Lecture "Analytical and Numerical Methods for Quantum Many-Body Systems from a Quantum Information Perspective" — Exercise Sheet #2

1. Determine the transfer operator $\mathbb{E} = \sum_i A^i \otimes \bar{A}^i$ for the AKLT state, the GHZ state, and the W state. What is the correlation length of these states?

   (Recall that the GHZ had tensors $A^0 = |0\rangle\langle 0|$, $A^1 = |1\rangle\langle 1|$, the AKLT state – up to basis change – $A^0 = \sigma_x$, $A^1 = \sigma_y$, $A^2 = \sigma_z$, and the W state $A^0 = \sigma_z$ and $A^1 = \sigma^+$).

2. Check that the transfer operator does not change under basis transformations of the physical system, $A^i \mapsto \sum A^i u_{ij}$ with $U = (u_{ij})$ a unitary matrix. Also try to understand the proof graphically.

3. Write a variational code for MPS with OBC, as described in the lecture, which can minimize the energy given some local nearest neighbor Hamiltonian. You might follow the steps below (but you definitely don't have to!). You might want to use a programming language which can deal with matrix operations natively, such as MATLAB or python with numpy/scipy. At each step, you should try to test the new functions.

   - In the following, N denotes the length of the chain.
   - First, think about the data structure for your tensors. Take into account that the very left and the very right tensor have different sizes – there might be ways to still treat all tensors on the same footing.
   - Write code to implement the isometric gauge around site 1, as well as a code to move the "center" of the isometric gauge by one site to the left and right. (You can probably use the "moving" function to perform the initialization.) Note that the "moving" function should only act locally and not scale with N.
   - Write a function which, given a tensor, computes the transfer operator, as well as a function to compute $\mathbb{E}_O = \sum_{ij} A^i \otimes \bar{A}^j \langle j|O|i\rangle$. Alternatively, you can write a function which given the tensor A, the operator O, and a vector $\vec{\rho}$, applies the transfer operator $\mathbb{E}_O$ to $\vec{\rho}$ (this will make improving the performance by changing the contraction order easier).
   - At this point, you might use the transfer operator to write a function for computing the normalization of the MPS, as well as the energy for a given Hamiltonian. (This is not necessary for later, but might be very useful for verifying that the program does what it should!) Hamiltonians of interest might be the transverse Ising model

     $$H = \sum_i \sigma_i^z \sigma_{i+1}^z + h\sigma_i^x$$

     or the Heisenberg model

     $$H = \sum_i \vec{\sigma}_i \cdot \vec{\sigma}_{i+1} = \sum_i \sigma_i^x \sigma_{i+1}^x + \sigma_i^y \sigma_{i+1}^y + \sigma_i^z \sigma_{i+1}^z \ .$$

   - Next, implement the routines to initialize and update the contributions of the Hamiltonians left and right of a certain position k, $L[k]$ and $R[k]$. (During the algorithm, when we are optimizing at position s, we want to have all $L[k]$ for $k < s$ and all $R[k]$ for $k > s$ available.) Write an algorithm which updates $L[k]$ when s is moved to the right (and $R[k]$ when s is moved to the left). This can also be used to initialize the $R[k]$ for the initial choice $s = 1$.
   - Write the routine to determine the quadratic form which gives the dependency of the energy on the tensor X at site s. This is given by $L[s]$, $R[s]$, and the two (groups of) terms in the Hamiltonian which overlap with the site s, and which have to be included separately. Next, you can use this funciton to build a routine which determines the X which minimizes the energy (by diagonalizing the quadratic form; you might also use Lanczos to only determine the smallest eigenvalue and its eigenvector which might be faster). (You might want to check that this works correctly by plugging in the optimal X and computing the energy: It should be equal to the smallest eigenvalue.)

- Now you can put things together: Initialize all the A^k's randomly (or with some guess you might have), initialize the isometric gauge, R and L (in this order, i.e., the gauge first! – why?) to $s = 1$, find the optimal X, replace $A^{[1]}$ by X, then move s one site to the right, repeat until you reach the right end, then turn around, etc..
- Store the energy after each step in the iteration. You should see that the energy decreases in each step, and after a few sweeps converges.
- Congratulations!

4. Things you might want to try with your code:

   - For a given model (such as Ising or Heisenberg), check the dependence of the final (converged) energy as a function of the bond dimension D (i.e., the size of the matrices). How is the convergence in D? For the Ising model, does it depend on the value of h?
   - If you want, you can exactly diagonalize the Hamiltonian for a small system and compare your numbers. (Using sparse matrices, sth. like $N = 20$ sites should be possible – this also means that using MPS only gets really interesting once N is considerably larger.)
   - For the Ising model, measure and plot the magnetization $M = \sum \sigma_z^i$ of the ground state as a function of h. What do you see? Can you explain this behavior? (The routine for such expecation values is essentially the same as the one for computing energies!)
   - Compute the energy per site for the Heisenberg model: I.e., solve the model for chains of various length and plot the ground state energy divided by the length as a function of the length.
   - Compute the staggered magnetization $M_\alpha = \sum (-1)^i \sigma_\alpha^i$ of the ground state as a function of h, for $\alpha = x, y, z$. (Why do we add the factor $(-1)^i$ in this case?) Try this starting from different initial configurations? What do you find? Also compute $\sum_\alpha M_\alpha^2$. Can you explain these observations?