You are now going to write a DMRG code using your code from HW3. The general process is going to be as follows:

Loop over sites and at each site: * Canonize your state * Produce many matrix-product states, each one which is the derivative of the current matrix-product state with one of the parameters on site $i$. Call it $\Psi[\alpha_{ij}]$ * Compute the overlap matrix of $S_{jk} = \langle \Psi[\alpha_{ij}] | \Psi[\alpha_{ik}] \rangle$. Verify that it’s the identity. * Compute the Hamiltonian matrix $H_{jk} = \langle \Psi[\alpha_{ij}] | H | \Psi[\alpha_{ik}] \rangle$ * Diagonalize the Hamiltonian. Choose the eigenvector with the smallest eigenvalue. Replace the parameters in site $i$. At each step you should check that your energy is going down. Then you should run until you’ve hit a fixed point. You can start essentially at any matrix.

As a first step, figure out analytically how to compute the derivatives of the MPS with respect to one of the parameters.

In [1]: function TakeDerivative(myMPS, mySite, myParameter) # myParameter can span over 1 to 2D^2 where

   end

Out[1]: TakeDerivative (generic function with 1 method)

Now, canonize over site $i$ and loop over all $2D^2$ parameters and produce the matrix-product states which are the derivatives on site $i$.

Verify they are the derivatives using finite differences. Do this by changing the parameter you think you’re taking the derivative of by adding $\delta$ to that parameter and evaluate some configuration $|c\rangle$, subtract $\delta$ from that parameter and evaluate the same configuration $|c\rangle$. Make sure that the $(\Psi[c;p + \delta] - \Psi[c;p - \delta])/(2\delta) = d\Psi/dp(c;p)$

Now go ahead and compute overlap of all your derivative MPS. Make sure the overlap matrix is identity!

For the next step you’re going to have to compute $\langle MPS_1 | H | MPS_2 \rangle$. Write some code that does this and build the full effective Hamiltonian matrix.

In [4]: function HOverlap(MPS1,MPS2)

   end

function HOverlapMatrix(MPS)

   end

Out[4]: HOverlapMatrix (generic function with 1 method)

Now put it all together into a DMRG step which works on site $i$, produces the effective Hamiltonian, diagonalizes it and then replaces the parameters.

In [3]: function DMRGStep(MPS, site_i)

   end

Out[3]: DMRGStep (generic function with 1 method)
Finally, write a complete DMRG code that sweeps back and forth lowering the energy at each step. Remember you can compute the energy by looking at $\langle MPS | H | MPS \rangle$.

In [ ]: