ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computational Quantum Physics
Exercise 13

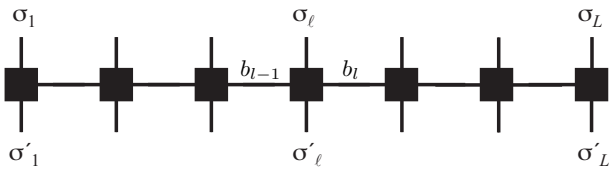
FS 16
Prof. S. Huber

Problem 13.1 Measure the Energy of a MPS using a MPO

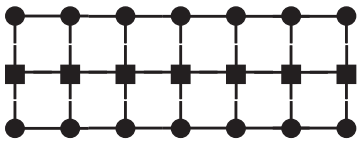
In this exercise we will measure the energy of a Matrix Product State (MPS) given its Hamiltonian in Matrix Product Operator (MPO) form. An MPO is the matrix product of four-legged matrices of the form $W^{\sigma_i'\sigma_i}_{b_{i-1}b_i}$

$$\hat{O}^{\{\sigma'\},\{\sigma\}} = W^{\sigma_1'\sigma_1} W^{\sigma_2'\sigma_2} \dots W^{\sigma_L'\sigma_L}. \tag{1}$$

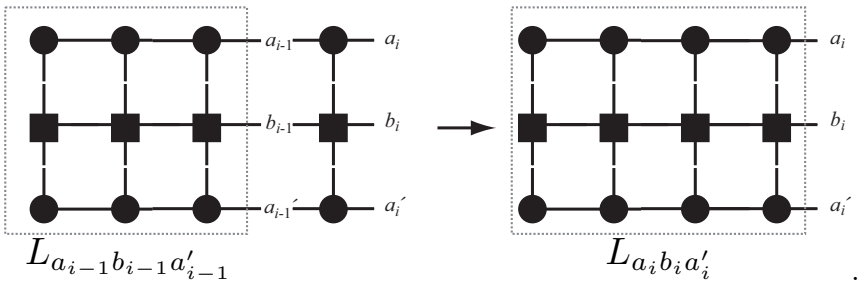
In the graphical notation it is represented as below:



We want to evaluate the energy expectation value of a Hamiltonian given in MPO form

$$\langle\psi|H|\psi\rangle \hat{=}$$  . (2)

For an efficient evaluation of above contractions use L-expressions (or equivalently R-expressions). The L-expression is build iteratively from the left via adding a single site in each step



.

When the last site is added to the L-expression it is equivalent to the energy expectation value in Eq. (2).

- We provide you with a working code `heisenberg_imaginary_time_evolution.ipynb` that calculates the groundstate of a Heisenberg chain via imaginary time evolution. Use this code to obtain the groundstate of the Heisenberg model as a MPS state.

- In our code the energy is measured as a sum of two-site operators. We want to measure the energy via a MPO. The Heisenberg model

$$H = \sum_{i=1}^{L-1} \left(\frac{J}{2} S_i^+ S_{i+1}^- + \frac{J}{2} S_i^- S_{i+1}^+ + J S_i^z S_{i+1}^z \right), \tag{3}$$

can be represented in MPO language by the following operator valued matrices

$$\hat{W}^{[i]} = \begin{bmatrix} \hat{I} & 0 & 0 & 0 & 0 \\ S_i^+ & 0 & 0 & 0 & 0 \\ S_i^- & 0 & 0 & 0 & 0 \\ S_i^z & 0 & 0 & 0 & 0 \\ 0 & (J/2)S_i^- & (J/2)S_i^+ & JS_i^z & \hat{I} \end{bmatrix}, \tag{4}$$

and for the first and last site

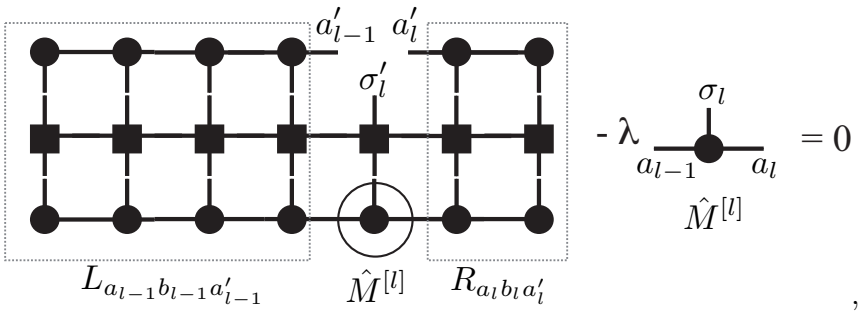
$$\hat{W}^{[1]} = \begin{bmatrix} 0 & (J/2)S_0^- & (J/2)S_0^+ & JS_0^z & \hat{I} \end{bmatrix}, \quad \hat{W}^{[L]} = \begin{bmatrix} \hat{I} \\ S_L^+ \\ S_L^- \\ S_L^z \\ 0 \end{bmatrix}. \tag{5}$$

Multiplication of above matrices, containing single-site operators, reveals how the Hamiltonian in Eq. (3) emerges. Implement the above MPO in our code example as a list of arrays.

- Measure the energy of the MPS obtained from imaginary time evolution using your MPO. First construct the L-expression iteratively. Check that you get the same energy as we in our sample code. When you succeed try to do the same using a R-expression build iteratively from the right.

Problem 13.2 Iterative Ground State Search using DMRG

The imaginary time evolution we provide is slow, takes many steps to converge and is plagued by a systematic error coming from the Trotter decomposition. In practice one would do DMRG, which converges much faster and is also more accurate.
In the single-site DMRG algorithm[1] one iteratively sweeps through the sites of the system and for each site minimizes the total energy with respect to the single site matrix $M^{\sigma_l}_{a_{l-1}al}$. The local energy minimization is mapped onto an eigenvalue problem which can be graphically depicted as below



,

where the lowest lying eigenvalue λ is the estimate for the groundstate energy. You can recognize that we used the iteratively build L- and R-expressions from the last exercise. The above graphical representation can be written down as an equation

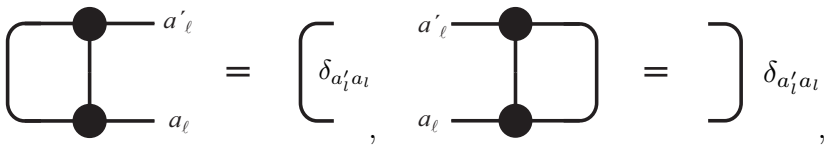
$$\sum_{\sigma_l, a_{l-1}, a_l} \left(L_{a_{l-1}b_{l-1}a'_{l-1}} W^{\sigma'_l \sigma_l}_{b_{l-1}b_l} R_{a_l b_l a'_l} \right) M^{\sigma_l}_{a_{l-1}a_l} - \lambda M^{\sigma_l}_{a_{l-1}a_l} = 0. \tag{6}$$

[1] Schollwöck, U. 2011, Annals of Physics, 326, 96

For an iterative eigenvalue solver, like the Lanzcos algorithm, you only need the "matrix-vector" product (the first part of above equation). If you decide to use the provided `scipy.sparse.linalg.eigsh` keep in mind that it expects $\hat{M}^{[l]}$ to be a vector, or one-dimensional array, not a three-dimensional array. This can be easily achieved via "combining indices" $M^{\sigma_l}_{a_{l-1}a_l} \hat{=} M_{(\sigma_l a_{l-1} a_l)}$ and you can use the provided `numpy.reshape` function for that.
However, Eq. (6) is only valid if we assume the MPS in a so-called "mixed canonical" form

$$|\psi\rangle \hat{=} \hat{A}^{[1]}\hat{A}^{[2]}\dots\hat{A}^{[l-1]}\hat{M}^{[l]}\hat{B}^{[l+1]}\dots\hat{B}^{[L]}, \tag{7}$$

where the A and B matrices are in a special canonized form such that



the A matrices are "left-normalized" and the B matrices "right normalized", or in equations for A matrices

$$\sum_{a'_{l-1},a_{l-1},\sigma_l} \delta_{a'_{l-1},a_{l-1}} A^{*\sigma_l}_{a'_{l-1}a'_l} A^{\sigma_l}_{a_{l-1}a_l} = \delta_{a'_l,a_l}, \tag{8}$$

and for B matrices correspondingly

$$\sum_{a'_l,a_l,\sigma_l} \delta_{a'_{l-1},a_{l-1}} B^{*\sigma_l}_{a'_{l-1}a'_l} B^{\sigma_l}_{a_{l-1}a_l} = \delta_{a'_{l-1},a_{l-1}}. \tag{9}$$

You can either use the functions for canonization provided in `heisenberg_imaginary_time_evolution.ipynb` or try it yourself. [2]

- Implement the "matrix-vector multiplication" from Eq. (6). Try again evaluating the total energy with the state of our sample code `heisenberg_imaginary_time_evolution.ipynb`, but now via solving the eigenvalue problem Eq. (6) using Lanczos. You can start with the first site, because the state is in right-canonized form when it comes out from our imaginary time evolution. When you get the correct total energy at the first site, try to do the same on an arbitrary site, bringing the state in a mixed canonical form first.

- Now we have everything to implement the full DMRG algorithm. The optimal algorithm is then:

  - Use an initial guess for $|\psi\rangle$ with random matrices. You need to right-normalize it, i.e. such that it consists of B-matrices only.
  - Calculate the R-experssion iteratively for all site positions $L-1$ through 1 iteratively.
  - *Right sweep:* Starting from site $l = 1$ through site $L-1$, sweep through the lattice to the right as follows: solve the standard eigenproblem by an iterative eigensolver for $M^{[l]}$, taking its current value as starting point. Once the solution is obtained, left-normalize $M^{[l]}$ into $A^{[l]}$, such that when before you had $\dots A^{[l-1]}M^{[l]}B^{[l+1]}\dots$ you then have $\dots A^{[l-1]}A^{[l]}M^{[l+1]}\dots$. The remaining matrices from the left-normalization are multiplied to $B^{[l+1]}$, such that it

[2] For detailed information how to do so read chapter 4.4 of Schollwöck, U. 2011, Annals of Physics, 326, 96

becomes the no more right-normalized $M^{[l+1]}$. It willl serve as the starting guess for the eigensolver for the next site. Build iteratively the L-expression by adding one more site. Move on by one site, $l \to l+1$, and repeat.

- *Left sweep:* Starting from site $l = L$ through site 2, sweep through the lattice to the left as follows: solve the standard eigenproblem by an iterative eigensolver for $M^{[l]}$, taking its current value as starting point. Once the solution is obtained, right-normalize $M^{[l]}$ into $B^{[l]}$, such that when before you had $\ldots A^{[l-1]}M^{[l]}B^{[l+1]}\ldots$ you then have $\ldots M^{[l-1]}B^{[l]}B^{[l+1]}\ldots$. Build iteratively the R-expression by adding one more site. Move on by one site, $l \to l-1$, and repeat.
- Repeat right and left sweeps until the energy is converged.