Slide 1:

Consider trying to find the ground state of this Hamiltonian for a 1D chain of spin-1/2’s particles. As you may recognize, this is the Ising model with a transverse field (not too important for now).

The Hilbert space of the ground state has dimension 2^N. The Hamiltonian is therefore a 2^N x 2^N matrix. To solve this (by which I mean to find the ground state and ground state wavefunction), I could employ exact diagonalization. While (as Eric knows) I like to use this method whenever I can, it is only good up to N=20 or so before my computer runs out of memory.

When N becomes large, especially when we are interested in the thermodynamic limits, finding the ground state and ground state wavefunction will be like finding a needle in the haystack.

---------------

So are we doomed? No fortunately! It turns out that for relevant Hamiltonians (more specifically Hamiltonians that contain local interactions that are gapped), the haystack is actually quite small. In fact it is infinitesimally small compared to the full Hilbert space. To be more precise, it is rigorously proven that for the local gapped Hamiltonians that I just talked about, the “haystack” is a small subspace of the Hilbert space that contains states with low entanglement entropy – i.e. states whose entanglement entropy follows the area law: entanglement entropy scales like the size of the boundary of the “partition”.

This means that if we could devise some clever parameterization of our guesses for the ground state wavefunction and come up with some efficient algorithm, then we will be able to solve this seemingly intractable optimization problem above.

---------------

Slide 2:

Okay, so we do start to try to come up with these tricks? One thing we could try is to answer the following. Since we want a efficient way to represent a state with some low entanglement entropy, given some full wavefunction |\psi>:

* How do approximate |\psi> well without keeping track of all d^N coefficients? (Here, d is the number of internal states. For the case of spin-1/2, d=2)
* Is it possible to reduce the entanglement entropy by this approximation?

Idea: if we could come up with some way to accomplish both goals, then we could construct some ansatz/guess states with both properties: (a) “compact” and (b) "having low entanglement entropy”

---------------

Slide 3:

The answer, as you might have sensed coming, has a lot to do with the single value decomposition – a well known technique in image processing, especially for image compression (also a very powerful result in linear algebra with many applications). Here I will provide a brief summary of the theory of SVD, specifically two results:

* The first is just the statement of the theorem: singular value decomposition, which states that any matrix M can be decomposed into the form U S V where U,V are unitary matrices and S is a diagonal matrix (not necessarily a square) whose diagonal entries are the *singular values* of M. And just to be complete, the singular values of M are the eigenvalues of the matrix Sqrt[adj(M) M], which means that the singular values are nonnegative.
  + Just to connect to quantum mechanics, in the case where adj(M)M is a density matrix \rho, then |s\_i|^2 are the eigenvalues of \rho, and we can think of them as probabilities (of what you might ask? Coming soon)
* The second result is the well-known *low-rank approximation.* This states that if I am given some matrix A with rank n, then the HS-distance from A its best approximation with rank k, is the sum of the squares of the n-k smallest singular values of A. Here, the HS-distance is simply the sum of the square root of modulus squares of the entries.

---------------

Slide 4:

Just to quickly show you what second result means, I have prepared this slide. Suppose I have some matrix A which I have decomposed into a product of U, Sigma, and V, where Sigma is the rectangular diagonal matrix with singular values highlighted in orange. The theorem says that the closest matrix to A of rank k is this one on the right where you keep k largest singular values of A. It is obvious then that the distance from A to this new matrix is the sum of the squares of the singular values that we just dropped.

As a quick application: If we treat an image as a matrix then we immediately have an application: image compression? On the left is a full-res image, and on the right is a compressed image.

---------------

Slide 5:

Now, it turns out that we can also do this for wavefunctions. The idea is to represent the wavefunction as a matrix, then SVD it.

To do this “matrixification,” we can basically split the system into two parts LEFT and RIGHT, each with their own orthonormal basis. Then, the full wavefunction will be given in *this* form (on the slide), where the coefficients here have two indices – which means we can treat it as a matrix. One thing to note is that this matrix is NOT an operator.

Now we apply SVD on this matrix \psi\_{lr} = [U D V]\_{lr}. Here, U, V are unitary and D is diagonal. s\_i are the singular values of \psi\_{lr} like I mentioned before. They are the eigenvalues of \sqrt{density matrix}, as we have discussed before.

---------------

Slide 6:

When I said that s\_i^2 could be interpreted as probabilities, we could have asked “okay, probabilities for what?” and this is where we go a little further. If we look at the SVD of \psi more carefully, we will notice that the unitary matrices U and V have (in the background) transformed the orthonormal bases into new ones which makes the wavefunction take on a clean and “diagonal” form. We call this the **Schmidt decomposition**, and the states |i>\_{L,R} are called the Schmidt states. We now interpret the s\_i^2 as the probabilities for the Schmidt states.

One really nice thing about the Schmidt decomposition is that from it we could immediately read off the reduced density matrix for each subsystem.

And the fact that we can interpret the s\_i^2 as probabilities is cemented by the normalization condition on \psi!

---------------

Slide 7:

Let’s do a quick example. Consider the following wavefunction.

Let’s matrixify by splitting the system down the middle. So we have two small Hilbert spaces, each with one spin. The system can then be written in this form, with the matrix \psi\_{ij} being…

Okay, doing a SVD on this matrix, we find the following decomposition. We could also calculate the Schmidt states, but I won’t show the results here.

---------------

Slide 8:

So, coming back to our timeline. So far, we have answer our first question of whether there is some procedure that can let us approximate some complicated wavefunction. But about the second one about entanglement entropy? Does \psi approximation using SVD and Schmidt decomposition reduce the entanglement entropy? Is for this very reason that SVD is the absolute best candidate.

To see this, let us look at how von Neumann entanglement entropy is calculated. By definition, the von Neumann entanglement entropy can be calculated using the reduced density matrices (turns out that either one can be used… since they have the same spectrum).

Now do we calculate this? Well, instead of using the usual definition of these reduced density matrices where you would trace out the density matrix of the entire system, we also know what these are in terms of the Schmidt states, from which we know that the spectrum of \rho\_L, \rho\_R are nothing more than the s\_i^2 themselves. And so the von Neumann entanglement entropy is simply this.

Under compression, what people typically do is keep the first 100s significant singular values or so and drop ~2^N of the remaining values (which are on the order of 10^-6 or something like that). This immediately reduces the entanglement entropy and give us exponential compression!

Truly amazing!

---------------

Slide 9:

Here’s the old example. Let us calculate the entanglement entropy in this case. It is obvious that because ln(1) = 0, S = 0. This is as expected, because \psi is a product state.

---------------

Slide 10:

Okay, so far we have seen how the SVD gives us a way to generate a low-entanglement entropy approximation of some given some wavefunction \psi.

But wait, you might ask… why are we doing this exactly? Isn’t our goal to find the ground state wavefunction of some complicated Hamiltonian? If we don’t have some wavefunction to start, what is the point of this compression procedure. There’s nothing to compress? So we have kind of a chicken-and-egg problem.

---------------

Slide 11:

This brings me to the second part of my ZGS which will most be on the matrix product state and finally on the DMRG (roughly).

What we have done so far with SVD and Schmidt decomposition is to set up the scene for introducing MPS, which stands for matrix product state, which is the basis for this DMRG method for very efficiently finding ground states of 1d Hamiltonians.

As a rough idea, MPS is a way of expressing wavefunctions, but instead of in terms of coefficients, it is in terms of product of matrices. Any wavefunction can be expressed in terms of a matrix product state. However, the advantage of matrix product states is that it is a very natural way to express, or more importantly, to generate states with low entanglement entropy, which is suitable for ground state searches (just to continue with the haystack analogy… basically by starting in the MPS, we are already making our haystack very small)

---------------

And just a few works on DMRG:

* It is a numerical, variational method for ground state searches
* Most efficient method for 1d systems
* Works with MPS, not \psi. So these guys go hand in hand

---------------

Slide 12:

To show you what a matrix product state, I feel like it is best to do this constructively… in the sense that I will show you have to convert a \psi in wavefunction form into an MPS. Let us begin with some wavefunction written in the Schmidt decomposition form. Here, I’ve added the subscript m to indicate where the L-R split happens on the chain of N particles. So m here is a physical index.

What I want to do next is to write the Schmidt states in terms of the standard basis with spins. And I will do this recursively. If the cut is at m=1, then the Schmidt state is just some superposition of the internal levels of a single particle. So, we just have this expression. Here, A \sigma\_1 i\_1 is a row vector of size (1 x d), where the d the number of possible values for \sigma\_1. The i\_1 subscript is just a label for the site location. So far so good.

Next, we basically move the cut over, or can also think of this as adding a spin. Then, we have a similar construction as before, where we are combining two systems: i\_1 and \sigma\_2. And we now need a matrix. By inserting the first expression and summing over i\_1, we’re left with the expression on the far right. Here, the first matrix has dimension (1xd), the second matrix has dimension (dxd^2).

Now we continue the pattern:… Here, A^\sigma\_3 has dimension (d^2xd^3), and so on

---------------

Slide 13:

Ultimately, we have that the Schmidt states for the L/R subsystems have the form: (slide)

The full wavefunction can now be written as (slide) where the sum is over all spin configurations. And S is the diagonal matrix with all the singular values s\_i = S\_{i,i}.

Here, the matrix dimensions are:

(1xd), (dxd^2), … (d^2,d), (dx1),

So that the entire expression AAAAASBBBBBB is a scalar!

This is \psi in the MPS form.

---------------

Slide 14:

Just to give an example, consider the GHZ state, which we say should take this form in the MPS language. What would the matrices be? It turns out that they are:

(slide)

And as you can see, the product of these matrices is 1 exactly when all of the \sigma’s are 0 or 1.

---------------

Slide 14:

Okay, so in what sense is MPS a natural language