Spring 2024 Rotation Report

Haylen Gerhard

October 24, 2024

Contents

1 Introduction

My Spring rotation was with Dr. Wladimir Benalcazar. It was very enjoyable and helped me to gain a better understanding of the density matrix renormalization group (DMRG) method that is prevalent in condensed-matter physics.

The bulk of our work has been working through sections 4.1 and 4.2.1 of [1] to get a good theoretical understanding of the machinery of matrix product states (MPSs) and what it means to ”do DMRG.” In the last few weeks I spent some time trying to recreate Figure 1a from [2], but ultimately I was unable to match their figure in the time span of the rotation.

2 Matrix Product States

Here, we present a crash course on matrix product states (MPSs) and their relation to the density matrix renormalization group (DMRG) approach that has been very fruitful for the analysis of 1-D and smaller 2-D quantum systems.

A short way to say why DMRG is so useful is that it assumes there are a certain number of (usually local) particles in the system that a given particle can entangle with. So, all states where there is entanglement between a two particles that are assumed to be not entangled are null, reducing the size of our Hilbert space immensely.

The details of this section are from or heavily informed by [1, 3]. Additionally, useful graphical representations can be found in [1].

2.1 A Preliminary Example

To prepare for the sections ahead, we look at a two-body spin system as a MPS. Traditionally, we have the general state

$$|\psi\rangle = a\,|\uparrow\uparrow\rangle + b\,|\uparrow\downarrow\rangle + c\,|\downarrow\uparrow\rangle + d\,|\downarrow\downarrow\rangle \tag{1}$$

So, for an arbitrary state of two spins, we have 4 parameters to completely define the state. There is no fundamental reason for organizing the coefficients this way. We could just as easily represent the state (as we often do!) with a column vector

$$|\psi\rangle = \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix} \tag{2}$$

or a row vector

$$|\psi\rangle = \begin{pmatrix} a & b & c & d \end{pmatrix} \tag{3}$$

or even a matrix!

$$|\psi\rangle = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \tag{4}$$

In each case, we need only define the right basis and operations to have machinery that is equivalent to the representation in Eq. (1) we know and love.[1] We could also choose a different basis for the matrix representation, such as the singlet and triplet spin states, each represented as matrices. This would yield

$$|\psi\rangle = \sum_i a_i M^i, \quad i \in \{s, 0, -, +\} \tag{5}$$

where i cycles through the singlet and triplet state matrices in the same basis as before.

But, as is in the title, we still are only representing the state as a single matrix, where we want to represent our states by a product of matrices. Let's say we wanted to represent the singlet state as a product of matrices. We could do something like

$$|\psi\rangle = AB = \begin{pmatrix} 0 & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{6}$$

where we have introduced a "virtual" degree of freedom[2] to encode the connection between the two spins. In tensor/Einstein notation, this is a little more apparent,

$$|\psi\rangle = A_{\sigma_1 a_1} B_{a_1 \sigma_2} |\sigma_1 \sigma_2\rangle \tag{7}$$

where σ_i represents the physical degrees of freedom and a_1 is the virtual bond/DOF that connects them.

[1]For the matrix above, we were in the basis of $\{|\uparrow\rangle, |\downarrow\rangle\}$.

[2]We will refer to these virtual legs of our tensors as either virtual or bond DOFs. In the literature you will often see "bond dimension" discussed, which is just the number of singular values we choose to keep

This representation likely seems superfluous at this point, but it will be useful in the sections to come.

2.2 Background

2.2.1 Singular Value Decomposition

Here, we analyze the singular value decomposition (SVD) of matrices. For an arbitrary matrix M of dimension $N_A \times N_B$, we can decompose it as

$$M = USV^\dagger \tag{8}$$

where

- U is $(N_A \times \min(N_A, N_B))$ and has orthonormal columns, which we refer to as the left singular vectors
- S is $(\min(N_A, N_B) \times \min(N_A, N_B))$ and is diagonal with the entries s_i being the singular values. We assume a descending order for the singular values.
- $V^\dagger$ is $(\min(N_A, N_B) \times N_B)$ and has orthonormal rows, which we refer to as the right singular vectors.
- For both U and $V^\dagger$, we have $X^\dagger X = I$. If $N_A \leq N_B$, then also $UU^\dagger = I$ and vice versa for V (unitarity condition).

In tensor notation, we have

$$M_{\sigma_1\sigma_2} = U_{\sigma_1 a} S_{aa} V_{a\sigma_2} \tag{9}$$

2.2.2 Schmidt Decomposition and Entanglement Entropy

Here, we use the SVD decomposition to form a Schmidt decomposition of our system and relate it to the entanglement entropy of our system. We begin with the pure state

$$|\psi\rangle = \sum_{ij} \Psi_{ij} |i\rangle_A |j\rangle_B \tag{10}$$

where we can think of $i = \sigma_1$ and $j = \sigma_2$ for a system of 2 spins, but more generally they are the pieces from some bipartition of our system into parts A and B. We now take the SVD of Ψ_{ij}, yielding[3]

[3]On notation, we will often have tensors that look like V^*_{ja} or $V^\dagger_{aj}$. The difference here is that the asterisk denotes that there is some form of conjugation/transposing that needs to be done, while the dagger denotes that the conjugation/transposing has been done. A more informative notation might be something like $(V_{ja})^*$ or $(V^\dagger)_{aj}$, but we are following the notation of [1]

$$|\psi\rangle = \sum_{ija} U_{ia} S_{aa} V^*_{ja} |i\rangle_A |j\rangle_B \tag{11}$$

We can separate the summation as

$$|\psi\rangle = \sum_a S_{aa} \left(\sum_i U_{ia} |i\rangle_A \right) \left(\sum_j V^*_{ja} |j\rangle_B \right) \tag{12}$$

and then rename the vectors in parentheses

$$|\psi\rangle = \sum_a s_a |a\rangle_A |a\rangle_B \tag{13}$$

From here, we can see that the density operator is easily available in the diagonal form

$$\rho = |\psi\rangle \langle\psi| = \sum_a s_a^2 |a\rangle_A \langle a|_A |a\rangle_B \langle a|_B \tag{14}$$

and the entanglement entropy of our system follows as

$$S = -\sum_i s_i^2 \log s_i^2; \quad s_i \neq 0 \tag{15}$$

2.2.3 Trace and Tensor Product Identities

We will need some identities of the trace operation and tensor (Kronecker) product to work through the steps in Eq. (57), so we list them here.

- The first idenity used is that the product of two traces is the trace of a tensor product, i.e.,

$$\mathrm{tr}(A)\,\mathrm{tr}(B) = \mathrm{tr}(A \otimes B) \tag{16}$$

- The second identity is that the tensor product of two products is the product of two tensor products, i.e.,

$$(AB) \otimes (CD) = (A \otimes C)(B \otimes D) \tag{17}$$

  This is a direct manifestation of the bilinearity of the tensor product.

- The last identity we will need is that the trace of a matrix is the sum of the eigenvalues of that matrix, i.e.,

$$\operatorname{tr} A = \sum_i \lambda_i \tag{18}$$

This identity can be seen from the property that the trace of a product is the same as the trace of cyclic permutations of that product, i.e.,

$$\operatorname{tr}(ABC) = \operatorname{tr}(BCA) = \operatorname{tr}(CAB) \tag{19}$$

combined with the fact that we can always find a similarity transformation that diagonalizes a matrix, i.e.,

$$\operatorname{tr} A = \operatorname{tr}(SAS^{-1}) = \operatorname{tr} A_{\text{diag}} \tag{20}$$

2.3 Arbitrary Matrix Product States

2.3.1 How to Arrive at an MPS Form

In this section, we begin the discussion of general matrix product states, where we assume a full Hilbert space of the form

$$\mathcal{H} = \bigotimes_i \mathcal{H}_i \tag{21}$$

where $\mathcal{H}_i$ are d dimensional Hilbert spaces of the individual lattice sites[4] of our system. We then have the wave function

$$|\psi\rangle = \sum_{\{\sigma\}} c_{\sigma_1\sigma_2...\sigma_L} |\sigma_1\sigma_2...\sigma_L\rangle \tag{22}$$

Since we assume our basis vectors $|\sigma_1\sigma_2...\sigma_L\rangle$ to be orthonormal, we can just think about the coefficients alone. We start by organizing them as a matrix[5]

$$c_{\sigma_1\sigma_2...\sigma_L} = \Psi_{\sigma_1,(\sigma_2...\sigma_L)} \tag{23}$$

and then we take the SVD of the matrix $\Psi_{\sigma_1,(\sigma_2...\sigma_L)}$, yielding

$$c_{\sigma_1\sigma_2...\sigma_L} = \sum_{a_1} U_{\sigma_1,a_1} S_{a_1,a_1} V^\dagger_{a_1,(\sigma_2...\sigma_L)} \tag{24}$$

[4]These need not be physical lattice sites, i.e., we could represent an N particle gas with N "lattice sites." Regardless, the machinery we develop here will benefit from this representation.

[5]Here, the legs of a given tensor are separated by commas, and leg indices like $(\sigma_2...\sigma_L)$ denote all d^{L-1} combinations of the σ_i

We now rename[6] $U_{\sigma_1,a_1} = A^{\sigma_1}_{1,a_1}$ and collect the other two matrices as $S_{a_1,a_1} V^{\dagger}_{a_1,(\sigma_2 \ldots \sigma_L)} = \Psi_{(a_1\sigma_2),(\sigma_3 \ldots \sigma_L)}$ where we have essentially just rearranged our indices. All together, this yields

$$c_{\sigma_1\sigma_2\ldots\sigma_L} = \sum_{a_1} A^{\sigma_1}_{1,a_1} \Psi_{(a_1\sigma_2),(\sigma_3\ldots\sigma_L)} \tag{25}$$

Once again, we take the SVD of our Ψ tensor, yielding

$$c_{\sigma_1\sigma_2\ldots\sigma_L} = \sum_{a_1,a_2} A^{\sigma_1}_{1,a_1} U_{(a_1\sigma_2),a_2} S_{a_2,a_2} V^{\dagger}_{a_2,(\sigma_3\ldots\sigma_L)} \tag{26}$$

then we move the physical index on our U matrix upstairs and we collect our S and $V^{\dagger}$ matrices into a new Ψ, yielding

$$c_{\sigma_1\sigma_2\ldots\sigma_L} = \sum_{a_1,a_2} A^{\sigma_1}_{1,a_1} A^{\sigma_2}_{a_1,a_2} \Psi_{(a_2\sigma_3),(\sigma_4\ldots\sigma_L)} \tag{27}$$

From here, we can see a pattern beginning to emerge where we will always have our A matrices of the form $A^{\sigma_l}_{a_l,a_{l+1}}$[7] where $a_0 = a_{L+1} = 1$. Enumerating this process through all of the physical indices, we get

$$c_{\sigma_1\sigma_2\ldots\sigma_L} = \sum_{\{a\}} A^{\sigma_1}_{1,a_1} A^{\sigma_2}_{a_1,a_2} \ldots A^{\sigma_{L-1}}_{a_{L-2},a_{L-1}} A^{\sigma_L}_{a_{L-1},1} \tag{28}$$

Bringing back in our basis vectors to represent the wave function and not just the coefficients, we have

$$|\psi\rangle = \sum_{\{\sigma\}} \sum_{\{a\}} A^{\sigma_1}_{1,a_1} A^{\sigma_2}_{a_1,a_2} \ldots A^{\sigma_{L-1}}_{a_{L-2},a_{L-1}} A^{\sigma_L}_{a_{L-1},1} |\sigma_1\sigma_2\ldots\sigma_L\rangle \tag{29}$$

We have now represented our wave function as a MPS! Specifically, this form is referred to as *left-canonical*.

2.3.2 Why "Left-Canonical"?

We call the MPS form presented in Eq. (29) "left-canonical" because of how we would go about taking the inner product (or overlap) of an MPS wave function(s). Recall the form of the A^{σ_i} matrices

$$U_{(a_{i-1}\sigma_i),a_i} = A^{\sigma_i}_{a_{i-1},a_i} \tag{30}$$

[6] When working with tensors, the same degree of freedom appearing as an upper or lower index is connected by a metric tensor, i.e, $M_j = \sum_i g_{i,j} M^i$. Here, we make no distinction between upper and lower indices (which is to say that $g_{i,j} = \delta_{i,j}$)

[7] Often, we will omit the a_j legs of our A^{σ_i} matrices and assume they follow this form

where although U is not always unitary, $U^\dagger U = I$ is always satisfied. Matrices that satisfy this condition are referred to as "left-normalized."[8] Since our A matrices are just reorganized U matrices, we should be able to find a way to represent this left-normality for our A matrices. Formally, we have

$$I = U^\dagger U = \sum_{\sigma_i, a_{i-1}} U^*_{(a_{i-1}\sigma_i),a'_i} U_{(a_{i-1}\sigma_i),a_i} = \sum_{\sigma_i, a_{i-1},} U^\dagger_{a'_i,(a_{i-1}\sigma_i)} U_{(a_{i-1}\sigma_i),a_i} \tag{31}$$

then moving to the A matrix representation yields

$$I = \sum_{\sigma_i, a_{i-1},} U^\dagger_{a'_i,(a_{i-1}\sigma_i)} U_{(a_{i-1}\sigma_i),a_i} = \sum_{\sigma_i, a_{i-1},} A^{\sigma_i \dagger}_{a'_i,a_{i-1}} A^{\sigma_i}_{a_{i-1},a_i} = \sum_{\sigma_i} A^{\sigma_i \dagger} A^{\sigma_i} = I \tag{32}$$

We now see that the A^{σ_i} matrices are indeed left-normalized[9] under the condition $\sum_{\sigma_i} A^{\sigma_i \dagger} A^{\sigma_i} = I$.

2.3.3 Other Canonical Forms

In the previous section, we discussed the notion of left-canonical MPSs and left-normalized matrices. A natural question from here on is "what are the other canonical MPS forms?" Indeed, one could have just as easily isolated the $V^\dagger$ matrices from the SVD of coefficients, e.g.,

$$\begin{aligned}
c_{\sigma_1\sigma_2\ldots\sigma_L} &= \Psi_{(\sigma_1\ldots\sigma_{L-1}),\sigma_L} \\
&= U_{(\sigma_1\ldots\sigma_{L-1}),a_{L-1}} S_{a_{L-1},a_{L-1}} V^\dagger_{a_{L-1},\sigma_L} \\
&= U_{(\sigma_1\ldots\sigma_{L-1}),a_{L-1}} S_{a_{L-1},a_{L-1}} B^{\sigma_L}_{a_{L-1},1} \\
c_{\sigma_1\sigma_2\ldots\sigma_L} &= \Psi_{(\sigma_1\ldots\sigma_{L-2}),(\sigma_{L-1}a_{L-1})} B^{\sigma_L}_{a_{L-1},1}
\end{aligned}$$

Where we now have these B^{σ_i} matrices that are "right-normalized," i.e.,[10,11] $\sum\limits_{\sigma_i} B^{\sigma_i} B^{\sigma_i \dagger} = I$. Enumeration of this process leads us to what we call the "right-canonical" MPS form

$$|\psi\rangle = \sum_{\{\sigma\}} \sum_{\{a\}} B^{\sigma_1}_{1,a_1} B^{\sigma_2}_{a_1,a_2} \ldots B^{\sigma_{L-1}}_{a_{L-2},a_{L-1}} B^{\sigma_L}_{a_{L-1},1} |\sigma_1\sigma_2\ldots\sigma_L\rangle \tag{33}$$

Additionally, one could imagine doing a combination of right/left decompositions, where we left-decompose up to the site ℓ, and right-decompose down to the site $\ell+1$. The initial left-decomposition to site ℓ would result in

[8] What condition would a right-normalized matrix satisfy?

[9] There is one set of A^{σ_i} matrices that is not necessarily left-normalized. Which one is it? Why?

[10] If this condition is not clear, review the properties of the $V^\dagger$ matrices that come from the SVD of a matrix and our previous discussion of how the A^{σ_i} matrices.

[11] Just like with the left-canonical form, there is one set of B^{σ_i} matrices that are not necessarily right-normalized. Which one is it?

$$c_{\sigma_1\sigma_2...\sigma_L} = \sum_{a_\ell}(A^{\sigma_1}...A^{\sigma_\ell})_{1,a_\ell} S_{a_\ell,a_\ell} V^\dagger_{a_\ell,(\sigma_{\ell+1}...\sigma_L)} \tag{34}$$

we then reshape $V^\dagger_{a_\ell,(\sigma_{\ell+1}...\sigma_L)} = \Psi_{(a_\ell\sigma_{\ell+1}...\sigma_{L-1}),\sigma_L}$ and decompose it from the right, yielding

$$c_{\sigma_1\sigma_2...\sigma_L} = \sum_{a_\ell}(A^{\sigma_1}...A^{\sigma_\ell})_{1,a_\ell} S_{a_\ell,a_\ell}(B^{\sigma_{\ell+1}}...B^{\sigma_L})_{a_\ell,1} \tag{35}$$

where $B^{\sigma_{\ell+1}}_{a_\ell,a_{\ell+1}} = U_{(a_\ell\sigma_{\ell+1}),a_{\ell+1}} S_{a_{\ell+1},a_{\ell+1}}$. Bringing in the basis vectors, we have the "mixed-canonical" MPS

$$|\psi\rangle = \sum_{\{\sigma\}}\sum_{a_\ell}(A^{\sigma_1}...A^{\sigma_\ell})_{1,a_\ell} S_{a_\ell,a_\ell}(B^{\sigma_{\ell+1}}...B^{\sigma_L})_{a_\ell,1} |\sigma_1\sigma_2...\sigma_L\rangle \tag{36}$$

This form is useful for a plethora of reasons, one of which is that we have put the state vector into Schmidt form. We can see this by rearranging the summation over physical indices as

$$|\psi\rangle = \sum_{a_\ell} S_{a_\ell,a_\ell}\left[\sum_{\sigma_1...\sigma_\ell}(A^{\sigma_1}...A^{\sigma_\ell})_{1,a_\ell}|\sigma_1...\sigma_\ell\rangle\right]\left[\sum_{\sigma_{\ell+1}...\sigma_L}(B^{\sigma_{\ell+1}}...B^{\sigma_L})_{a_\ell,1}|\sigma_{\ell+1}...\sigma_L\rangle\right] \tag{37}$$

and then introduce the vectors

$$\begin{aligned} |a_\ell\rangle_A &= \sum_{\sigma_1...\sigma_\ell}(A^{\sigma_1}...A^{\sigma_\ell})_{1,a_\ell}|\sigma_1...\sigma_\ell\rangle \\ |a_\ell\rangle_B &= \sum_{\sigma_{\ell+1}...\sigma_L}(B^{\sigma_{\ell+1}}...B^{\sigma_L})_{a_\ell,1}|\sigma_{\ell+1}...\sigma_L\rangle \end{aligned} \tag{38}$$

resulting in

$$|\psi\rangle = \sum_{a_\ell} s_{a_\ell}|a_\ell\rangle_A|a_\ell\rangle_B \tag{39}$$

which is the same as the form in Eq. (13). We then by extension have easy access to all sorts of things via the (reduced) density operator.

Additionally, we now have an easy way to "scan" our MPS. For example, if we want the singular values associated with $a_{\ell+1}$ instead of a_ℓ, we can just absorb the singular values into $|a_\ell\rangle_B$, contract all of our B^σ matrices to get back to the form of Eq. (34), make a new $A^{\sigma_{\ell+1}}$ matrix and then decompose the right side into B^σ matrices again, yielding

$$|\psi\rangle = \sum_{a_{\ell+1}} s_{a_{\ell+1}} |a_{\ell+1}\rangle_A |a_{\ell+1}\rangle_B \tag{40}$$

This is, to a simple approximation, the process that DMRG algorithms use to find the ground state. They are essentially a variational algorithm over the singular values of different MPS representations of our state. Of course, the machinery behind something like TenPy [1] is much more complex, but that's the basic idea.

2.4 Advantages of MPS Representation

At this point, it is fair to feel that we are just playing a rearranging game with our coefficients. In this section, we look at some advantages to the MPS representation we developed previously.

2.4.1 Truncation of The Hilbert Space

If one tracks the dimension of our virtual legs, a_i, we see that it grows exponentially to $d^{L/2}$ around $a_{L/2}$ and then goes back down. This exponential growth is also evident in the traditional view of wave functions as we have d^L coefficients for a system with L lattice sites.

Fortunately, in the mixed-canonical view we now have a way to truncate/trim the irrelevant spaces of coefficients. Recall that when calculating the entanglement entropy of a given bipartition, as in Eq. (15), we discarded the singular values that were zero. So, if we have a system where most of the singular values are zero, we can truncate them with no loss of information about our system. Formally, the number of non-zero singular values represent the number of pure states[12] that are required to construct our wave function, which gives a rough measurement of how entangled our system is.[13] Of course, anytime we truncate singular values,[14] we are going to introduce some error into our system. Thus, the question arises, how much can we truncate? It can be shown (see [4]) that the error caused by truncation is

$$\| |\psi\rangle - |\psi_{\text{trunc}}\rangle \|^2 \leq 2 \sum_i s_i^2 \tag{41}$$

where s_i are the singular values that we have truncated.[15] So, we can safely truncate singular values to whatever accuracy we desire. From here on out, we assume all matrices in the bulk to be of

[12]By pure state, what we mean is a state made from a single tensor product of the left and right Hilbert spaces, i.e., $|\psi\rangle = |\psi\rangle_L \otimes |\psi\rangle_R$, where $|\psi\rangle_L$ and $|\psi\rangle_R$ need not be pure states on their own as we are calculating the entanglment between our subsystems, not the entanglement in our subsystems themselves. More than one non-zero singular value indicates that we cannot represent the wave function with a single tensor product of $|\psi\rangle_L$ and $|\psi\rangle_R$, i.e, it is a mixed state.

[13]For a bipartition with one non-zero singular value, $s^2 = 1$, what is the entanglement entropy? What about a system (of size L) with L singular values $s_i^2 = \frac{1}{L}$. Is there anything special about these two cases?

[14]i.e., project into a subspace of possible states

[15]Note here that we truncate at each site individually, so there are $L-1$ (L for periodic/infinite boundary conditions) sets of singular values that we truncate

size $D \times D$ where D is the number of singular values we truncate to.

2.4.2 Efficient Inner Products

We will now step away from the SVD view of MPSs, and consider two general (arbitrary) MPSs

$$
\begin{aligned}
|\psi\rangle &= M^{\sigma_1} M^{\sigma_2} ... M^{\sigma_{L-1}} M^{\sigma_L} |\sigma_1 ... \sigma_L\rangle \\
|\phi\rangle &= \tilde{M}^{\sigma'_1} \tilde{M}^{\sigma'_2} ... \tilde{M}^{\sigma'_{L-1}} \tilde{M}^{\sigma'_L} |\sigma'_1 ... \sigma'_L\rangle
\end{aligned} \tag{42}
$$

If we want to take find an overlap of two states, we have

$$
\langle\phi|\psi\rangle = \sum_{\{\sigma\},\{\sigma'\}} (\tilde{M}^{\sigma'_L \dagger} \tilde{M}^{\sigma'_{L-1} \dagger} ... \tilde{M}^{\sigma'_2 \dagger} \tilde{M}^{\sigma'_1 \dagger})(M^{\sigma_1} M^{\sigma_2} ... M^{\sigma_{L-1}} M^{\sigma_L}) \langle\sigma'_1 ... \sigma'_L|\sigma_1 ... \sigma_L\rangle \tag{43}
$$

where we get a $\delta_{\sigma,\sigma'}$ from the orthonormality of our Hilbert space, yielding

$$
\langle\phi|\psi\rangle = \sum_{\{\sigma\}} (\tilde{M}^{\sigma_L \dagger} \tilde{M}^{\sigma_{L-1} \dagger} ... \tilde{M}^{\sigma_2 \dagger} \tilde{M}^{\sigma_1 \dagger})(M^{\sigma_1} M^{\sigma_2} ... M^{\sigma_{L-1}} M^{\sigma_L}) \tag{44}
$$

From here we just need to sum the physical and (implied) virtual DOFs. A straightforward/naive way of perform this sum would be to contract each state separately[16] and then contract along the physical DOFs. However, after contracting the virtual DOFs, we are left with

$$
\langle\phi|\psi\rangle = \sum_{\{\sigma\}} \tilde{c}^*_{\sigma_1 ... \sigma_L} c_{\sigma_1 ... \sigma_L} \tag{45}
$$

Here, we have recovered our original inner product form. However, the number of terms to sum over is now d^L, resulting in a complexity that scales exponentially with system size.

A efficient approach would be to sum over the matrices corresponding to the sites.[17]. We start by reorganizing our summations and matrices as

$$
\langle\phi|\psi\rangle = \sum_{\sigma_L} \tilde{M}^{\sigma_L \dagger} \left(... \left(\sum_{\sigma_2} \tilde{M}^{\sigma_2 \dagger} \left(\sum_{\sigma_1} \tilde{M}^{\sigma_1 \dagger} M^{\sigma_1} \right) M^{\sigma_2} \right) ... \right) M^{\sigma_L} \tag{46}
$$

Now, let's think about the complexity involved in the calculation of the innermost sum. We have d copies of matrix-matrix-multiplication and then we sum them together. Calling the matrix leftover from the σ_1 summation C_1, we now have

$$
\langle\phi|\psi\rangle = \sum_{\sigma_L} \tilde{M}^{\sigma_L \dagger} \left(... \left(\sum_{\sigma_2} \tilde{M}^{\sigma_2 \dagger} C_1 M^{\sigma_2} \right) ... \right) M^{\sigma_L} \tag{47}
$$

[16]i.e., sum the virtual legs first, resulting in $2(L-1)$ matrix multiplications or a complexity of $\mathcal{O}(L$

[17]i.e., to alternate sums over physical and virtual degrees of freedom

where now for the σ_2 summation we have $2d$ copies of matrix multiplication to sum over as a result of C_1 being between the two σ_2 matrices. From here, one can see that the process becomes repetitive as the sum for σ_ℓ will make some C_ℓ matrix that we need to consider in the sum for $\sigma_{\ell+1}$. Enumerating this for all the sites, we end up with $2(L-1)d$ matrix multiplications for a complexity of $\mathcal{O}(LD^3d)$.[18] Wow! Just by choosing how we take the inner product in our MPS representation we have been able to go from an exponential complexity to a polynomial[19] complexity in system size.

One can imagine moving this analysis to expectation values, where are operators are of the form

$$\hat{O} = O^{\sigma'_1,\sigma_1}O^{\sigma'_2,\sigma_2}...O^{\sigma'_{L-1},\sigma_{L-1}}O^{\sigma'_L,\sigma_L} \tag{48}$$

In the end, the analysis yields a complexity of essentially[20] $\mathcal{O}(LD^3d)$ again for an arbitrary operator and $\mathcal{O}(D^3d) + \mathcal{O}(D^2d^2)$ for a single site operator.[21]. This second complexity makes single site expectation values or two-site correlations particularly fast to compute for a given system, as it is only dependent on the virtual and physical DOF dimensions, not the system size.

2.5 Spin Chain MPS Example

Here, we present a discussion of the AKLT ground state as a MPS. The AKLT Hamiltonian is

$$\hat{H} = \sum_i \mathbf{S}_i \cdot \mathbf{S}_{i+1} + \frac{1}{3}(\mathbf{S}_i \cdot \mathbf{S}_{i+1})^2 \tag{49}$$

Here, we look at the $S = 1$ spin system. The ground state has been shown [5] to be represented by pairs of symmetrized/triplet spin 1/2 pairs[22] connected on either side by a singlet connection.[23] Treating the system of L spin 1 particles as $2L$ spin 1/2 particles, we have a matrix product state of the form

$$|\psi\rangle = \sum_{\{\sigma\}} \sum_{\{a\},\{b\}} M^{\sigma_1}_{a_1,b_1}\Sigma_{b_1,a_2}M^{\sigma_2}_{a_2,b_2}\Sigma_{b_2,a_3}...M^{\sigma_L}_{a_L,b_L}\Sigma_{b_L,a_1}|\sigma_1...\sigma_L\rangle \tag{50}$$

where σ_i represents the physical (spin 1) degrees of freedom from the triplet spin 1/2 particles ($\sigma \in (0,+,-)$). Additionally, the a and b virtual DOFs represent the singlet connections between pairs of spin 1/2 particles.[24] Explicitly, these matrices are represented as

[18] where D is the truncated dimension of our M^σ matrices
[19] and linear at that!
[20] See section 4.2.1 of [1] for a detailed discussion
[21] That is, an operator where all but one $O^{\sigma'_i,\sigma_i}$ is the identity
[22] which make up a spin 1 particle
[23] See Figure 2.1 of [5] or Figure 19 of [1] for a graphical representation.
[24] The b indices represent the rightward connections and the a indices represent leftward connections, as implied by the ordering in Eq. (50)

$$M^+ = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} \quad M^0 = \begin{pmatrix} 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 \end{pmatrix} \quad M^- = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \quad \Sigma = \begin{pmatrix} 0 & \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} & 0 \end{pmatrix} \tag{51}$$

where they are all in a given spin 1/2 basis $\{|\uparrow\rangle, |\downarrow\rangle\}$

2.5.1 An Aside on Periodic Boundary Conditions

Note that, in Eq. (50), we explicitly have our a_1 as the left virtual leg of M^{σ_1} and we have a_1 again as the right leg of our leftmost Σ matrix. Before, such as in Eq. (36), we had set $a_1 = 1$. This was an example of an open boundary condition MPS form, where the connected form in Eq. (50) is a periodic BC form. The periodic representation is obviously going to be advantageous when we have periodic BC on our lattice, but there is no reason that we can't encode periodic BC in an open MPS form.[25] To omit the bond dimensions, as in Eq. (42), we now have

$$|\psi\rangle = \sum_{\{\sigma\}} \mathrm{tr}(M^{\sigma_1}\Sigma M^{\sigma_2}\Sigma ... M^{\sigma_L}\Sigma) |\sigma_1...\sigma_L\rangle \tag{52}$$

This formalism is common in the study of systems with periodic BC, but to numerically implement models in something like TenPy, this formalism isn't really used.[26]

2.5.2 As Normal As We Can Get

While we can't put the MPS in Eq. (52) into a canonical (normalized) form, we can normalize the state in the thermodynamic limit. First, we combine our M^σ and Σ matrices as

$$\tilde{A}^{\sigma_i} = M^{\sigma_i}\Sigma \tag{53}$$

explicitly yielding

$$\tilde{A}^+ = \begin{pmatrix} 0 & \frac{1}{\sqrt{2}} \\ 0 & 0 \end{pmatrix} \quad \tilde{A}^0 = \begin{pmatrix} -\frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{pmatrix} \quad \tilde{A}^- = \begin{pmatrix} 0 & 0 \\ -\frac{1}{\sqrt{2}} & 0 \end{pmatrix} \tag{54}$$

Analysis of these $\tilde{A}^{\sigma_i}$ matrices shows that $\sum_i \tilde{A}^{\sigma_i\dagger}\tilde{A}^{\sigma_i} = \frac{3}{4}I$, leading us to define $A^{\sigma_i} = \frac{2}{\sqrt{3}}\tilde{A}^{\sigma_i}$, yielding

$$A^+ = \begin{pmatrix} 0 & \sqrt{\frac{2}{3}} \\ 0 & 0 \end{pmatrix} \quad A^0 = \begin{pmatrix} -\frac{1}{\sqrt{3}} & 0 \\ 0 & \frac{1}{\sqrt{3}} \end{pmatrix} \quad A^- = \begin{pmatrix} 0 & 0 \\ -\sqrt{\frac{2}{3}} & 0 \end{pmatrix} \tag{55}$$

[25] It might take a bit higher of a bond dimension to encode these connections, though.
[26] The issue here arises that we cannot put the state into a canonical form. Why isn't this possible? This issue is ultimately solved by the implementation of an "infinite" MPS state and things like iDMRG.

We now have the AKLT ground state in the form

$$|\psi\rangle = \sum_{\{\sigma\}} \mathrm{tr}(A^{\sigma_1} A^{\sigma_2} ... A^{\sigma_L})\, |\sigma_1 \sigma_2 ... \sigma_L\rangle \tag{56}$$

The left-normalization of A^{σ_i} normalizes the state in the thermodynamic limit. We can see this through

$$\begin{aligned}
\langle\psi|\psi\rangle &= \sum_{\{\sigma\}} \mathrm{tr}(A^{\sigma_1} A^{\sigma_2} ... A^{\sigma_L})^* \, \mathrm{tr}(A^{\sigma_1} A^{\sigma_2} ... A^{\sigma_L}) \\
&= \mathrm{tr}\left(\left[\sum_{\sigma_2} A^{\sigma_1 *} \otimes A^{\sigma_1} \right] \left[\sum_{\sigma_2} A^{\sigma_2 *} \otimes A^{\sigma_2} \right] ... \left[\sum_{\sigma_L} A^{\sigma_L *} \otimes A^{\sigma_L} \right] \right) \\
&= \mathrm{tr}\, E^L \\
&= \sum_{i=1}^{4} \lambda_i^L \\
\langle\psi|\psi\rangle &= 1 - 3(\frac{1}{3})^L
\end{aligned} \tag{57}$$

where

$$E = \sum_{\sigma} A^{\sigma *} \otimes A^{\sigma} \tag{58}$$

and λ_i are the eigenvalues of E. Here, we used the identities of the trace and tensor product discussed in section 2.2.3. It is clear from here that, in the thermodynamic limit, we have normalized our MPS for the ground state of the AKLT Hamiltonian.

3 TenPy

3.1 Introduction

Here, we present an introduction to the use of Python's TenPy package, which is useful for implementations of algorithms like density matrix renormalization group (DMRG) and time-evolving block decimation (TEBD).

All code (and maybe more) presented in this section can be found here. Just save a copy of the notebook to your personal drive and get coding!

In this section, we show some of the basics for how to use TenPy, namely building a model for a spin chain and using DMRG to find and analyze properties of the ground state. First off, we need to import some stuff, which we show here.

```
# importing

from tenpy.models.lattice import Chain # bare chain package

from tenpy.networks import site # package that has all sorts of site structures

from tenpy.models.spins import SpinModel # general model for spin lattices

from tenpy.models.spins import SpinChain # special model for a spin chain that
                                         # allows us to bypass the construction
                                         # of a chain lattice

from tenpy.networks.mps import MPS # package to construct a matrix product state
                                   # for a given model

from tenpy.algorithms import dmrg # you know what it be
```

3.2 Lattice

Here, we construct a Chain lattice object to feed into our model. We use the SpinSite as the sites of our lattice, where the S parameter used below in our site initialization defines the spin value.

```
L = 10 # number of sites in our lattice
bc = 'open' # open boundary conditions term, for periodic we put 'periodic'

spin_site = site.SpinSite(0.5, 'None') # we want a spin 1/2 site with no
                                       # conservation laws

lat = Chain(L=L, site=spin_site, bc=bc) # building our lattice!
```

3.3 Model

Here, we insert out lattice built above into the SpinModel to create our model. Here, we implement the transverse field Ising model

$$\hat{H} = \sum_{\langle i,j \rangle} J S_i^z S_j^z - \sum_i g S_i^x$$

Here, we use the SpinModel, but if we know we want a SpinChain, there is a SpinChain model already premade for us, neat!

The SpinModel module can be used in a couple different ways. For example, we can just insert our lattice into the model.

```
J = 1
g = 1

model_params = {
    'lattice' : lat,
    'Jz' : J,
    'hx' : g
}

model = SpinModel(model_params=model_params)
```

Or we can insert parameters into the model like

```
model_params = {
    'L' : L,
    'bc_x' : bc,
    'S' : 0.5,
    'lattice' : 'Chain',
    'Jz' : J,
    'hx' : g*J
}

model = SpinModel(model_params=model_params)
```

Either way, we get the same result!

3.4 Initial MPS

Now that we have a model, we can make our initial MPS and then we will be ready for DMRG. This initial MPS can be random, uniform, or a structured guess based on symmetries of our model. Here, we use the MPS module, where we can construct our matrix product state using just strings!

```
product_state = ["up"]*L # yep, it's that easy!


psi = MPS.from_product_state(model.lat.mps_sites(),
                             product_state,
                             bc=model.lat.bc_MPS) # okay, we had to do this
                                                  # extra step, still easy!
```

3.5 DMRG

Now we have all the pieces to run a DMRG algorithm. We will need to set some parameters for the DMRG, but other than that we are home free! The only thing here that I think could be unclear is the Mixer in dmrg_params. But, you can think of the Mixer as an object that controls the "annealing rate," to make a comparison with the simulated annealing method used in Metropolis Monte-Carlo methods.

```
dmrg_params = {


'trunc_params' : { # parameters for truncation of stuff
    'chi_max' : 100, # maximum virtual/bond dimension
    'svd_min' : 1e-7 # minimum value of singular values before we truncate them
},


'max_E_err' : 1e-5, # maximum error in energies as we make variations


'mixer' : True # something that helps us avoid getting stuck in local minima
               # by expanding the local bond dimension for the some number
               # of initial sweeps. Think of this like annealing in
               # Metropolis (Monte-Carlo) algorithms. True gives the default
               # class of Mixer
}


info = dmrg.run(psi,model,dmrg_params) # dmrg returns all sorts of statistics
                                       # to info
```

And that's it! We now can run a DMRG algorithm for an arbitrary spin chain.

References

[1] Ulrich Schollwöck. The density-matrix renormalization group in the age of matrix product states. *Annals of Physics*, 326(1):96–192, January 2011.

[2] Frank Pollmann, Ari M. Turner, Erez Berg, and Masaki Oshikawa. Entanglement spectrum of a topological phase in one dimension. *Physical Review B*, 81(6), February 2010.

[3] Johannes Hauschild and Frank Pollmann. Efficient numerical simulations with tensor networks: Tensor network python (tenpy). *SciPost Physics Lecture Notes*, October 2018.

[4] F. Verstraete and J. I. Cirac. Matrix product states represent ground states faithfully. *Physical Review B*, 73(9), March 2006.

[5] Ian Affleck, Tom Kennedy, Elliott H. Lieb, and Hal Tasaki. Rigorous results on valence-bond ground states in antiferromagnets. *Phys. Rev. Lett.*, 59:799–802, Aug 1987.