Exercise 10.1: TEBD for infinite MPS and Transfer Matrix

This exercise uses the provided files `a_mps.py,b_model.py,c_tebd.py,tfi_exact.py` from last week[1]. As discussed in the lecture, one can adjust TEBD to work directly on infinite, translation invariant systems with a unit cell of a few sites; L labels now the number of sites in the unit cell.

a) Make the necessary adjustments to the provided MPS and TEBD code such that it works for an infinite system.

Hint: You can either make the adjustments in the files directly, or (more elegantly) define classes derived from the existing ones, only overwriting the methods which need to be changed (and for `c_tebd.py` define new functions to replace the previous ones).

- Where we act on two sites i and $j = i+1$, we should take the $j = i+1$ modulo L to enforce the periodic structure.
  (`a_mps.MPS.get_theta2`, `c_tebd.update_bond`)
- The MPS has now L instead of $L-1$ bonds.
  (`a_mps.MPS.get_chi`, `bond_expectation_value`, `entanglement_entropy`)
- Similarly, we have to adjust `TFIModel.init_H_bonds` to generate L terms.
- Since the energy is extensive, we are interested in the energy *density* and should calculate the energy per site.
  (`TFIModel.energy`)

b) Run imaginary time evolution (as was done for the finite case) to find the ground state of the infinite system with a $L = 2$ unit cell. Compare your results for the energy with the function[2] `tfi_exact.infinite_gs_energy`.

c) One advantage of the infinite formulation is that one can extract the correlation length directly from the eigenvalues of the transfer matrix. For the $L = 2$ unit cell and right-canonical B^0, B^1 tensors (as saved in `MPS.Bs`), we define the transfer matrix as the contraction

$$T_{(aa'),(cc')} = \sum_{i_0,i_1,b,b'} B^{[0]i_0}_{ab} \overline{B^{[0]i_0}_{a'b'}} B^{[1]i_1}_{bc} \overline{B^{[1]i_1}_{b'c'}} \tag{1}$$

Write a function to contract the transfermatrix T for a given MPS.

d) Find the ground state of the infinite system for a few values of $g \in [0.5, 1.5]$ and plot the absolute value of the 3 largest (in magnitude) eigenvalues of the transfermatrix versus g. How can you interprete the result?

Hint: Use `scipy.sparse.linalg.eigs(..., which='LM')`.

[1] Download them again if you modified them last week!

[2] It uses an analytical formula obtained by mapping the system to free fermions, see P. Pfeuty, *The one-dimensional Ising model with a transverse field*, Annals of Physics **57**, 79 (1970).

Exercise 10.2: PEPS with simplified update and boundary MPS

This exercise uses the provided file `e_tps.py`, which imports a few parts parts of `a_mps.py`, `c_tebd.py` and should work regardlessw whether you modified these files in the previous exercise or not.

a) Read the file `e_tps.py`. It defines a class representing a tensor product state and functions to run the simplified update to find the ground state of the transverse field Ising model on a honeycomb lattice (`run_simplified_update`). Given the TPS, it is nontrivial to evaluate epxectation values, which can be done with `evaluate_exp_vals` using the method of infinite boundary MPS and TEBD. Run a whole simulation using the function `example_run_ising_honeycomb`.

b) Define the bond-operators for measuring the magnetization in x and z direction and write a function similar to `example_run_ising_honeycomb` measuring these operators as well.

c) Check the convergence with the different parameters: for different `chi_tps` = 2, 3, 4, ... check convergence with `chi_mps` and plot the energy and/or magnetization versus `chi_tps`.

d) Can you find the transition with g? Compare to the result of a Quantum-Monte carlo methods (Henk W. J. Bloete and Youjin Deng Phys. Rev. E 66, 066110), which yields a critical field $g_c = 2.13250(4)$.

Bonus) In case you still have plenty of time: Use the function `evaluate_exp_vals` to evaluate the partition function and magnetization of the 2D *classical* Ising model.