in our basis, and hence the model reduces to the classical Ising Model in the case that $h = 0$. We can also see that for this case we will introduce Ferromagnetism (spins tend to align with each other) when $J < 0$ and anti-Ferromagnetism (spins tend to anti-align with each other) when $J > 0$.

Qualitatively we know that the ground state of the Hamiltonian[4], $\hat{H}$, depends only on the dimensionless quantity $g = |h/J|$. This allows us to study two opposing limits, $g \gg 1$ and $g \ll 1$.

Let us first consider the case where $g \gg 1$. In this case the second term in (1.1) dominates the interaction and we find that the ground state is simply,

$$|0\rangle = \prod_i |\rightarrow\rangle_i \qquad h < 0$$
$$\text{or} \qquad |0\rangle = \prod_i |\leftarrow\rangle_i \qquad h > 0, \tag{1.2}$$

where we define superposition states,

$$|\rightarrow\rangle_i = \frac{1}{\sqrt{2}}(|\uparrow\rangle_i + |\downarrow\rangle_i)$$
$$|\leftarrow\rangle_i = \frac{1}{\sqrt{2}}(|\uparrow\rangle_i - |\downarrow\rangle_i).$$

These states are the eigenstates of the $\hat{S}_x^{[i]}$ operators with, $\hat{S}_x^{[i]}|\rightarrow\rangle = |\rightarrow\rangle$ and $\hat{S}_x^{[i]}|\leftarrow\rangle = -|\leftarrow\rangle$. For the states in eq. (1.2) we find that the expectation values of $\hat{S}_i^z$ on different sites, i, are totally uncorrelated (ie. $\langle 0|\hat{S}_z^{[i]}\hat{S}_z^{[j]}|0\rangle = \delta_{ij}$). Now making perturbative corrections in g will build in correlations in $\hat{S}_i^z$ that increase in range. For small enough g these correlations are expected to remain short range and that the correlation decays exponentially with increasing distance between spins,

$$\langle 0|\hat{S}_z^{[i]}\hat{S}_z^{[j]}|0\rangle \sim e^{-|x_i - x_j|/\xi}, \tag{1.3}$$

where ξ is some correlation length[4].

Now let us consider the case $g \ll 1$. In this case we have the first term in (1.1) dominating the interaction. Here we find that the ground state will be simply,

$$|0\rangle = \prod_{i=\text{even}} |\uparrow\rangle_i|\downarrow\rangle_{i+1} \qquad \text{or} \qquad |0\rangle = \prod_{i=\text{even}} |\downarrow\rangle_i|\uparrow\rangle_{i+1} \qquad J > 0$$
$$\text{or} \qquad |0\rangle = \prod_i |\uparrow\rangle_i \qquad \text{or} \qquad |0\rangle = \prod_i |\downarrow\rangle_i. \qquad J < 0 \tag{1.4}$$

Small increases in g are expected to mix in a small fraction of spins in the $|\rightarrow\rangle$ or $|\leftarrow\rangle$ states, but in an infinite system the ground state degeneracy will survive. This is because there is an exact global Z_2 symmetry transformation (generated by $\prod_i \hat{S}_x^{[i]}$) mapping the two ground states to each other, under

which $\langle H \rangle$ remains invariant. It is known that a thermodynamic system will always choose one of the two states as its ground state. The nature of the states described in 1.4 when $J < 0$ and small g perturbation theory suggests that,

$$\lim_{|j-i|\to\infty} \langle 0| \hat{S}_z^{[i]} \hat{S}_z^{[j]} |0\rangle = N_0^2, \tag{1.5}$$

where N_0 is the "spontaneous magnetisation" of the ground state. We can now observe that it is not possible for a state that obeys (1.3) to transform analytically into a state that obeys (1.5) as a function of g. We can therefore conclude that there must be some critical value $g = g_c$ at which the large separation limit of the two point correlator changes and this position is a quantum phase transition[4].

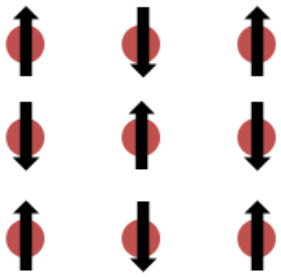
1.3 Frustrated Systems

Frustration in many-body quantum mechanics is defined by a system that is incapable of simultaneously minimizing all the terms of the Hamiltonian. One example of such frustration is Geometric Frustration. This occurs when the geometry of a lattice of spins is such that all interaction terms cannot be minimised for a given lattice model.

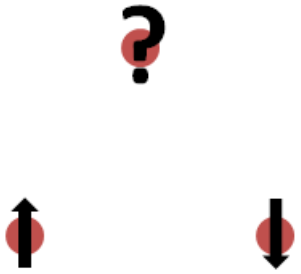
The novel case of such geometric frustration is an anti-ferromagnetic Ising Model on a triangular lattice. Let us first look at the square lattice to understand what is meant by this. In the square lattice all interactions can be simultaneously minimised as in figure 1.2(a). In fact there are always just two possible ground state configurations for the Ising Model on a square lattice regardless of the number of particles (choose a single spin and then all other spins are determined).

By contrast if we consider what happens when there is a triangular lattice we see that it is not possible to choose a configuration where minimisation of all exchange interactions is satisfied. We can see in figure 1.2(b) that even when there are just three spins we cannot find a configuration which minimises all exchange interactions. In fact this configuration has six states which are all the ground state of the system. This ground state degeneracy is a key characteristic of frustrated systems. This implies that such systems can never be completely frozen, as there is still a non-zero entropy at zero temperature.

Frustrated systems such as these are of great interest in the field of quantum computation as well as condensed matter physics[2, 22, 19].



(a) One of two possible ground states for an Anti-Ferromagnetic Ising Model on a square lattice in two dimensions.



(b) Representation of the inability to minimise all exchange interactions for an Anti-Ferromagnetic Ising Model on a triangular lattice in two dimensions.

Figure 1.2: Anti-Ferromagnetic interactions in two dimensions.

1.4 Triangular Lattice Ising Model

The triangular lattice Ising Model has been studied for a long time in the field of condensed matter physics[17], but has as recently as within the last year been looked at in the context of quantum information

theory and quantum computation. Experiments involving trapped ions in triangular lattices have been created to study these frustrated quantum magnetic systems[2].

Analytical methods have been applied to the Classical Ising Model applied to the triangular lattice, and it is known that the ground states have no long range order[17]. There is known to be a transition between an ordered and disordered phase in this model, the nature of the transition is, however, not fully understood and there is still no quantitative consensus as to the exact point of the transition[23, 24, 25, 26]. Furthermore, while these results tell us something about the thermodynamic limit they do not give insight into the effects of boundary conditions on such triangular Lattice Models. There have also been recent proposals for experiments involving trapped ions in frustrated ladder systems[27].

1.5 Numerical Methods

There are a number of numerical methods that can be employed to study spin systems under the Ising Model such: exact diagonalisation, Density Matrix Renormalisation Group (DMRG)[28], Matrix Product States (MPS)[29, 6], Multi-scale Entanglement Renormalisation Ansatz (MERA)[13] and Quantum Monte Carlo[30].

New approximate numerical methods such as the iTEBD, variatonal MPS and MERA allow us to efficiently describe the ground state behaviour of spin lattice systems by properly representing the inherent entanglement. The MPS representation, along with the iTEBD algorithm, in particular has been used in this research in order to study the ground state behaviour of the frustrated Ising model on triangular ladder systems. This has allowed efficient simulations of otherwise intractable infinite spin systems. This thesis will present the results of these simulations and give quantitative results relating to the phase diagram of the infinite triangular double ladder chain. The following chapter will describe an exact numerical method for finding the ground state energy of finite size spin lattices which has allowed for accurate benchmarking of our MPS and iTEBD algorithms that we will introduce in chapter 3.

Chapter 2

Exact Diagonalisation

Exact diagonalisation is a method for exactly solving many-body Quantum Mechanical systems. It is typically thought of as being the "brute force" approach to studying such systems as it involves explicitly calculating then diagonalising the Hamiltonian. Exact diagonalisation can be used to find the ground state energy of spin lattices of finite numbers of particles. This technique, however, can become intractable even for small lattices. For example exact diagonalisation of the Ising model on a triangular lattice is intractable for lattices exceeding ~ 30 spins even for the most powerful supercomputers. In this research project exact diagonalisation methods have been developed, and finite size scaling, which involves extrapolating results of increasingly large systems, has allowed us to compare ground state energies with the main numerical method of this research, the iTEBD algorithm, which gives results for infinite one-dimensional systems.

2.1 The Method

We start with the Ising model defined by (1.1). We then generate all classical configurations,$\{|k\rangle\}$, of N particles, of which there are 2^N. This becomes the algebraic basis for our representation of the system. We then generate all matrix elements of our Hamiltonian, H, by the following relationship,

$$H_{m,n} = \langle m| \hat{H} |n\rangle . \tag{2.1}$$

Once all matrix elements have been generated we can then use a numerical diagonalisation, such as the Lanczos Algorithm[31], to find the eigenvector, $|GS\rangle$, with the lowest eigenvalue, ϵ_0. The state $|GS\rangle$ is the ground state of the system and ϵ_0 is the ground state energy. Given that the number of classical configurations is 2^N it is then clear that an arbitrary quantum state needs to be represented by 2^N numbers and the matrix is of size 2^N by 2^N. This exponential complexity limits such numerical techniques to tens of particles as diagonalisation becomes intractable for very large matrices.

2.2 Using Symmetries

If we consider the alternate form of our Ising Model described by,

$$\hat{H} = J \sum_{\langle i,j \rangle} \hat{S}_x^{[i]} \hat{S}_x^{[j]} + h \sum_i \hat{S}_z^{[i]}, \tag{2.2}$$

we find that there exists a $\mathbb{Z}_2$ symmetry. Let us first note that the second term of the Hamiltonian only contributes to diagonal matrix elements in our basis (2.2) (ie. $\langle m| \hat{S}_i{}^z |n\rangle = \alpha_m \delta_{m,n}$ for some complex scalar α_m.) From this we consider now just the first term. Now let us define the parity of a configuration to be simply the number of up spins modulo two. Given that $\hat{S}_i^x |\uparrow\rangle_i = |\downarrow\rangle_i$ and $\hat{S}_i^x |\downarrow\rangle_i = |\uparrow\rangle_i$ we find that parity is conserved under application of the Hamiltonian. We can therefore conclude that configurations

of with even parity will be mapped to configurations of even parity and those with odd parity will be mapped to configurations with odd parity.

This tells us something about the matrix elements of our Hamiltonian, namely that there exists some ordering of a specific basis such that the Hamiltonian matrix will have a block diagonal structure. More specifically the Hamiltonian will be given by (all non-coloured parts are zero),

$$H = \begin{pmatrix} H_{1,1} & \cdots & H_{1,M} & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ H_{M,1} & \cdots & H_{M,M} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & H_{M+1,M+1} & \cdots & H_{M+1,2^N} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & H_{2^N,M+1} & \cdots & H_{2^N,2^N} \end{pmatrix}. \tag{2.3}$$

From this we can now simply find the eigenvector with the lowest eigenvalue for the blue and red matrix separately then pick from these the eigenvector, $|GS\rangle$, with the lowest eigenvalue, ϵ_0. This reduces the computational complexity of the problem by utilising the symmetries of the system. This technique is known as block diagonalisation and is represented in figure 2.1.

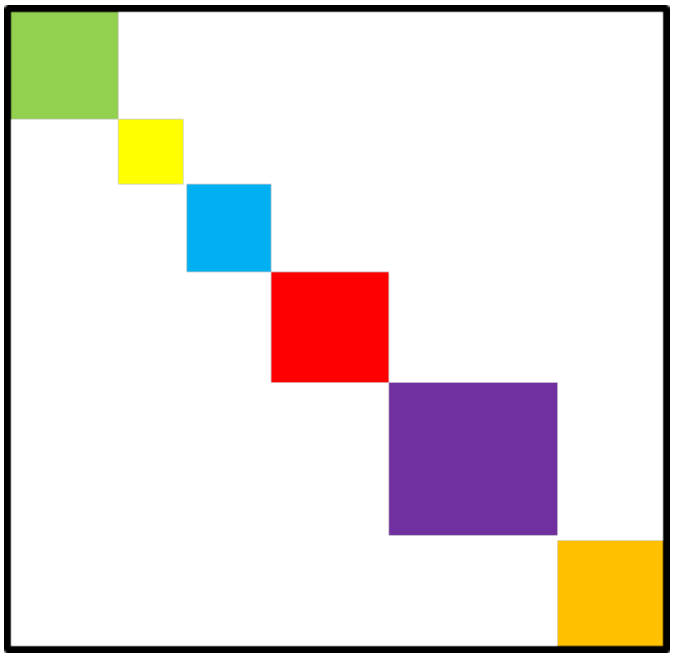


Figure 2.1: Colour visualisation of an arbitrary block diagonal matrix. White regions are zero values. The eigenvector corresponding to the lowest eigenvalue of the whole matrix is the eigenvector corresponding to the lowest eigenvalue out of all the eigenvectors of the matrices composing the coloured regions.

The exact diagonalisation algorithm implemented as part of this research project (see appendix B.1) has allowed for simulations up to $\sim$ 25 particles and for one-dimensional spin chains this is already sufficiently converged to the ground state behaviour in the thermodynamic limit (see section 5.3.1). This has allowed for benchmarking the results of the iTEBD algorithm on infinite ladder chains.

Chapter 3

MPS and Infinite Time Evolving Block Decimation

Tensor networks offer a way of describing many-body quantum systems using tensor algebra. Pure quantum states can be described by a vector in a Hilbert space[32]. We described an N-body quantum system by the tensor product of N Hilbert Spaces. We can therefore represent the wavefunction, $|\psi\rangle$, of any N-body quantum system in the following way,

$$|\psi\rangle = \sum_{i_1}^{d_1} \sum_{i_2}^{d_2} ... \sum_{i_N}^{d_N} C_{i_1,i_2,...,i_N} |i_1, i_2, ..., i_N\rangle$$

Where the summation takes all possible values of i_k (ie. the dimension, d_k, of the Hilbert space of the k^{th} quantum particle). Overall we find the number of different terms in this representation to be $\prod_k^N d_k$. The tensor used as a representation of the system is the tensor C in the expression.

Clearly we can see that the number of coefficients (ie. the size of the Hilbert Space) scales exponentially and the size of the Hilbert space, $\mathcal{H}$, is 2^N in an N-Body spin half system. For this reason we have exponential memory complexity if we wish to fully represent the state of a given quantum system on a classical computer. Also in order to find exact solutions it would require solving a matrix (the Hamiltonian) who's size scales exponentially with the number of particles and since numerical methods for diagonalising a matrix rely on matrix multiplication[33] we expect time complexity to exceed $O(2^{2N})$.

Given this it is clear we need some approximation methods in order to gain an understanding of large quantum systems. This is where Tensor Networks can be useful. While the Hilbert Space of the system can be very large, most of this is resulting from high-energy states of the system which are not physically realisable at low temperatures, which is what we are mostly interested in. For certain systems it can be shown that there exists some smaller subspace that may offer us a complete representation of the lower energy levels of the quantum system. Tensor Networks such as the Matrix Product State exploit this fact in order to reduce the problem to a smaller subspace, $\mathcal{H}'$, of the Hilbert Space (figure 3.1).

3.1 Schmidt Decomposition

The Schmidt decomposition of a pure quantum state, $|\psi\rangle$, can be considered as a separation of the system into two subsystems, A and B.

$$|\psi\rangle = \sum_{i}^{\chi} \lambda_i \left|\varphi_i^A\right\rangle \left|\varphi_i^B\right\rangle \tag{3.1}$$

If there is no entanglement between the two subsystems A and B then we can represent the state with $\chi = 1$, but if there is some entanglement then we need a $\chi > 1$. This value of χ can be used to give us some

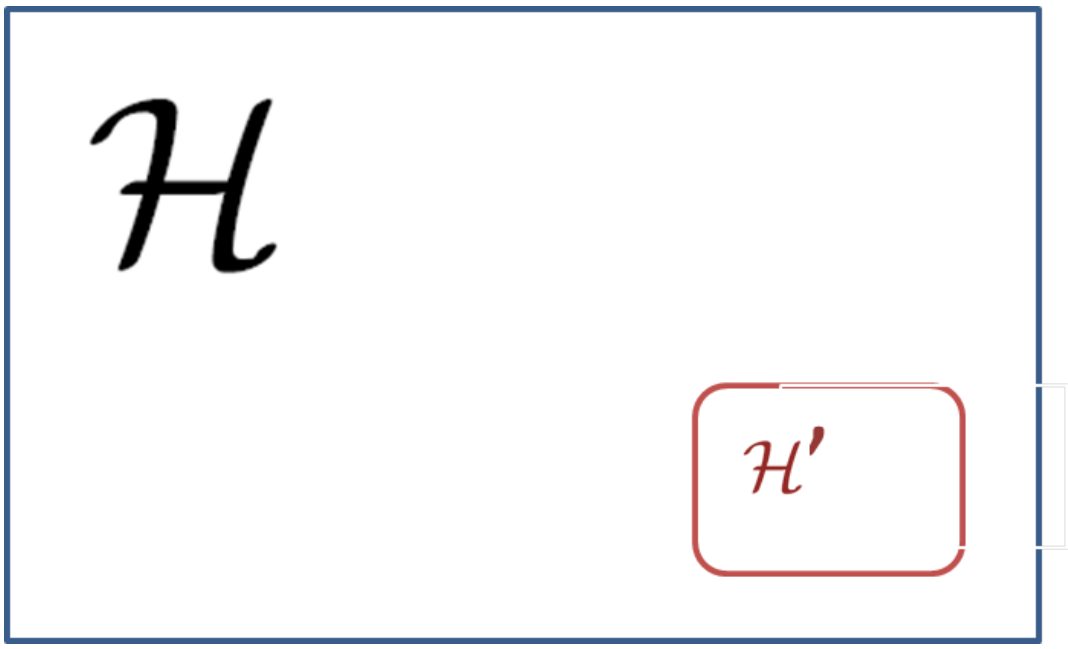


Figure 3.1: Diagrammatic representation of reduced Hilbert Space.

measure of entanglement in a many-body system[32]. Recursively applying the Schmidt decomposition to a many-body state results in the Matrix Product State representation as introduced in 3.3. It is expected that for the ground state of many spin lattice systems we are able to truncate this parameter χ and still give an accurate representation of the system. The Matrix Product State representation relies on the truncation of this summation to some maximum finite χ in order to efficiently represent ground state behaviour. This is done by only including the largest weight λ_i terms up to some χ terms.

3.2 Tensor Networks

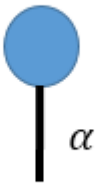
A tensor is defined as multi-dimensional array of complex numbers. It is the generalisation (or extension) of scalars, vectors and matrices to higher dimensions. We can define the rank of a tensor to be the number of free indices (or dimensions) it has. A scalar therefore is a tensor of rank 0 and vectors and matrices are tensors of rank 1 and 2, respectively.

3.2.1 Tensor Network Diagrams

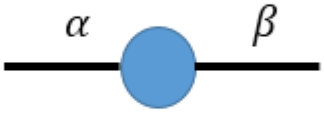
Due to the complexity of representing collections of tensors and contractions over different indices in the algebraic representation it is often clearer to use a graphical notation. For this purpose there is a pictorial representation used for tensor networks. The basic idea is to represent every tensor by some simple shape (eg. circle, square, triangle etc.) and every free index of that tensor by some outgoing line (see figure 3.2). It is common practice to give the indices labels.

Similar to Einstein notation we can also introduce a convention for indicating a summation over common indices of different tensors, we will hereafter call this contracting over indices. This is done by connecting two lines of different tensors together (see figure 3.3). The assumption being that the number of values taken by the connected indices is the same. Hence we can describe the matrix multiplication of a matrix A with a matrix B as shown in figure 3.4.

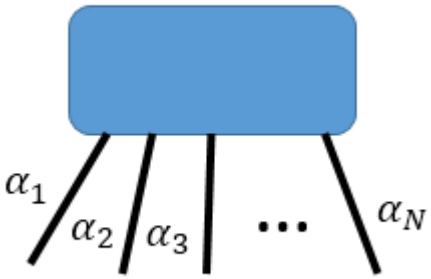
(a) Tensor network diagrammatic representation of a scalar.



(b) Tensor network diagrammatic representation of a vector.



(c) Tensor network diagrammatic representation of a matrix.



(d) Tensor Network Diagrammatic representation of a single tensor of rank N.

Figure 3.2: Tensor Diagrams.

3.2.2 Tensor Operations

Now that we have a neat way of expressing tensors we need to define the tensor operations performed in our simulations. Here we will describe the operations; reshape and permute, and then describe how these can be used in combination with matrix multiplication to perform tensor contractions (a MATLAB implementation can be found in the appendix).

The reshape operation can be thought of as fusing together or breaking up free indices. If two indices are fused together the number of values taken by the free index formed is simply the product of the number of values taken by the two free indices that compose it. This results in a tensor with a rank one less than the tensor had to begin with. If a fused leg is split back into two legs the rank of the tensor increases by one and the product of the number of values taken by the two free is equal to the number of values taken by the free index from which they were produced. The breaking is just reversing a fusing and these operations are shown diagrammatically in figure 3.5.

The permutation essentially just reorders the free indices of the tensor. This process is summarised in figure 3.6.

Since software packages like MATLAB are purposely built for efficient matrix multiplication it is often practical to perform tensor contractions through a process of reshapes, permutations and matrix multiplications. The process is as follows:

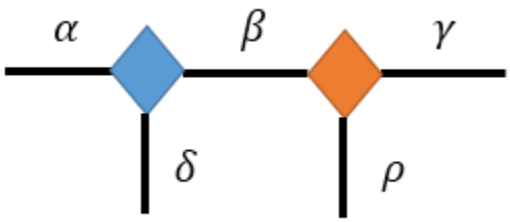


Figure 3.3: Tensor Network Diagrammatic representation of contracting two tensors of rank three together.

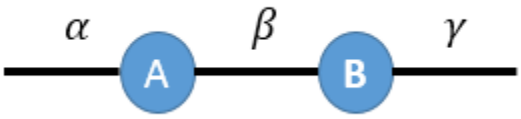


Figure 3.4: Tensor Network Diagrammatic representation of matrix multiplication of a matrix A with a matrix B.

1. take all uncontracted indices of the first tensor and permute them all to be the first free indices of that tensor then fuse these indices together using a reshape
2. take all contracted indices of the first tensor (these should now be the last free indices of the tensor) and fuse them together
3. take all uncontracted indices of the second tensor and permute them all to be the last free indices of that tensor then fuse these indices together using a reshape
4. take all contracted indices of the second tensor (these should now be the first free indices of the tensor) and fuse them together
5. perform a matrix multiplication between the first and second tensor (these should both be matrices now)
6. split the first free index of the resultant tensor to the original uncontracted indices of the first tensor
7. split the second free index of the resultant tensor to the original uncontracted indices of the second tensor

An example of a tensor contraction is depicted in figure 3.7. The pictured process corresponds to the

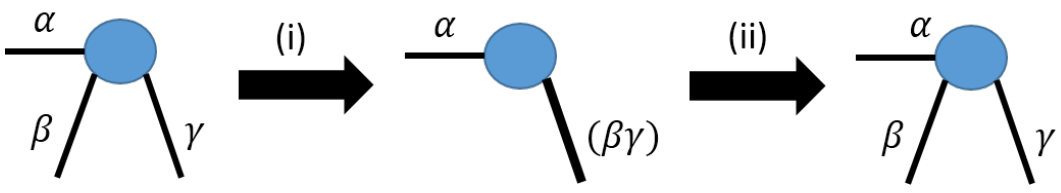


Figure 3.5: Reshape operation on a rank three tensor. (i) The free indices β and γ are fused together to form the single free index $(\beta\gamma)$. (ii) The free index $(\beta\gamma)$ is split back into the the two free indices β and γ.

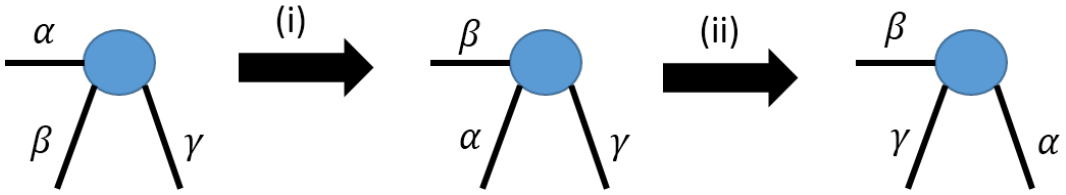


Figure 3.6: Permutation operation on a rank three tensor. (i) The free indices α and β are swapped. (ii) The free indices α and γ are swapped.

following algebraic equation,

$$C_{i_1,i_3,\alpha_1,\alpha_2} = \sum_{i_2=\alpha_3} A_{i_1,i_2,i_3} B_{\alpha_1,\alpha_2,\alpha_3}. \tag{3.2}$$

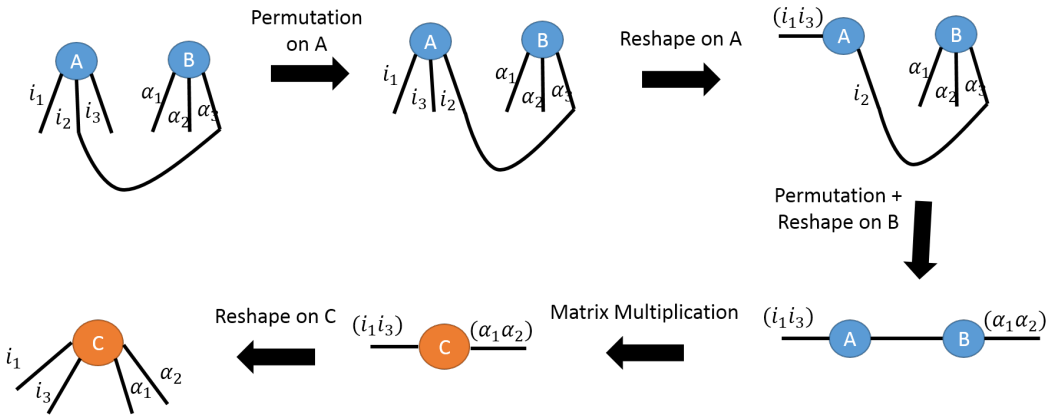


Figure 3.7: Permutation operation on a rank three tensor. (i) The free indices α and β are swapped. (ii) The free indices α and γ are swapped.

3.3 The Matrix Product State Representation

The Matrix Product State[5, 34, 29, 6] is an alternative representation of quantum states that allows for a controlled truncation of the Hilbert Space of one-dimensional many-body Quantum systems. Before explaining the Matrix Product State representation let us first consider an example.

Consider a system consisting of two spins. Then the most generic wave function can be represented as,

$$|\psi\rangle = \sum_{i_1=0}^{d} \sum_{i_2=0}^{d} C_{i_1,i_2} |i_1\rangle |i_2\rangle , \tag{3.3}$$

where d is the local Hilbert space dimension for each spin (for simplicity we are assuming that both spins have the same dimension). If the system is a product state then components of the tensor C, could be expressed as the product of two complex scalars, such that $C_{i_1,i_2} = c_{i_1} c_{i_2}$. For example, consider that the spins are spin half particles ($d = 2$) then the general wavefunction can be written as,

$$|\psi\rangle = C_{0,0} |00\rangle + C_{0,1} |01\rangle + C_{1,0} |10\rangle + C_{1,1} |11\rangle . \tag{3.4}$$

Now if the state were such that $C_{0,0} = ax, C_{0,0} = ay, C_{0,0} = bx, C_{0,0} = by$, then the state could be expressed as a product state as follows,

$$|\psi\rangle = ax\,|00\rangle + ay\,|01\rangle + bx\,|10\rangle + by\,|11\rangle \tag{3.5}$$
$$= (a\,|0\rangle + b\,|1\rangle) \otimes (x\,|0\rangle + y\,|1\rangle). \tag{3.6}$$

In this product state representation the spins are not entangled. This can be thought of by considering what would happen if one of the spins were measured. When it is a product state this measurement would have no affect on the outcome of any future measurement on the second spin. Alternatively we can see in the maximally entangled Bell state defined by,

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), \tag{3.7}$$

we find that there is no way to factor this into a tensor product of the two separate spins and is therefore not a product state. In this state any measurement of either spin will subsequently completely determine the result of future measurements on the second spin.

The important distinction between the product state and the general state relates to the number of coefficients required to represent the system. When there are N spins each with local dimension, d, we need d^N coefficients to represent a general state. However if it were a product state we would need only dN coefficients. While not all wavefunctions can be decomposed into product states, it is always possible to decompose the system into a product of matrices. This is the idea behind the Matrix Product State (MPS) representation.

The construction of the MPS[5] is as follows, we start with the bipartite splitting of the N particle the system into the two subsystems containing 1 and the $N-1$ remaining spins,

$$|\psi\rangle = \sum_{\alpha_1}^{\chi_{\alpha_1}} \lambda_{\alpha_1}^{[1]} \left|\Phi_{\alpha_1}^{[1]}\right\rangle \left|\Phi_{\alpha_1}^{[2\ldots N]}\right\rangle \tag{3.8}$$
$$= \sum_{i_1}^{d_{i_1}} \sum_{\alpha_1}^{\chi_{\alpha_1}} \Gamma_{\alpha_1}^{[1]i_1} \lambda_{\alpha_1}^{[1]} |i_1\rangle \left|\Phi_{\alpha_1}^{[2\ldots N]}\right\rangle, \tag{3.9}$$

where we have expanded $\left|\Phi_{\alpha_1}^{[1]}\right\rangle$ in terms of the basis vectors $\{|i_1\rangle\}$ for the first spin. Here χ_{α_1} gives the number of states in $\left|\Phi_{\alpha_1}^{[1]}\right\rangle$ that are entangled with states in $|i_1\rangle \left|\Phi_{\alpha_1}^{[2\ldots N]}\right\rangle$. We can then recursively apply this procedure to the remaining spins to arrive at the Matrix Product State representation. This alternate representation is depicted diagrammatically in figure 3.8.

$$|\psi\rangle = \sum_{i_1,i_2,\ldots,i_N} C_{i_1,i_2,\ldots,i_n} |i_1, i_2, \ldots, i_N\rangle \tag{3.10}$$
$$C_{i_1,i_2,\ldots,i_n} = \sum_{\alpha_1,\alpha_2,\ldots,\alpha_{N-1}} \Gamma_{\alpha_1}^{[1]i_1} \lambda_{\alpha_1}^{[1]} \Gamma_{\alpha_1,\alpha_2}^{[2]i_2} \lambda_{\alpha_2}^{[2]} \Gamma_{\alpha_2,\alpha_3}^{[3]i_3} \ldots \lambda_{\alpha_{N-1}}^{[N-1]} \Gamma_{\alpha_{N-1}}^{[N]i_N} \tag{3.11}$$

In this representation we then make an approximation. In order to truncate the Hilbert Space to the low-energy states it is reasonable to choose an approximation that limits the entanglement that can exist between blocks of spins in the system[32]. This can be done by truncating the number of values that each α_k can take to some number χ such that,

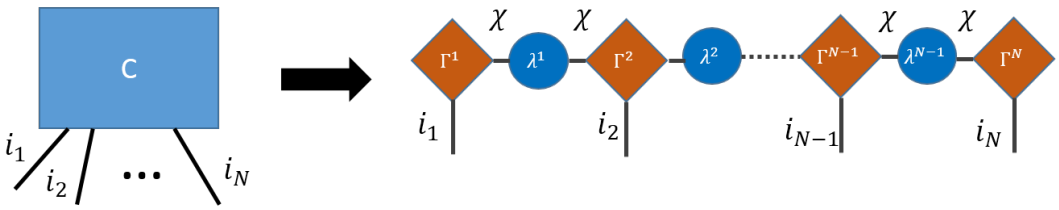


Figure 3.8: Tensor network diagram showing how the rank N tensor of coefficients, C, of an N-body quantum state, is represented by a set of N Γ tensors and $N-1$ λ tensors.

$$C_{i_1,i_2,\ldots,i_n} = \sum_{\alpha_1,\alpha_2,\ldots,\alpha_{N-1}}^{\chi} \Gamma^{[1]i_1}_{\alpha_1} \lambda^{[1]}_{\alpha_1} \Gamma^{[2]i_2}_{\alpha_1,\alpha_2} \lambda^{[2]}_{\alpha_2} \Gamma^{[3]i_3}_{\alpha_2,\alpha_3} \ldots \lambda^{[N-1]}_{\alpha_{N-1}} \Gamma^{[N]i_N}_{\alpha_{N-1}}. \tag{3.12}$$

Using this approximation we find that as the system grows the entanglement growth is bounded by this parameter χ. It was proven in [35, 36] that in order to represent ground states of one-dimensional systems at criticality (at phase transitions), we only require a χ that grows polynomially in the system size. Ground states of non-critical one-dimensional local Hamiltonians, however, can be represented by a fixed finite χ.

Given this approximation we now find that the number of coefficients scales polynomially (in the non-critical case it is O(N)) with the size of the system, as opposed to exponentially, and as a result simulations become tractable. We can also note that increasing the value of χ should result in a more exact solution. This is useful as it allows us to test the convergence of our algorithm by simply increasing χ and determining if the results change in any way.

3.4 Infinite Time Evolution Block Decimation

The infinite Time Evolution Block Decimation (iTEBD)[6] is an algorithm based on the Time Evolution Block Decimation[5] to simulate one-dimensional quantum lattice systems in the thermodynamic limit. For most algorithms the cost of simulation grows with the system size and thus the thermodynamic limit can only be achieved by extrapolation of results for increasing system size. The iTEBD algorithm, however, exploits the invariance under translations of systems and parallelisability of local updates in TEBD, in order to simulate infinite systems directly.

Evolving a system in imaginary time provides a reliable method of evolving an arbitrary MPS to gapped ground states. Consider starting with an MPS $|\psi(0)\rangle$ then evolving in imaginary time leads to,

$$|\psi(t)\rangle = e^{-Ht} |\psi(0)\rangle = \sum_{k=1}^{n} e^{-\lambda_k t} |\psi_k\rangle \langle\psi_k| \psi(0)\rangle \tag{3.13}$$

where we have taken the eigenvalue decomposition of $H = \sum_k^\infty \lambda_k |\psi_k\rangle \langle\psi_k|$. Therefore as long as the ground state is not degenerate $|\psi(t)\rangle$ will converge to the ground state exponentially fast[37]. This makes the iTEBD algorithm very robust, because no matter what initial state we choose, convergence is always exponentially fast. Even in cases with ground state degeneracy we still find that the system will converge until some superposition of lowest energy states is reached. The rate of convergence for this method is dependent only on the size of the gap between the ground and excited state. Time evolution in real time has also been used to study dynamical properties of spin lattices[6].

The algorithm is described in figure 3.9. We first contract together the MPS with the unitary time evolution operator, U, to get a rank four tensor, Θ. A Singular Value decomposition is then used in order

to give the three tensors. We then contract the X and Y tensors with the matrix inverse λ^B in order to recover the updated versions of Γ^B and Γ^A. We have now done one update on all our tensors in the MPS except for λ^B. We then repeat the entire process except the Θ is given by figure 3.10. It is important that the weights of the elements in in the Λ tensors be arranged in descending order (usually standard for SVD functions) such that when the MPS bonds are truncated to the largest χ terms only the states contributing the largest to the state are kept and those that are removed have the smallest weights.

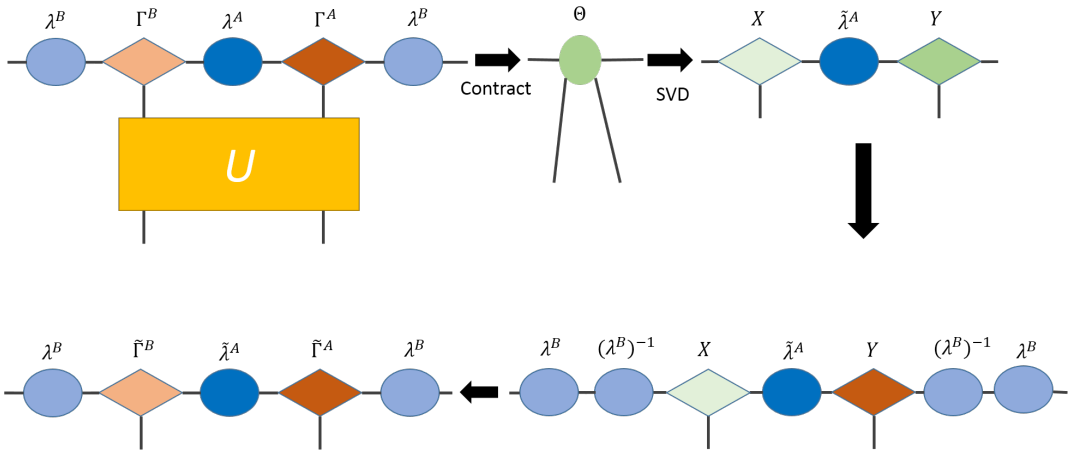


Figure 3.9: Graphical Representation of the iTEBD algorithm. This represents the first half of every time evolution iteration.

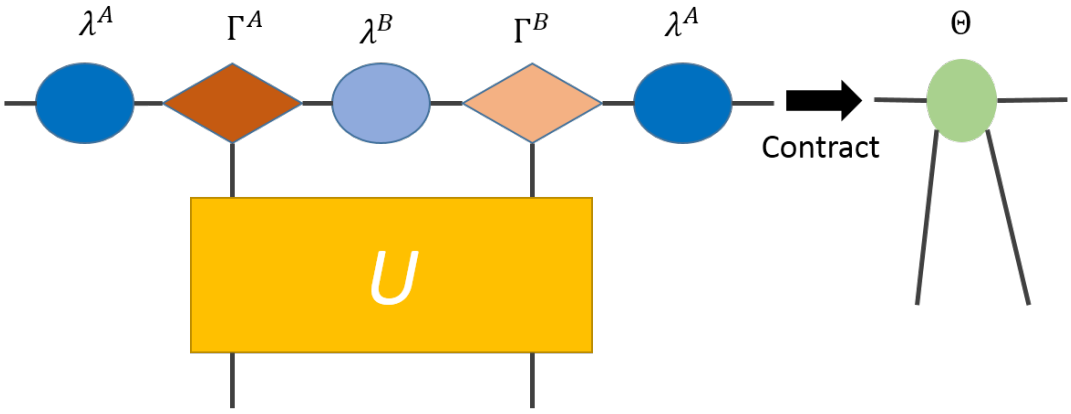


Figure 3.10: Graphical Representation of the iTEBD algorithm. This represents how to calculate Θ for the second half of every time evolution iteration. All other steps are identical except now A is replaced with B and vice versa in all the pictures.

In order to calculate, $\hat{U}$, we first take a two site Hamiltonian constructed from a single term of the summations in $\hat{H}$ and generate all eigenstates $\{|\psi_k\rangle\}$ as well as their corresponding eigenvalues $\{E_k\}$ and we then obtain $\hat{U}$ through the following relation,

$$\hat{U} = \sum_k e^{-E_k dt/\hbar} |\psi_k\rangle \langle\psi_k| . \tag{3.14}$$

Algorithmically it is practical to adjust the time step τ to be smaller and smaller over time to get closer to the ground state.

3.5 Trotter Decomposition

Performing the time evolution for this process is not completely trivial because the different terms that compose the Hamiltonian don't commute[37] and can't simply be applied one after the other. A commonly used technique to perform the time evolution is to use the Trotter decomposition[38] defined by,

$$e^{A+B} = \lim_{n\to\infty} \left(e^{\frac{A}{n}} e^{\frac{B}{n}} \right)^n.$$

If we can split the Hamiltonian into two parts A and B ($-H = A + B$) we can then approximate the desired evolution by evolving first under $e^{\tau A}$ then under $e^{\tau B}$ and then repeat this process K times until we have evolved for a total imaginary time $T = iK\tau$. The time step, τ, can be chosen such that the error produced at each evolution is arbitrarily small.

3.6 Implementation

The iTEBD algorithm was implemented as part of this research project to simulate infinite one-dimensional systems under and Ising model Hamiltonian (see appendix B.2). We have been able to determine suitable truncation parameters χ for the MPS for different systems as well as time steps τ for the Trotter Decomposition in the iTEBD algorithm. The results were successfully benchmarked against known results for the infinite one-dimensional chain. These methods were then extended to allow for simulations of ladder chains and new results relating to the phase diagram of the infinite triangular double ladder chain were calculated. The following chapter will describe how we were able to benchmark our implementation.

Chapter 4

The One-Dimensional Ising model in a Transverse Field

The infinite Time Evolution Block Decimation algorithm described in the previous chapter was implemented in order to study infinite one-dimensional spin lattices. In order to benchmark these methods we have compared our results to known analytical results of the infinite one-dimensional Ising model in a transverse field[16]. Several benchmarking results are presented in this chapter and we can see that the iTEBD algorithm is reliably reproducing the ground state results of the systems we are studying.

4.1 Theory

The Ising model is an interesting model, because, although it is a highly simplified model, it stills reproduces interesting physical phenomena. The one-dimensional infinite Ising model can be described by the Hamiltonian, $\hat{H}$, given by,

$$\hat{H} = J\sum_{i=1}^{\infty} \hat{S}_x^{[i]}\hat{S}_x^{[i+1]} + h\sum_i \hat{S}_z^{[i]}. \tag{4.1}$$

This system has been solved analytically[16]. The analytical solution allows for the calculations of various properties of the ground state of the system such as energy and spin-spin correlation values. All the plots in this chapter will refer directly to the infinite one-dimensional Ising model defined by (4.1) and analytical results will be derived from [16].

4.2 Benchmarking Our Results With the One-Dimensional Ising Model

The results that follow are based on simulations utilising a code that was written specifically as part of this research project. In particular an implementation of the iTEBD algorithm was created and benchmarked against known results.

4.2.1 Energy

The first benchmarking done using the iTEBD code was to compare the energy per spin of the ground states found with the analytical results. Figure 4.1 compares the simulated ground state energy with the known analytical result. The agreement in these two curves is good enough that the difference can't be seen on the plot. The actual variations is always less than 10^{-4} even at the critical point ($h = 1$). These results imply strongly that the algorithm is finding the correct ground state for the system.

The interesting thing to note about these results is that the errors are larger near the critical point and the greatest error 2.1×10^{-5} occurs when $h = 1$. This confirms the result that the MPS is not capable

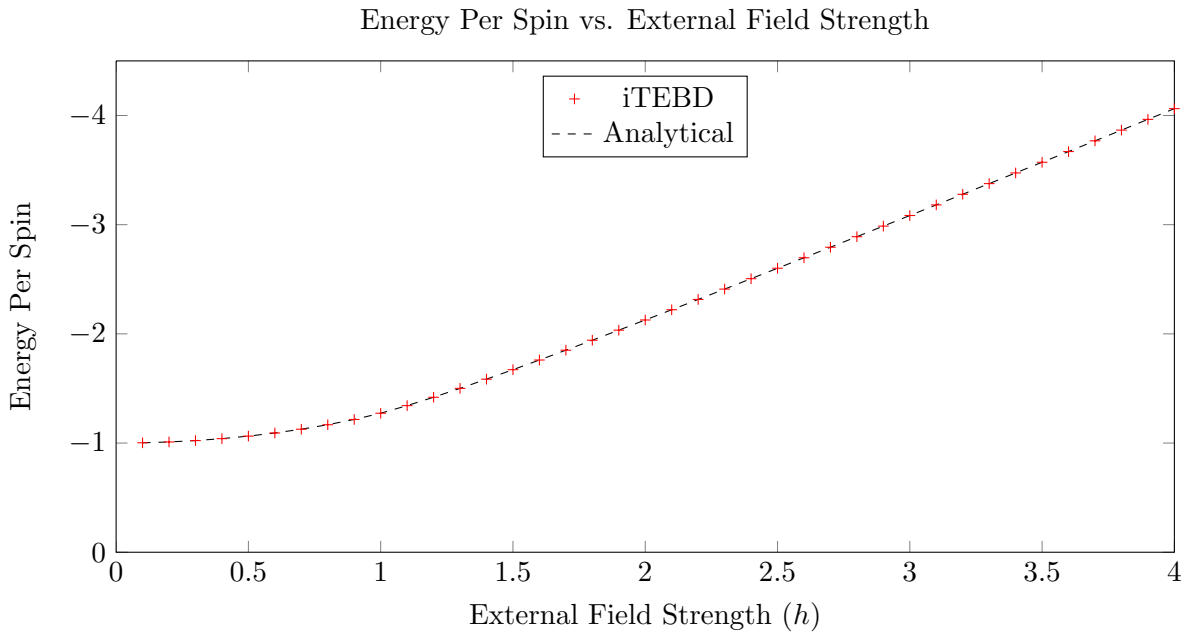


Figure 4.1: Plot showing the energy per spin vs. the external field strength for the iTEBD algorithm as well as the analytical result. Both curves represent the one-dimensional infinite Ising model with exchange interaction strength $J = 1$. The simulations were based on $\chi = 70$ and a minimum time step $\tau = 10^{-7}$.

of precisely capturing infinite critical systems. This is because as the system becomes infinitely large, an infinitely large χ would be required to give a complete representation at the critical point.

4.2.2 Magnetisation

We define the magnetisation M of a spin chain in the z direction (ie. along the axis of the external field in (4.1)) by,

$$M = \langle \hat{S}_z^{[i]} \rangle + \langle \hat{S}_z^{[i+1]} \rangle,$$

where the angle braces denote the expectation value. This quantity describes how well the two spins are aligned with the external magnetic field and, given the translational invariance, how well aligned all spins are with the external magnetic field. This property describes how the system will interact magnetically with other magnetic materials. Here we wish to see how this magnetisation varies with external field strength. From figure 4.3 we can see that as the external field strength increases the alignment with the external field increases monotonically. This is to be expected as the ground state becomes dominated by interactions with this field. Since our h values in this plot are positive we expect that the spins will be aligned in the negative z direction, which is what we see also. We can also see that as h becomes very large M saturates toward the maximum possible magnetisation $M = 2$.

4.2.3 Correlation

We define our short range correlations C by,

$$C = \langle \hat{S}_x^{[i]} \hat{S}_x^{[i+1]} \rangle$$

which gives a measure of the nearest neighbour alignment along the x axis (ie. the axis along which exchange interactions occur in 4.1). Given the translational invariance we can look at just the interaction between any adjacent pair of spins in our infinitely long Ising chain to learn about all interactions. We

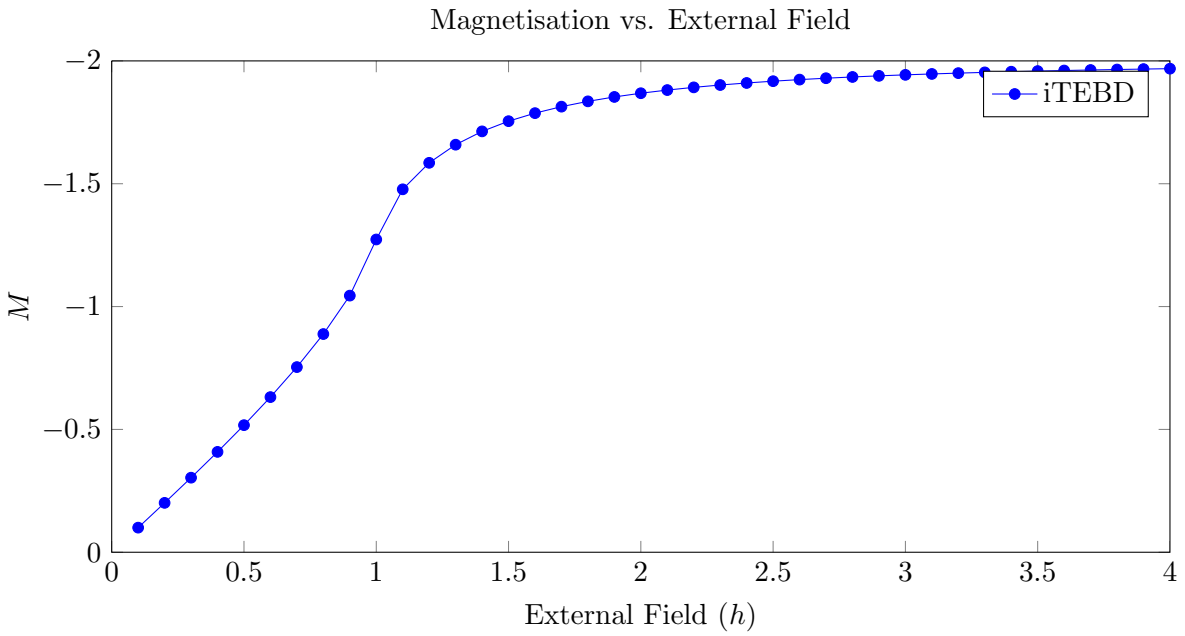


Figure 4.2: Plot showing the magnetisation given by $M = \langle \hat{S}_z^{[i]} \rangle + \langle \hat{S}_z^{[i+1]} \rangle$ vs. the external field strength for the iTEBD algorithm. The curve represents the ground state of the one-dimensional infinite Ising model with exchange interaction strength $J = 1$. The simulations were based on $\chi = 70$ and a minimum time step $\tau = 10^{-7}$.

wish to study again how varying the strength of the external field effects this quantity. From figure 4.3 we see that as the Hamiltonian is dominated by the external field the correlation decreases. This behaviour is to be expected as the system is finding the ground state that increasingly minimises the external field terms of the Hamiltonian rather than the non-commuting interaction terms.

Given that the iTEBD algorithm is finding the ground state of infinite one-dimensional spin chains it is actually possible to study the correlation of spins separated by large distances. In order to quantify how correlated two spins are at a separation distance we introduce the two-point correlator, C_2, defined by,

$$C_2(r) = \langle \psi | \, S_z^{[0]} S_z^{[r]} \, | \psi \rangle - (\langle \psi | \, S_z^{[0]} \, | \psi \rangle)^2, \tag{4.2}$$

$$\tag{4.3}$$

where r represents the separation distance. It is known[16] that at the critical point for the one-dimensional Ising chain this C_2 function decays as a power law for the ground state. Away from the critical point, however, we expect to see an exponential decay in this quantity.

In figure 4.4 we can see that the ground state behaviour of the critical point, $h = 1$, is being very accurately reproduced by this method with visible differences only becoming noticeable at $r \sim 1000$. The numerical accuracy of the results are expected to be only 10^{-4} for energy and yet as this plot and the data shows these results have error values less that 10^{-7}. The line representing $h = 1.001$ also gives quite a good qualitative indication of near critical results with the slight divergence from power law implying the expected exponential decay.

4.2.4 Numerical Considerations

As mentioned earlier the MPS relies on truncating the dimensions of tensor bonds to some finite value χ. Since increasing χ increases the size of the tensors we are considering it is important to find a value χ that can reliably reproduce the state while still being sufficiently low that computing ground states does not

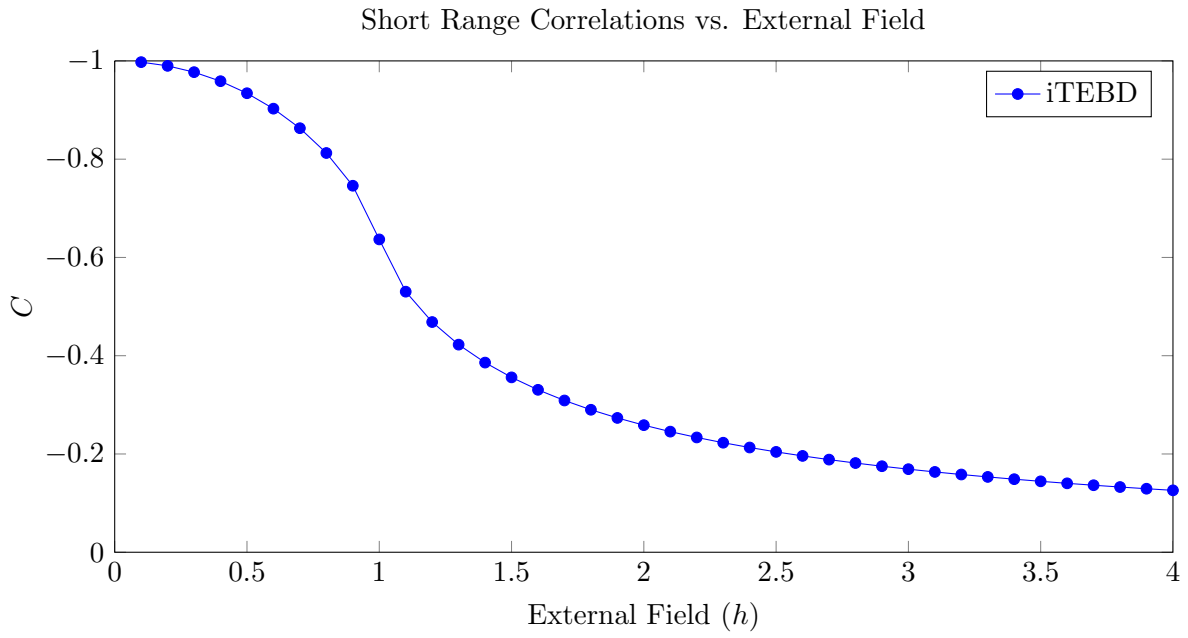


Figure 4.3: Plot showing the short range correlation given by $C = \langle \hat{S}_x^{[i]} \hat{S}_x^{[i+1]} \rangle$ vs. the external field strength for the iTEBD algorithm. The curve represents the ground state of the one-dimensional infinite Ising model with exchange interaction strength $J = 1$. The simulations were based on $\chi = 70$ and a minimum time step $\tau = 10^{-7}$.

take too long. It was found through testing that for the one-dimensional infinite Ising chain increasing the χ beyond as little as 30 did not increase the accuracy. For this reason it was determined that a χ of 50 or more was sufficient to reproduce most results away from the critical point.

In order to quantify the importance of increasing χ we are able to analyse the Schmidt weights of our λ tensors. From figure 4.5 we see that for all systems sufficiently far from the critical point the smallest Schmidt weights become very insignificant. This suggests that increasing χ would not necessarily have any measurable quantitative change on the system. This plot also makes very clear that for values approaching the critical point a much larger χ is required to accurately represent the state.

It is known that the for the iTEBD algorithm the computational time complexity scales as $\mathrm{O}(d^3\chi^3)$ where d is the local dimension of the particles being considered and χ is the truncation parameter in the MPS. Figure 4.6 gives some qualitative support for this prediction, as we see the time taken is increasing at least quadratically, but χ is not large enough for the exact power law to be deduced. This plot shows also that the time taken is longer for the critical case than others.

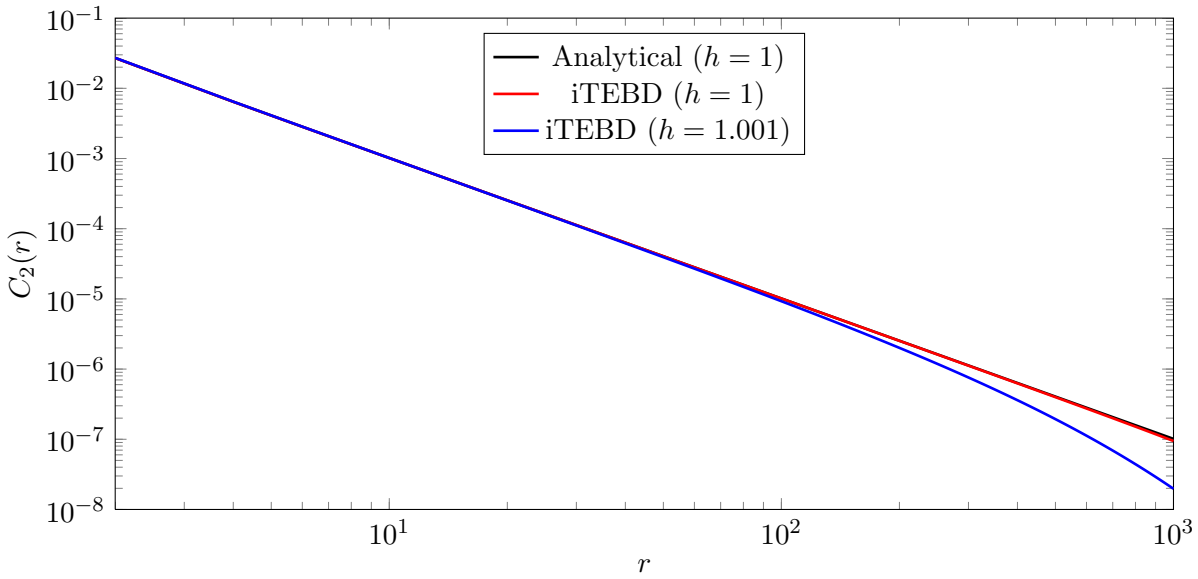


Figure 4.4: Long range correlation values for the infinite one-dimensional Ising model at and near the critical point. These values are all corresponding to $\chi = 150$ and minimum time step $\tau = 10^{-12}$.

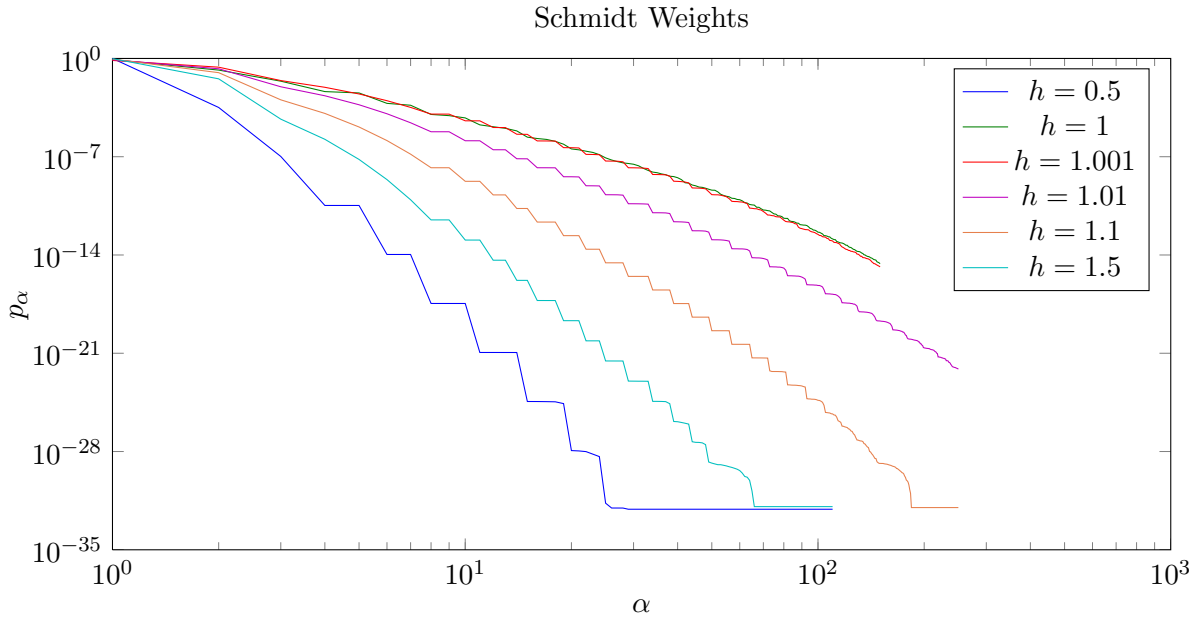


Figure 4.5: Plot of the weights of the elements of λ^A. Here we plot $p_\alpha = (\lambda_{\alpha\alpha})^2$ for different values of the external field strength h. The curves represent the ground state of the one-dimensional infinite Ising model with exchange interaction strength $J = 1$.

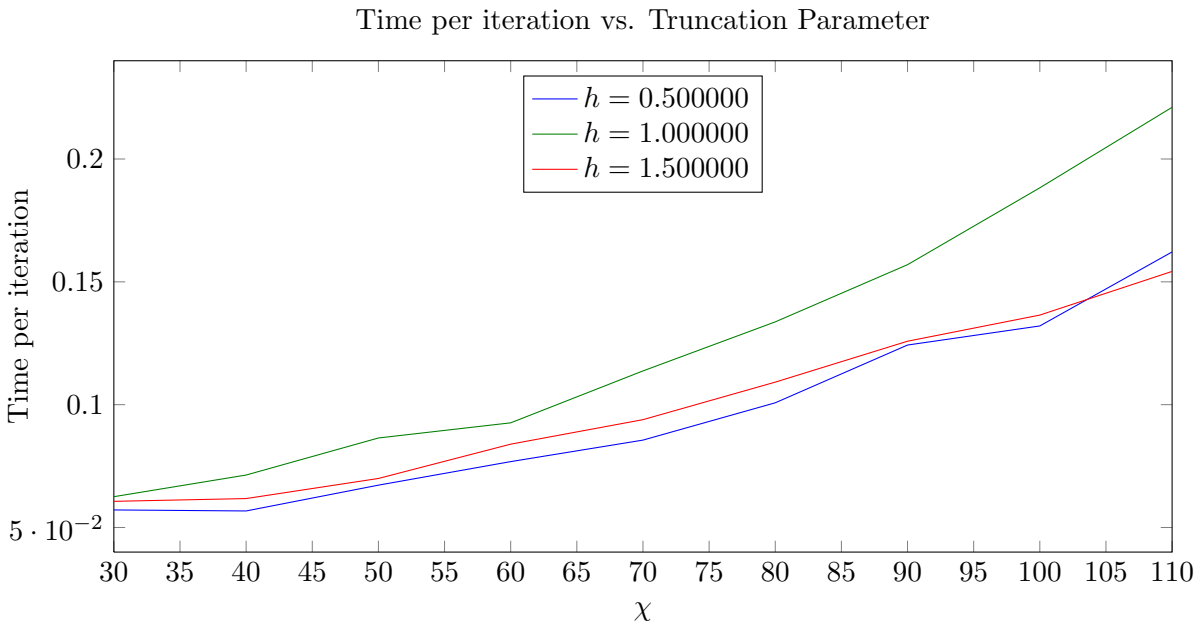


Figure 4.6: Plot showing the time taken for one iteration of the iTEBD algorithm for different values of h. The curves represent the ground state of the one-dimensional infinite Ising model with exchange interaction strength $J = 1$.

Chapter 5

The Two-Dimensional Ising Model in a Transverse Field

After benchmarking the iTEBD algorithm for a number of one-dimensional systems in the previous chapter we will now discuss how this can allow us to study properties of two-dimensional spin lattices. In particular we are interested in studying the triangular lattice introduced in the first chapter. In order to do this we will introduce ladder systems and look at new scientific results achieved by this research project.

5.1 Ladder Lattices

Infinite ladder lattices as we will call them refer to lattices created by adjacent infinitely long One-Dimension spin chains. Using the successfully benchmarked iTEBD code some properties of these systems have been studied. In order to extend an algorithm that was applicable to one-dimensional infinite chains to such systems we can group pairs of adjacent particles in the ladder together to form a pseudo-particle now with a local Hilbert space whose basis states are formed by the tensor product of the two particles. This means that the local dimension of the site becomes the produce of the local dimensions of the sites that were grouped to form it. The method for calculating ground states and calculating expectation values then is the same as the one-dimensional case.

5.1.1 Double Chain

Let us now introduce the Double spin chain depicted in figure 5.1. The Hamiltonian, H, for this system is defined by,

$$\hat{H} = \sum_{i=1}^{\infty} \left[J_1 \hat{S}_x^{[2i-1]} \hat{S}_x^{[2i]} + J_2 \hat{S}_x^{[2i-1]} \hat{S}_x^{[2i+2]} + J_3 (\hat{S}_x^{[2i-1]} \hat{S}_x^{[2i+1]} + \hat{S}_x^{[2i]} \hat{S}_x^{[2i+2]}) + h(\hat{S}_z^{[2i-1]} + \hat{S}_z^{[2i]}) \right]. \quad (5.1)$$

In practice the iTEBD algorithm used involved grouping pairs of spins $2i-1$ with $2i$ and $2i+1$ with $2i+2$ to pseudo-spin three-half particles in order to perform the evolution to the ground state through the same method as the one-dimensional system. As shown in figure 5.1 the spins are grouped in pairs with one spin from each chain.

5.2 Theory

Since there is no analytical solution to the two-dimensional Ising model in a transverse field on the triangular lattice it was necessary to look at certain extreme cases in order to benchmark this updated

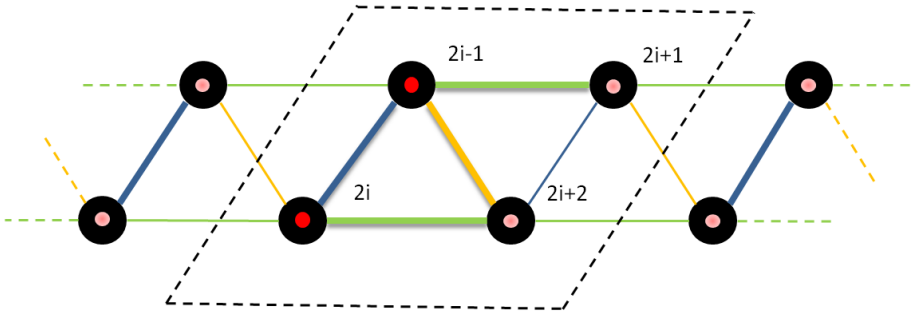


Figure 5.1: Two by infinite spin triangular lattice.

algorithm. Exact diagonalisation with Finite Size Scaling also allowed for comparison of ground state energies for these ladder systems.

5.3 Results - Double Chain

The results that follow are based on simulations utilising a code that was written specifically as part of this research project. In particular a general code for exact diagonalisation of the Two-Dimension Ising model on a triangular lattice as well as an extension of the iTEBD code from the previous chapter to these ladder systems.

5.3.1 Finite Size Scaling

Finite size scaling involves finding properties for increasingly large systems and then extrapolating these results to the thermodynamic limit. This technique is a widely used method to avoid diagonalising intractably large matrices but still study systems in the thermodynamic limit [39, 40].

In order to compare results between the iTEBD algorithm applied to (5.1) and the exact diagonalisation, ground state energies of 2 by L lattices of increasing length, L, were determined by exact diagonalisation. We can see from figure 5.2 that the energy per spin of the lattices does converge quite well at a lengths as little as 12. It should also be noted that the energy per spin at every odd length chain is slightly higher, due to the effects of the periodic boundary conditions combined with the Anti-Ferromagnetic interactions. This means in practice that for extrapolation of energies we choose to only consider chains of even length.

We can also see by looking at the different curves in figure 5.2 that larger external fields cause the odd-even variations to be smaller. This is to be expected as the external field acts only on single sites. When the ground state of the system has spins completely aligned with the external field then we would predict that the energy per spin would just be proportional to the external field and would not vary with chain length, which is a trend that can be seen for increasing h values in figure 5.2.

From the results of the Finite Size Scaling the ground state energies of the double ladder chain could be fairly accurately predicted by a chain of length 12. Using these results we were able to further validate the accuracy of the iTEBD algorithm when applied to these ladder systems. From figure 5.3 we can see that there is a very strong agreement between the ground state found by the iTEBD algorithm and exact diagonalisation with Finite Size Scaling.

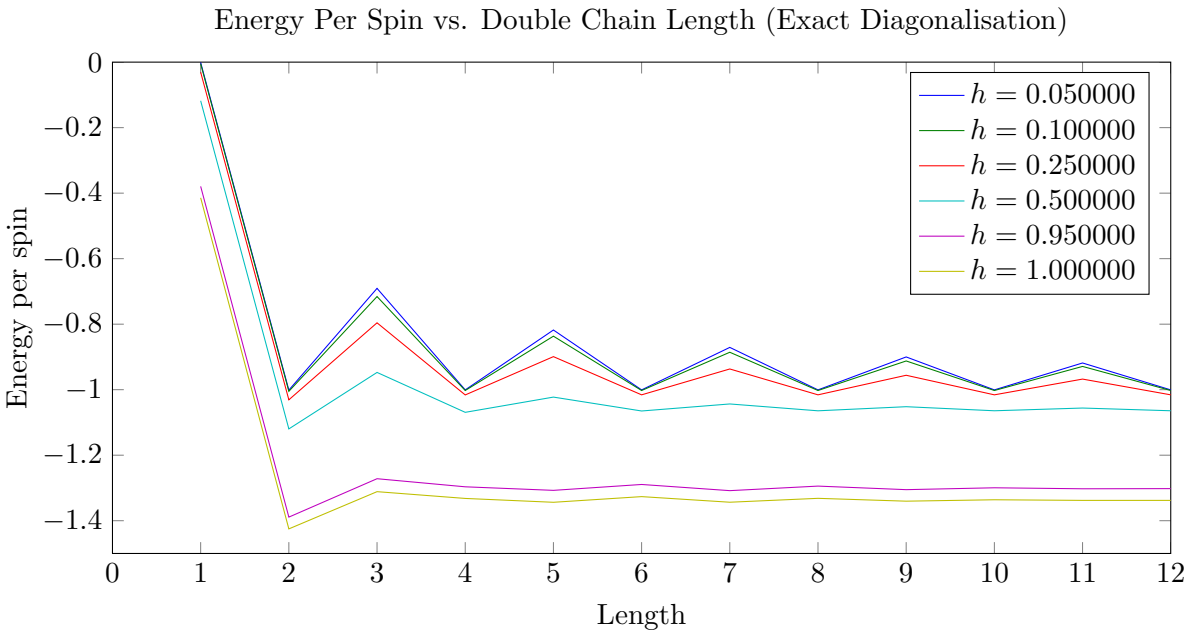


Figure 5.2: Plot showing the energy per spin for an Ising model on a 2 by L triangular lattice vs. the length, L. These results are from exact diagonalisation of the Hamiltonian where all exchange interactions are 1 and different lines are for different external field strength, h.

5.3.2 Energy

Further support for the accuracy of the iTEBD algorithm applied to the Hamiltonian defined by (5.1) can be found by considering the case in which we have $J_1 = J_2 = 1$ and $J_3 = 0$. From this it can be noted that this reproduces exactly the same system as the infinite one-dimensional Ising model. For this reason we were able to compare results from the known analytical values with those produced by the iTEBD algorithm. We can see in figure 5.4 that there is, indeed, good agreement between the analytical results and the numerics.

Given that the infinite one-dimensional Ising model does not have geometric frustration and also that this corresponds to having $J_1 = J_2 = 1$ and $J_3 = 0$ in (5.1) and that the triangular lattice is expected to be geometrically frustrated, the increase in frustrating bonds could be simulated by increasing the strength of the J_3 interaction term in the Hamiltonian. Results of this nature could give some insight into the affects of geometric frustration on triangular Lattice Ising systems.

We can also look at certain limiting cases in our Hamiltonian. Firstly we noted that if we were to have $J_3 \ll J_1, J_2$ then we should simply reproduce the physics of the one-dimensional spin chain. Now also if we were to consider the case in which $J_3 \gg J_1, J_2$ then we expect again to be reproducing the physics of the one-dimensional Ising chain as this is equivalent to having two one-dimensional Ising chains. In between these two limits, however, the behaviour of the system is non-trivial.

Figure 5.5 shows how the ground state energy changes for increasing J_3. We can see that the system has least binding energy around $J_3 = 0.5$ for the different values of external field strength. The system also seems to fall to some minimum binding energy at this value then slowly rise back to the same binding energy as $J_3 = 0$ when $J_3 = 1$. This behaviour seems to be indicative of the geometric frustration of the Anti-Ferromagnetic triangular lattice. The shape of the curve seems to suggest that this frustration is greatest at $J_3 = 0.5$ as this point gives the lowest binding energy.

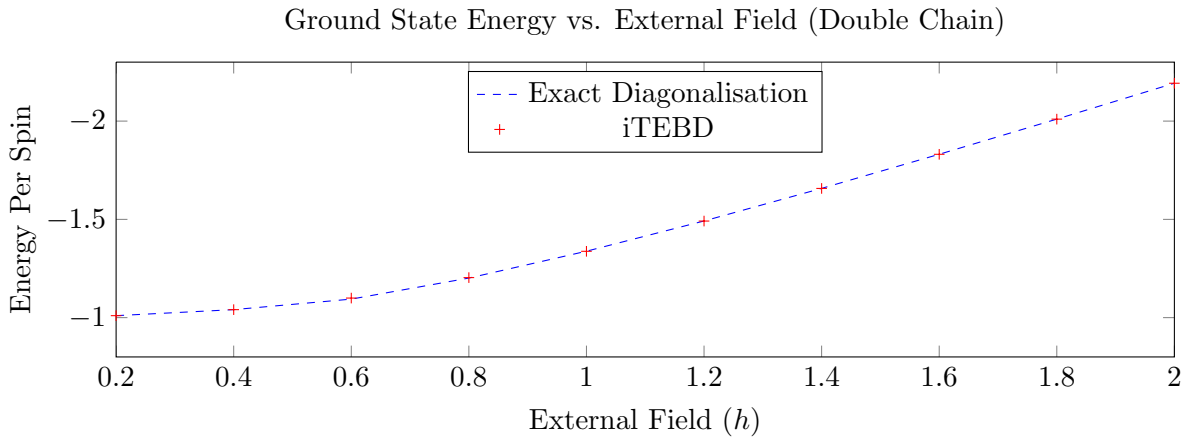


Figure 5.3: Plot showing comparison of the energy per spin of exact diagonalisation of a 2 by 12 triangular lattice and the iTEBD code. All results are for ground state of the Hamiltonian defined by (5.1) where all exchange interactions, J_i, are 1.

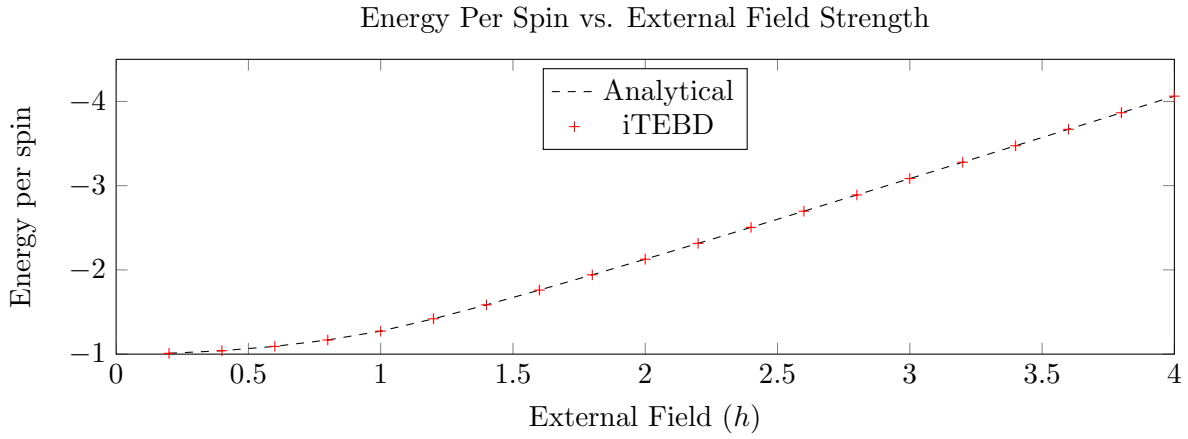


Figure 5.4: Plot showing comparison of analytical results to iTEBD results for Hamiltonian defined by (5.1) with $J_1 = J_2 = 1$ and $J_3 = 0$. Analytical results are the same as the infinite one-dimensional Ising model.

5.3.3 Spin Expectation Values

In order to understand the ground states chosen in these geometrically frustrated spin systems we study the expectation values of different Pauli spin operators. Since we have nearest neighbour interactions along the x axis and external field that is directed along the z axis it is sensible to consider the expectation values of these terms that contribute to the Hamiltonian.

From figure 5.6 we can see that when the interaction term J_3 is small the system chooses a ground state which minimises expectation values of x spin-spin correlations between the spins whose interaction strength is given by J_1 or J_2. Such behaviour is to be expected as we know that the external field strength in these systems is small and thus the ground state must minimise its energy by minimising the strongest exchange interactions. It is also clear by looking at figure 5.1 it is not possible to simultaneously minimise all of the interaction terms.

As the J_3 interaction strength becomes sufficiently large spins flip to minimise interactions defined by the J_3 term in the Hamiltonian. This then results in the minimising all exchange terms in the Hamiltonian except for those given by J_1. We can therefore see that there is essentially two interactions defined by the J_3 term competing with one interaction defined by the J_1 term. Hence it seems sensible to see the

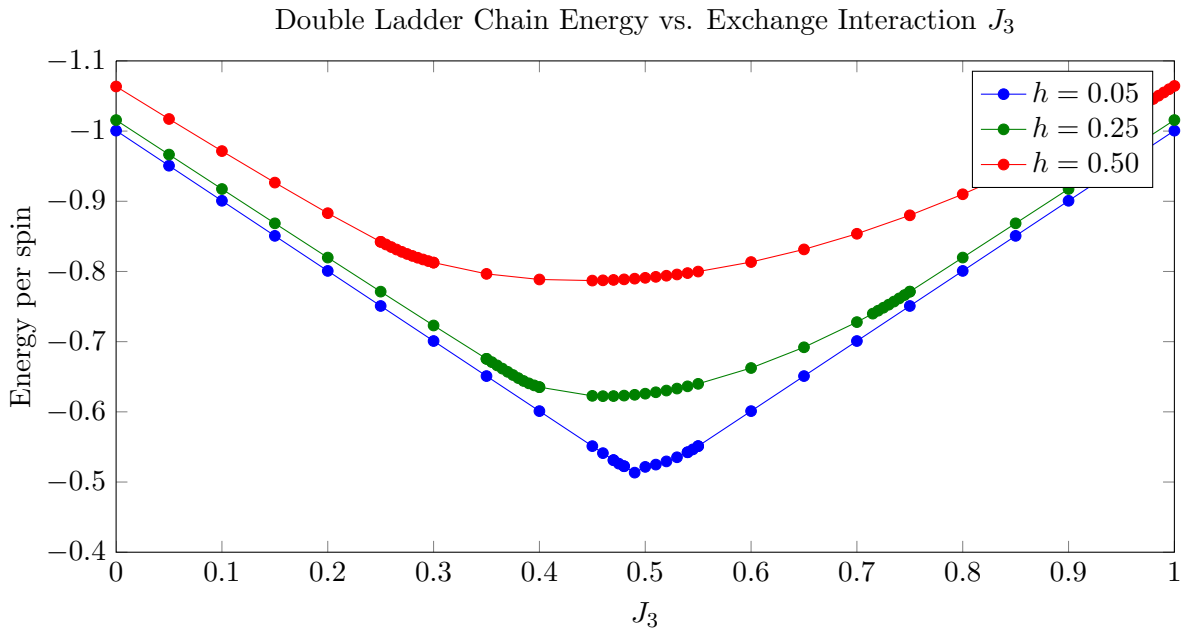


Figure 5.5: The ground state energy per spin of the Hamiltonian defined by (5.1) with $J_1 = J_2 = 1$ from iTEBD.

transition between ground states occurring around $J3 = J1/2 = 0.5$ as we do in figure 5.6.

By looking at the z expectation values it is also evident that the width of the transition region in the middle is caused by the system choosing a ground state that actually minimises its energy by aligning more with the external magnetic field. Evidence for this hypothesis can be obtained by comparing figures 5.6, 5.7 and 5.8. We see that as the external field strength increases the width of the region in which a greater binding energy is achieved by alignment with the external field also increases.

The phase transition behaviour is actually most evident in the $\hat{S}_x$ expectation values of individual spins. We can see that for all the different external field values the transition between different ground state behaviour has been very sudden when looking at these measurements.

5.3.4 Phase Transitions

As we have discovered it is possible to see phase transition behaviour in our double ladder chain system defined by (5.1) by looking at $\hat{S}_x$ expectation values of individual spins. We can therefore consider the h-J_3 phase diagram of such a model by finding phase transitions in these observables. The approach to finding highly accurate predictions for the position of these phase transitions is to pick a certain value of h and then sweep J_3 over smaller and smaller ranges and determine the point at which the expectation values suddenly change.

From figure 5.9 we can see that for small h and J_3 the system is in the so called zig-zag phase (this is the region below and to the left of the red line). This corresponds to the simultaneous minimisation of exchange interactions given by J_1 and J_2 (see figure 5.10). This corresponds to the regions in figures 5.6, 5.7 and 5.8 in which $\hat{S}_x^{[2i-1]} \approx -1$ and $\hat{S}_x^{[2i]} \approx 1$. As the J_3 gets larger (for some finite h) the system moves into the Ordered phase in which energy is being minimised by aligning more with the external field, corresponding to the regions in which all $\hat{S}_x \approx 0$. And as J_3 becomes sufficiently large the system moves to the Striped phase, in which $\hat{S}_x^{[2i-1]} \approx -1$ and $\hat{S}_x^{[2i]} \approx -1$.

As mentioned previously we expect that when we have $J_1 = J_2 = 1, J_3 = 0$ we are reproducing the physics of the one-dimensional infinite chain, which has a phase transition when $h = 1$. This suggests that we expect to see the phase boundary intersect the h axis at $h = 1$ which is consistent with the shape

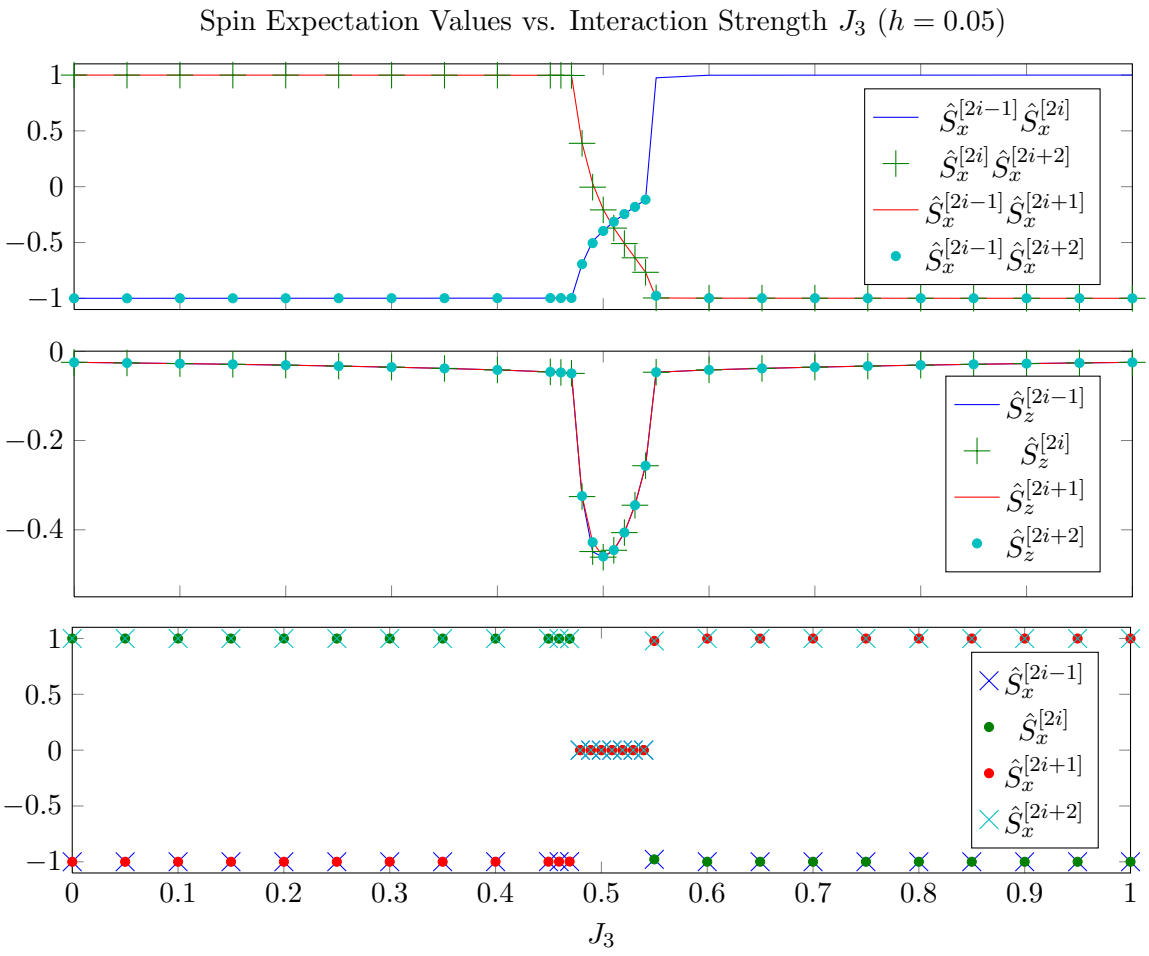


Figure 5.6: Expectation values of various operators on the ground state of the Hamiltonian defined by (5.1) with $J_1 = J_2 = 1$.

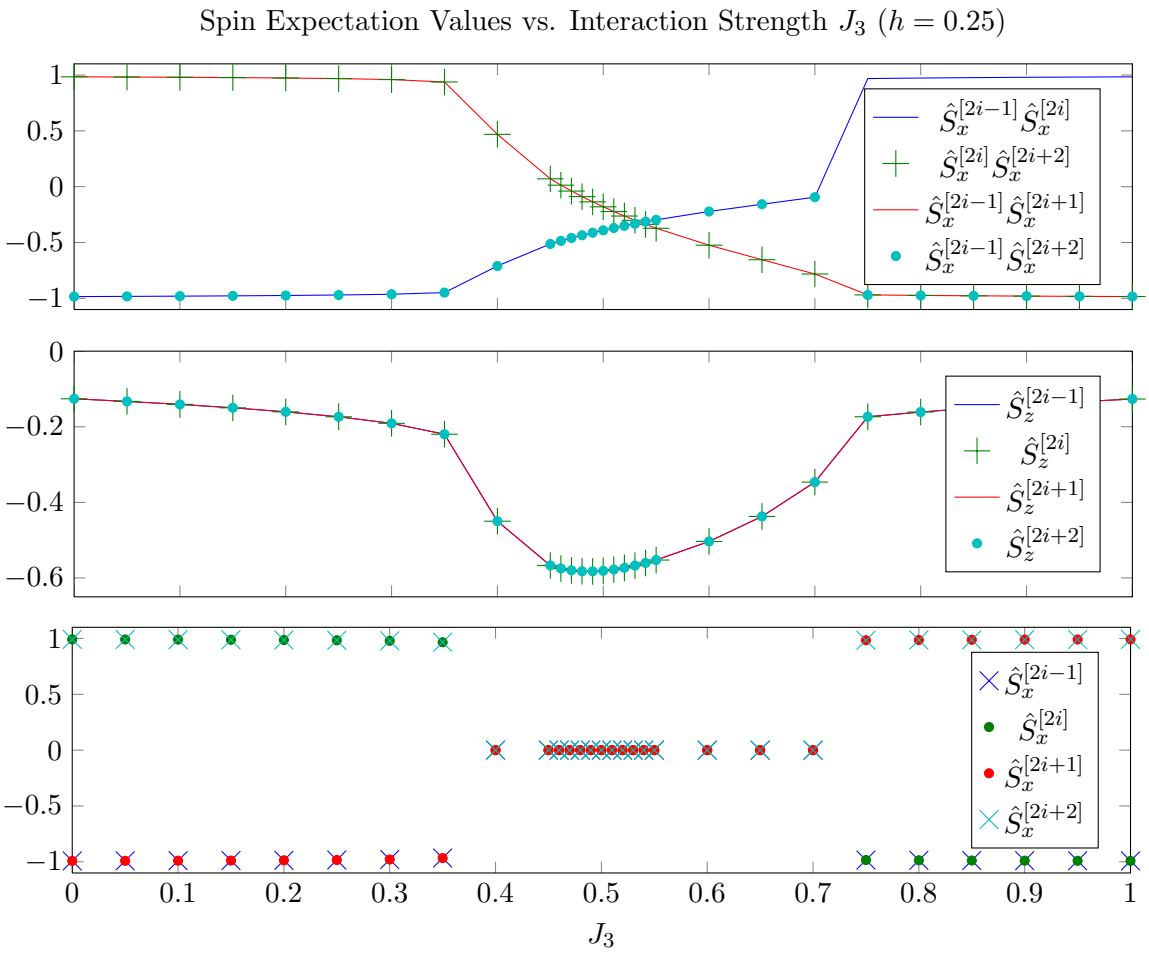


Figure 5.7: Expectation values of various operators on the ground state of the Hamiltonian defined by (5.1) with $J_1 = J_2 = 1$.

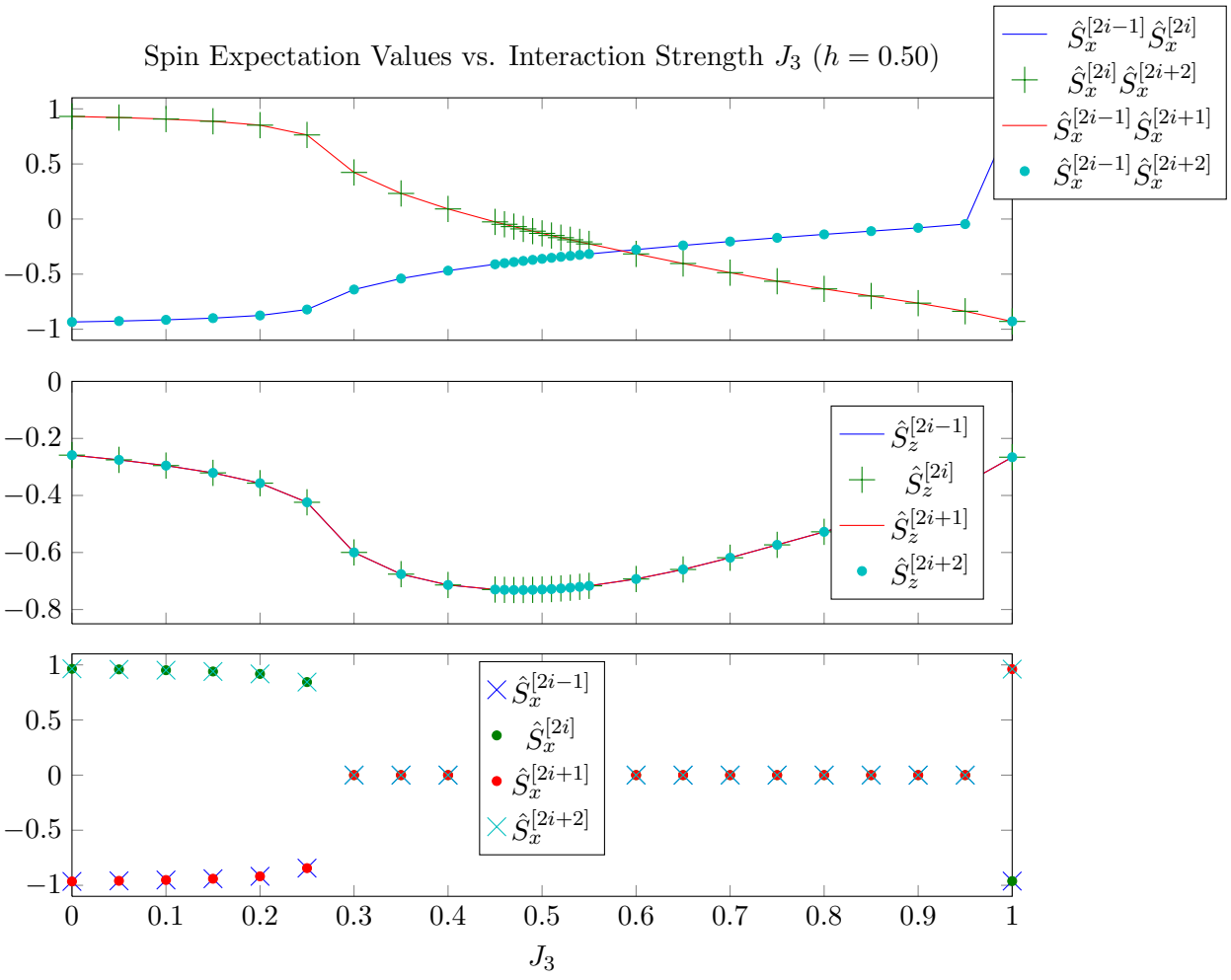


Figure 5.8: Expectation values of various operators on the ground state of the Hamiltonian defined by (5.1) with $J_1 = J_2 = 1$.

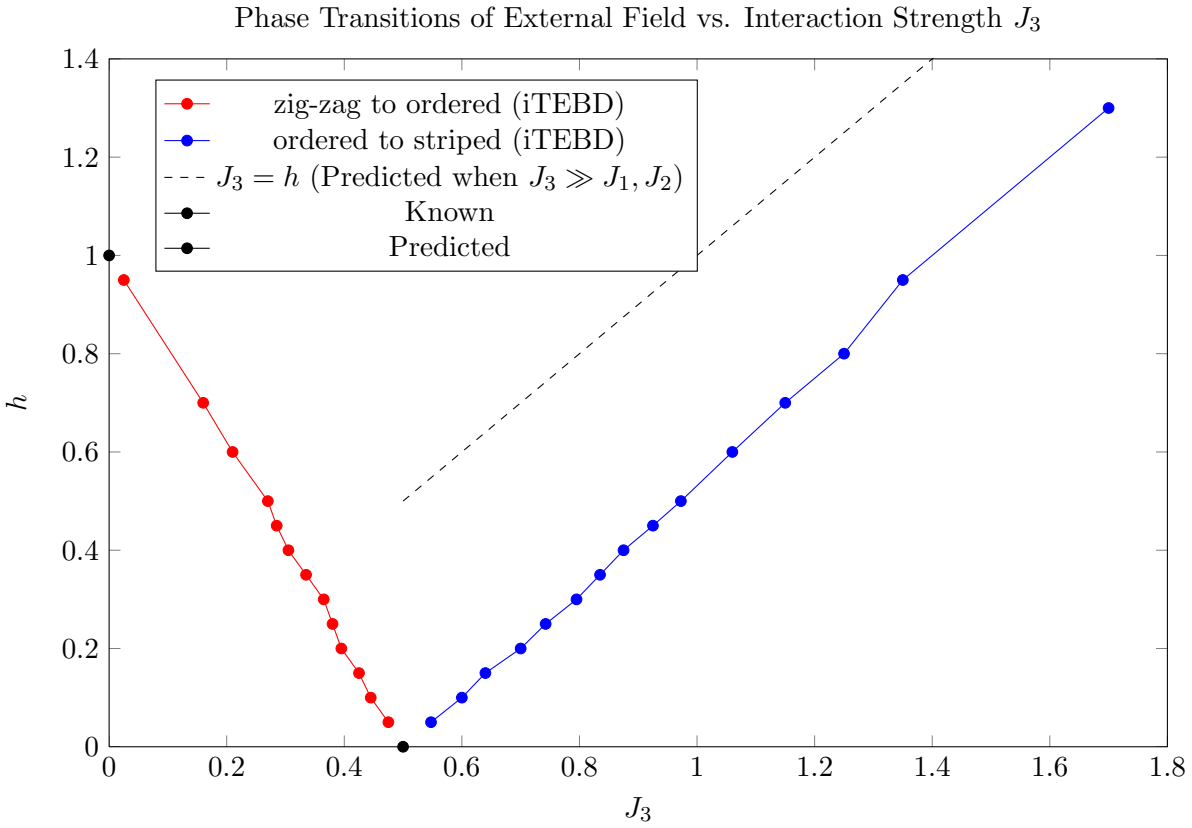


Figure 5.9: Phase transitions on the ground state of the Hamiltonian defined by (5.1) with $J_1 = J_2 = 1$. Here we are plotting, for various values of h the J_3 values at which we see a sudden transition for measurements of individual S_x expectation values similar to those depicted in figures 5.6, 5.7 and 5.8. The black dot at (0,1) refers to the known phase transition in the one-dimensional Ising model in a transverse field, since we know that when $J_3 = 0$ we are reproducing the one-dimensional model. The black dot at (0.5,0) refers to the expected competition between the two J_3 interactions and one of the other J interactions. Finally the dashed line gives the large J_3 limit prediction for the phase boundary between the ordered and striped phases. We expect that when $J_3 \gg J_1, J_2$ we will again reproduce the one-dimensional system which will then have a phase transition at $J_3 = h$.

and position of phase boundary we see in 5.9. It should also be noted that when $J_3 \gg J_1, J_2$ then again we are reproducing the one-dimensional infinite chain. In this regime the system should have a phase boundary $J_3 = h$ which means we should expect the phase boundary to tend toward this line as $J_3 \to \infty$.

By extending the iTEBD algorithm to the infinite double chain system we were able to accurately determine several points along the phase boundaries of the h-J_3 phase diagram. In particular we discovered distinct phases that we called the zig-zag phase, in which the energy of the anti-Ferromagnetic interactions were minimised along a zig-zag pattern of the chain, the ordered phase in which ground state energy was minimised by the spins aligning more with the external magnetic field, and finally the striped phase in which the energy of the anti-Ferromagnetic was minimised along stripes of the ladder.

We expect these results may give some insight into phases of the infinite two-dimensional triangular lattice, however the lattice is not yet wide enough to give quantitative predictions for such systems. Similar to the finite size scaling already presented we expect that as these ladders become sufficiently large they will begin to represent bulk behaviour of two-dimensional triangular lattices. It could therefore be useful to study wider ladder chains to learn about the infinite two-dimensional lattices. Also by studying ladders of varying widths and boundary conditions we may begin to understand the effect of boundaries on two-

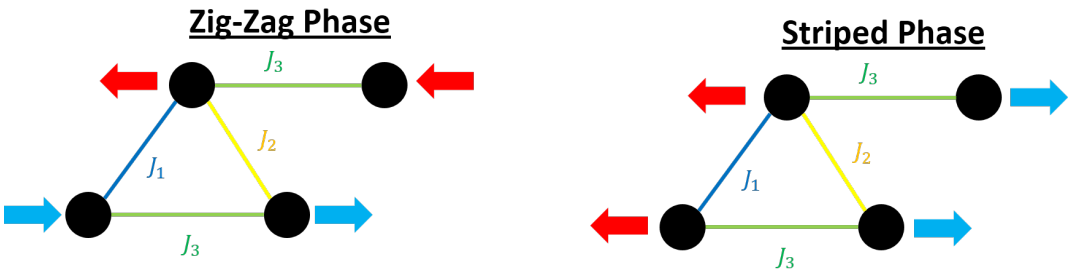


Figure 5.10: Figure showing the phases of the system defined by (5.1) in the extreme limits. The ground state of the system found by the iTEBD algorithm is expected to consist of infinitely repeating these four spins along a chain. The labels next to the bonds indicates the parameter corresponding to the strength of the exchange interaction.

dimensional systems. Although we expect the time complexity to be exponential with increasing ladder width as well as the Hamiltonian being more difficult to construct we will attempt to study triple ladder chain systems in the following section.

5.4 Results - Triple Chain

The methods used were then extended to apply to a three-chain system (see figure 5.11), this time with periodic boundary conditions. The purpose of this was to apply finite size scaling arguments to predict bulk behaviour of two-dimensional triangular lattices under and Ising model. Various properties were calculated for this new system with an emphasis on benchmarking the validity of the extension as well comparing the phase structure of the system to that of the two-chain system. The Hamiltonian, H, for the triple chain system is defined by,

$$\begin{aligned}\hat{H} =& J_1(\hat{S}_x^{[3i-2]}\hat{S}_x^{[3i-1]} + \hat{S}_x^{[3i]}\hat{S}_x^{[3i+2]} + \hat{S}_x^{[3i-2]}\hat{S}_x^{[3i+3]}) \\ &+ J_2(\hat{S}_x^{[3i-2]}\hat{S}_x^{[3i+2]} + \hat{S}_x^{[3i-1]}\hat{S}_x^{[3i]} + \hat{S}_x^{[3i-2]}\hat{S}_x^{[3i]}) \\ &+ J_3(\hat{S}_x^{[3i-2]}\hat{S}_x^{[3i+1]} + \hat{S}_x^{[3i-1]}\hat{S}_x^{[3i+2]} + \hat{S}_x^{[3i]}\hat{S}_x^{[3i+3]}) \\ &+ h(\hat{S}_z^{[3i-2]} + \hat{S}_z^{[3i-1]} + \hat{S}_z^{[3i]})\end{aligned} \tag{5.2}$$

The comparison to results from Finite Size Scaling (see figure 5.12) of exact diagonalisation again show that the iTEBD algorithm is very likely finding the ground state of the system and the physics being simulated is accurate. The difference in energies for all values are of order no more than 10^{-2} for all energies and we expect this is due to the inherent approximations made in the Finite Size Scaling of the exact diagonalisation. It should be noted we expect that the iTEBD algorithm is able to produce more accurate results relating to the thermodynamic limit of the ladder systems we are looking at. The iTEBD algorithm also allows us to compute long range correlations (see appendix A) and measure individual spin expectation values very simply.

Phase transitions again are expected in this model similar to those of the Double Chain system. For this reason it is useful to look again at the expectation values of different spins for different values of h and J_3. From figure 5.13 we can see again similar phase transitional behaviour to that observed in the two-chain system. It is expected that the phase diagram of this system is more complex to that of the two-chain system but given the reliability of the iTEBD algorithm we expect that a great deal can be predicted using the code we have already created. Although we are limited by exponential complexity for increasing number of chains in our ladder systems, similar to the Finite Size Scaling used with exact diagonalisation, we expect to be able to gain insight into bulk behaviour of two-dimensional triangular lattices as well as make predictions about boundary effects in finite size lattices.

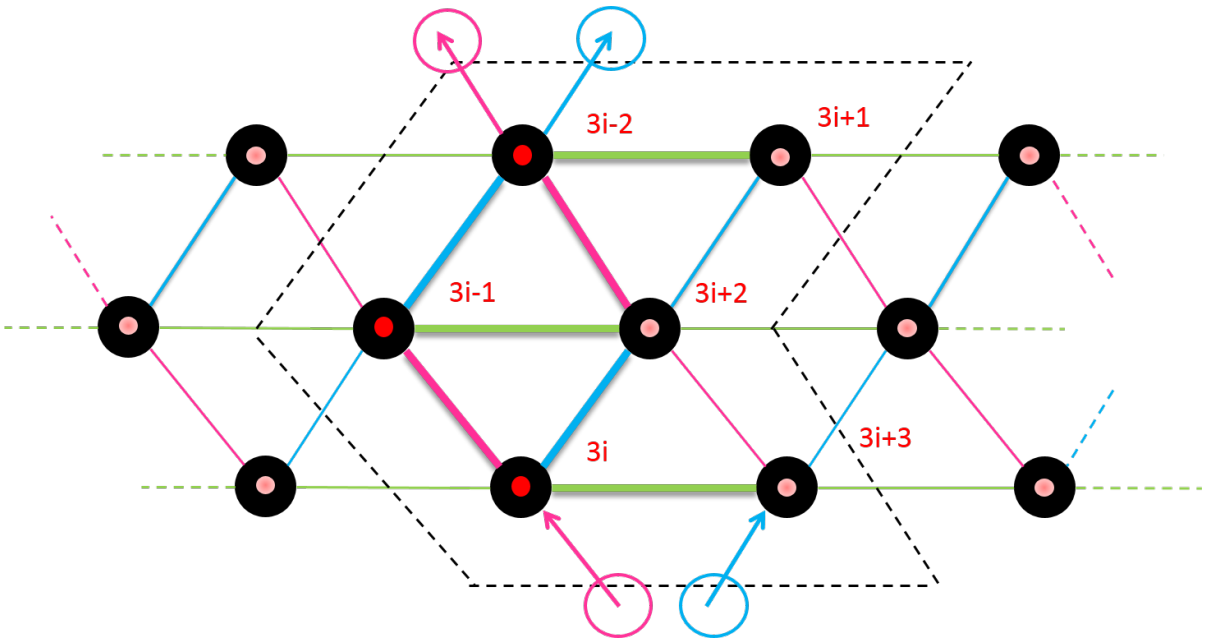


Figure 5.11: Three by infinite spin triangular lattice.

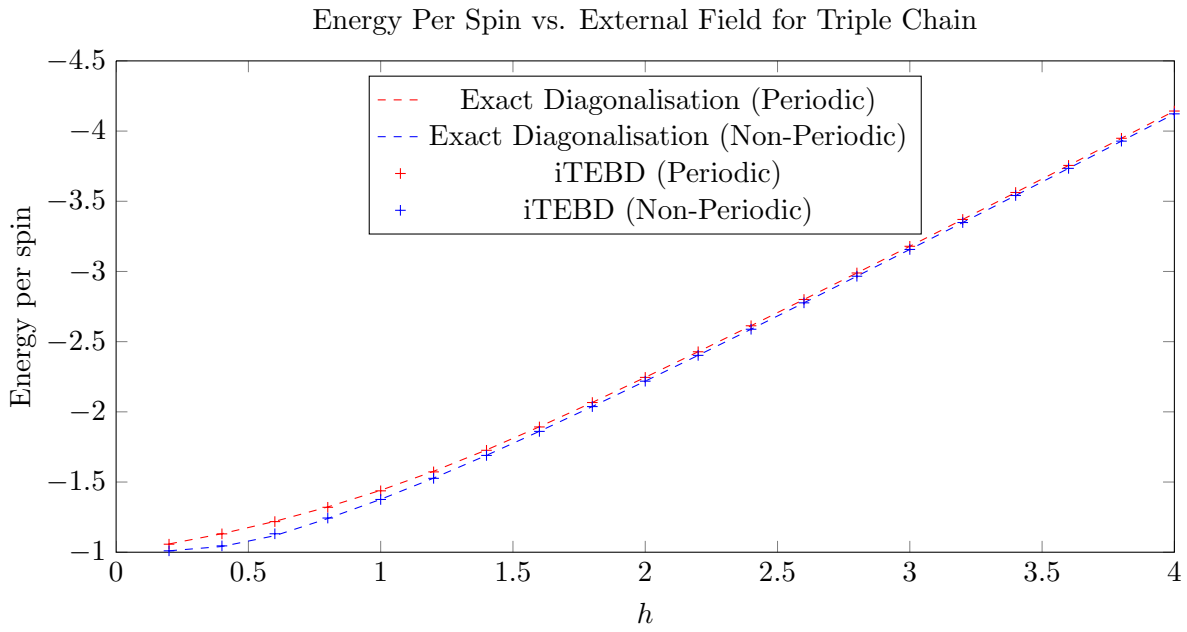


Figure 5.12: Comparison of the energy per spin of exact diagonalisation of a 3 by 8 triangular lattice and the iTEBD code. All results are for ground state of the Hamiltonian defined by (5.2) where all exchange interactions strengths, J_i, are 1. Non-periodic boundaries relate to removing the terms $\hat{S}_x^{[3i-2]}\hat{S}_x^{[3i+3]}$ and $\hat{S}_x^{[3i-2]}\hat{S}_x^{[3i]}$ from the Hamiltonian.

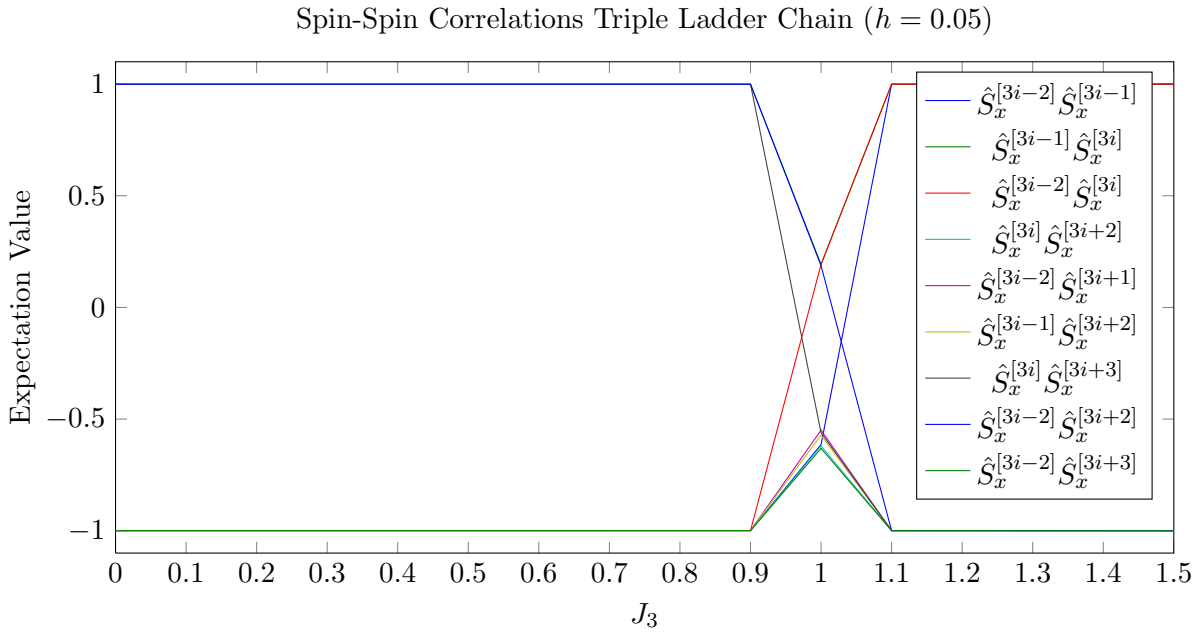


Figure 5.13: Spin-spin correlations along x-axis for ground state of triple ladder chain (Hamiltonian defined by (5.2)) with $J_1 = J_2 = 1$ and $h = 0.05$.

Chapter 6

Conclusion and Outlook

The goal of this project was to study the effects of frustration in the Ising model on a triangular lattice. In order to achieve this goal a code for simulations using the iTEBD algorithm on infinite one-dimensional spin systems was developed. The algorithm was successfully benchmarked with a wide range of known quantities as well as numerical results from exact diagonalisation with Finite Size Scaling code that was also written as part of this project. The iTEBD algorithm was then extended to simulate ladder chains and was successfully benchmarked against results from the exact diagonalisation.

As a direct result of this research we have, with high accuracy, determined the location of phase transitions in the triangular double ladder system. We have discovered three distinct phases occurring as we vary the parameters present in the Hamiltonian of the two-chain system, including: a zig-zag phase representing minimisation of Anti-Ferromagnetic interactions along a zig-zag chain pattern, an ordered phase representing a sudden tendency to minimisation of external field interactions, and finally a Striped phase in which Anti-Ferromagnetic interactions along stripes of the ladder are minimised.

Finally we were able to calculate various properties in the three-chain ladder system and compare energies again to exact diagonalisation. It is expected that the iTEBD is producing reliable results and is able to calculate the kinds of properties necessary for analysing the phase diagram of this more complex system.

A a result of this research project we now know the phase diagram of the double ladder chain quite accurately and we have a good indication that our algorithms could be applied to the triple ladder chain and even larger ladders. Collating our results with wider ladders we expect to be able to gain qualitative insight into bulk behaviour as well as boundary effects of a two-dimensional Ising model on a triangular lattice. We have also presented results relating to the effects of boundary conditions in such systems. We expect that by increasing the width of the ladders, it should be possible to observe how the nature of the phase diagram changes as we approach the infinite two-dimensional case.

Appendix A

Long Range Correlations (The C2 function)

The following figures describe, diagrammatically, how the C_2 function was calculated. Note that the C_2 value calculated needs to be divided by the norm (see the code correlation.m for exactly how all this is done).

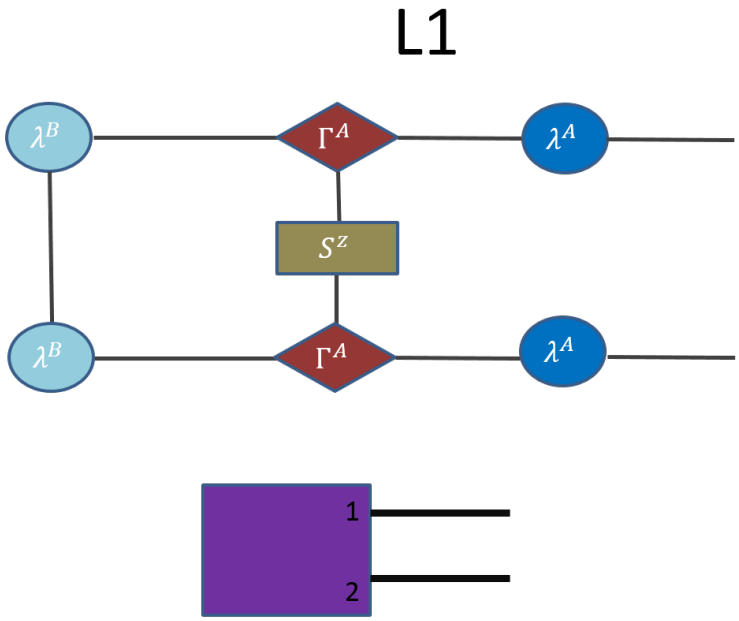


Figure A.1: L1 tensor.

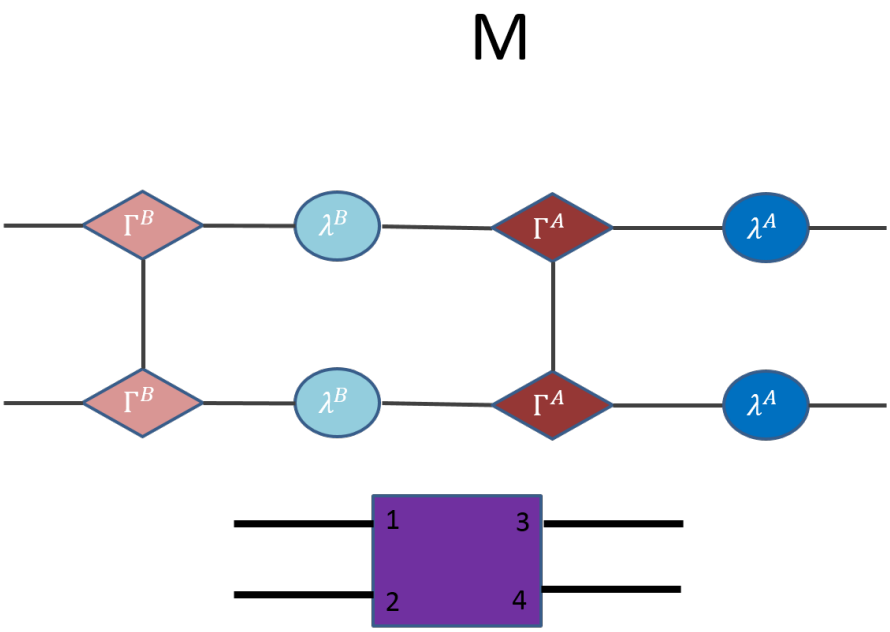


Figure A.2: M tensor.

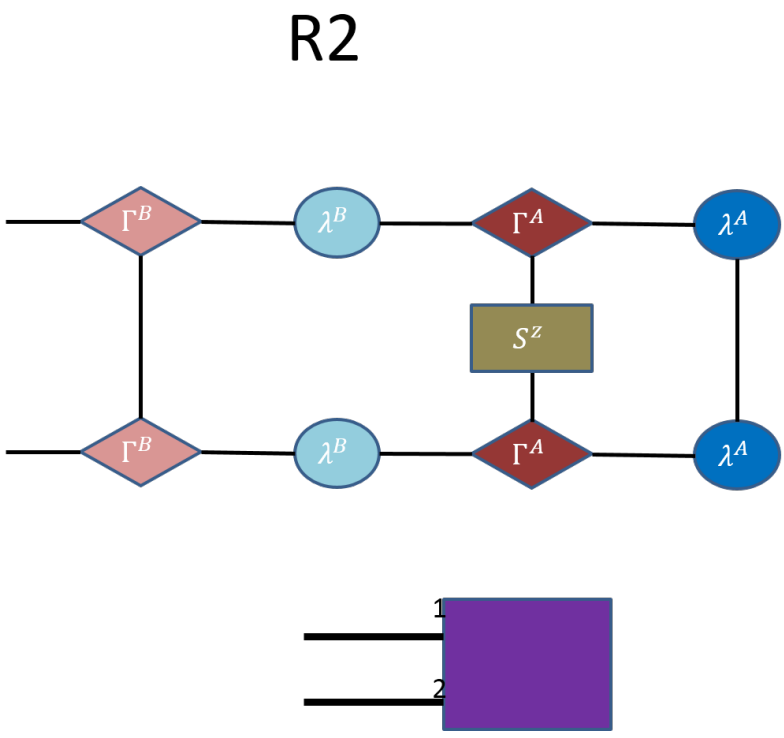


Figure A.3: R2 tensor.

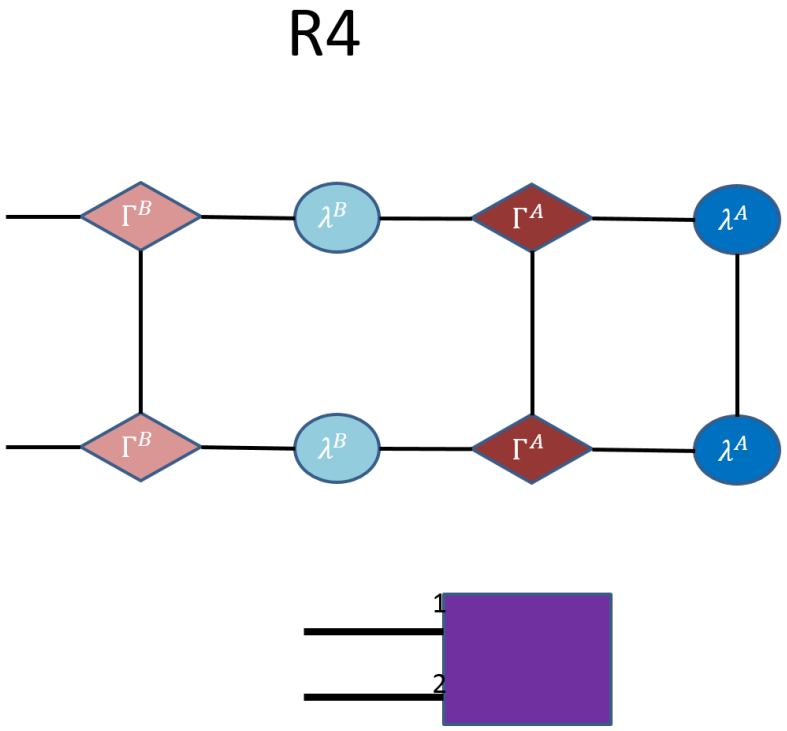


Figure A.4: R4 tensor.

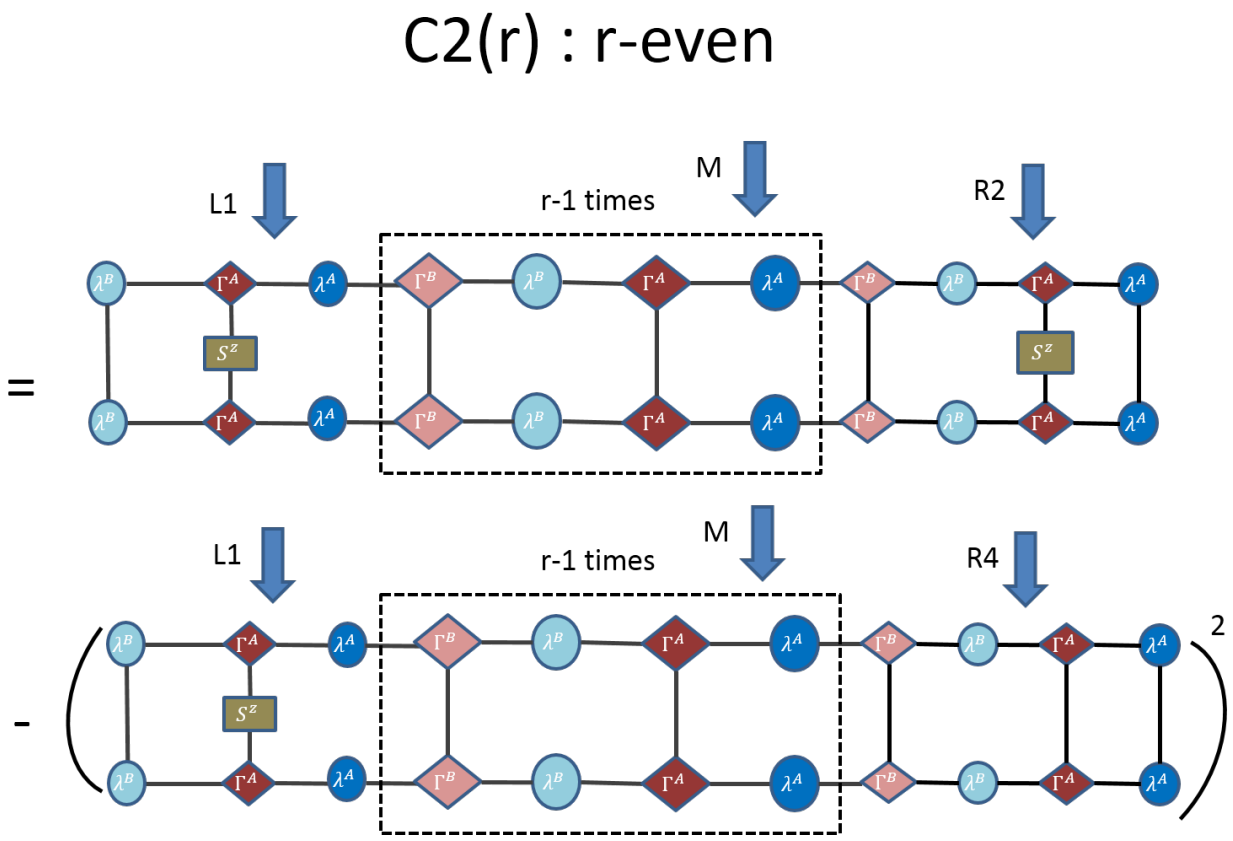


Figure A.5: C2 calculation (Note: this number still needs to be divided by the norm in order to get properly normalised expectation values).

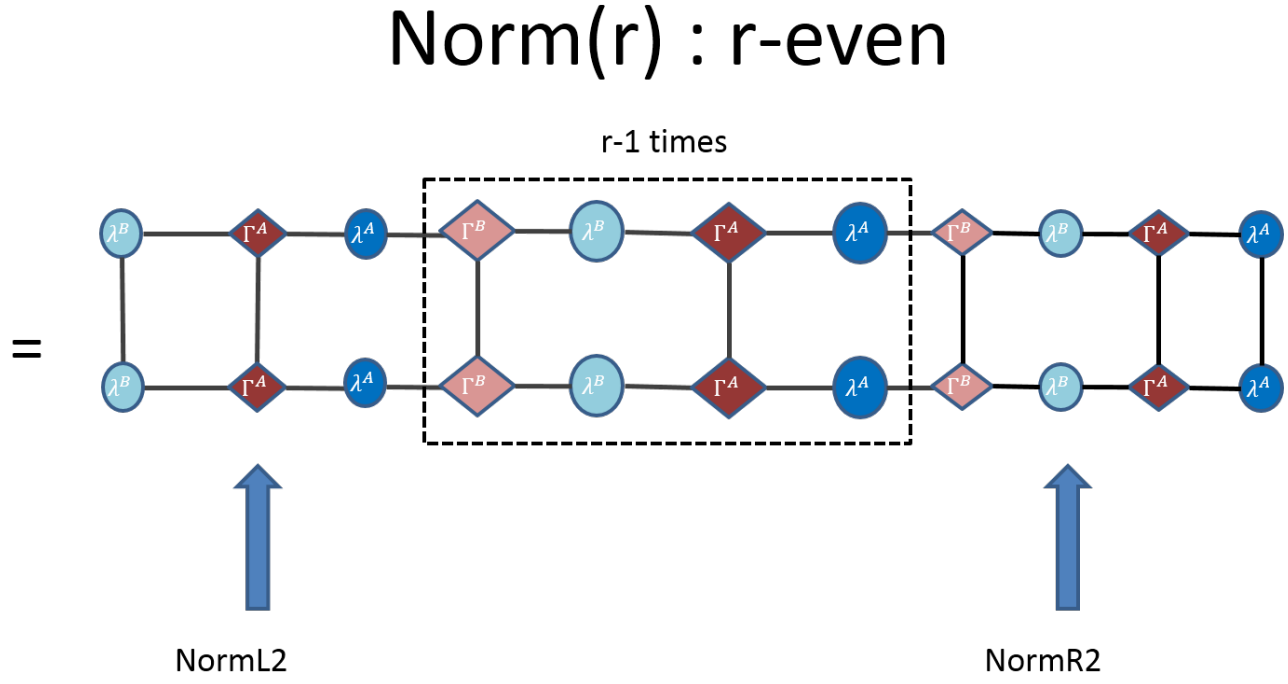


Figure A.6: Norm calculation.

Appendix B

Code

B.1 Exact Diagonalisation

The following is the code used to generate the sparse matrix to be diagonalised for the exact diagonalisation algorithm. The Matlab code included simply reads in the sparse matrix to be diagonalised. A perl script is included which simply automates the entire process and allows for easily setting a range of parameters. Finally a C code is given which is similar to the first exactDiagonalisation.c, but this implements the symmetry described in the chapter on exact diagonalisation. In the end this code was not actually used to calculate any results included in this paper, however it is included to show that the symmetry was implemented.

```
/*
 * C Program to generate all matrix elements for the Hamiltonian
 * of the Two Dimensional External Field Ising Model on a
 * Triangular Lattice.
 * This code gives options for periodic boundary conditions along
 * both axes, as well as variable interaction and external field
 * strength.
 * The matrix is printed to STDOUT in the sparse format.
 * Usage: ./exactDiagonalisation height width J h periodicEnds periodicTop
 *       height - is the number of rows of spins in the lattice
 *       width - is the number of columns of spins in the lattice
 *       J - is the exchange term in the hamiltonian
 *       h - is the transverse term in the hamiltonian
 *       periodicEnds - is a boolean indicating if it is periodic
 *                      on the left and right
 *       periodicTop - is a boolean indicating if it is periodic on the
 *                     top and bottom
 *
 *
 * By Dylan Griffith
 *
 */


#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#define FALSE 0
#define TRUE 1
#define MAXSIZE 1000

int hash(int lattice[][MAXSIZE], int height, int width);
void increment(int lattice[][MAXSIZE], int height, int width);
```

```
int triangleIsingExchange(int lattice[][MAXSIZE], int height, int width, int periodicEnds,
    int periodicTop);
void triangleIsingTransverse(int lattice[][MAXSIZE], int height, int width, double
   transverse);

int main(int argc, char *argv[]) {
    int lattice[MAXSIZE][MAXSIZE];
    int height = atoi(argv[1]);
  int width = atoi(argv[2]);
    double exchange = atof(argv[3]);
    double transverse = atof(argv[4]);
  int periodicEnds = atoi(argv[5]);
  int periodicTop = atoi(argv[6]);
  int nParticles = height*width;
    int permutations = 1 << nParticles;
    int i,j;
  // Initialise system
    for(i=0;i<height;i++) {
        for(j=0;j<width;j++) {
            lattice[i][j] = 1;
        }
    }
  // Loop over all basis states and print hamiltonian in sparse format
    printf("%d %d %f\n", 0, 0, exchange*((double) triangleIsingExchange(lattice, height,
        width, periodicEnds, periodicTop)));
    triangleIsingTransverse(lattice, height, width, transverse);
    for(i=1;i<permutations;i++) {
        increment(lattice, height, width);
        printf("%d %d %f\n", i, i, exchange*((double) triangleIsingExchange(lattice,
            height, width, periodicEnds, periodicTop)));
        triangleIsingTransverse(lattice, height, width, transverse);
    }
    return 0;
}

/**
  Calculates the total energy contribution
  of the exchange force for the given lattice.
  */
int triangleIsingExchange(int lattice[][MAXSIZE], int height, int width, int periodicEnds,
    int periodicTop) {
    int energy = 0;
    int i,j;
  int upshift;
    for(i=0;i<height;i++) {
        for(j=0;j<width;j++) {
      // Creates zig zag pattern down lattice
      upshift = 2*(i % 2) - 1;
      if (periodicTop && periodicEnds) {
        energy += lattice[i][j]*lattice[(i+1)%height][j];
        energy += lattice[i][j]*lattice[(i+1)%height][(j + upshift + width)%width];
        energy += lattice[i][j]*lattice[i][(j+1)% width];
      }else if (periodicTop) {
        energy += lattice[i][j]*lattice[(i+1)%height][j];
        if((j+1) < width) {
          energy += lattice[i][j]*lattice[i][(j+1)];
        }
        if(((j + upshift) >= 0) && ((j + upshift) < width)) {
          energy += lattice[i][j]*lattice[(i+1)%height][(j + upshift)];
        }
      }else if (periodicEnds) {
```

```
        if((i+1) < height) {
          energy += lattice[i][j]*lattice[(i+1)][j];
          energy += lattice[i][j]*lattice[(i+1)][(j + upshift + width)%width];
        }
        energy += lattice[i][j]*lattice[i][(j+1)%width];
      }else {
        if((i+1) < height) {
          energy += lattice[i][j]*lattice[(i+1)][j];
          if (((j + upshift) >= 0) && ((j + upshift) < width)) {
            energy += lattice[i][j]*lattice[(i+1)][(j + upshift)];
          }
        }
        if ((j+1) < width) {
          energy += lattice[i][j]*lattice[i][(j+1)%width];
        }
          }
    }
    }
    return energy;
}

/**
    Calculates the terms in the hamiltonian corresponding
    to the transverse operator. The terms are outputted
    as a sparce matrix in the format "i j H_ij" where
    i and j indicate the row and column of the matrix,
    respectively and H_ij is the ij component of the
    matrix.
    */
void triangleIsingTransverse(int lattice[][MAXSIZE], int height, int width, double
    transverse){
    int i,j;
    int configuration = hash(lattice, height, width);
    for(i=0;i<height;i++) {
        for(j=0;j<width;j++) {
            lattice[i][j] *= -1;
            printf("%d %d %lf\n", configuration, hash(lattice, height, width), transverse)
                ;
            lattice[i][j] *= -1;
        }
    }
}

/**
  Calculates the next configuration based on a
  binary number increment method.
  */
void increment(int lattice[][MAXSIZE], int height, int width){
  int done = FALSE;
  int i = 0;
  int j = 0;
  while(!done) {
      if(lattice[i][j] == -1) {
      lattice[i][j] = -1*lattice[i][j];
      if(j == (width-1)) {
        j = 0;
        i = i + 1;
      }else {
        j = j + 1;
      }
    }else {
```

```
149         lattice[i][j] = -1*lattice[i][j];
            done = TRUE;
151         }
        }
153 }

155 /**
      Determines the binary number corresponding to
157   the configuration. This is the configuration
      number and is used for the ordering of basis
159   states.
      */
161 int hash(int lattice[][MAXSIZE], int height, int width){
      int hash = 0;
163   int i,j;
      for(i = 0; i < height; i++) {
165     for(j = 0;j < width;j++) {
          hash += pow(2,(i*width + j))*(1 - lattice[i][j])/2;
167     }
      }
169   return hash;
    }
```

Code/exactDiagonalisation.c

```
1  #!/usr/bin/perl -w
   # Script for automating the exact diagonalisation algorithm
3  #
   #
5  # By Dylan Griffith
   #
7

9  @widths = (8);
   system "gcc exactDiagonalisation.c -lm -Wall -Werror -O -o triangleIsingGeneral";
11 initialise_strings();
   open F, ">$outfile" or die;
13 print F "J,h,height,width,E_ground\n";
   close F;
15 $height = 3;
   $periodic_top = 1;
17 $periodic_ends = 1;
   @Js = (1);
19 @hs = (0.4,0.8,1.2,1.4,1.8,2.2,2.4,2.8,3.2,3.4,3.8);
   foreach $J (@Js) {
21   foreach $h (@hs) {
       foreach $width (@widths) {
23       open F, ">>$outfile" or die;
         print F "$J,$h,";
25       close F;
         print "Calculating hamiltonian for width=$width...\n";
27       $time = time();
```

```
        system "./triangleIsingGeneral $height $width $J $h $periodic_ends $periodic_top >
            hamiltonian_$height\_$width.dat";
        $time = time() - $time;
        print "Finished in $time seconds.\n";
        print "Solving hamiltonian for width=$width...\n";
        $time = time();
        open F, ">temp.m" or die;
        printf F $matlab_solve, $height, $width, $height, $width, $height, $width, $int_id,
            $int_id, $float_id;
        close F;
        system "matlab -nodesktop -nosplash -r temp >matlab_error.log 2>&1";
        $time = time() - $time;
        print "Finished in $time seconds.\n\n";
    }
  }
}

# Email the results
system "uuencode $outfile $outfile | mail -s '$outfile' dyl.griffith\@gmail.com";

sub initialise_strings {
$time = localtime();
$time =~ s/ //g;
our $outfile = "groundStateEnergies_$time.csv";
our $int_id = '%d';
our $float_id = '%f';
our $matlab_solve = <<EOF;
H = readsparsegeneral('hamiltonian_%d_%d.dat',%d,%d);
height = %d;
width = %d;
energy = eigs(H, 1, 'sa');
file = fopen('$outfile', 'a');
fprintf(file, '%s,%s,%s\\n', height, width, energy);
fclose(file);
exit;
EOF
}
```

Code/exact_diagonalisation.pl

```
function Matrix = readsparsegeneral(filename, height, width)
% Function to read in a matrix from a file that is in the sparse format
%
%
% By Dylan Griffith
%


file = dlmread(filename, ' ');
dimension = 2^(height*width);

index_row = file(:,1) + 1;
index_column = file(:,2) + 1;
```

```
value = file(:,3);
Matrix = sparse(index_row, index_column, value, dimension, dimension);
```

Code/readsparsegeneral.m

```
/**
    Usage: ./exactDiagonalisationXXZ n J h outEven outOdd
        n - is the side length of the square array of spins
            (ie. total spins = n^2)
        J - is the exchange term in the hamiltonian
        h - is the transverse term in the hamiltonian
        outEven - is the file to output terms for the
                  matrix of even parity configurations
        outOdd - is the file to output terms for the
                  matrix of odd parity configurations

        Note: that parity is defined by the sum over all spins
        (with spin up counted as zero and spin down counted as
        one) modulo 2.
*/
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<assert.h>
#define FALSE 0
#define TRUE 1
#define MAXSIZE 1000
#define EVEN 0
#define ODD 1


int hash(int lattice[][MAXSIZE], int n);
void increment(int lattice[][MAXSIZE], int n);
int parity(int lattice[][MAXSIZE], int n);
int sum(int lattice[][MAXSIZE], int n);

void hamiltonianDiagonal(int lattice[][MAXSIZE], int n);
void hamiltonianOffDiagonal(int lattice[][MAXSIZE], int n);

/**
    configurationNumber will point to an array
    that maps the hash value of a lattice
    configuration to its new identification
    number for the even and odd states. The
    counter variables are to keep count of
    the odd and even states discovered.
*/
int *configurationNumber;
int configurationCounterEven;
int configurationCounterOdd;
int J;
int h;
FILE *fEven, *fOdd;
```

```
49
51 int main(int argc, char *argv[]) {
       int lattice[MAXSIZE][MAXSIZE];
53
       int n = atoi(argv[1]);
55     J = atoi(argv[2]);
       h = atoi(argv[3]);
57
59     // Open necessary output files
       char *outEven = argv[4]; // file to output even matrix to
61     char *outOdd = argv[5]; // file to output odd matrix to
       fEven = fopen(outEven, "w");
63     fOdd = fopen(outOdd, "w");
       assert(fEven != NULL);
65     assert(fOdd != NULL);

67     hamiltonianDiagonal(lattice, n);
       hamiltonianOffDiagonal(lattice, n);
69
       fclose(fEven);
71     fclose(fOdd);

73     return 0;
   }
75
   /**
77     Calculates all the diagonal elements of the hamiltonian and
       outputs them to the corresponding files.
79     The following occurs too: Initialise spins to all up state.
       Also loops through every possible configuration and populates
81     the configurationNumber hash table.
   */
83 void hamiltonianDiagonal(int lattice[][MAXSIZE], int n){
       int i,j;
85     int permutations = (int) pow((double) 2,(double) n*n);

87
       // Initialise lattice to all spins up
89     for(i=0;i<n;i++) {
           for(j=0;j<n;j++) {
91             lattice[i][j] = 1;
           }
93     }

95     // Create enough memory for configurationNumber hashes
       configurationNumber = malloc(permutations*sizeof(int));
97     assert(configurationNumber != NULL);
       configurationNumber[0] = 0;
99     configurationCounterEven = 0;
       configurationCounterOdd = 0;
101
       // Loop over all permutations and output diagonal elements of hamiltonian
103    fprintf(fEven, "%d %d %d\n", configurationCounterEven, configurationCounterEven, h*sum
           (lattice, n));
       configurationCounterEven++;
105    for(i=1;i<permutations;i++){
           increment(lattice, n);
107        if(parity(lattice, n) == EVEN) {
               configurationNumber[hash(lattice,n)] = configurationCounterEven;
```

```
109             fprintf(fEven, "%d %d %d\n", configurationCounterEven,
                    configurationCounterEven, h*sum(lattice, n));
                configurationCounterEven++;
111         }else {
                configurationNumber[hash(lattice,n)] = configurationCounterOdd;
113             fprintf(fOdd, "%d %d %d\n", configurationCounterOdd, configurationCounterOdd,
                    h*sum(lattice, n));
                configurationCounterOdd++;
115         }
    }
117
    // Re-initialise lattice to all spins up
119 for(i=0;i<n;i++) {
        for(j=0;j<n;j++) {
121         lattice[i][j] = 1;
        }
123 }
}
125
/**
127     Returns the sum over all spins where spin up is defined as +1 and spin
    down defined as -1.
129 */
int sum(int lattice[][MAXSIZE], int n) {
131 int sum = 0;
    int i,j;
133 for(i=0;i<n;i++) {
        for(j=0;j<n;j++) {
135         sum += lattice[i][j];
        }
137 }
    return sum;
139 }

141 /**
    Takes a lattice initialised to all spin up then generates
143 all off diagonal terms of the hamiltonian by applying
    S_i^x S_j^x operator to all spin pairs (i,j) in all possible
145 lattice configurations. These terms are output to
    the file titled outEven or outOdd if they correspond to
147 configurations with even or odd parity, respectively.
*/
149 void hamiltonianOffDiagonal(int lattice[][MAXSIZE], int n){
    int p,i,j;
151 int permutations = (int) pow((double) 2,(double) n*n);


153
    for(p = 0;p < permutations;p++) {
155     for(i=0;i<n;i++) {
            for(j=0;j<n;j++) {
157             /**energy += exchange*lattice[i][j]*lattice[(i+1)%n][j];
                energy += exchange*lattice[i][j]*lattice[(i+1)%n][(j+1)%n];
159             energy += exchange*lattice[i][j]*lattice[i][(j+1)% n];
                */
161             lattice[i][j] *= -1;
                lattice[(i+1)%n][j] *= -1;
163             if(parity(lattice, n) == EVEN) {
                    fprintf(fEven, "%d %d %d\n", configurationNumber[p],
                        configurationNumber[hash(lattice, n)], J);
165             }else {
```

```
                    fprintf(fOdd, "%d %d %d\n", configurationNumber[p],
                        configurationNumber[hash(lattice, n)], J);
                }
                lattice[(i+1)%n][j] *= -1;

                lattice[(i+1)%n][(j+1)%n] *= -1;
                if(parity(lattice, n) == EVEN) {
                    fprintf(fEven, "%d %d %d\n", configurationNumber[p],
                        configurationNumber[hash(lattice, n)], J);
                }else {
                    fprintf(fOdd, "%d %d %d\n", configurationNumber[p],
                        configurationNumber[hash(lattice, n)], J);
                }
                lattice[(i+1)%n][(j+1)%n] *= -1;

                lattice[i][(j+1)%n] *= -1;
                if(parity(lattice, n) == EVEN) {
                    fprintf(fEven, "%d %d %d\n", configurationNumber[p],
                        configurationNumber[hash(lattice, n)], J);
                }else {
                    fprintf(fOdd, "%d %d %d\n", configurationNumber[p],
                        configurationNumber[hash(lattice, n)], J);
                }
                lattice[i][(j+1)%n] *= -1;

                lattice[i][j] *= -1;
            }
        }
        if(p < permutations - 1)
            increment(lattice, n);
    }
}


/**
    Permutates lattice configuration to next lattice configuration
    based on binary enumeration of spins.
*/
void increment(int lattice[][MAXSIZE], int n) {
  int done = FALSE;
  int i = 0;
  int j = 0;
  while(!done) {
      if(lattice[i][j] == -1) {
    lattice[i][j] = -1*lattice[i][j];
    if(j == (n-1)) {
        j = 0;
        i = i + 1;
    }else {
        j = j + 1;
    }
  }else {
      lattice[i][j] = -1*lattice[i][j];
      done = TRUE;
      }
  }
}

/**
    Returns the parity of a lattice configuration as defined
    by the sum over all spins (with spin up counted as zero
```

```
    and spin down counted as one) modulo 2.
*/
int parity(int lattice[][MAXSIZE], int n) {
    int parity = EVEN;
    int i,j;
    for(i = 0; i < n; i++) {
        for(j=0;j<n;j++) {
            parity += (1 - lattice[i][j])/2;
        }
    }
    return parity % 2;
}

int hash(int lattice[][MAXSIZE], int n) {
  int hash = 0;
  int i,j;
  for(i = 0; i < n; i++) {
    for(j = 0;j < n;j++) {
      hash += pow(2,(i*n + j))*(1 - lattice[i][j])/2;
    }
  }
  return hash;
}
```

Code/exactDiagonalisationXXZ.c

B.2 iTEBD

The following gives the code for the iTEBD algorithm including the functions used to generate the Hamiltonians and Unitary operators used in the algorithm. Some functions are also given that were used for calculating all the spin expectation values. Finally the functions used for tensor operations are given.

```
function [EnergyFinal,GamA,GamB,LamA,LamB] = itebd(H0,Chi,TIMEMAX,NumStepSizes)
% Function performs the iTEBD algorithm for the system defined by the Hamiltonian
% H0. TIMEMAX is the number of iterations at each time step size dt. NumStepSizes
% is the number of different dt values are used (Note: it is reduced by a factor of
% ten after every TIMEMAX time steps). Chi is self-explanatory if you read the way
% the MPS is described.
%
%
% By Dylan Griffith
%


%% Define some parameters
format long;
E = zeros(NumStepSizes,TIMEMAX+1);
Norm = zeros(NumStepSizes,TIMEMAX+1);
%% Get hamiltonian
%% Determine the dimensions of the hilbert space
Hilbert = sqrt(size(H0,1));
```

```
%% Generate an initial ansatz for the wavefunction
LamA = eye(Chi,Chi);
LamB = eye(Chi,Chi);
GamA = zeros(Chi, Hilbert, Chi);
GamB = zeros(Chi, Hilbert, Chi);
GamA(:,:,:)=1/(sqrt(Hilbert)*Chi);
GamB(:,:,:)=1/(sqrt(Hilbert)*Chi);
%% Time step size
dt = 0.1;
%% Loop over different time step sizes
for I=1:NumStepSizes
    %% Get the Unitary (for 1D Ising on two spins) and decrease step size
    U = unitary(H0, dt);
    dt = dt/10;
    %% Re-express Hamiltonian and Unitary as tensors
    U = reshape(U, [Hilbert Hilbert Hilbert Hilbert]);
    H = reshape(H0, [Hilbert Hilbert Hilbert Hilbert]);

    %%%%%%%%%%%%%%%%%%%%%%%%%%%% OBSERVABLES %%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %% Contract tensors of wavefunction
    Psi = contract(GamA, LamA, 3, 1);
    Psi = contract(Psi, GamB, 3, 1);
    Psi = contract(LamB, Psi, 2, 1);
    Psi = contract(Psi, LamB, 4, 1);

    %% Calculate Energy and add to array
    HPsi = contract(Psi, H, [2 3], [1 2]);
    HPsi = permute(HPsi, [1 3 4 2]);
    E(I,1) = contract(HPsi, Psi, [1 2 3 4], [1 2 3 4]);

    %% Calculate Norm 1 and add to arrays
    Theta = contract(GamA, LamA, 3, 1);
    Theta = contract(Theta, GamB, 3, 1);
    Theta = contract(LamB, Theta, 2, 1);
    Theta = contract(Theta, LamB, 4, 1);
    Norm(I,1) = contract(Theta, Theta, [1 2 3 4], [1 2 3 4]);

    for k=2:(TIMEMAX+1)

        %%%%%%%%%%%%%%%%%%%%%%%%%%%% FIRST UNITARY %%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %% Contract the wave function to rank 4 tensor
        Theta = contract(GamA, LamA, 3, 1);
        Theta = contract(Theta, GamB, 3, 1);
        Theta = contract(LamB, Theta, 2, 1);
        Theta = contract(Theta, LamB, 4, 1);
        %% Contract with the time evolution unitary on GammaB GammaA
        Theta = contract(Theta, U, [2 3], [1 2]);
        Theta = permute(Theta, [1 3 4 2]);
        Theta = Theta/sqrt(contract(Theta, Theta, [1 2 3 4], [1 2 3 4])); % Normalise
            state

        %% Use singular value decomposition to recover LamA
        Theta = reshape(Theta, [Chi*Hilbert Hilbert*Chi]); % Reshape to get into form of a
             matrix
        [X , LamA, Y] = svd(Theta);
        Y = Y'; % SVD returns transpose of right matrix
        LamA = LamA(1:Chi,1:Chi); % Truncate to largest Chi schmidt weights (Chi by Chi
            matrix)
        X = X(:,1:Chi); % Truncate to Chi*Hilbert by Chi
        Y = Y(1:Chi,:); % Truncate to Chi by Chi*Hilbert
```

```
78          X = reshape(X, [Chi Hilbert Chi]); % Restore rank 3 tensor form
            Y = reshape(Y, [Chi Hilbert Chi]); % Restore rank 3 tensor form
80
            %% Contract X and Y with LamB^-1 to recover GamA and GamB
82          GamA = contract(invertsingular(LamB), X, 2, 1);
            GamB = contract(Y, invertsingular(LamB), 3, 1);
84
            %%%%%%%%%%%%%%%%%%%%%%%%%%%%%% SECOND UNITARY %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
86
            %% Contract the wave function to rank 4 tensor
88          Theta = contract(GamB, LamB, 3, 1);
            Theta = contract(Theta, GamA, 3, 1);
90          Theta = contract(LamA, Theta, 2, 1);
            Theta = contract(Theta, LamA, 4, 1);
92          %% Contract with the time evolution unitary on GammaB GammaA
            Theta = contract(Theta, U, [2 3], [1 2]);
94          Theta = permute(Theta, [1 3 4 2]);
            Theta = Theta/sqrt(contract(Theta, Theta, [1 2 3 4], [1 2 3 4])); % Normalise
                state
96
            %% Use singular value decomposition to recover LamB
98          Theta = reshape(Theta, [Chi*Hilbert Hilbert*Chi]); % Reshape to get into form of a
                matrix
            [X , LamB, Y] = svd(Theta);
100         Y = Y'; % SVD returns transpose of right matrix
            LamB = LamB(1:Chi,1:Chi); % Truncate to largest Chi schmidt weights (Chi by Chi
                matrix)
102         X = X(:,1:Chi); % Truncate to Chi*Hilbert by Chi
            Y = Y(1:Chi,:); % Truncate to Chi by Chi*Hilbert
104         X = reshape(X, [Chi Hilbert Chi]); % Restore rank 3 tensor form
            Y = reshape(Y, [Chi Hilbert Chi]); % Restore rank 3 tensor form
106
            %% Contract X and Y with LamA^-1 to recover GamA and GamB
108         GamB = contract(invertsingular(LamA), X, 2, 1);
            GamA = contract(Y, invertsingular(LamA), 3, 1);
110
            %% Eliminate very low schmidt weights
112         LamB(LamB < 10e-13) = 0;
114         %%%%%%%%%%%%%%%%%%%%%%%%%%%%%% OBSERVABLES %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
116         %% Contract tensors of wavefunction
            Psi = contract(GamA, LamA, 3, 1);
118         Psi = contract(Psi, GamB, 3, 1);
            Psi = contract(LamB, Psi, 2, 1);
120         Psi = contract(Psi, LamB, 4, 1);
122         %% Calculate Energy and add to array
            HPsi = contract(Psi, H, [2 3], [1 2]);
124         HPsi = permute(HPsi, [1 3 4 2]);
            E(I,k) = contract(HPsi, Psi, [1 2 3 4], [1 2 3 4]);
126
            %% Calculate Norm 1 and add to arrays
128         Theta = contract(GamA, LamA, 3, 1);
            Theta = contract(Theta, GamB, 3, 1);
130         Theta = contract(LamB, Theta, 2, 1);
            Theta = contract(Theta, LamB, 4, 1);
132         Norm(I,k) = contract(Theta, Theta, [1 2 3 4], [1 2 3 4]);
134         %% Eliminate very low schmidt weights
            LamB(LamB < 10e-13) = 0;
```

```
136
    end
138     EnergyFinal = E(I,TIMEMAX+1);
end
```

Code/itebd.m

```
function out = invertsingular(in)
% Performs a pseudo-inverse on the diagonal matrix in.
% All zero entries remain zero entries in the pseudo-inverse.
%
%
% By Dylan Griffith
%


for c=1:max(size(in))
    if(in(c,c) == 0)
        continue;
    else
        in(c,c) = 1/in(c,c);
    end
end
out = in;
```

Code/invertsingular.m

```
function E0 = infinite1dground(J,h)
% Calculates the ground state energy for the infinite one-dimensional Ising
% model. The solution is based on Pfeuty 1970
%
%
% By Dylan Griffith
%

% External field strength as a ratio to J
B = abs(h/J);
% Convert to parameters used in Pfeuty
Gam = 2*B;
lam=2/Gam;
% Integral parts
dp = pi*1e-6;
p = 0:dp:pi/2;
```

```
th = sqrt(4*lam/(1+lam)^2);
f = sqrt(1-th^2*sin(p).^2);
% Numerically integrate
Integral = sum(f)*dp;
% Calculate ground state energy
E0 = -(Gam*2/pi)*(1+lam)*(Integral/2);
% Rescale for the parameters I used
E0 = J*E0;
```

Code/infinite1dground.m

```
function H = hamiltoniandoubleladder(J1,J2,J3,h)
% Hamiltonian for a double ladder triangle system
%
%
% By Dylan Griffith
%

%% Pauli spin matrices
X=[0 1;1 0];
Z=[1 0;0 -1];
I = eye(2);
II = kron(I,I);
%% Inside interaction (Blue)
H1 = J1*kron(X,X);
H1 = kron(H1,II);
%% Across interaction (Orange)
H2 = kron(X,II);
H2 = J2*kron(H2,X);
%% Adjacent interaction (Green)
H31 = kron(X,I);
H31 = kron(H31,X);
H31 = kron(H31,I);
H32 = kron(I,X);
H32 = kron(H32,I);
H32 = kron(H32,X);
H3 = J3*(H31 + H32);
%% Transverse field (Red)
H41 = kron(Z,I);
H41 = kron(H41,II);
H42 = kron(I,Z);
H42 = kron(H42,II);
H4 = h*(H41 + H42);
%% Total hamiltonian
H= H1 + H2 + H3 + H4;
```

Code/hamiltoniandoubleladder.m

```
function H = hamiltoniantripleladder(J1,J2,J3,J4,J5,J6,J7,h)
% Hamiltonian for a triple ladder triangular lattice system with periodic
% exchange conditions given by J6 and J7. Note that the convention used
% in the report had only J1,J2,J3 because the number of parameters was
% reduced for clarity. The J1,J4,J7 are effectively J1 in the report,
% J2,J5,J6 are J2 and J3 is J3.
%
%
% By Dylan Griffith
%

H = zeros(2^6);
%% One site operators
X=[0 1;1 0];
Z=[1 0;0 -1];
I = eye(2);
%% Two site operators
XX = kron(X,X);
XI = kron(X,I);
IX = kron(I,X);
ZI = kron(Z,I);
IZ = kron(I,Z);
II = kron(I,I);
%% Three site operators
XIX = kron(XI,X);
IXX = kron(I,XX);
XXI = kron(XX,I);
XII = kron(XI,I);
IXI = kron(IX,I);
IIX = kron(II,X);
ZII = kron(ZI,I);
IZI = kron(IZ,I);
IIZ = kron(II,Z);
III = kron(II,I);
%% Exchange terms
H = H + J1*kron(XXI,III); % Blue
H = H + J2*kron(XII,IXI); % Yellow
H = H + J3*(kron(XII,XII) + kron(IXI,IXI) + kron(IIX,IIX)); % Green
H = H + J4*kron(IIX,IXI); % Purple
H = H + J5*kron(IXX,III); % Pink
H = H + J6*kron(XIX,III); % Gray
H = H + J7*kron(XII,IIX); % Reddish Brown
% Transverse field terms
H = H + h*(kron(ZII,III) + kron(IZI,III) + kron(IIZ,III)); % Red
```

Code/hamiltoniantripleladder.m

```
function [Z1, Z2, Z1Z2, X1, X2, X1X2] = spins(GamA,GamB,LamA,LamB)
% function calculates the spin observables for the 1D MPS defined by the
% function parameters
%
```

```
%
% By Dylan Griffith
%


Hilbert = 2;
%% Get sigma operators (pauli matrices) and Re-express as tensors
X=[0 1;1 0];
Z=[1 0;0 -1];
SigX1 = kron(X,eye(2));
SigX2 = kron(eye(2),X);
SigX1X2 = kron(X,X);
SigZ1 = kron(Z,eye(2));
SigZ2 = kron(eye(2),Z);
SigZ1Z2 = kron(Z,Z);
SigX1 = reshape(SigX1, [Hilbert Hilbert Hilbert Hilbert]);
SigX2 = reshape(SigX2, [Hilbert Hilbert Hilbert Hilbert]);
SigX1X2 = reshape(SigX1X2, [Hilbert Hilbert Hilbert Hilbert]);
SigZ1 = reshape(SigZ1, [Hilbert Hilbert Hilbert Hilbert]);
SigZ2 = reshape(SigZ2, [Hilbert Hilbert Hilbert Hilbert]);
SigZ1Z2 = reshape(SigZ1Z2, [Hilbert Hilbert Hilbert Hilbert]);
%% Contract together the MPS
Psi = contract(GamA, LamA, 3, 1);
Psi = contract(Psi, GamB, 3, 1);
Psi = contract(LamB, Psi, 2, 1);
Psi = contract(Psi, LamB, 4, 1);
%% Calculate the observables
% Calculate Z1
Temp = contract(Psi, SigZ1, [2 3], [1 2]);
Temp = permute(Temp, [1 3 4 2]);
Z1 = contract(Temp, Psi, [1 2 3 4], [1 2 3 4]);
% Calculate Z2
Temp = contract(Psi, SigZ2, [2 3], [1 2]);
Temp = permute(Temp, [1 3 4 2]);
Z2 = contract(Temp, Psi, [1 2 3 4], [1 2 3 4]);
% Calculate Z1Z2
Temp = contract(Psi, SigZ1Z2, [2 3], [1 2]);
Temp = permute(Temp, [1 3 4 2]);
Z1Z2 = contract(Temp, Psi, [1 2 3 4], [1 2 3 4]);
% Calculate X1
Temp = contract(Psi, SigX1, [2 3], [1 2]);
Temp = permute(Temp, [1 3 4 2]);
X1 = contract(Temp, Psi, [1 2 3 4], [1 2 3 4]);
% Calculate X2
Temp = contract(Psi, SigX2, [2 3], [1 2]);
Temp = permute(Temp, [1 3 4 2]);
X2 = contract(Temp, Psi, [1 2 3 4], [1 2 3 4]);
% Calculate X1X2
Temp = contract(Psi, SigX1X2, [2 3], [1 2]);
Temp = permute(Temp, [1 3 4 2]);
X1X2 = contract(Temp, Psi, [1 2 3 4], [1 2 3 4]);
```

Code/spins.m

```
function ExpectationValues = spinsdoubleladder(GamA,GamB,LamA,LamB)
% This function calculates the expectation values for the spin matrices on
% the given MPS. The order is as follows:
% [<XIII>,<IXII>,<IIXI>,<IIIX>,<XXII>,<IXXI>,<IIXX>,<XIXI>,<IXIX>,<XIIX>,
% <ZIII>,<IZII>,<IIZI>,<IIIZ>,<ZZII>,<IZZI>,<IIZZ>,<ZIZI>,<IZIZ>,<ZIIZ>]
%
%
% By Dylan Griffith
%


%% Number of expectation values to calculate
NumResults = 20;
Hilbert = size(GamA,2);
%% Spin half operators
X=[0 1;1 0];
Z=[1 0;0 -1];
I = eye(2);
%% Two site operators
XI = kron(X,I);
IX = kron(I,X);
XX = kron(X,X);
ZI = kron(Z,I);
IZ = kron(I,Z);
ZZ = kron(Z,Z);
II = kron(I,I);
%% Generate operators
Ops = cell(1,NumResults);
% X Operators
Ops{1} = kron(XI,II);
Ops{2} = kron(IX,II);
Ops{3} = kron(II,XI);
Ops{4} = kron(II,IX);
Ops{5} = kron(XX,II);
Ops{6} = kron(IX,XI);
Ops{7} = kron(II,XX);
Ops{8} = kron(XI,XI);
Ops{9} = kron(IX,IX);
Ops{10} = kron(XI,IX);
% Z Operators
Ops{11} = kron(ZI,II);
Ops{12} = kron(IZ,II);
Ops{13} = kron(II,ZI);
Ops{14} = kron(II,IZ);
Ops{15} = kron(ZZ,II);
Ops{16} = kron(IZ,ZI);
Ops{17} = kron(II,ZZ);
Ops{18} = kron(ZI,ZI);
Ops{19} = kron(IZ,IZ);
Ops{20} = kron(ZI,IZ);
%% Contract together the MPS
Psi = contract(GamA, LamA, 3, 1);
Psi = contract(Psi, GamB, 3, 1);
Psi = contract(LamB, Psi, 2, 1);
Psi = contract(Psi, LamB, 4, 1);
%% Calculate expectation values
ExpectationValues = zeros(1,NumResults);
for C=1:NumResults
    Op = reshape(Ops{C},[Hilbert Hilbert Hilbert Hilbert]);
    Temp = contract(Psi, Op, [2 3], [1 2]);
```

```
    Temp = permute(Temp, [1 3 4 2]);
    ExpectationValues(C) = contract(Temp, Psi, [1 2 3 4], [1 2 3 4]);
end
```

Code/spinsdoubleladder.m

```
function ExpectationValues = spinstripleladder(GamA,GamB,LamA,LamB)
% This function calculates the expectation values for the spin matrices on
% the given MPS. The order is as follows:
% [<XIIIII>,<IXIIII>,<IIXIII>,<IIIXII>,<IIIIXI>,<IIIIIX>,<XXIIII>,<IXXIII>,
% <IIXXII>,<IIIXXI>,<IIIIXX>,<XIXIII>,<IXIXII>,<IIXIXI>,<IIIXIX>,<XIIXII>,
% <IXIIXI>,<IIXIIX>,<XIIIXI>,<IXIIIX>,<XIIIIX>,
% <ZIIIII>,<IZIIII>,<IIZIII>,<IIIZII>,<IIIIZI>,<IIIIIZ>,<ZZIIII>,<IZZIII>,
% <IIZZII>,<IIIZZI>,<IIIIZZ>,<ZIZIII>,<IZIZII>,<IIZIZI>,<IIIZIZ>,<ZIIZII>,
% <IZIIZI>,<IIZIIZ>,<ZIIIZI>,<IZIIIZ>,<ZIIIIZ>]
%
%
% By Dylan Griffith
%


%% Number of expectation values to calculate
NumResults = 42;
Hilbert = size(GamA,2);
%% Spin half operators
X=[0 1;1 0];
Z=[1 0;0 -1];
I = eye(2);
%% Two site operators
XI = kron(X,I);
XX = kron(X,X);
IX = kron(I,X);
ZI = kron(Z,I);
IZ = kron(I,Z);
ZZ = kron(Z,Z);
II = kron(I,I);
%% Three site operators
XII = kron(XI,I);
IXI = kron(IX,I);
IIX = kron(II,X);
XIX = kron(XI,X);
IXX = kron(I,XX);
XXI = kron(XX,I);
ZII = kron(ZI,I);
IZI = kron(IZ,I);
IIZ = kron(II,Z);
ZIZ = kron(ZI,Z);
IZZ = kron(I,ZZ);
ZZI = kron(ZZ,I);
III = kron(II,I);
%% Generate operators
Ops = cell(1,NumResults);
% X Operators
```

```
Ops{1} = kron(XII,III);
Ops{2} = kron(IXI,III);
Ops{3} = kron(IIX,III);
Ops{4} = kron(III,XII);
Ops{5} = kron(III,IXI);
Ops{6} = kron(III,IIX);
Ops{7} = kron(XXI,III);
Ops{8} = kron(IXX,III);
Ops{9} = kron(IIX,XII);
Ops{10} = kron(III,XXI);
Ops{11} = kron(III,IXX);
Ops{12} = kron(XIX,III);
Ops{13} = kron(IXI,XII);
Ops{14} = kron(IIX,IXI);
Ops{15} = kron(III,XIX);
Ops{16} = kron(XII,XII);
Ops{17} = kron(IXI,IXI);
Ops{18} = kron(IIX,IIX);
Ops{19} = kron(XII,IXI);
Ops{20} = kron(IXI,IIX);
Ops{21} = kron(XII,IIX);
% Z Operators
Ops{22} = kron(ZII,III);
Ops{23} = kron(IZI,III);
Ops{24} = kron(IIZ,III);
Ops{25} = kron(III,ZII);
Ops{26} = kron(III,IZI);
Ops{27} = kron(III,IIZ);
Ops{28} = kron(ZZI,III);
Ops{29} = kron(IZZ,III);
Ops{30} = kron(IIZ,ZII);
Ops{31} = kron(III,ZZI);
Ops{32} = kron(III,IZZ);
Ops{33} = kron(ZIZ,III);
Ops{34} = kron(IZI,ZII);
Ops{35} = kron(IIZ,IZI);
Ops{36} = kron(III,ZIZ);
Ops{37} = kron(ZII,ZII);
Ops{38} = kron(IZI,IZI);
Ops{39} = kron(IIZ,IIZ);
Ops{40} = kron(ZII,IZI);
Ops{41} = kron(IZI,IIZ);
Ops{42} = kron(ZII,IIZ);
%% Contract together the MPS
Psi = contract(GamA, LamA, 3, 1);
Psi = contract(Psi, GamB, 3, 1);
Psi = contract(LamB, Psi, 2, 1);
Psi = contract(Psi, LamB, 4, 1);
%% Calculate expectation values
ExpectationValues = zeros(1,NumResults);
for C=1:NumResults
    Op = reshape(Ops{C},[Hilbert Hilbert Hilbert Hilbert]);
    Temp = contract(Psi, Op, [2 3], [1 2]);
    Temp = permute(Temp, [1 3 4 2]);
    ExpectationValues(C) = contract(Temp, Psi, [1 2 3 4], [1 2 3 4]);
end
```

Code/spinstripleladder.m

```
function rank = trank(A)
% Returns the rank of the tensor A. The rank of a tensor is the number of
% indices (or free parameters).
% eg.
%       A = ones(2,2,2,2);
%       trank(A)
%       ans = 4
%
%       B = ones(2,3,2,5,6,2);
%       trank(B)
%       ans = 6
%
%
% By Dylan Griffith
%


rank = numel(size(A));
```

Code/trank.m

```
function C = contract(A,B, IndA, IndB)
% This function is used to contract the tensors A and B along the indices
% IndA and IndB, respectively. This function returns the result of the
% contraction, C.
% eg.
%           C = contract(A, B, [1 2], [2 3]);
%           C = contract(A, B, 1, 1);
%
%
% By Dylan Griffith
%


%% Determine indices for contracting and not contracting in each tensor.
IndsAllA = 1:trank(A);
IndsAllB = 1:trank(B);

IndsContA = IndA;
IndsKeepA = setxor(IndsAllA, IndsContA);

IndsContB = IndB;
IndsKeepB = setxor(IndsAllB, IndsContB);

%% Determine the dimension of indices in each tensor and hence the total
% dimension to be contracted over and the total dimension to keep.
DimsAllA = size(A);
DimsKeepA = DimsAllA(IndsKeepA);
DimsContA = DimsAllA(IndsContA);
TotalDimKeepA = prod(DimsKeepA);
TotalDimContA = prod(DimsContA);
```

```
DimsAllB = size(B);
DimsKeepB = DimsAllB(IndsKeepB);
DimsContB = DimsAllB(IndsContB);
TotalDimKeepB = prod(DimsKeepB);
TotalDimContB = prod(DimsContB);

%% Permute the tensors such that the indices to be contracted over are at
%% the end for A and the beginning for B.
A = permute(A, [IndsKeepA IndsContA]);
B = permute(B, [IndsContB IndsKeepB]);

%% Reshape A and B to matrices and get the product A*B.
A = reshape(A, [TotalDimKeepA TotalDimContA]);
B = reshape(B, [TotalDimContB TotalDimKeepB]);
C = A*B;

%% Reshape C back into tensor of correct rank
if (~(isempty(DimsKeepA) && isempty(DimsKeepB)))
    C = reshape(C, [DimsKeepA DimsKeepB]);
end
```

Code/contract.m

Bibliography

[1] A. Micheli, G. Brennen, and P. Zoller, "A toolbox for lattice-spin models with polar molecules," *Nature Physics*, vol. 2, no. 5, pp. 341–347, 2006.

[2] J. W. Britton, B. C. Sawyer, A. C. Keith, C. C. J. Wang, J. K. Freericks, H. Uys, M. J. Biercuk, and J. J. Bollinger, "Engineered two-dimensional Ising interactions in a trapped-ion quantum simulator with hundreds of spins," *NATURE*, vol. 484, pp. 489–492, APR 26 2012.

[3] D. Poilblanc, N. Schuch, D. Pérez-García, and J. Cirac, "Topological and entanglement properties of resonating valence bond wave functions," *Physical Review B*, vol. 86, no. 1, p. 014404, 2012.

[4] S. Sachdev, *Quantum Phase Transitions*. Cambridge University Press, 2011.

[5] G. Vidal, "Efficient classical simulation of slightly entangled quantum computations," *Phys. Rev. Lett.*, vol. 91, p. 147902, Oct 2003.

[6] G. Vidal, "Classical simulation of infinite-size quantum lattice systems in one spatial dimension," *Phys. Rev. Lett.*, vol. 98, p. 070201, Feb 2007.

[7] J. Jordan, R. Orús, G. Vidal, F. Verstraete, and J. Cirac, "Classical simulation of infinite-size quantum lattice systems in two spatial dimensions," *Physical Review Letters*, vol. 101, no. 25, p. 250602, 2008.

[8] R. Orus and G. Vidal, "Infinite time-evolving block decimation algorithm beyond unitary evolution," *Physical Review B*, vol. 78, no. 15, p. 155117, 2008.

[9] U. Schollwock, "The density-matrix renormalization group in the age of matrix product states," *Annals of Physics*, vol. 326, no. 1, pp. 96 – 192, 2011.

[10] U. Schollwöck, "The density-matrix renormalization group," *Reviews of Modern Physics*, vol. 77, no. 1, p. 259, 2005.

[11] R. Chitra, S. Pati, H. Krishnamurthy, D. Sen, and S. Ramasesha, "Density-matrix renormalization-group studies of the spin-1/2 heisenberg system with dimerization and frustration," *Physical Review B*, vol. 52, no. 9, p. 6581, 1995.

[12] S. White, "Strongly correlated electron systems and the density matrix renormalization group," *Physics Reports*, vol. 301, no. 1, pp. 187–204, 1998.

[13] R. N. C. Pfeifer, G. Evenbly, and G. Vidal, "Entanglement renormalization, scale invariance, and quantum criticality," *Phys. Rev. A*, vol. 79, p. 040301, Apr 2009.

[14] L. Cincio, J. Dziarmaga, and M. Rams, "Multiscale entanglement renormalization ansatz in two dimensions: Quantum ising model," *Physical review letters*, vol. 100, no. 24, p. 240603, 2008.

[15] V. Giovannetti, S. Montangero, M. Rizzi, and R. Fazio, "Homogeneous multiscale-entanglement-renormalization-ansatz states: An information theoretical analysis," *Physical Review A*, vol. 79, no. 5, p. 052314, 2009.

[16] P. Pfeuty, "The one-dimensional ising model with a transverse field," *ANNALS of Physics*, vol. 57, no. 1, pp. 79–90, 1970.

[17] G. H. Wannier, "Antiferromagnetism. the triangular ising net," *Phys. Rev.*, vol. 79, pp. 357–364, Jul 1950.

[18] R. Moessner, S. L. Sondhi, and P. Chandra, "Two-dimensional periodic frustrated ising models in a transverse field," *Phys. Rev. Lett.*, vol. 84, pp. 4457–4460, May 2000.

[19] K. Kim, M. Chang, S. Korenblit, R. Islam, E. Edwards, J. Freericks, G. Lin, L. Duan, and C. Monroe, "Quantum simulation of frustrated ising spins with trapped ions," *Nature*, vol. 465, no. 7298, pp. 590–593, 2010.

[20] D. A. Lidar and O. Biham, "Simulating ising spin glasses on a quantum computer," *Phys. Rev. E*, vol. 56, pp. 3661–3681, Sep 1997.

[21] P. Stelmachovic and V. Buzek, "Quantum-information approach to the ising model: Entanglement in chains of qubits," *Phys. Rev. A*, vol. 70, p. 032313, Sep 2004.

[22] A. W. Sandvik, "Ground states of a frustrated quantum spin chain with long-range interactions," *Phys. Rev. Lett.*, vol. 104, p. 137204, Mar 2010.

[23] R. Moessner and S. Sondhi, "Ising models of quantum frustration," *Physical Review B*, vol. 63, no. 22, p. 224401, 2001.

[24] Y. Jiang and T. Emig, "Exotic phases in geometrically frustrated triangular ising magnets," *Journal of Physics: Condensed Matter*, vol. 19, no. 14, p. 145234, 2007.

[25] S. Isakov and R. Moessner, "Interplay of quantum and thermal fluctuations in a frustrated magnet," *Physical Review B*, vol. 68, no. 10, p. 104409, 2003.

[26] Y. Jiang and T. Emig, "Ordering of geometrically frustrated classical and quantum triangular ising magnets," *Physical Review B*, vol. 73, no. 10, p. 104452, 2006.

[27] A. Bermudez, J. Almeida, K. Ott, H. Kaufmann, S. Ulm, U. Poschinger, F. Schmidt-Kaler, A. Retzker, and M. Plenio, "Quantum magnetism of spin-ladder compounds with trapped-ion crystals," *New Journal of Physics*, vol. 14, no. 9, p. 093042, 2012.

[28] S. R. White, "Density matrix formulation for quantum renormalization groups," *Phys. Rev. Lett.*, vol. 69, pp. 2863–2866, Nov 1992.

[29] D. Perez-Garcia, F. Verstraete, M. M. Wolf, and J. I. Cirac, "Matrix Product State Representations," May 2007.

[30] S. K. Baek, J. Um, S. D. Yi, and B. J. Kim, "Quantum monte carlo study of the transverse-field quantum ising model on infinite-dimensional structures," *Phys. Rev. B*, vol. 84, p. 174419, Nov 2011.

[31] J. Cullum and R. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations: Theory*. Progress in Scientific Computing, Birkhäuser, 1985.

[32] J. Jordan, *Studies of Infinite Two-Dimensional Quantum Lattice Systems with Projected Entangled Pair States*. PhD thesis, 2003.

[33] A. Weiße and H. Fehske, "Exact diagonalization techniques," *Computational Many-Particle Physics*, pp. 529–544, 2008.

[34] G. Vidal, "Efficient simulation of one-dimensional quantum many-body systems," *Phys. Rev. Lett.*, vol. 93, p. 040502, Jul 2004.

[35] F. Verstraete and J. Cirac, "Matrix product states represent ground states faithfully," *Physical Review B*, vol. 73, no. 9, p. 094423, 2006.

[36] L. Tagliacozzo, T. de Oliveira, S. Iblisdir, and J. Latorre, "Scaling of entanglement support for matrix product states," *Physical Review B*, vol. 78, no. 2, p. 024410, 2008.

[37] F. Verstraete, V. Murg, and J. Cirac, "Matrix product states, projected entangled pair states, and variational renormalization group methods for quantum spin systems," *Advances in Physics*, vol. 57, no. 2, pp. 143–224, 2008.

[38] M. Suzuki, "General theory of higher-order decomposition of exponential operators and symplectic integrators," *Physics Letters A*, vol. 165, no. 5, pp. 387–395, 1992.

[39] K. J. Runge and G. T. Zimanyi, "Exact-diagonalization study of the one-dimensional disordered *XXZ* model," *Phys. Rev. B*, vol. 49, pp. 15212–15222, Jun 1994.

[40] A. Läuchli, J. Sudan, and E. Sørensen, "Ground-state energy and spin gap of spin-1/2 kagomé-heisenberg antiferromagnetic clusters: Large-scale exact diagonalization results," *Physical Review B*, vol. 83, no. 21, p. 212401, 2011.