

Chapter 6

Finite DMRG, Conserved Quantities, and other Tricks

A. Matrix Product Operators

In our DMRG code, we needed to keep track of two matrices: Z_j^L and H_j^L . As we add more terms to our Hamiltonian we need to keep track of more and more things. Moreover, we would like our code to be easily modifiable to work with different Hamiltonians.

The book-keeping can be simplified if we rewrite our Hamiltonian as a "Matrix Product Operator." We want to find tensors H_1, \dots, H_5 that satisfy:

$$\begin{array}{c} \sigma_1 \downarrow \quad \sigma_2 \downarrow \quad \sigma_3 \downarrow \quad \sigma_4 \downarrow \quad \sigma_5 \downarrow \\ \boxed{\text{H}} \\ \uparrow \sigma_1 \quad \uparrow \sigma_2 \quad \uparrow \sigma_3 \quad \uparrow \sigma_4 \quad \uparrow \sigma_5 \end{array} = \begin{array}{c} \sigma_1 \downarrow \quad \sigma_2 \downarrow \quad \sigma_3 \downarrow \quad \sigma_4 \downarrow \quad \sigma_5 \downarrow \\ \boxed{H_1} \xrightarrow{\alpha_1} \boxed{H_2} \xrightarrow{\alpha_2} \boxed{H_3} \xrightarrow{\alpha_3} \boxed{H_4} \xrightarrow{\alpha_4} \boxed{H_5} \\ \uparrow \sigma_1 \quad \uparrow \sigma_2 \quad \uparrow \sigma_3 \quad \uparrow \sigma_4 \quad \uparrow \sigma_5 \end{array} \cdot \quad (6.1)$$

We then just need to feed the H_j into our code. The object we need to minimize

$$Q^\dagger H_{\text{eff}} Q = \begin{array}{c} \begin{array}{c} \text{L}_1 \text{L}_2 \text{L}_3 \\ \text{L}_1^* \text{L}_2^* \text{L}_3^* \end{array} \begin{array}{c} \text{Q}_4 \\ \text{Q}_4^* \end{array} \begin{array}{c} \text{R}_3 \text{R}_2 \text{R}_1 \\ \text{R}_3^* \text{R}_2^* \text{R}_1^* \end{array} \\ \boxed{\text{H}} \end{array} \quad (6.2)$$

is then constructed as

$$\begin{array}{c} \begin{array}{c} \text{L}_1 \text{L}_2 \text{L}_3 \\ \text{L}_1^* \text{L}_2^* \text{L}_3^* \end{array} \begin{array}{c} \text{Q}_4 \\ \text{Q}_4^* \end{array} \begin{array}{c} \text{R}_3 \text{R}_2 \text{R}_1 \\ \text{R}_3^* \text{R}_2^* \text{R}_1^* \end{array} \\ \boxed{\text{H}} \end{array} = \begin{array}{c} \boxed{\text{HL}_3} \xrightarrow{\alpha_3} \boxed{H_4} \xrightarrow{\alpha_4} \boxed{H_5} \xrightarrow{\alpha_5} \boxed{\text{HR}_3} \\ \boxed{Q_3} \xrightarrow{\alpha_3} \boxed{H_4} \xrightarrow{\alpha_4} \boxed{H_5} \xrightarrow{\alpha_5} \boxed{Q_3^*} \end{array} \cdot \quad (6.3)$$

We update the left-block operator via

$$(6.4)$$

A.1. Explicit Construction

Just as any wavefunction can be expressed as a matrix product wavefunction, and operator can be expressed as a matrix product operator. Not all can be encoded efficiently though. The sort of local Hamiltonians which describe physical systems can be.

Limiting ourselves to 5 sites, the operator we want is

$$(6.5)$$

One can readily verify that you get exactly that operator by multiplying out

$$\begin{aligned}
 H = & \left(\begin{array}{ccc} 1 & -J & -h \\ \downarrow \sigma_1 & \downarrow \sigma_1 & \downarrow \sigma_1 \\ \boxed{Z_1} & \boxed{X_1} & \end{array} \right) \left(\begin{array}{ccc} 1 & -J & -h \\ \downarrow \sigma_2 & \downarrow \sigma_2 & \downarrow \sigma_2 \\ \boxed{Z_2} & \boxed{X_2} & \\ \downarrow \sigma_2 & & \\ 1 & & \end{array} \right) \left(\begin{array}{ccc} 1 & -J & -h \\ \downarrow \sigma_3 & \downarrow \sigma_3 & \downarrow \sigma_3 \\ \boxed{Z_3} & \boxed{X_3} & \\ \downarrow \sigma_3 & & \\ 1 & & \end{array} \right) \\
 & \times \left(\begin{array}{ccc} 1 & -J & -h \\ \downarrow \sigma_4 & \downarrow \sigma_4 & \downarrow \sigma_4 \\ \boxed{Z_4} & \boxed{X_4} & \\ \downarrow \sigma_4 & & \\ 1 & & \end{array} \right) \left(\begin{array}{ccc} -h & & \\ \downarrow \sigma_5 & & \\ \boxed{X_5} & & \\ \downarrow \sigma_5 & & \\ 1 & & \end{array} \right) \left(\begin{array}{ccc} \boxed{Z_5} & & \\ \downarrow \sigma_5 & & \\ & & 1 \end{array} \right)
 \end{aligned} \tag{6.6}$$

The generalization to other Hamiltonians should be clear.

B. Conserved Quantities

There are a number of important tricks which are important for efficiency of a DMRG code. One is taking advantage of conservation laws – a trick which can speed up code by several orders of magnitude. It is also a deeply physical story.

There are not a lot of conservation laws to take advantage of with the transverse field Ising model, so the example to have in mind is the XXZ model,

$$H = \sum_j J_x (\sigma_j^x \sigma_{j+1}^x + \sigma_j^y \sigma_{j+1}^y) + J_z (\sigma_j^z \sigma_{j+1}^z). \tag{6.7}$$

This has a global $U(1)$ symmetry, corresponding to an invariance under rotations about the \hat{z} axis:

$$\sigma^z \rightarrow \sigma^z \tag{6.8}$$

$$\sigma^y \rightarrow \sigma^y \cos(\phi) - \sigma^x \sin(\phi) \tag{6.9}$$

$$\sigma^x \rightarrow \sigma^x \cos(\phi) + \sigma^y \sin(\phi). \tag{6.10}$$

This invariance leads to conservation of total S_z : No terms in the Hamiltonian changes the total number of up-spins. Consequently the eigenstates are also eigenstates of S_Z , with quantum number m .

A further consequence is that the reduced density matrix of part of the system will be block diagonal, with each block corresponding to a different value of m . To explicitly see that, imagine that $|m_R, \alpha\rangle_R$ form an arbitrary orthonormal basis for the right half of the system – this arbitrary basis can always be chosen so that any given state within it is an eigenstate of S_z . We then consider the reduced density matrix,

$$\hat{\rho}_L = \sum_{m_R} \sum_{\alpha} R \langle m_R, \alpha | \psi \rangle \langle \psi | m_R, \alpha \rangle_R. \tag{6.11}$$

Since the total spin of the system is m , then ${}_R\langle m_R, \alpha | \psi \rangle$ is a state of the left part of the system, with total spin $m_L = m - m_R$. We see that $\hat{\rho}_L$ only involves sums of terms where each ket-bra product has the same m_L for the ket and the bra. Hence ρ_L does not mix sectors with different m_L .

This means that the Schmidt states are all eigenstates of S_z : Every bond index (which encode the Schmidt states) can be labeled with a quantum number m . Similarly, every spin index can also be labeled by m . Here is where the arrows in our notation come in. The sum of all the incoming m 's minus the outgoing m 's is denoted the “valence” of a tensor. For an eigenstate, all of the non-zero terms for any given matrix will all have the same valence. An example will help. Consider the most general four-site wavefunction where there are two up-spins, and two down-spins. This is clearly analogous to the 2-particle wavefunctions from the first class, and can be written

$$|\psi\rangle = \left(\begin{array}{cc} \downarrow & \uparrow \end{array} \right) \left(\begin{array}{cc} \downarrow & \uparrow \\ & \downarrow \uparrow \end{array} \right) \left(\begin{array}{cc} \psi_{34} \uparrow & \\ \psi_{24} \downarrow & \psi_{23} \uparrow \\ \psi_{14} \downarrow & \psi_{13} \uparrow \\ & \psi_{12} \downarrow \end{array} \right) \left(\begin{array}{c} \uparrow \\ \downarrow \end{array} \right) \quad (6.12)$$

(This can of course be recognized as the mixed Canonical Form.) To make book-keeping simpler, we give an \uparrow a valence of $+1$ – rather than the more standard $1/2$ – similarly \downarrow gets a -1 . The dummy indices for the columns of the first matrix (and the rows of the second) are denoted a_-, a_+ . The indices for the columns of the second (and the rows of the third) are b_-, b_0, b'_0, b_{+2} , the columns of the third (and the rows of the fourth) are c_-, c_+ . We will label the matrices A, B, C, D . We will write in-coming indices down-stairs, and outgoing upstairs. The non-zero elements are

$$\begin{array}{llll} & B_{a_-\downarrow}^{b_{-2}} = 1 & C_{b_{-2}\uparrow}^{c_{-1}} = \psi_{34} & D_{c_{-1}\uparrow} = 1 \\ & B_{a_-\uparrow}^{b_0} = 1 & C_{b_0\downarrow}^{c_{-1}} = \psi_{24} & D_{c_1\downarrow} = 1 \\ A_{\downarrow}^{a_-} = 1 & B_{a_+\downarrow}^{b'_0} = 1 & C_{b'_0\uparrow}^{c_1} = \psi_{23} & \\ A_{\uparrow}^{a_+} = 1 & B_{a_+\uparrow}^{b_2} = 1 & C_{b'_0\downarrow}^{c_{-1}} = \psi_{14} & \\ & & C_{b'_0\uparrow}^{c_1} = \psi_{13} & \\ & & C_{b_2\downarrow}^{c_1} = \psi_{12} & \end{array}$$

As you can see, each tensor has a valence of zero: The upstairs indices – representing outgoing bonds – have the same net spin as the downstairs indices. The sum of the valence of all of the tensors will be the net spin.

This sparsity helps in several ways: (1) The tensors take up less memory – one only needs to store blocks with a fixed valence. (2) Tensor operations (adding, multiplying, contracting) are faster, since one can take advantage of the sparsity. (3) The eigensolver is more efficient – as you can drop the components of the wavefunction that are in blocks with different valence. (4) The SVD's can be done block by block. All of this technology works in the background – one just needs to define the methods for your tensor class to take advantage of it.

Any Abelian symmetry can be dealt with in this way. Non-abelian symmetries (such as the rotational invariance of the Heisenberg model) are a bit more complicated.

C. Finite DMRG

If you need to deal with finite or inhomogeneous systems, the iDMRG algorithm does not work. There is, however, a very efficient way to optimize finite matrix product states – which is usually just called the DMRG. Historically, the lore was that iDMRG was not as efficient as the finite DMRG. I believe with modern algorithms they are roughly equivalent in cost. Nonetheless, when people say DMRG they mean the finite algorithm.

Imagine that we have a MPS in the right canonical form. Given n sites, one first generates and stores the right-most “environment” matrix

$$HR_n = \begin{array}{c} \begin{array}{|c|} \hline \text{ } \\ \hline \end{array} \begin{array}{c} t_{n-1} \\ a_{n-1} \\ t_{n-1} \end{array} \end{array} = \begin{array}{c} \begin{array}{|c|} \hline \text{ } \\ \hline \end{array} \begin{array}{c} t_{n-1} \\ a_{n-1} \\ t_{n-1} \end{array} \end{array} \begin{array}{c} \begin{array}{|c|} \hline R_n \\ \hline \end{array} \begin{array}{c} t_{n-1} \\ \sigma_n \end{array} \\ \begin{array}{|c|} \hline H_n \\ \hline \end{array} \begin{array}{c} a_{n-1} \\ \sigma_n \end{array} \\ \begin{array}{|c|} \hline R_n^* \\ \hline \end{array} \begin{array}{c} \sigma_n \\ t_{n-1} \end{array} \end{array} . \quad (6.13)$$

One then sequentially generates further matrices via

$$HR_j = \begin{array}{c} \begin{array}{|c|} \hline \text{ } \\ \hline \end{array} \begin{array}{c} t_{j-1} \\ a_{j-1} \\ t_{j-1} \end{array} \end{array} = \begin{array}{c} \begin{array}{|c|} \hline \text{ } \\ \hline \end{array} \begin{array}{c} t_{j-1} \\ a_{j-1} \\ t_{j-1} \end{array} \end{array} \begin{array}{c} \begin{array}{|c|} \hline R_j \\ \hline \end{array} \begin{array}{c} t_{j-1} \\ \sigma_j \end{array} \\ \begin{array}{|c|} \hline H_j \\ \hline \end{array} \begin{array}{c} a_{j-1} \\ \sigma_j \end{array} \\ \begin{array}{|c|} \hline R_j^* \\ \hline \end{array} \begin{array}{c} \sigma_j \\ t_{j-1} \end{array} \end{array} \begin{array}{c} \begin{array}{|c|} \hline \text{ } \\ \hline \end{array} \begin{array}{c} t_j \\ a_j \\ t_j \end{array} \end{array} = HR_{j+1} . \quad (6.14)$$

One continues on until $j = 3$ – so that we have the environment for the left two sites. If the bond dimension is very large these would be stored on disk, rather than in memory.

One then generates

$$H_{12}^{\text{eff}} = \begin{array}{c} \begin{array}{|c|} \hline \text{ } \\ \hline \end{array} \begin{array}{c} a_1 \\ \sigma_1 \end{array} \end{array} \begin{array}{c} \begin{array}{|c|} \hline H_1 \\ \hline \end{array} \begin{array}{c} a_1 \\ \sigma_1 \end{array} \end{array} \begin{array}{c} \begin{array}{|c|} \hline \text{ } \\ \hline \end{array} \begin{array}{c} a_2 \\ \sigma_2 \end{array} \end{array} \begin{array}{c} \begin{array}{|c|} \hline H_2 \\ \hline \end{array} \begin{array}{c} a_2 \\ \sigma_2 \end{array} \end{array} \begin{array}{c} \begin{array}{|c|} \hline \text{ } \\ \hline \end{array} \begin{array}{c} t_2 \\ a_2 \\ t_2 \end{array} \end{array} = HR_3 . \quad (6.15)$$

One then solves the eigenvalue problem

$$\begin{array}{c} \begin{array}{|c|} \hline Q_1 \\ \hline \end{array} \begin{array}{c} t_1 \\ \sigma_1 \end{array} \end{array} \begin{array}{c} \begin{array}{|c|} \hline H_1 \\ \hline \end{array} \begin{array}{c} a_1 \\ \sigma_1 \end{array} \end{array} \begin{array}{c} \begin{array}{|c|} \hline \text{ } \\ \hline \end{array} \begin{array}{c} a_2 \\ \sigma_2 \end{array} \end{array} \begin{array}{c} \begin{array}{|c|} \hline H_2 \\ \hline \end{array} \begin{array}{c} a_2 \\ \sigma_2 \end{array} \end{array} \begin{array}{c} \begin{array}{|c|} \hline \text{ } \\ \hline \end{array} \begin{array}{c} t_2 \\ a_2 \\ t_2 \end{array} \end{array} = E \begin{array}{c} \begin{array}{|c|} \hline Q_1 \\ \hline \end{array} \begin{array}{c} t_1 \\ \sigma_1 \end{array} \end{array} . \quad (6.16)$$

Next one uses a SVD to decompose Q_{12} , putting our wavefunction into the mixed canonical form,

$$|\psi\rangle = \text{Diagram (6.17)} \quad (6.17)$$

One also constructs

$$HL_1 = \text{Diagram (6.18)} \quad (6.18)$$

and stores it to disk. One then retrieves the stored HR_4 , and updates the second two sites by solving

$$\text{Diagram (6.19)} = E \text{Diagram (6.19)} \quad (6.19)$$

Another SVD gives

$$|\psi\rangle = \text{Diagram (6.20)} \quad (6.20)$$

We make the next left environment via

$$HL_2 = \text{Diagram (6.21)} \quad (6.21)$$

We continue in this manner until we have updated all of the tensors. We then sweep back the other way. At each step using our stored environments, updating the right-environments as we sweep left.

We keep sweeping back and forth until we have converged. For many models the convergence is surprisingly fast – taking only 2-3 sweeps.

D. Delayed Evaluation

One simple trick is that one never actually calculates matrices such as

$$H_{\text{eff}} = \begin{array}{c} \text{HL}_1 \quad \text{H}_2 \quad \text{H}_3 \quad \text{HR}_4 \\ \begin{array}{c} \text{---} t_1 \text{---} \text{---} t_2 \text{---} \text{---} t_3 \text{---} \\ \text{---} \sigma_1 \text{---} \text{---} \sigma_2 \text{---} \text{---} \sigma_3 \text{---} \\ \text{---} t'_1 \text{---} \text{---} t'_2 \text{---} \text{---} t'_3 \text{---} \end{array} \end{array} \cdot \quad (6.22)$$

Rather, one stores the components HL, H_2, H_3, HR . In your iterative eigenvalue solver you efficiently contract the network

$$H_{\text{eff}} Q = \begin{array}{c} \text{HL}_1 \quad \text{H}_2 \quad \text{H}_3 \quad \text{HR}_4 \quad Q_{23} \\ \begin{array}{c} \text{---} t_1 \text{---} \text{---} t_2 \text{---} \text{---} t_3 \text{---} \\ \text{---} \sigma_1 \text{---} \text{---} \sigma_2 \text{---} \text{---} \sigma_3 \text{---} \\ \text{---} t'_1 \text{---} \text{---} t'_2 \text{---} \text{---} t'_3 \text{---} \end{array} \end{array} \cdot \quad (6.23)$$

When the bond dimension is large this can give substantial speedup.

E. State Prediction

An even more important trick is to use “state prediction.” One uses an iterative eigensolver – and the key to efficiency is to start with a good guess for the eigenvector. In the finite algorithm, it is trivial to find a good guess. Consider, for example, trying to optimize Q_{23} in the second step of our algorithm,

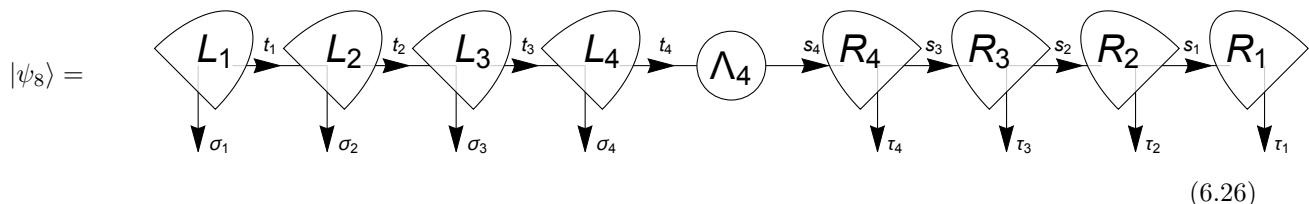
$$|\psi\rangle = \begin{array}{c} \text{---} t_1 \text{---} \text{---} t_2 \text{---} \text{---} t_3 \text{---} \text{---} t_4 \text{---} \text{---} t_5 \text{---} \text{---} t_6 \text{---} \text{---} t_7 \text{---} \\ \text{---} \sigma_1 \text{---} \text{---} \sigma_2 \text{---} \text{---} \sigma_3 \text{---} \text{---} \sigma_4 \text{---} \text{---} \sigma_5 \text{---} \text{---} \sigma_6 \text{---} \text{---} \sigma_7 \text{---} \end{array} \quad (6.24)$$

the previous step of the algorithm gave us the matrix M_2 , and we already had R_3 . Therefore we already have a pretty good $Q_{23} = M_2 R_3$. It will be refined by solving the eigenproblem, but nonetheless is a good starting point.

State prediction in the iDMRG algorithm is a bit more complicated. Suppose we have just found the following representation of the wavefunction,

$$|\psi_6\rangle = \begin{array}{c} \text{---} t_1 \text{---} \text{---} t_2 \text{---} \text{---} t_3 \text{---} \text{---} t_4 \text{---} \text{---} t_5 \text{---} \text{---} t_6 \text{---} \text{---} t_7 \text{---} \\ \text{---} \sigma_1 \text{---} \text{---} \sigma_2 \text{---} \text{---} \sigma_3 \text{---} \text{---} \sigma_4 \text{---} \text{---} \sigma_5 \text{---} \text{---} \sigma_6 \text{---} \text{---} \sigma_7 \text{---} \end{array} \quad (6.25)$$

We want to make a good guess for $L_4\Lambda_4R_4$ in



$$|\psi_8\rangle = \text{Diagram showing the contraction of tensors } L_1, L_2, L_3, L_4, \Lambda_4, R_4, R_3, R_2, R_1 \text{ with indices } \sigma, t, s, \tau. \quad (6.26)$$

The obvious guess would be $L_4 = L_3, R_4 = R_3, \Lambda_4 = \Lambda_3$. It turns out that this doesn't work. The problem is that L_3 takes you from the space described by t_2 into t_3 . You can't put a t_2 socket into a t_3 hole.

The solution is to do a SVD on R_3 , writing $\Lambda_3 R_3 = \tilde{L}_4 \tilde{\Lambda}_4$. This new matrix \tilde{L}_4 takes one from the t_3 space to the t_2 space. We use it for the guess for L_4 , we use $\tilde{\Lambda}_4$ as our guess for Λ_4 , and we use the transpose of $\tilde{\Lambda}_4$ as the guess for R_4 .

F. Noise

This section is copied from some notes I had for another purpose. I don't think we will have time to go through this, but I copy it here in case you find it useful. I'll edit it to make it more clear when I have a chance.

F.1. Getting Stuck

You will find that in some cases the DMRG algorithm gets stuck. To illustrate this, consider a strange model where we have hard-core bosons hopping on a lattice, but where the hopping is between next-nearest neighbors. For concreteness, let's take 5 sites and hard wall boundary conditions, and limit the occupation per site to 0 or 1. Our Hamiltonian is

$$H = -t \left(a_1^\dagger a_3 + a_3^\dagger a_1 + a_2^\dagger a_4 + a_4^\dagger a_2 + a_3^\dagger a_5 + a_5^\dagger a_3 \right). \quad (6.27)$$

Let's start initially with a product state 00100, which means there is one particle on the third site, and none on any other sites.

Let's now run the standard DMRG algorithm. We "expose" the first two sites. Our bond dimension is 1, so our effective Hamiltonian is $2^2 \times 2^2$ – just the Hamiltonian acting on the first two sites. By inspection, this effective Hamiltonian is a matrix where every element is 0. Thus our starting state 00 is an eigenstate, and we are happy. We do a SVD – which is trivial since we have a product state, and set $A_1 = |0\rangle$. We then go on to the next two sites. Again our effective Hamiltonian is 0... The algorithm never changes the wavefunction. We are stuck in 00100.

Steve White came up with a nice trick for getting un-stuck, which is usually referred to as a "noise" term. It involves modifying the SVD step.

F.2. SVDs and density matrices

To understand White's approach, I need to remind you how to do SVDs via density matrices.

Suppose we have a matrix M_{ab} , which we want to write as $\sum_{\mu} A_{a\mu} B_{\mu b}$, where A is left-normalized: $\sum_a A_{a\mu}^* A_{a\nu} = \delta_{\mu\nu}$. We could do this with a SVD, or alternatively we can construct

$$\rho_{aa'} = \sum_b M_{ab} M_{a'b}^*, \quad (6.28)$$

which if M is a wavefunction, is just the reduced density matrix.

We then do an eigen-decomposition of ρ , writing it as $\rho = U \Lambda^2 U^\dagger$, where Λ^2 is a diagonal matrix, and U is unitary, meaning that $U^\dagger U = U U^\dagger = 1$. Hence $M = U U^\dagger M$. We take $A = U$ and $B = U^\dagger M$. Clearly A is left normalized. I am calling the diagonal matrix Λ^2 to connect it later with the SVD. [And since the eigenvalues of a Hermitian matrix are always non-negative, this notation makes some sense.]

Now this procedure gives you a bond dimension equal to the dimension of the index a , which we will call d_a . Suppose you want to truncate. Well you just take the largest eigenvalues. Lets order the eigenvalue from largest to smallest,

$$\rho_{aa'} = \sum_{\mu=1}^{d_a} U_{a\mu} \lambda_{\mu}^2 U_{a'\mu}^* \quad (6.29)$$

$$\approx \sum_{\mu=1}^d U_{a\mu} \lambda_{\mu} U_{a'\mu}^*, \quad (6.30)$$

where $d \leq d_a$ is the bond dimension we are truncating to. We then take $A = U_{a\mu}$, with $\mu = 1, 2, \dots, d$, and

$$B_{\mu b} = \sum_{a'} U_{a'\mu}^* M_{a'b}. \quad (6.31)$$

The equivalence to the SVD is seen by writing $M = U \Lambda V$, where U and V are unitary. We can immediately calculate $\rho = M M^\dagger = U \Lambda V V^\dagger \Lambda U^\dagger = U \Lambda^2 U^\dagger$, and see that the matrices we have been working with are the same as in the SVD.

Clearly the indices a and b can be “compound indices” representing both site and link bonds.

F.3. Noise

step 1

In the very first step of our toy problem, we follow this procedure and generate the density matrix

$$\rho_1 = |0\rangle\langle 0|, \quad (6.32)$$

where these bra's and kets are acting on the first site. Now what White does is look at H and see what operators act on the first site: in this case a_1 and a_1^\dagger . He then applies each of these to the density matrix, taking

$$\tilde{\rho}_1 = \rho_1 + \epsilon \left(a_1 \rho_1 a_1^\dagger + a_1^\dagger \rho_1 a_1 \right) \quad (6.33)$$

$$= |0\rangle\langle 0| + \epsilon |1\rangle\langle 1|. \quad (6.34)$$

This clearly has two eigenvalues, 1 and ϵ , and get a bond dimension 2 (instead of the 1 we got without the noise term). We take

$$A_1 = \begin{pmatrix} |0\rangle & |1\rangle \end{pmatrix} \quad (6.35)$$

and

$$A_2 = \begin{pmatrix} |0\rangle \\ 0 \end{pmatrix}. \quad (6.36)$$

The product of these matrices is again just the state $|00\rangle$, but the higher bond dimension will help us in the next step. We will call $\mu = 0$ the first column of A_1 , and $\mu = 1$ the second.

step 2

In that next step, we expose sites 2 and 3. The effective Hamiltonian is now $2^3 \times 2^3$, since the bond dimension on the left is 2. The matrix is sparse, so rather than writing it out, I will just list the non-zero matrix elements. I have already defined the index μ . Lets take a to be the site index for site 2, and have $a = 0$ correspond to $|0\rangle_2$ and $a = 1$ correspond to $|1\rangle_2$. Lets take b to be the site index for site 3, with the same meaning.

The non-zero matrix elements in the sector we are interested in (which has one particle) are:

$$H_{\mu'=1, a'=0, b'=0}^{\mu=0, a=0, b=1} = -t \quad (6.37)$$

$$H_{\mu'=0, a'=0, b'=1}^{\mu=1, a=0, b=0} = -t. \quad (6.38)$$

All others are zero. We diagonalize this, and get (in the μ, a, b basis) $\psi_{1,0,0} = \psi_{0,0,1} = \sqrt{2}$, and all other elements zero. The non-zero elements of the reduced density matrix for site 2 are then

$$\rho_{\mu'=1, a'=1}^{\mu=1, a=0} = 1/2 \quad (6.39)$$

$$\rho_{\mu'=0, a'=1}^{\mu=0, a=0} = 1/2. \quad (6.40)$$

Not worrying about the noise term, we get

$$A_2 = \begin{pmatrix} |0\rangle & 0 \\ 0 & |0\rangle \end{pmatrix} \quad (6.41)$$

and

$$A_3 = \frac{1}{\sqrt{2}} \begin{pmatrix} |1\rangle \\ |0\rangle \end{pmatrix}. \quad (6.42)$$

and you see that we are “unstuck” as $\psi = (|10000\rangle + |00100\rangle)/\sqrt{2}$ is closer to the ground state $\psi_{\text{GS}} = (|10000\rangle + \sqrt{2}|00100\rangle + |0001\rangle)/2$ than our starting state was. In fact after 1 full sweep we will be there (while without the noise we would never get there).

G. Periodic Boundaries

In most of our theoretical work, we use periodic boundary conditions. It should be clear that the DMRG's efficiency is really based upon the fact that you can start at one end of the chain and move from there. This whole scheme is not compatible with periodic boundary conditions.

Nonetheless one can use matrix product states with periodic boundary conditions:

$$|\psi\rangle = \text{Tr} M M M \cdots M M M. \quad (6.43)$$

The trace makes the system periodic. Unfortunately there is no canonical form for such a periodic state that is as nice as the Canonical forms we used for the system with boundaries, or for the infinite system. The issue is that if we want to divide the ring into two pieces we need to cut it in two places. The Schmidt states will then be labeled by two sets of indices corresponding to degrees of freedom at each end. I don't think anyone has come up with a systematic way of dividing up the degrees of freedom.

There are a number of ad-hoc approaches which appear to work. I don't have time to go into them.

H. Homework 4

Problem 23. For Credit Write a MPO which describes the XXZ Hamiltonian

$$H = \sum_j J_x (\sigma_j^x \sigma_{j+1}^x + \sigma_j^y \sigma_{j+1}^y) + J_z (\sigma_j^z \sigma_{j+1}^z). \quad (6.44)$$

Problem 24. Bonus. Write a finite DMRG code for the transverse field Ising model. Take 20 sites. Start in a product state of all spins in the \hat{z} -direction. Restrict bond dimension to be no larger than 20. Plot $\langle Z \rangle$ as a function of sweep number at $h = 0.5$, $h = 0.9$, and $h = 1$. How does convergence compare to the iDMRG code?