A Practical Introduction to Quantum Many-Body Simulation and Matrix Product States

Ahmed Akhtar
Prof. Nicolas Regnault

May 3, 2016

Abstract

This paper offers a practical introduction to numerical techniques in Quantum Many-Body simulation. We cover exact diagonalization examples for the Heisenberg and AKLT models. We also cover the Matrix Product State representation of the AKLT ground state. The paper also discusses theoretical details behind MPS and how it can be used to efficiently calculate physical quantities of a system.

1 Introduction

One does not have to know a lot of Quantum Mechanics to understand that Hilbert spaces can get big very, very quickly. Consider a system of N interacting spin-half particles. Without any additional assumption, each state of the system lives in a Hilbert space $\mathcal{H}$ with dimension $\dim \mathcal{H}$ that goes like 2^N. This is because each particle can be in either the $+\hbar/2$ or $-\hbar/2$ eigenstate of the z-angular momentum operator S_z.

$$\dim \mathcal{H} = 2^N \tag{1}$$

In general, the dimension of the Hilbert space of a lattice of N spin s particles is

$$\dim \mathcal{H} = (2s+1)^N \tag{2}$$

This is the heart of the problem in condensed matter simulations of spin-systems. If we want to represent the exact state of a system numerically, memory and time scale exponentially. If the system size was small, this wouldn't be much of a problem. For example, finding the eigenstates of the AKLT-Hamiltonian (Affleck, Lieb, Kennedy and Tasaki) for $N = 8$ is not too difficult computationally even on a laptop, but what about for $N = 100, 1000, 10000$? Often-times we want to simulate large systems. If we were restricted to exponential algorithms, simulating large systems would be impossible.

Matrix product states (MPS) are a feasible method for representing the states of a variety of systems, including the AKLT-model, which is an extension of the Heisenberg model. Each model comprises of a spin chain, or a 1d lattice, of N sites, with each site occupied by a particle with a finite state space of dimension d. In the Heisenberg model, we imagine a spin chain of N spin-half particles ($d = 2$). In the AKLT-model, we imagine a spin chain of N spin-one particles ($d = 3$). MPS provides a way to calculate a number of important system variables (e.g. ground state energy, correlation length, overlaps, etc.) in time linear in N, d with a high degree of accuracy. As such, MPS has a wide array of applications in condensed matter simulations[1].

What about 2d lattices or 3d lattices of spin particles? The natural generalization of the Matrix Product State is the tensor network. While this paper only discusses spin chains, the concepts and implementations could be generalized to higher dimensions.

A number of factors play a role in whether an efficient MPS representation of the ground state exists in a particular system. MPS relies on the system having little entanglement between the different parts of the system. As we will see, for the AKLT-spin chain, the Von Neumann Entropy has a finite upper bound. This is an example of the area law, which suggests that the Von Neumann entropy of the reduced density matrix of some subvolume of a system is proportional to the surface area of the subvolume. It was also shown that for a 1d system with a finite gap[2] (i.e. a finite energy difference between the ground state and the excited state as $N \to \infty$), its ground state has an efficient MPS description. The AKLT model is also an example of a system with a finite gap [4].

This paper will explore simulations using exact diagonalization, where we generate the Hamiltonian and then diagonalize to find the energy spectrum, and also MPS, where we write the state as its matrix product representation. The codes used to implement each calculation are included in the appendix and will occasionally be referenced in the paper. This is a practical introduction to quantum many-body simulations, so programming is a large part of the work completed in this paper. The language used is Python. NumPy classes were used for the linear algebra aspects of the code. MatPlotLib was used for plotting the data.

This paper will also discuss the Von Neumann Entropy, which is a measure of the entanglement in a system. We will consider some of the theory behind MPS, involving reduced density operators and Schmidt decomposition. We will conclude the paper with how MPS can be used to efficiently

[1] See [2], [3].

[2] Also known as a Haldane gap.

compute the squared norm of a vector $|\Psi\rangle$. This is a remarkable example of how MPS can be used to circumvent the fact that Hilbert spaces grows exponentially with system size N.

2 The Heisenberg Hamiltonian and Straightforward Simulations

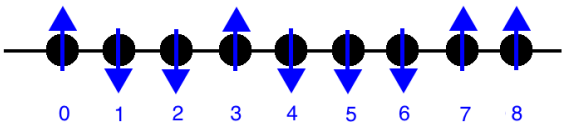


Figure 1: The first 9 sites in a spin chain consisting of spin-half particles.

To get an introduction to numerical calculations on spin chains, we will first study the Heisenberg Hamiltonian. Suppose that you have a spin chain with N sites. At each site, labeled 0 to $N-1$, there is a spin-half particle. Thus, the size of the Hilbert space is $\dim \mathcal{H} = 2^N$. Suppose that the z-spin angular momentum operator of the particle at site i is represented as S_z^i. Then the Heisenberg Hamiltonian is given by

$$H = \sum_{i=0}^{N-2} S_x^i S_x^{i+1} + S_y^i S_y^{i+1} + S_z^i S_z^{i+1} \tag{3}$$

The notation here can be simplified by defining $\vec{S^i}$ as the collection of operators (S_x^i, S_y^i, S_z^i). Using this notation, the above Hamiltonian becomes

$$H = \sum_{i=0}^{N-2} \vec{S^i} \cdot \vec{S^{i+1}} \tag{4}$$

We will find the energy spectrum, or the eigenvalues of the Hamiltonian, by first constructing the Hamiltonian as a matrix. Then we will use Python's NumPy linear algebra classes to calculate the eigenvalues. Note that this algorithm is fundamentally exponential, because there are $d^2 = 4^n$ elements of the Hamiltonian. Although it is not necessary to actually calculate every element of the Hamiltonian, since as we will see the Hamiltonians we discuss in this paper are sparse and can be made block-diagonal in the appropriate basis, calculating the non-zero terms still goes exponentially. This will be an issue for large sized systems, but to get an idea of what numerical computations are like we can proceed with this example. We will find that the implementation of the Heisenberg Hamiltonian is very similar to the implementation of the AKLT Hamiltonian, for which we will construct the matrix product states.

To compute that elements of H, we need a way to compute $\langle i| H |j\rangle \equiv H_{ij}$ without already knowing the matrix elements. But first, we need a representation of $|i\rangle$. What are the state vectors of a spin-half chain?

The state at each site can be described completely by a pair of numbers s and m, where s is the intrinsic spin at each site and m is the S_z eigenvalue. In this case, $s = 1/2$ and[3] $m = \pm 1/2$. Thus, we can write the local basis vectors as $|\frac{1}{2}\ \frac{1}{2}\rangle$ for $m = 1/2$ and $|\frac{1}{2}\ -\frac{1}{2}\rangle$ for $m = -1/2$. The basis vectors of the whole system are given by the direct product of the basis vectors at one site. If we write the state at each site in terms of the S_z eigenvectors, we can write an arbitrary basis element of the *system* as

$$|\frac{1}{2}\ m_0\rangle \otimes |\frac{1}{2}\ m_1\rangle \ldots \otimes |\frac{1}{2}\ m_{N-1}\rangle \tag{5}$$

where $m_i = \pm 1/2$ represents the S_z eigenvalue at the i-th site. To simplify this notation, we can write the $m = -1/2$ local basis state as $|0\rangle$ and the $m = 1/2$ local basis state as $|1\rangle$ and drop the tensor product symbol. This allows us to view each state as a binary integer.

[3] As usual, we leave out the $\hbar$. In most numerical calculations on quantum systems, $\hbar$ is taken to be 1.

$$|00010010...01\rangle \tag{6}$$

All 2^N consecutive integers from 00...0 to 11...1, written in binary, map bijectively to the set of basis states of the system. Thus, to simplify our notation even further, we can write each state as its corresponding integer from $\{0, 1, ...2^N - 1\}$:

$$|\alpha\rangle \quad \alpha \in \{0...d-1\} \tag{7}$$

For example, the state $|5\rangle$ has as its binary representation $|000...101\rangle$. It is the state where all of the particles are spin down except for the ones at site $N-1$ and $N-3$ which are spin up.

To represent linear combinations of states would be rather difficult, because we would have to keep track of each state vector separately along with the coefficient. To compute H_{ij} it would help greatly to break H up into operators that map from exactly one basis vector to another, with a constant multiplier. We can do this by utilizing the raising and lowering operators[4].

$$S^i_\pm \equiv S^i_x \pm iS^i_y \tag{8}$$

The annihilation and creation operators act on a spin state with intrinsic spin s and z-spin m according to the following formula

$$S_\pm |s\ m\rangle = \hbar\sqrt{s(s+1) - m(m\pm 1)}\,|s\ m\pm 1\rangle \tag{9}$$

where the i in the exponent tells you which site the operator acts on. For example, S^0_+ is the raising operator for the spin-half particle at location site 0. By expanding out the terms in $\frac{1}{2}(S^i_+S^{i+1}_- + S^i_-S^{i+1}_+)$, we can see how to rewrite $S^i_xS^{i+1}_x + S^i_yS^{i+1}_y$ in the Hamiltonian in terms of the raising and lowering operators, which map one basis state to exactly one other basis state.

$$\frac{1}{2}((S^i_x + iS^i_y)(S^{i+1}_x - iS^{i+1}_y) + (S^i_x - iS^i_y)(S^{i+1}_x + iS^{i+1}_y)) \tag{10}$$

$$= \frac{1}{2}(S^i_xS^{i+1}_x - iS^i_xS^{i+1}_y + iS^i_yS^{i+1}_x + S^i_yS^{i+1}_y + S^i_xS^{i+1}_x + iS^i_xS^{i+1}_y - iS^i_yS^{i+1}_x + S^i_yS^{i+1}_y) \tag{11}$$

$$= \frac{1}{2}(2S^i_xS^{i+1}_x + 2S^i_yS^{i+1}_y) \tag{12}$$

$$= S^i_xS^{i+1}_x + S^i_yS^{i+1}_y \tag{13}$$

Thus, the Heisenberg Hamiltonian of system becomes

$$H = \sum_{i=0}^{N-2} \frac{1}{2}(S^i_+S^{i+1}_- + S^i_-S^{i+1}_+) + S^i_zS^{i+1}_z \tag{14}$$

Each of the terms in the sum have a neat mapping between basis vectors of the system. For example, suppose that $N = 4, i = 2, \alpha = 5$. Then $S^i_zS^{i+1}_z\,|5\rangle = \frac{-\hbar}{2}\frac{\hbar}{2}\,|0101\rangle = \frac{-\hbar^2}{4}\,|5\rangle$ and $S^i_+S^{i+1}_-\,|5\rangle = S^i_+S^{i+1}_-\,|0101\rangle = \frac{3\hbar^2}{4}\,|0110\rangle = \frac{3\hbar^2}{4}\,|6\rangle$ and $S^i_-S^{i+1}_+\,|5\rangle = 0$. Armed with this new, convenient form of the Hamiltonian, we can write code that calculates the matrix elements of H. First, we can write an algorithm that gives you the binary representation of a state given α. This form is particularly useful for determining the effect of the operators on a given state, since each operator acts on two bits of the whole state at once. We give a simple, intuitive (though not efficient) algorithm for finding the N bit binary representation of an integer in line 20 of Section 5.1. One can

[4]Also known as the creation and annihilation operators.

also go the other way: find the integer representation $|\alpha\rangle$ given the binary representation (see line 13). The array element state[i] has the value 1.0 if the spin at position i is up and 0.0 otherwise.

One can compute $S_z^i S_z^j |\alpha\rangle$ by looking at the binary representation, and multiplying the provided state vector by $\frac{1}{4}$ if the z-spin at i and j are the same and $-\frac{1}{4}$ if they are different[5]. See line 29 of section 5.1. We represent this numerically by two double type integers: one the coefficient (result[0]) and one for α (result[1]).

Computing $S_+^i S_-^j |\alpha\rangle$ (line 39 of Section 5.1) is also easier using the binary representation. If the state at i is $|1\rangle$ or if the state at j is $|0\rangle$, then the resulting state will be annihilated. This is represented by setting the coefficient equal to zero. We can find the resulting α from acting on the state by just taking the original α and subtracting the bit at the j and adding the bit at i.

With these functions defined, it is a simple matter to compute the elements of the Heisenberg Hamiltonian (see the Appendix for a complete implementation) and we can move on to computing the spectrum using NumPy. Since $S_z^{total} \equiv \sum_{i=0}^{N-1} S_z^i$ commutes with the Heisenberg Hamiltonian, we can plot the spectrum of each fixed S_z^{total} eigenvalue. To do that, we need to generate the basis that has a given fixed total z-spin. From a computation perspective, that is not too difficult. One method is to go through the complete basis, and keep track of the basis states whose total z-spin adds up to the fixed value. We present an alternative approach in the appendix.

The alternative approach is to find the number of spins which are "up" if the S_z^{total} eigenvalue is, say, s_z^t, and then recursively generate the sub-basis by moving around the ones in the state array. Let x be the number of sites that have $S_z = \hbar/2$. Then, $N - x$ must be the number of remaining sites which have z spin $-\hbar/2$. Thus, in units of $\hbar/2$

$$s_z^t = x - (N - x) = 2x - N \tag{15}$$

$$x = \frac{1}{2}(s_z^t + N) \tag{16}$$

How do we know we will always get an integer for x? It turns out that s_z^t has a number of interesting properties, such as that if N is even, then s_z^t will always be even. Furthermore, if N is odd, then s_z^t will be odd. Thus their sum will always be even, and therefore x will be an integer. We can prove this by recognizing that the parity of s_z^t is an invariant: Suppose we have N spins originally all spin up. In this configuration, s_z^t has the same parity as N since they are equal. We can generate all configurations (or states) possible for N spins by flipping bits. If we flip a bit, we always change s_z^t by 2–if the spin is down and we flip it up, then we increase s_z^t by 2, and if the spin is up and we flip it down, we decrease s_z^t by two. Thus, the parity of s_z^t is an invariant for a given N.

Another interesting thing that the previous set of equations tells us is that there are exactly $N + 1$ unique s_z^t, one for each unique $x \in \{0...N\}$, ranging from $-N$ to $+N$. Additionally, one can ask, what is the size of the sub-basis with $S_z^{total} = s_z^t$? Let this subspace be $\mathcal{S}_{s_z^t}$. Since there are exactly x spin ups in a given configuration and there are N sites that we can choose from to be spin up

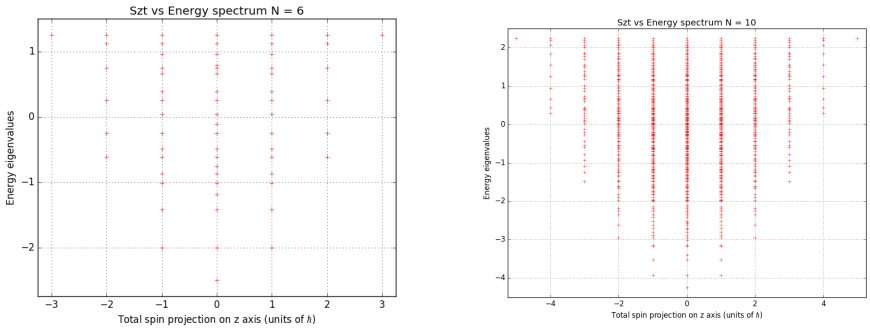
$$\dim \mathcal{S}_{s_z^t} = \binom{N}{x} = \binom{N}{\frac{1}{2}(s_z^t + N)} \tag{17}$$

We can sum over $\dim \mathcal{S}_{s_z^t}$ to show that it partitions the space. Since there is a bijection between x and s_z^t we can just as well sum over x and then invoke the binomial theorem:

$$\sum \dim \mathcal{S}_{s_z^t} = \sum_{x=0}^{N} \binom{N}{x} = (1+1)^N = 2^N \quad \square \tag{18}$$

[5]Note that we are working in units of $\hbar^2$

We plot the energy spectrum vs total z-spin s_z^t in the following figures.



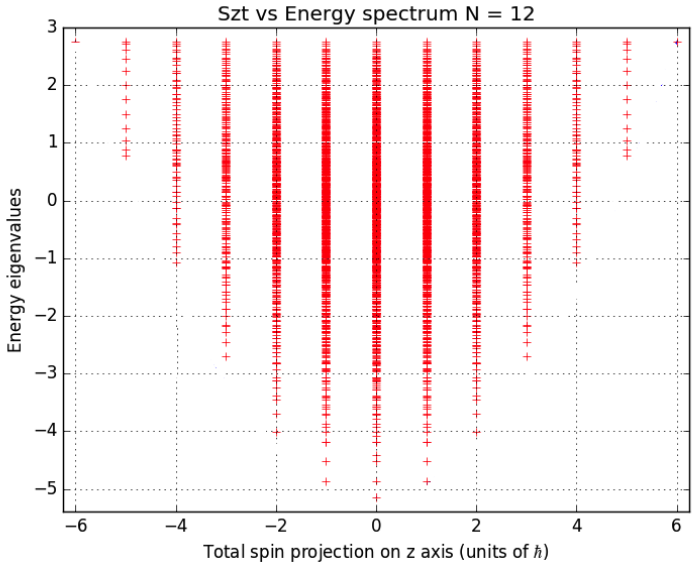


Figure 2: Energy is in units of $\hbar^2$ and spin angular momentum is in units of $\hbar$.

There are a number of interesting features in these plots. One interesting property that we notice is that the energy spectra are not evenly spaced vertically. In fact, at lower energies, the energy levels are farther apart, whereas for higher energies, they get closer together. This effect is also emblematic of the energy levels of ideal fermionic gasses. As the energy increases, the gaps between energy levels become smaller and smaller. Another interesting thing worth noting is that each horizontal line of energies represents a $\mathbf{S}^2 \equiv (\sum \vec{S^i})^2$ degenerate subspace, where the $\mathbf{S}^2$ eigenvalue is given by $l(l+1)$ where l is the maximum value of s_z^t for a given energy. Thus, one could present the same data by just plotting the energy and the total s_z^t value associated with it without losing any information.

To run the simulation for $N = 12$ required 40 minutes on a MacBook Pro (2012). We are quickly reaching an upper limit on the size of the system N that we compute the spectrum of. A quick inspection of the code reveals that even for a sparse matrix with only 1 non-zero value for every column, we would still have to compute the result of an operator on a state $3(N-1)$ times, which results in at least $3(N-1)2^N$ calculations of $S_z^i S_z^j \left|\alpha\right\rangle$ or $S_+^i S_-^j \left|\alpha\right\rangle$ just to construct the Hamiltonian. This is unfeasible for any large input size. This is where MPS comes in. But first, we must learn another Hamiltonian: The AKLT-Hamiltonian.

3 Features of the AKLT Hamiltonian

The AKLT-Hamiltonian[6] acts on a spin chain of length N with sites labeled 0 to $N-1$. Typically, the spin chain consists of spin 1. One interesting feature of a chain of spin ones is that each site can be represented as the triplet states of a pair of spin halves. The AKLT-Hamiltonian is given by

$$H = \sum_{i=0}^{N-2} \vec{S^i} \cdot \vec{S^{i+1}} + \frac{1}{3}(\vec{S^i} \cdot \vec{S^{i+1}})^2 \tag{19}$$

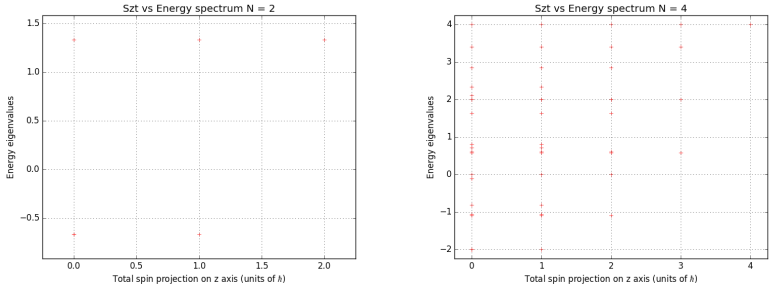
The size of the Hilbert space in the spin-one chain is

$$\dim \mathcal{H} = 3^N \tag{20}$$

The spectrum of a single pair of spin-ones is not difficult to calculate, but as N gets big, the computation becomes impossible to do by hand. As an exercise, we can write down what the correct eigenvalues would be for a pair of spin ones labeled 1 and 2. In this case, $\vec{S^1} \cdot \vec{S^2} = \frac{1}{2}((\mathbf{S})^2 - (\vec{S^1})^2 - (\vec{S^2})^2)$ where $\mathbf{S}^2 \equiv (\vec{S^1} + \vec{S^2})^2$. Since every spin is a spin-one, the last two terms have eigenvalues of $2\hbar^2$. From the rules of the addition of angular momenta for two spins with intrinsic spin s_1 and s_2, the total spin $\mathbf{S}$ ranges from $s_1 + s_2$ to $|s_1 - s_2|$. Thus, the eigenvalues of $(\mathbf{S})^2$ are $0, 2\hbar^2$ and $6\hbar^2$. Plugging in each possible eigenvalue of every operator gives the complete spectrum for $N = 2$: $\{-\frac{2}{3}, -\frac{2}{3}, \frac{4}{3}\}\hbar^2$. The ground state energy shows up twice, and is half the energy of the excited state. By adjusting the energy eigenvalues for the Hamiltonian for $N = 2$ we see that it is a projector onto the singlet and triplet states.

The implementation follows conceptually from the implementation for the Heisenberg spin half chain, so it may not be as instructive to go through it here[7]. This time around, we generated the basis first and assigned to each state an integer from 0 to $3^N - 1$. Since each spin can be in one of three states, the natural representation of each spin is as a base 3 integer string. If there is a 0.0, 1.0, or 2.0 at position i, then the site has $S_z^i = -\hbar, 0$, and $\hbar$, respectively. We label the state $|0\,0\,0...0\rangle$ as the integer 0, and follow the rules of addition base 3 to generate the rest of the basis in $O(3^N N)$ (generating the basis means generating $\dim \mathcal{H}$ length-N arrays).

We plotted again the energy spectrum vs. total z spin below. One of the more striking features we notice is that there is a gap in the ground state Hamiltonian. This gap guarantees a ground state MPS representation, which we will implement in the next section [7, 4]. In fact, it is conjectured that there exists gaps for spin chains with an integer spin but not for a half-integer spin, which is why we did not observe a gap between the Heisenberg ground state and the first excited state [1]. Another interesting feature we notice is that there is a four-fold degeneracy in the ground state, regardless of N. See figure 4 for a conceptual understanding of why this is.



[6] Affleck, Lieb, Kennedy and Tasaki

[7] See the appendix for details.

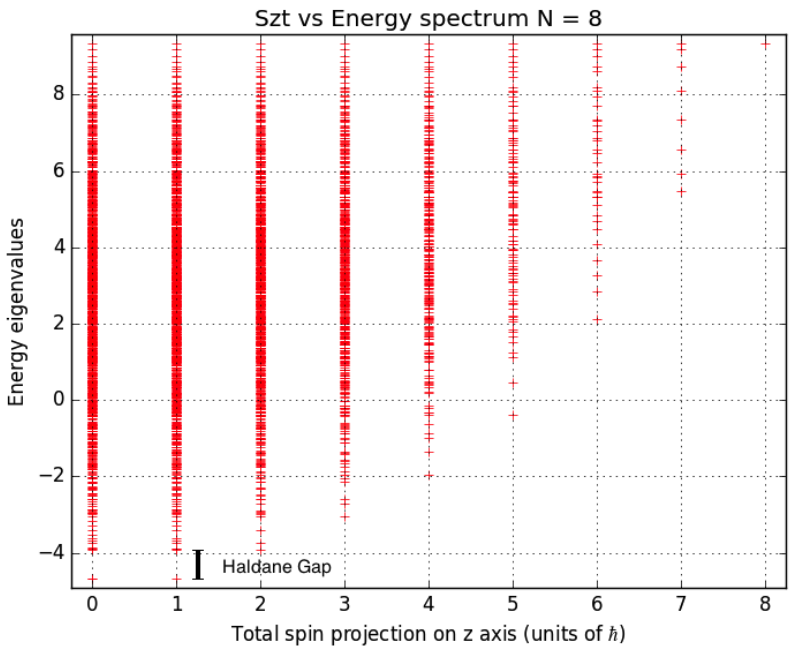


Figure 3: Energy is in units of $\hbar^2$ and spin angular momentum is in units of $\hbar$.

Figure 4: The AKLT Spin-1 Chain represented for $N = 5$. Each dot is a spin-half, and each line is a singlet bond. Each pair of spin-half particles connected by a line is a valence bond state. Note that we have drawn the system with open boundary conditions, as opposed to periodic boundary conditions where a bond would connect the first and last sites. The open boundary condition suggests a pair of free spin-halfs. This is significant because when looking at the plots, the ground state energy spectrum contains 4 points: two for $s_z^t = 0$, one for $s_z^t = 1$, and one for $s_z^t = -1$. This four-fold degeneracy is explained by the fact that the two outer spins can be in four states (three triplet and one singlet), where two of the states have $s_z^t = 0$, and one has $s_z^t = 1$ and one has $s_z^t = -1$.

$S_z^t = 0$								
0.0	0.0	-0.913	0.0	0.408	0.0	0.555	0.0	0.0
0.0	0.0	0.365	0.0	0.816	0.0	-0.577	0.0	0.0
0.0	0.0	0.183	0.0	0.408	0.0	0.599	0.0	0.0
$S_z^t = 1$								
0.0	0.0	0.0	0.0	0.0	0.707	0.0	-0.707	0.0
0.0	0.0	0.0	0.0	0.0	0.707	0.0	0.707	0.0
$S_z^t = 2$								
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0

Figure 5: The S_z^t eigenstates are also generated for $N = 2$. The states are written in the ordered basis where the number in the first column represents the coefficient to the state $|1\ -1\rangle \otimes |1\ -1\rangle$ and the final column represents the coefficient to the state $|1\ 1\rangle \otimes |1\ 1\rangle$.

4 Matrix Product State Representation of the AKLT Ground State

4.1 What does a MPS look like?

Instead of beginning with the usual theoretical derivation of the MPS using singular value decomposition, we can begin with the concrete example of the ground state for the AKLT Hamiltonian. In the previous section, we were able to find the ground state vectors for the AKLT-Hamiltonian by first constructing the Hamiltonian and then finding its eigenvectors using NumPy. This process took time exponential in the number of sites, and thus quickly became infeasible, requiring a running time of 40 minutes to compute the energy spectrum for $N = 8$.

As discussed prior, the MPS representation of the ground state takes time linear in the number of sites and the size of the local state space. The efficiency of the MPS also depends on the auxiliary space dimension χ. The auxiliary space dimension, also known as the bond dimension, is the largest matrix dimension of the product state. The larger we make χ, the more accurate our representation. In the case of the AKLT ground state, there exists an exact representation with $\chi = 2$. This means that using a small number of 2×2 matrices, we can efficiently calculate the AKLT ground state for arbitrary N.

Suppose that the state we want to represent as a product state is given by

$$|\Psi\rangle = \sum_{\{m_i\}} c_{\{m_i\}} |m_0 m_1 ... m_{N-1}\rangle \tag{21}$$

where as usual, N is the number of sites and each site can take on d different states. As in the Heisenberg example, m_i is the z angular momentum at each site. For the AKLT, $m_i = 0, \pm 1$ so $d = 3$. These indices, with $m_i \in \{-1, 0, 1\}$, are referred to as the physical indices because they refer to the possible results of an S_z measurement on a site. Without loss of generality[8], suppose we write $|\Psi\rangle$ as

$$|\Psi\rangle \sum_{\{m_i\}} (A^{[m_0]} ... A^{[m_{N-1}]})_{\alpha_L, \alpha_R} |m_0 ... m_{N-1}\rangle \tag{22}$$

A number of changes have happened here. Apparently, we have replaced the coefficient in our first representation of $|\Psi\rangle$ with an element of a product of N matrices: $A^{[m_0]} ... A^{[m_{N-1}]}$. In general, each site has d matrices associated with it, one for each physical index. Depending on the physical

[8]We haven't stated any explicit condition on the size of the matrices, so at this stage there isn't any loss of generality in writing the state as a matrix product state.

situation, the matrices could vary from site to site though in systems with translation symmetry cases they won't. Additionally, we have defined indices α_L, α_R that give the location of the coefficient in the matrix. At first, this seems like a roundabout way of representing a particular state. To calculate the coefficient for a single basis state, we would have to do $N-1$ matrix multiplications. The sum runs over all of the 3^N states available to the system, so on the surface it seems as though even if a Matrix Product State representation of a particular state exists, it would still take use exponential time to calculate it.

Let's explore a few examples of MPS. Suppose that we want to represent the highest S_z^t state using MPS. Intuitively, we know this state will be (in the physical indices) $|+1...+1\rangle$. How can we represent this state using Matrix Product States? If we let $A^{[-1]} = A^{[0]} = \begin{pmatrix}0\end{pmatrix}$ and let $A^{[+1]} = \begin{pmatrix}1\end{pmatrix}$ and use the same set of matrices at each site, then

$$|\Psi\rangle = \sum_{\{m_i\}} (A^{[m_0]}...A^{[m_{N-1}]})_{\alpha_L,\alpha_R} |m_0...m_{N-1}\rangle = |+1...+1\rangle \tag{23}$$

since the only term in the sum that will survive is the one where $m_0 = ... = m_{N-1} = +1$. The bond dimension of this system is 1, since all the matrices in this case are 1 dimensional.

Suppose we want to find the MPS representation for the CAT state of a system. The CAT state is an equal superposition of the highest and lowest S_z^t states.

$$|\Psi_{CAT}\rangle = \frac{1}{\sqrt{2}}(|+1\ ...\ +1\rangle + |-1\ ...\ -1\rangle) \tag{24}$$

Consider now the matrices

$$A^{[-1]} = \begin{pmatrix}0 & 0\\ 0 & 1\end{pmatrix} \quad A^{[0]} = \begin{pmatrix}0 & 0\\ 0 & 0\end{pmatrix} \quad A^{[+1]} = \begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix} \tag{25}$$

Note that the product of any two of the matrices that are not the same is zero (as expected). Similarly, any state that has 0 as a physical index will make the matrix product zero. Thus, the matrix product will not be zero only if the state in the sum is $|+1\ ...\ +1\rangle$ or $|-1\ ...\ -1\rangle$.

$$|\Psi_{CAT}\rangle = (A^{[+1]})^N_{\alpha_L,\alpha_R} |+1\ ...\ +1\rangle + (A^{[-1]})^N_{\alpha_L,\alpha_R} |-1\ ...\ -1\rangle \tag{26}$$

$$= (A^{[+1]})_{\alpha_L,\alpha_R} |+1\ ...\ +1\rangle + (A^{[-1]})_{\alpha_L,\alpha_R} |-1\ ...\ -1\rangle \tag{27}$$

However, the issue is that $A^{[+1]}$ and $A^{[-1]}$ are not non-zero in the same location. Thus, we can't get an equal superposition using only one pair of indices. To work around this, we can use the trace of the product of the matrices. This approach is often used any time the system as periodic boundary conditions. Since the trace of both matrices is 1, we get the CAT state up to a global normalization factor.

$$|\Psi\rangle = \sum_{\{m_i\}} Tr(A^{[m_0]}...A^{[m_{N-1}]}) |m_0...m_{N-1}\rangle = |+1\ ...\ +1\rangle + |-1\ ...\ -1\rangle \tag{28}$$

$$= \sqrt{2}\,|\Psi_{CAT}\rangle \tag{29}$$

4.2 AKLT Ground State

We will examine the matrices

$$A^{[0]} = \begin{pmatrix} -\sqrt{\frac{1}{3}} & 0 \\ 0 & \sqrt{\frac{1}{3}} \end{pmatrix} \quad A^{[+1]} = \begin{pmatrix} 0 & \sqrt{\frac{2}{3}} \\ 0 & 0 \end{pmatrix} \quad A^{[-1]} = \begin{pmatrix} 0 & 0 \\ -\sqrt{\frac{2}{3}} & 0 \end{pmatrix} \tag{30}$$

These three matrices can be used to generate the ground state manifold for the AKLT Hamiltonian. To see this, we can work out the example for $N = 2$ by hand. There are four ground state vectors for $N = 2$ with energy (as determined in a previous section) $-\frac{2}{3}$. In the total spin $\mathbf{S}^2$ and S_z^t basis, the states are $|1\,0\rangle, |1\ \pm 1\rangle, |00\rangle$ (where the first number is the $\mathbf{S}^2$ eigenvalue and the second is the S_z^t eigenvalue). There are four spots in the matrix which can be specified by the α_L, α_R, and therefore we should be able to recover 4 independent states. Let's see which of the matrix elements of the product correspond to the coefficients for a given state. For two sites, here are all the matrix products that can be evaluated.

$$A^{[0]}A^{[+1]} = \begin{pmatrix} -\sqrt{\frac{1}{3}} & 0 \\ 0 & \sqrt{\frac{1}{3}} \end{pmatrix} \begin{pmatrix} 0 & \sqrt{\frac{2}{3}} \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & -\sqrt{2}/3 \\ 0 & 0 \end{pmatrix} \quad A^{[+1]}A^{[0]} = \begin{pmatrix} 0 & \sqrt{\frac{2}{3}} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} -\sqrt{\frac{1}{3}} & 0 \\ 0 & \sqrt{\frac{1}{3}} \end{pmatrix} = \begin{pmatrix} 0 & \sqrt{2}/3 \\ 0 & 0 \end{pmatrix}$$

$$A^{[0]}A^{[0]} = \begin{pmatrix} -\sqrt{\frac{1}{3}} & 0 \\ 0 & \sqrt{\frac{1}{3}} \end{pmatrix} \begin{pmatrix} -\sqrt{\frac{1}{3}} & 0 \\ 0 & \sqrt{\frac{1}{3}} \end{pmatrix} = \begin{pmatrix} 1/3 & 0 \\ 0 & 1/3 \end{pmatrix} \quad A^{[+1]}A^{[+1]} = \begin{pmatrix} 0 & \sqrt{\frac{2}{3}} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & \sqrt{\frac{2}{3}} \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

$$A^{[+1]}A^{[-1]} = \begin{pmatrix} 0 & \sqrt{\frac{2}{3}} \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ -\sqrt{\frac{2}{3}} & 0 \end{pmatrix} = \begin{pmatrix} -2/3 & 0 \\ 0 & 0 \end{pmatrix} \quad A^{[-1]}A^{[+1]} = \begin{pmatrix} 0 & 0 \\ -\sqrt{\frac{2}{3}} & 0 \end{pmatrix} \begin{pmatrix} 0 & \sqrt{\frac{2}{3}} \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & -2/3 \end{pmatrix}$$

$$A^{[-1]}A^{[-1]} = \begin{pmatrix} 0 & 0 \\ -\sqrt{\frac{2}{3}} & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ -\sqrt{\frac{2}{3}} & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \quad A^{[-1]}A^{[0]} = \begin{pmatrix} 0 & 0 \\ -\sqrt{\frac{2}{3}} & 0 \end{pmatrix} \begin{pmatrix} -\sqrt{\frac{1}{3}} & 0 \\ 0 & \sqrt{\frac{1}{3}} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ -\sqrt{2}/3 & 0 \end{pmatrix}$$

$$A^{[0]}A^{[-1]} = \begin{pmatrix} -\sqrt{\frac{1}{3}} & 0 \\ 0 & \sqrt{\frac{1}{3}} \end{pmatrix} \begin{pmatrix} 0 & 0 \\ -\sqrt{\frac{2}{3}} & 0 \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ -\sqrt{2}/3 & 0 \end{pmatrix}$$

From now on, we will write everything using the physical indices. We define $|\Psi^{AKLT}_{\mathbf{s}^2, s_z^t}\rangle$ as the AKLT ground state with total spin $\mathbf{s}^2$ and total z angular momentum s_z^t. The state $|\Psi^{AKLT}_{1\,1}\rangle$ in the physical indices is given by $\frac{1}{\sqrt{2}}(|1\,0\rangle - |0\,1\rangle)$. Let's examine what vector we recover when looking at $(\alpha_L, \alpha_R) = (1, 2)$ index of the products. We know that any product involving $A^{[-1]}$ will give 0 at the position (1, 2) of the product. Thus, we only need to examine the products of $A^{[+1]}$ and $A^{[0]}$. Referring to the block of equations above, we ascertain the state formed by $\alpha_L = 1, \alpha_R = 2$.

$$\sum_{\{m_i\}} (A^{[m_0]}...A^{[m_{N-1}]})_{1,2} |m_0...m_{N-1}\rangle = \sqrt{2}/3\,(|1\,0\rangle - |0\,1\rangle) = 3/2\,|\Psi^{AKLT}_{1\,1}\rangle \tag{31}$$

Note that the matrices give the exact $S_z^t = 1$ ground state of the AKLT Hamiltonian for $N = 2$ within a constant factor of 3/2. We can generate the remaining basis elements by picking different matrix elements of the product:

$$(\alpha_L, \alpha_R) = (2, 1) \rightarrow \quad |\Psi\rangle = -\sqrt{2}/3\,(|0\ -1\rangle + |-1\,0\rangle) = 3/2\,|\Psi^{AKLT}_{1\,-1}\rangle \tag{32}$$

$$(\alpha_L, \alpha_R) = (1, 1) \rightarrow \quad |\Psi\rangle = \frac{1}{3}\,(|0\,0\rangle - 2\,|+1\ -1\rangle) \tag{33}$$

$$(\alpha_L, \alpha_R) = (2, 2) \rightarrow \quad |\Psi\rangle = \frac{1}{3}\,(|0\,0\rangle - 2\,|-1\ +1\rangle) \tag{34}$$

The first equation gives the total spin $S^2 = 1$, $S_z^t = -1$ eigenstate of the AKLT. The second and third equation of the set span the $S_z^t = 0$ space and are linear combinations of $|\Psi_{0\,0}^{AKLT}\rangle$ and $|\Psi_{1\,0}^{AKLT}\rangle$. We can extract $|\Psi_{1\,0}^{AKLT}\rangle$ by subtracting the second and third equation.

We can easily write some code that generates the ground state given N, α_L, α_R and a set of MPS matrices. In under a minute, we can generate one of the ground state eigenvectors for $N = 8$ and measure its energy using a simple code that takes a vector and multiplies it by the Hamiltonian. Since the $N = 8$ vector has $3^8 = 6561$ components (if we don't use any symmetry), we won't display it here. To confirm that our MPS approximation is accurate, we can compare the inner product of the exact diagonalization state for $N = 4$ calculated using the code derived in the last section with the MPS state. If the inner product is 1, we know the MPS gives the correct state.

$\alpha_L = 1, \alpha_R = 1$								
0.0	0.0	0.0	0.0	0.33	0.0	-0.66	0.0	0.0
$\alpha_L = 1, \alpha_R = 2$								
0.0	0.0	0.0	0.0	0.0	-0.47	0.0	0.47	0.0
$\alpha_L = 2, \alpha_R = 1$								
0.0	0.47	0.0	-0.47	0.0	0.0	0.0	0.0	0.0
$\alpha_L = 2, \alpha_R = 2$								
0.0	0.0	-0.66	0.0	0.33	0.0	0.0	0.0	0.0

Figure 6: MPS Representation for $N = 2$ in basis ordered from $|-1\ -1\rangle$ to $|+1\ +1\rangle$. Note that every ground state for arbitrary N consists of 4 degenerate energy eigenstates.

As mentioned earlier, it is not clear how having an MPS representation results in more efficient calculations of pertinent system variables e.g. scalar products between states and the ground state energy. In this section, we will explore some mathematical ideas behind MPS such as Singular Value Decomposition and Schmidt Decomposition. We will also talk about the Von Neumann Entanglement entropy and why it is relevant to MPS. We will conclude the paper with an example of a polynomial time computation of the overlap between two vectors using Matrix Product States.

4.3 SVD and Schmidt Decomposition of Bipartite Systems

The Singular Value Decomposition Theorem (often contracted to *SVD*) says that any rectangular matrix M can be written as

$$M = USV^\dagger \tag{35}$$

where U and V are semi-orthonormal and S is diagonal with non-negative entries. Let's break that down. Suppose M has n rows and m columns. Then U is a $n \times \min(n, m)$ matrix with orthonormal columns i.e. $U^\dagger U = \mathbb{I}$. S is a $\min(n, m) \times \min(n, m)$ diagonal matrix with nonnegative entries written s.t. $S_{1,1} \geq S_{2,2} \geq ...S_{r,r} > 0$ and $S_{t,t} = 0\ \forall t > r$. The rank of the diagonal matrix S is also r. In order for the dimensions to work out, $V^\dagger$ is then $\min(n, m) \times m$. It also has orthogonal rows, so $V^\dagger V = \mathbb{I}$ [8].

A bipartite system is a system that consists of two parts, where the combined Hilbert space of each part forms the full Hilbert space. If the two parts are $\mathcal{H}_L$ and $\mathcal{H}_R$, then the full Hilbert space is $\mathcal{H} = \mathcal{H}_L \otimes \mathcal{H}_R$. Using SVD, we can write any bipartite system in a much more illuminating form. This form is known as the *Schmidt decomposition*.

Suppose that $\dim \mathcal{H}_L = n$ and $\dim \mathcal{H}_R = m$. We can write any state of the combined system $\mathcal{H}$ as

$$|\Psi\rangle = \sum_{i=1}^{n} \sum_{j=1}^{m} M_{ij} |i\rangle_L |j\rangle_R \tag{36}$$

where $\{|i\rangle_L\}$ and $\{|i\rangle_R\}$ are orthogonal bases for $\mathcal{H}_L$ and $\mathcal{H}_R$, respectively. We can think of each coefficient of the basis state M_{ij} as a matrix element of some matrix M. With some algebraic manipulation, we can deduce that the reduced density matrices of the sub-systems ρ_L and ρ_R are given by $MM^\dagger$ and $M^\dagger M$, respectively. Recall that the reduced density matrix for a system or subsystem is a sum of projectors onto the states of the system weighted by their probability of being measured. The reduced density matrix for the full state is then $\rho = |\Psi\rangle\langle\Psi|$, and the reduced density matrix for one part is found by tracing ρ over the states of the other part [5]. We derive the relationship for ρ_L below (ρ_R follows similarly).

$$\rho_L = Tr_R(|\Psi\rangle\langle\Psi|) \tag{37}$$

$$= \sum_{j=1}^{m} \langle j|_R \left(\sum_{i'j'} M_{i'j'} |i'\rangle_L |j'\rangle_R \sum_{i''j''} M^*_{i''j''} \langle i''|_L \langle j''|_R \right) |j\rangle_R \tag{38}$$

$$= \sum_{j=1}^{m} \left(\sum_{i'j'} M_{i'j'} |i'\rangle_L \langle j|_R |j'\rangle_R \sum_{i''j''} M^*_{i''j''} \langle i''|_L \langle j''|_R |j\rangle_R \right) \tag{39}$$

$$= \sum_{j=1}^{m} \left(\sum_{i'} M_{i'j} |i'\rangle_L \sum_{i''} M^*_{i''j} \langle i''|_L \right) \tag{40}$$

In the second line of the equations, we exploited the orthogonality of $\{|j\rangle_R\}$. To complete the derivation, let's observe what an element of ρ_L, $(\rho_L)_{ij}$, would be.

$$(\rho_L)_{ij} = \langle i|_L \sum_{k=1}^{m} \left(\sum_{i'} M_{i'k} |i'\rangle_L \sum_{i''} M^*_{i''k} \langle i''|_L \right) |j\rangle_L \tag{41}$$

$$= \sum_{k=1}^{m} M_{ik} M^*_{jk} = \sum_{k=1}^{m} M_{ik} (M^\dagger)_{kj} = (MM^\dagger)ij \tag{42}$$

$$\rho_L = MM^\dagger \quad \blacksquare \tag{43}$$

Suppose that we apply SVD to the matrix M. M is a rectangular $n \times m$ matrix, and so SVD applies, however it is unclear at this stage what decomposing the matrix of coefficients would yield. Following through with the calculation, we arrive at an interesting result.

$$M = USV^\dagger \tag{44}$$

$$M_{ij} = \sum_{k=1}^{\min(n,m)} U_{ik} (SV^\dagger)_{kj} \tag{45}$$

$$= \sum_{k=1}^{\min(n,m)} U_{ik} S_{kk} V^\dagger_{kj} \tag{46}$$

Since S is a nonnegative diagonal matrix, we can keep the terms in the sum where the diagonal elements are non-zero and discard the ones which are zero. The last nonzero diagonal element is, by definition of r, $S_{rr} \equiv s_r \neq 0$.

$$M_{ij} = \sum_{k=1}^{r} U_{ik} s_k V_{kj}^{\dagger} \tag{47}$$

$$|\Psi\rangle = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{r} U_{ik} s_k V_{kj}^{\dagger} |i\rangle_L |j\rangle_R \tag{48}$$

$$= \sum_{k=1}^{r} s_k \left(\sum_{i=1}^{n} U_{ik} |i\rangle_L \right) \left(\sum_{j=1}^{m} V_{kj}^{\dagger} |j\rangle_R \right) \tag{49}$$

$$= \sum_{k=1}^{r} s_k |\alpha_k\rangle_L |\alpha_k\rangle_R \tag{50}$$

where

$$|\alpha_k\rangle_L \equiv \sum_{i=1}^{n} U_{ik} |i\rangle_L \quad |\alpha_k\rangle_R \equiv \sum_{j=1}^{m} V_{kj}^{\dagger} |j\rangle_R \tag{51}$$

To complete the derivations, we will show that the bases $\{|\alpha_k\rangle_L\}$ and $\{|\alpha_k\rangle_R\}$ are orthonormal. If $n = m$, then both sets are orthonormal bases for their respective systems [8].

$$\langle \alpha_{k'} | \alpha_k \rangle_L = \sum_{i'=1}^{n} U_{i'k'}^{*} \langle i' |_L \sum_{i=1}^{n} U_{ik} |i\rangle_L \tag{52}$$

$$= \sum_{i=1}^{n} U_{ik'}^{*} U_{ik} = \delta_{k',k} \tag{53}$$

where in the first line, we utilize the orthogonality of $\{|i\rangle_L\}$ and in the last line, we utilize the orthogonality of the columns of U. The proof for the orthonormality of $\{|\alpha_k\rangle_R\}$ is identical. Note also that since the above represents a state, the magnitude squared of each coefficient must add up to one.

$$\sum_{k=1}^{r} |s_k|^2 = 1 \tag{54}$$

What is the merit of writing the combined state in this fashion? Firstly, the Von Neumann entropy, which we will see is a measure of the entanglement of the system, is easily calculable in this form. The Von Neumann entropy of the $\mathcal{H}_L$ subsystem is [5]

$$\mathcal{S}_L \equiv -Tr_L(\rho_L \ln \rho_L) \tag{55}$$

In the form of the Schmidt decomposition, we can see ρ is a sum of pure states of the form $|\alpha_k\rangle_L |\alpha_k\rangle_R \langle \alpha_k|_L \langle \alpha_k|_R$ each of which has a probability of $s_k^* s_k \equiv |s_k|^2$ of being measured. Thus, the Von Neumann entropy simplifies. Additionally, the mixed density matrices ρ_L and ρ_R are also easily calculable [8].

$$\mathcal{S}_L = \mathcal{S}_R = -\sum_{k=1}^{r} |s_k|^2 \ln(|s_k|^2) \tag{56}$$

$$\rho_L = Tr_R(|\Psi\rangle \langle\Psi|) \tag{57}$$

$$= \sum_{k=1}^{r} \langle\alpha_k|_R \left(\sum_{k'=1}^{r} s_{k'} |\alpha_{k'}\rangle_L |\alpha_{k'}\rangle_R \sum_{k''=1}^{r} s^*_{k''} \langle\alpha_{k''}|_L \langle\alpha_{k''}|_R \right) |\alpha_k\rangle_R \tag{58}$$

$$= \sum_{k=1}^{r} s_k s^*_k |\alpha_k\rangle_L \langle\alpha_k|_L = \sum_{k=1}^{r} |s_k|^2 |\alpha_k\rangle_L \langle\alpha_k|_L \tag{59}$$

$$\rho_R = \sum_{k=1}^{r} |s_k|^2 |\alpha_k\rangle_R \langle\alpha_k|_R \tag{60}$$

4.4 Entanglement Entropy of a Spin Chain

We consider a similar exercise on an arbitrary state $|\Psi\rangle$ of a familiar system: a spin chain of N particles. Suppose that we decide to divide the system into a left and a right half: $\mathcal{H} = \mathcal{H}_L \otimes \mathcal{H}_R$. The left half of the system consists of the first N_L particles and is spanned by the basis $\{|m_0 ... m_{N_L-1}\rangle\}$. The right half consists of the remaining particles and is spanned by the basis $\{|m_{N_L} ... m_{N-1}\rangle\}$. We can write the coefficients of the state as matrix elements of the entanglement matrix M, which is defined by the relationship $M_{\{m_0 ... m_{N_L-1}\},\{m_{N_L} ... m_{N-1}\}} = c_{\{m_1 ... m_{N-1}\}}$. We define n and m as the dimensions of the left and right Hilbert spaces, respectively. It follows that $\dim \mathcal{H} = nm$ and M is $n \times m$.

$$|\Psi\rangle = \sum_{\{m_i\}} c_{\{m_i\}} |m_0 ... m_{N-1}\rangle \tag{61}$$

$$= \sum_{\{m_i\}} M_{\{m_0 ... m_{N_L-1}\},\{m_{N_L} ... m_{N-1}\}} |m_0 ... m_{N_L-1}\rangle |m_{N_L} ... m_{N-1}\rangle \tag{62}$$

As we can see, the system above is bipartite and therefore we can write it as a Schmidt decomposition. We use slightly different notation to express that we are now specifically looking at the example of a spin chain.

$$|\Psi\rangle = \sum_{i=1}^{\min(n,m)} e^{-\mathcal{E}_i/2} |L : i\rangle \otimes |R : i\rangle \tag{63}$$

where some of the $e^{-\mathcal{E}_i/2}$ are allowed to be zero[9], in accordance with the fact that some of the s_i in the Schmidt decomposition in equation (50) can be 0. The states $\{|L : i\rangle\}$ and $\{|R : i\rangle\}$ are orthonormal bases given the SVD of $M = UDV^\dagger$.

$$|L : i\rangle = \sum_{\{m_0 ... m_{N_L-1}\}} U_{\{m_0 ... m_{N_L-1}\},i} |m_0 ... m_{N_L-1}\rangle \tag{64}$$

$$|R : i\rangle = \sum_{\{m_{N_L} ... m_{N-1}\}} V^*_{\{m_{N_L} ... m_{N-1}\},i} |m_{N_L} ... m_{N-1}\rangle \tag{65}$$

[9] Or equivalently some of the $\mathcal{E}_i \to \infty$

Using the fact that $\rho_L = MM^\dagger$, we see that the reduced density matrix of the left half L is given by

$$\rho_L = \sum_{i=1}^{\min(n,m)} e^{-\mathcal{E}_i} |L:i\rangle \langle L:i| \tag{66}$$

The spectrum of ρ_L is the set of $\{e^{-\mathcal{E}_i}\}$. This is clear the diagonalization of ρ_L above. The rank will be the number of non-zero coefficients $e^{-\mathcal{E}_i}$. The reduced density matrix for the right half ρ_R has the same spectrum and rank.

As we saw before, the Von Neumann Entropy, or the entanglement entropy, of the left half is given by $\mathcal{S}_L = -Tr_L[\rho_L \ln \rho_L]$. How do we compute the logarithm of a matrix? First, note that the square of ρ_L amounts to squaring the coefficient of each term in the sum of equation (67). Generalizing this, we notice that any power a of ρ_L is given by $(\rho_L)^a = \sum_{i=1}^{\min(n,m)} e^{-a\mathcal{E}_i} |L:i\rangle \langle L:i|$. The natural logarithm function has a polynomial expansion and writing $\ln \rho_L$ in terms of this expansion reveals that the logarithm is just applied to the coefficients $e^{-\mathcal{E}_i}$. Thus, the entanglement entropy is given by

$$\mathcal{S}_L = \sum_{i=1} \langle L:i| \left(\sum_{i'} e^{-\mathcal{E}_{i'}} |L:i'\rangle \langle L:i'| \sum_{i''} -\mathcal{E}_{i''} |L:i''\rangle \langle L:i''| \right) |L:i\rangle \tag{67}$$

$$= -\sum_{i=1}^{\min(n,m)} \mathcal{E}_i e^{-\mathcal{E}_i} \tag{68}$$

If $|\Psi\rangle$ is a product state, there is only one non-zero $e^{-\mathcal{E}_i}$, which must equal 1 due to normalization. Thus, $\mathcal{S}_L = -\sum_{i=1}^{\min(n,m)} \mathcal{E}_i e^{-\mathcal{E}_i} = 0$ for a product state. It will be greater than 0 for any entangled state.

4.4.1 An example with Two Spin Halfs

$$|\Psi\rangle = c_{11}|11\rangle + c_{10}|10\rangle + c_{01}|01\rangle + c_{00}|00\rangle \tag{69}$$

A generic state for two spin halfs is given by the equation above using the same notation as in section 2. The entanglement matrix M from the previous section has 4 entries is given by

$$M = \begin{pmatrix} c_{11} & c_{10} \\ c_{01} & c_{00} \end{pmatrix} \tag{70}$$

The density matrix for the left spin is ρ_L which is given by tracing the density matrix $|\Psi\rangle\langle\Psi|$ over the right states, or by using the fact we proved in the previous section that $\rho_L = MM^\dagger$

$$\rho_L = MM^\dagger = \begin{pmatrix} c_{11} & c_{10} \\ c_{01} & c_{00} \end{pmatrix} \begin{pmatrix} c_{11}^* & c_{01}^* \\ c_{10}^* & c_{00}^* \end{pmatrix} \tag{71}$$

Suppose $|\Psi\rangle$ is any of the product states, such as $|11\rangle$. Then

$$\rho_L = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \tag{72}$$

$$\mathcal{S}_L = -Tr_L[\rho_L \ln \rho_L] = -Tr[\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}] = 0 \tag{73}$$

If $|\Psi\rangle$ is the singlet state, which definitely exhibits entanglement, then

$$\rho_L = \begin{pmatrix} 0 & 1/\sqrt{2} \\ -1/\sqrt{2} & 0 \end{pmatrix} \begin{pmatrix} 0 & -1/\sqrt{2} \\ 1/\sqrt{2} & 0 \end{pmatrix} = \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix} \tag{74}$$

$$\mathcal{S}_L = -Tr_L[\rho_L \ln \rho_L] = -Tr[\begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix} \begin{pmatrix} \ln \frac{1}{2} & 0 \\ 0 & \ln \frac{1}{2} \end{pmatrix}] = \ln 2 \tag{75}$$

In this case, ρ_L for the $S_z^t = 0$ triplet state is the same and therefore so is the entanglement entropy. As implied by the name, it seems that the more entangled a system is, the higher the entanglement entropy. In fact, we can show that the maximally entangled states for two spin halfs with $S_z^t = 0$ are in fact the singlet and triplet states. Indeed, $|\Psi\rangle = \beta |10\rangle + \alpha |01\rangle$ where $\beta^2 + \alpha^2 = 1$. The entanglement entropy would be given by $\mathcal{S}_L = -\beta^2 \ln \beta^2 - \alpha^2 \ln \alpha^2$. We can find the maximum by taking both derivatives, setting them equal to zero, and using the method of Lagrange multipliers to fix the normalization condition. Or we could also just infer from symmetry that the critical point could only occur at $\alpha = \beta = 1/\sqrt{2}$, which shows that an equal superposition of $|01\rangle$ and $|10\rangle$ states gives the maximal entropy. Additionally, it tells us that the maximum entanglement entropy of a pair of spin halfs is $\ln 2$.

4.5 MPS and the Reduced Density Matrix

$$\sum_{\{m_i\}} \left(A^{[m_0]} ... A^{[m_{N-1}]} \right)_{\alpha_L, \alpha_R} |m_0 ... m_{N-1}\rangle \tag{76}$$

We will now see how to get ρ_L from the MPS representation of a state. Suppose we have a state $|\Psi\rangle$ with an MPS representation given above. Let χ be the bond dimension of the MPS. Suppose that we divide the system up into a left side (which includes the first N_L particles of the system) and a right side. Then we can rewrite the MPS in a form that resembles the Schmidt decomposition. As we learned in the previous sections, this form is very illuminating for the entanglement related properties of the system.

$$|\Psi\rangle = \sum_{\{m_i\}} \sum_{\alpha=1}^{\chi} (A^{[m_0]} ... A^{[m_{N_L-1}]})_{\alpha_L, \alpha} (A^{[m_{N_L}]} ... A^{[m_{N-1}]})_{\alpha, \alpha_R} |m_0 ... m_{N_L-1}\rangle \otimes |m_{N_L} ... m_{N-1}\rangle \tag{77}$$

$$= \sum_{\alpha=1}^{\chi} \sum_{\{m_i\}} ((A^{[m_0]} ... A^{[m_{N_L-1}]})_{\alpha_L, \alpha} |m_0 ... m_{N_L-1}\rangle) \otimes ((A^{[m_{N_L}]} ... A^{[m_{N-1}]})_{\alpha, \alpha_R} |m_{N_L} ... m_{N-1}\rangle) \tag{78}$$

$$= \sum_{\alpha=1}^{\chi} |L : \alpha\rangle \otimes |R : \alpha\rangle \tag{79}$$

$$|L : \alpha\rangle \equiv \sum_{\{m_0 ... m_{N_L-1}\}} (A^{[m_0]} ... A^{[m_{N_L-1}]})_{\alpha_L, \alpha} |m_0 ... m_{N_L-1}\rangle \tag{80}$$

$$|R : \alpha\rangle \equiv \sum_{\{m_{N_L} ... m_{N-1}\}} (A^{[m_{N_L}]} ... A^{[m_{N-1}]})_{\alpha, \alpha_R} |m_{N_L} ... m_{N-1}\rangle \tag{81}$$

However, the equations above aren't quite a Schmidt decomposition. A Schmidt decomposition guarantees an orthonormal basis for each section of the Hilbert space. In this case, there's no

reason for $\{|L:\alpha\rangle\}$ or $\{|R:\alpha\rangle\}$ to be an orthonormal basis. However, we can always extract an orthonormal basis from a subspace in a number of ways, for example, by Gram-Schmidt. Suppose that we find χ-dimensional orthonormalized bases for each subspace, $\{|L:\bar{\alpha}\rangle\}$ and $\{|R:\bar{\beta}\rangle\}$, and we rewrite $|L:\alpha\rangle$ and $|R:\alpha\rangle$ in these bases using transformation matrices[10] U and V.

$$|L:\alpha\rangle = \sum_{\bar{\alpha}} U_{\alpha,\bar{\alpha}} |L:\bar{\alpha}\rangle \quad |R:\alpha\rangle = \sum_{\bar{\beta}} V_{\alpha,\bar{\beta}} |R:\bar{\beta}\rangle \tag{82}$$

$$|\Psi\rangle = \sum_{\alpha=1}^{\chi} |L:\alpha\rangle |R:\alpha\rangle \tag{83}$$

$$= \sum_{\alpha=1}^{\chi} \sum_{\bar{\alpha}} U_{\alpha,\bar{\alpha}} |L:\bar{\alpha}\rangle \sum_{\bar{\beta}} V_{\alpha,\bar{\beta}} |R:\bar{\beta}\rangle \tag{84}$$

$$= \sum_{\bar{\alpha}} \sum_{\bar{\beta}} (\sum_{\alpha=1}^{\chi} U_{\alpha,\bar{\alpha}} V_{\alpha,\bar{\beta}}) |L:\bar{\alpha}\rangle |R:\bar{\beta}\rangle \tag{85}$$

$$= \sum_{\bar{\alpha}} \sum_{\bar{\beta}} (\sum_{\alpha=1}^{\chi} U^T_{\bar{\alpha},\alpha} V_{\alpha,\bar{\beta}}) |L:\bar{\alpha}\rangle |R:\bar{\beta}\rangle \tag{86}$$

$$= \sum_{\bar{\alpha},\bar{\beta}} (U^T V)_{\bar{\alpha},\bar{\beta}} |L:\bar{\alpha}\rangle |R:\bar{\beta}\rangle \tag{87}$$

We almost have a Schmidt decomposition of the system. Note that since ρ_L is given by $MM^\dagger$, ρ_L will be a $\chi \times \chi$ matrix. By the rank-nullity theorem, χ is the sum of the rank and nullity of ρ_L, therefore the rank of ρ_L is upper bounded by χ.

This is a fascinating result, mainly because it tells us about the maximum of the entanglement entropy of a system. In the case of the AKLT-Hamiltonian, we saw that the ground states for all N could be generated exactly using the MPS representation, for which $\chi = 2$. Thus, the rank of ρ_L, which as we saw before was correlated with the entanglement of the system, can be at most 2. From this, we can infer that the maximum entropy of the AKLT ground state, for any N, is given by $\ln 2$.

$$\max(\mathcal{S}_L) = \ln 2 \tag{88}$$

However, to actually compute the spectrum of ρ_L we need to know the transformation matrices U, V. We will find U by diagonalizing the overlap matrix O, defined by $O_{\alpha,\alpha'} = \langle L:\alpha | L:\alpha' \rangle$. A similar method can be used to determine V. Given an efficient method to calculate O, we have an efficient method to calculate U and V. From the definition of $|L:\alpha\rangle$,

$$O_{\alpha,\alpha'} = \langle L:\alpha | L:\alpha' \rangle \tag{89}$$

$$= \sum_{\{m_0 \ldots m_{N_L-1}\}} (A^{[m_0]} \ldots A^{[m_{N_L-1}]})^*_{\alpha_L,\alpha} (A^{[m_0]} \ldots A^{[m_{N_L-1}]})_{\alpha_L,\alpha'} \tag{90}$$

We also know from the expansion of $|L:\alpha\rangle$ in the $|L:\bar{\alpha}\rangle$ basis that

[10] Note that U and V are $\chi \times \chi$ matrices.

$$O_{\alpha,\alpha'} = \sum_{\bar{\alpha}=1}^{\chi} U^*_{\alpha,\bar{\alpha}} U_{\alpha',\bar{\alpha}} \tag{91}$$

$$= \sum_{\bar{\alpha}=1}^{\chi} (U^\dagger)_{\bar{\alpha},\alpha} U_{\alpha',\bar{\alpha}} \tag{92}$$

$$O = UU^\dagger \tag{93}$$

If O we write all of matrices using real numbers, then O becomes a symmetric real matrix. In addition, O is going to be positive semi-definite, which tells us that all of the eigenvalues of O are non-negative. From Spectral Decomposition, we know that O has a diagonalization $Q^T DQ$, where Q is orthogonal and D is the matrix of eigenvalues. Since all the entries of D are non-negative, we can rewrite D as A^2, where A is the diagonal matrix with the square root of the eigenvalues.

$$O = Q^T DQ = Q^T A^2 Q = Q^T AAQ = Q^T A(Q^T A)^T \tag{94}$$

$$\tag{95}$$

Thus, we arrive at a form for O which is very similar to UU^T. Using the diagonalization of O, we can find U. If we diagonalize the overlap matrices for the $|R:\alpha\rangle$ vectors, we would be able to find V. We can then compute ρ_L simply.

$$\rho_L = U^T VUV^T \tag{96}$$

4.6 Using MPS to Efficiently Compute the Overlap Matrix

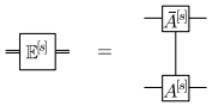


Figure 7: A graphical description of the transfer operator in MPS, diagram from [7]

Suppose that we want to compute the squared norm of a vector $|\Psi\rangle$: $\langle\Psi|\Psi\rangle$. In general, each scalar norm requires $\dim \mathcal{H}$ multiplications, thus the operation scales exponentially with system size N. In this section, we will give a concrete example of how MPS can be used to work around the exponential nature of many-body Hilbert spaces.

Consider a spin chain on $N = 2$ particles. Suppose that the set of matrices at each site is the same. This is the case for the AKLT-Hamiltonian. An arbitrary state of the system has an MPS representation. We can find the squared norm in the MPS representation as follows.

$$|\Psi\rangle = \sum_{\{m_0\ m_1\}} (A^{[m_0]}A^{[m_1]})_{\alpha_L,\alpha_R}\,|m_0\ m_1\rangle \tag{97}$$

$$\langle\Psi|\Psi\rangle = \sum_{\{m_0\ m_1\}}\sum_{\{m_0'\ m_1'\}} (A^{[m_0']}A^{[m_1']})^*_{\alpha_L,\alpha_R}(A^{[m_0]}A^{[m_1]})_{\alpha_L,\alpha_R}\,\langle m_0'\ m_1'|m_0\ m_1\rangle \tag{98}$$

$$= \sum_{\{m_0\ m_1\}} (A^{[m_0]}A^{[m_1]})^*_{\alpha_L,\alpha_R}(A^{[m_0]}A^{[m_1]})_{\alpha_L,\alpha_R} \tag{99}$$

$$= \sum_{m_0}\sum_{m_1}\sum_{i} A^{[m_0]^*}_{\alpha_L,i}\,A^{[m_1]^*}_{i,\alpha_R}\sum_{i'} A^{[m_0]}_{\alpha_L,i'}A^{[m_1]}_{i',\alpha_R} \tag{100}$$

$$= \sum_{i,i'}\left(\sum_{m_0} A^{[m_0]^*}_{\alpha_L,i}\,A^{[m_0]}_{\alpha_L,i'}\right)\left(\sum_{m_1} A^{[m_1]^*}_{i,\alpha_R}\,A^{[m_0]}_{i',\alpha_R}\right) \tag{101}$$

To simplify the expression, we can introduce a tensor called the Transfer Operator, E. The Transfer Operator E is a tensor because it takes four indices to retrieve an element, unlike a matrix which takes two indices to retrieve an element. In the case of the AKLT ground state, each matrix $A^{[m]}$ is $\chi \times \chi$ and $\chi = 2$. Thus, we can write the transfer operator as a $\chi^2 \times \chi^2$ matrix [7].

$$E \equiv \sum_m A^{[m]^*} \otimes A^{[m]} \tag{102}$$

$$E_{(\alpha,\alpha');(\beta,\beta')} = \sum_m A^{[m]^*}_{\alpha,\beta}\,A^{[m]}_{\alpha',\beta'} \tag{103}$$

Consider the operator E^N. If we take the expression for the tensor operator above and raise it to the Nth power, we get

$$E^N = \sum_{m_0}...\sum_{m_{N-1}} A^{[m_0]^*}...A^{[m_{N-1}]^*} \otimes A^{[m_0]}...A^{[m_{N-1}]} \tag{104}$$

$$= \sum_{\{m_0...m_{N-1}\}} A^{[m_0]^*}...A^{[m_{N-1}]^*} \otimes A^{[m_0]}...A^{[m_{N-1}]} \tag{105}$$

$$= \sum_{\{m_0...m_{N-1}\}} (A^{[m_0]}...A^{[m_{N-1}]})^* \otimes A^{[m_0]}...A^{[m_{N-1}]} \tag{106}$$

Using the second defining equation of an element of the transfer matrix, we can infer what an element of E^N would be

$$E^N_{(\alpha,\alpha');(\beta,\beta')} = \sum_{\{m_0...m_{N-1}\}} (A^{[m_0]}...A^{[m_{N-1}]})^*_{\alpha,\beta}(A^{[m_0]}...A^{[m_{N-1}]})_{\alpha',\beta'} \tag{107}$$

If we evaluate the above tensor at $(\alpha_L,\alpha_L);(\alpha_R,\alpha_R)$, we get exactly the norm of an N particle state. Thus, for the case of $N = 2$, $\langle\Psi|\Psi\rangle$ is just $E^2_{(\alpha_L,\alpha_L);(\alpha_R,\alpha_R)}$. This simplifies our notation greatly, while also telling us some useful things. For example, to compute E^N, we just need to do $N-1$ multiplications of a $\chi^2 \times \chi^2$ matrix, as opposed to a d^N multiplications. In fact, we could do it in $\ln N$ matrix multiplications by repeated squaring. Thus, if we calculate E, we might hope to calculate $\langle\Psi|\Psi\rangle$ efficiently. The calculation for E should go like $d\chi^4$, which implies that we can compute $\langle\Psi|\Psi\rangle$ in $O(d\chi^4 \ln N)$. We write a simple code (see the appendix) to generate the matrix elements for E for the AKLT model.

$$E = \frac{1}{3}\begin{pmatrix} 1 & 0 & 0 & 2 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 2 & 0 & 0 & 1 \end{pmatrix} \tag{108}$$

Figure 8: The transfer operator E, generated by the code. The code is provided in the appendix.

Note that while E is not in general symmetric, it is for the AKLT case because for each (α_L, α_R), $\sum_m (A^{[m]}_{\alpha_L,\alpha_R})^2 = \sum_m (A^{[m]}_{\alpha_R,\alpha_L})^2$. For the AKLT model, the ground state matrices seem to have this property, and so E and all its powers are symmetric[11]. The Nth power of E is the matrix of overlaps between the ground state vectors for the N particle AKLT-model.

The eigenvalues of E, which can be determined exactly, are $[1, -1/3, -1/3, -1/3]$. Writing E as a sum of projectors weighted by their eigenvalue λ_i, $\sum_i \lambda_i |\lambda_i\rangle \langle\lambda_i|$, we notice that as N gets big, the projectors with eigenvalues less than one die out and $E^N \to |\lambda_1\rangle \langle\lambda_1|$, where $|\lambda_1\rangle$ is the eigenvector with eigenvalue 1. Thus, for large N,

$$E^N \to \frac{1}{2}\begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \tag{109}$$

Thus as N goes to infinity, the squared norms of the MPS representations of the AKLT ground state go to $1/2$. Thus, finding the norm of one of the AKLT ground state vectors for large N, an operation that would one would think scales exponentially, can be approximated to $1/2$ in constant time and with a high degree of accuracy. For some other system, the operation would amount to taking the Nth power of the largest eigenvalue of the transfer matrix E, an operation that can be done in $\ln N$ time[12].

N	$\langle\Psi\|\Psi\rangle$
10	0.499991532395
20	0.499999999735
40	0.499999999756
80	0.499999999513

Figure 9: A table of the squared norms as N gets big. Note that the choice of ground state vector makes no difference for large N.

In computing E, we have also found a way to efficiently compute the inner products of states that have MPS representations. Referring to elements of the overlap matrix O in equation (90), $O_{\alpha,\alpha'} = E^{N_L}_{(\alpha_L,\alpha_L);(\alpha,\alpha')}$ where E is the transfer matrix for that system. In the previous section, we showed that one could determine U and V, and thus ρ_L, given an efficient way to compute the inner products of $\{|L:\alpha\rangle\}$ and $\{|R:\alpha\rangle\}$. Now that we have found that, we see how MPS allows one to compute in polynomial time various quantities, in this case the entanglement entropy, of systems whose Hilbert spaces grow exponentially with input size N. This is a remarkable and boundlessly useful fact. Thanks to MPS, simulating large many-body quantum systems is no longer impossible.

[11] The powers of a symmetric matrix are symmetric. The proof is straightforward.
[12] Not accounting for the time needed to diagonalize E, which is polynomial in χ.

5 Appendix

5.1 Heisenberg Hamiltonian Exact Diagonalization Code

```
import numpy as np
import numpy.linalg as la
import scipy.misc as sc
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
import sys
import numpy as np
import numpy.linalg as la
import scipy.misc as sc
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt

def getIntegerRepresentation(state) :
   alpha = 0
   N = len(state)
   for i in range(0, N) :
      alpha = alpha + state[i] * (2 ** (N - i - 1))
   return alpha

def getState(alpha, N) :
   state = np.zeros((N, 1))
   for i in range(0, N) :
   pw = 2 ** (N - i - 1)
   if pw <= alpha :
      state[i] = 1.0
      alpha = alpha - pw
   return state

def SziSzj(alpha, N, i, j) :
   result = np.zeros((2, 1))
   result[1] = alpha
   state = getState(alpha, N)
   if state[i] == state[j] :
      result[0] = 1.0 / 4.0
   else :
      result[0] = -1.0 / 4.0
   return result

def SpiSmj(alpha, N, i, j) :
   result = np.zeros((2, 1))
   state = getState(alpha, N)
   if state[i] == 1.0 or state[j] == 0.0 :
      return result
   result[0] = 1.0
   result[1] = alpha + 2**(N - i - 1) -  2**(N - j - 1)
   return result

def innerProd(alpha, beta) :
   if alpha == beta :
      return 1.0
   else :
```

```
52          return 0.0
53
54      #build the basis for N spin-halfs that has total Sz spin szt
55      spinhalfbasis = np.array([])
56
57      def buildBasis(szt, N) :
58          global spinhalfbasis
59          spinhalfbasis = np.array([])
60
61          k = (N + 2 * szt) / 2
62          #begin by placing 1's at all the first positions
63          positions = np.arange(0, k)
64
65          #compute current value of bit vector
66          val = 0.0
67          for i in range(0, len(positions)) :
68              val += 2**(N - positions[i] - 1)
69
70          #add additional basis elements recursively
71          createBasisElements(val, positions, N)
72
73      def advance(alpha, N, i) :
74          return alpha - 2**(N - i - 1) + 2**(N - (i + 1) - 1)
75
76      def createBasisElements(val, positions, N) :
77          global spinhalfbasis
78          if val not in spinhalfbasis[:] :
79              spinhalfbasis = np.append(spinhalfbasis, val)
80              for i in reversed(range(0, len(positions))) :
81                  #create a new basis state by moving a "1" to the right
82                  if i == len(positions) - 1 : #at the rightmost position
83                      if positions[i] < (N - 1) :
84                          #calculate and append the new basis element
85                          newval = advance(val, N, positions[i])
86                          #recursive step
87                          positions[i] += 1
88                          createBasisElements(newval, positions, N)
89                          positions[i] -= 1
90                  else : #not at the rightmost position, no other 1 in front
91                      if positions[i] != (positions[i+1] - 1) :
92                          #calculate and append the new basis element
93                          newval = advance(val, N, positions[i])
94                          #recursive step
95                          positions[i] += 1
96                          createBasisElements(newval, positions, N)
97                          positions[i] -= 1
98
99      #Construct the Heisenberg Hamiltonian for N spin-halfs
100     def buildHamiltonian(N) :
101         global spinhalfbasis
102         np.sort(spinhalfbasis)
103         dim = len(spinhalfbasis)
104         H = np.zeros((dim, dim))
105         for i in range(0, dim) :
106             for j in range(0, dim) :
```

```
107             val = 0.0
108             for k in range(0, N - 1) :
109                result1 = SziSzj(spinhalfbasis[j], N, k, k + 1)
110                result2 = SpiSmj(spinhalfbasis[j], N, k, k + 1)
111                result3 = SpiSmj(spinhalfbasis[j], N, k + 1, k)
112
113                val += result1[0] * innerProd(spinhalfbasis[i], result1[1]) \
114                      + 0.5 * (result2[0] * innerProd(spinhalfbasis[i], result2[1]) \
115                      + result3[0] * innerProd(spinhalfbasis[i], result3[1]))
116
117          H[i][j] = val
118
119       return H
120
121    #Generate the spectrum for H for each possible value of szt
122    N = 12
123    szt_data = np.array([])
124    energy_data = []
125    s = np.array([])
126
127    for i in range(0, N+1) :
128       energy_data.append([])
129
130    for i in range(0, N+1) :
131       #letting i be the number of spin-up qubits, we compute szt
132       szt = 0.5 * i - 0.5 * (N - i)
133       #build the basis and put it into var "spinhalfbasis"
134       buildBasis(szt, N)
135       H = buildHamiltonian(N)
136       #determine the eigenvalues
137       w, v = la.eig(H)
138
139       szt_data = np.append(szt_data, szt)
140       energy_data[i] = w
141
142       if szt >= 0 :
143          s = np.append(s, szt)
144
145    x = np.array([])
146    y = np.array([])
147
148    for i in range(0, len(szt_data)) :
149       for j in range(0, len(energy_data[i])) :
150          x = np.append(x, szt_data[i])
151          y = np.append(y, energy_data[i][j])
152
153    #Fit a parabola to the graph
154    para_a = (max(y) - min(y)) / (max(x)**2)
155    para_c = min(y)
156    def parabola(x) :
157    return para_a * x**2 + para_c
158
159    print('a = %f \nc = %f' % (para_a, para_c))
160
161    plt.figure(1)
```

```
plt.plot(x, parabola(x))
plt.plot(x, y, 'r+')
#plt.plot(z, w, 'bo')
plt.xlim(min(x) - .25, max(x) + .25)
plt.ylim(min(y) - .25, max(y) + .25)
plt.xlabel('Total spin projection on z axis (units of $\hbar$)')
plt.ylabel('Energy eigenvalues')
plt.title('Szt vs Energy spectrum N = %d' % N)
plt.grid(which = 'both')

plt.show()
```

5.2 AKLT Hamiltonian Exact Diagonalization Code

```
import numpy as np
import numpy.linalg as la
import scipy.misc as sc
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
import sys

#get N
N = int(sys.argv[1])

#Calculate the dimension of the Hilbert Space
d = 3**N

#build the complete basis for N spin ones.
def buildBasis() :
   basis = np.zeros((d, N))
   sztvals = np.zeros(d)
   sztvals[0] = -N
   for i in range(1, d) :
      added = 0
      for j in reversed(range(0, N)) :
         if added == 0 :
            if basis[i - 1][j] < 2 :
               basis[i][j] = basis[i - 1][j] + 1
               added = 1
         else :
            basis[i][j] = 0
         else :
            basis[i][j] = basis[i - 1][j]
         sztvals[i] = sztvals[i] + basis[i][j] - 1

   return basis, sztvals

basis, sztvals = buildBasis()

#Find the sub-basis for a given value of SzT
szt = 1.0

def findFixedSztSubspace(szt) :
   subspace = np.array([])
   for alpha in range(0, d) :
      if sztvals[alpha] == szt :
```

```
43            subspace = np.append(subspace, alpha)
44
45       return subspace
46
47    #Define the relevant operators
48
49    def SziSzj(alpha, i, j) :
50       result = np.zeros(2)
51       spini = basis[alpha][i]
52       spinj = basis[alpha][j]
53
54       result[0] = (spini - 1) * (spinj - 1)
55       result[1] = alpha
56
57       return result
58
59    def SpiSmj(alpha, i, j) :
60       result = np.zeros(2)
61       spini = basis[alpha][i]
62       spinj = basis[alpha][j]
63
64       if spini == 2 or spinj == 0 :
65          return result
66
67       result[0] = 2
68
69       result[1] = alpha + 3 ** (N - i - 1) - 3 ** (N - j - 1)
70
71       return result
72
73    def innerProd(alpha, beta) :
74       if alpha == beta :
75          return 1.0
76       else :
77          return 0.0
78
79
80    #Pass each operator the state c|a>  as [c, a]
81    #Returns <beta| SiDotSj |alpha>
82    def SiDotSj(alpha, i, j, beta) :
83       term1 = SziSzj(alpha, i, j)
84       term2 = SpiSmj(alpha, i , j)
85       term3 = SpiSmj(alpha, j, i)
86
87       sum = term1[0] * innerProd(beta, term1[1]) \
88          + 0.5 * (term2[0] * innerProd(beta, term2[1]) + term3[0] \
89                * innerProd(beta, term3[1]))
90
91       return sum
92
93    #Returns <beta| (SiDotSj)^2 |alpha>
94    def SiDotSjSquared(alpha, i, j, beta) :
95       SiSjOnAlpha = np.zeros((3, 2))
96       SiSjOnBeta = np.zeros((3, 2))
97
```

```
98       SiSjOnAlpha[0] = SziSzj(alpha, i, j)
99       term = SpiSmj(alpha, i , j)
100      term[0] = 0.5 * term[0]
101      SiSjOnAlpha[1] = term
102      term = SpiSmj(alpha, j , i)
103      term[0] = 0.5 * term[0]
104      SiSjOnAlpha[2] = term
105
106      SiSjOnBeta[0] = SziSzj(beta, i, j)
107      term = SpiSmj(beta, i , j)
108      term[0] = 0.5 * term[0]
109      SiSjOnBeta[1] = term
110      term = SpiSmj(beta, j , i)
111      term[0] = 0.5 * term[0]
112      SiSjOnBeta[2] = term
113
114      sum = 0
115
116      for i in range(0, 3) :
117         for j in range(0, 3) :
118            sum = sum + SiSjOnAlpha[j][0] * SiSjOnBeta[i][0] \
119                    * innerProd(SiSjOnBeta[i][1], SiSjOnAlpha[j][1])
120
121      return sum
122
123   #Construct the Heisenberg Hamiltonian for N spins
124   def buildHamiltonian() :
125      H = np.zeros((d, d))
126      for i in range(0, d) :
127         for j in range(0, d) :
128            val = 0.0
129            for k in range(0, N - 1) :
130               val = val + SiDotSj(j, k, k + 1, i) \
131                        + 1.0/3.0 * SiDotSjSquared(j, k, k + 1, i)
132
133            H[i][j] = val
134
135      return H
136
137   #Construct the Heisenberg Hamiltonian for N spins
138   def buildBlockHamiltonian(szt) :
139      subbasis = findFixedSztSubspace(szt)
140      dim = len(subbasis)
141      H = np.zeros((dim, dim))
142      for i in range(0, dim) :
143         for j in range(0, dim) :
144            val = 0.0
145            for k in range(0, N - 1) :
146               val = val + SiDotSj(subbasis[j], k, k + 1, subbasis[i]) \
147                  + 1.0/3.0 * SiDotSjSquared(subbasis[j], k, k + 1, subbasis[i])
148
149            H[i][j] = val
150
151      return H
```

5.3 AKLT MPS Ground State Generation Code

```
import numpy as np
import numpy.linalg as la
import scipy.misc as sc
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
import sys

#Store the program parameters, N and Szt
N = int(sys.argv[2])
Szt = int(sys.argv[3])
al = int(sys.argv[4])
ar = int(sys.argv[5])

#Store the matrices
file = open(sys.argv[1])
lines = file.readlines()

matrices = []

#First line tells us the dimension d, remove it after recording d.
d = int(lines[0])

lines[0:1] = []

tempMatrix = np.zeros((d, d))

for i in range(0, len(lines)) :
   splitline = lines[i].split()
   for j in range(0, d) :
      place = int(i % d)
      tempMatrix[place][j] = float(splitline[j])

   if (i + 1) % d == 0 :
      matrices.append(tempMatrix)
      tempMatrix = np.zeros((d, d))

print(matrices)

dim = 3 ** N
vector = np.zeros(dim)
state = np.zeros(N)
for i in range(0, dim) :
   prod = np.eye(2)
   for j in range(0, N) :
      prod = np.dot(prod, matrices[int(state[j])])
   vector[i] = prod[al][ar]
   added = 0
   # generate next state
   for k in reversed(range(0, N)) :
      if added == 1 :
         break
      if state[k] < 2 :
         state[k] = state[k] + 1
```

```
54              added = 1
55          else :
56              state[k] = 0
57
58  file = open('MPS_data/gs_eigenvectors.txt', 'a')
59  file.write('N=%d \nal = %d ar = %d \n' % (N, al, ar))
60  for val in vector :
61      file.write(str(val) + ' ')
62  file.write('\n')
```

5.4 AKLT Squared Norm Finding Code

```
1   import numpy as np
2   import numpy.linalg as la
3   import scipy.misc as sc
4   import matplotlib.mlab as mlab
5   import matplotlib.pyplot as plt
6   import sys
7
8   #Store the program parameters
9   N = int(sys.argv[2])
10  al = int(sys.argv[3])
11  ar = int(sys.argv[4])
12
13  #Store the matrices
14  file = open(sys.argv[1])
15  lines = file.readlines()
16
17  matrices = []
18
19  #First line tells us the dimension chi, remove it after recording chi.
20  chi = int(lines[0])
21
22  lines[0:1] = []
23
24  tempMatrix = np.zeros((chi, chi))
25
26  for i in range(0, len(lines)) :
27      splitline = lines[i].split()
28      for j in range(0, chi) :
29          place = int(i % chi)
30          tempMatrix[place][j] = float(splitline[j])
31
32      if (i + 1) % chi == 0 :
33          matrices.append(tempMatrix)
34          tempMatrix = np.zeros((chi, chi))
35
36
37  #Calculate E, E^N, and print out the norm
38  d = chi * chi
39  E = np.zeros((d, d))
40
41  for a1 in range(0, chi) :
42      for a2 in range(0, chi) :
43          for b1 in range(0, chi) :
44              for b2 in range(0, chi) :
```

```
45                sum = 0.0
46                for m in range(0, len(matrices)) :
47                   sum = sum + matrices[m][a1][b1] * matrices[m][a2][b2]
48                E[a2 + chi * a1][b2 + chi * b1] = sum
49
50    w1, v1 = la.eig(E)
51    EN = la.matrix_power(E, N)
52    print(str(N) + ' & ' + str(EN[al + chi * al][ar + chi * ar]) +
53          ' & ' + str(abs(max(w1) ** N - 2 * EN[al + chi * al][ar + chi * ar])))
```

References

[1] Affleck, I. "Quantum Spin Chains and the Haldane Gap." *Journal of Physics: Condensed Matter J. Phys.: Condens. Matter* 1.19 (1989): 3047-072. Web. 3 May 2016.

[2] Ayeni, Babatunde M. "Simulation of Braiding Anyons Using Matrix Product States." *ArXiv*. N.p., 20 Jan. 2016. Web. 2 May 2016.

[3] Eichler, C. "Exploring Interacting Quantum Many-Body Systems by Experimentally Creating Continuous Matrix Product States in Superconducting Circuits." *ArXiv*. N.p., 16 Dec. 2016. Web. 02 May 2016.

[4] Hastings, M. B. "Entropy and Entanglement in Quantum Ground States." *ArXiv*. N.p., 3 Jan. 2007. Web. 02 May 2016.

[5] Nielsen, Michael A., and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge: Cambridge UP, 2000. Print.

[6] Regnault, Nicolas. "Entanglement Spectroscopy and Its Application to the Quantum Hall Effects." *ArXiv* (2015): n. pag. *ArXiv*. 26 Oct. 2015. Web. 2 May 2016.

[7] Schuch, Norbert. "Condensed Matter Applications of Entanglement Theory." *ArXiv*. N.p., 24 June 2013. Web. 02 May 2016.

[8] Schollwock, Urlich. "The Density-matrix Renormalization Group in the Age of Matrix Product States." *ArXiv*. N.p., 3 Jan. 2011. Web. 02 May 2016.