

## Exercise 7: Time evolving block decimation (TEBD)

This exercise uses the provided files `a_mps.py`, `b_model.py`, `c_tebd.py` (and for comparison some exact diagonalization code in `tfi_exact.py`).

- a) Read the code in the file `a_mps.py`. This file defines the class `MPS` in an object-oriented approach. In short, defining the class is defining a “type” which collects data in attributes (e.g. `MPS.Bs`, `MPS.L`) and has methods (e.g. `MPS.site_expectation_value`) which can use the attributes (referenced with the special first argument `self`) for calculations. Generate an *instance* of the `MPS` class representing the state  $|\uparrow\uparrow \dots \uparrow\rangle$  with the function `init_spinup_MPS`, for the start with  $L = 14$  sites. Check that the (site) expectation values of the operators  $\sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$  and  $\sigma^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$  give the expected values.
- b) Write a function similar to `init_spinup_MPS`, but initialize an `MPS` for the state  $|\rightarrow\rightarrow \dots \rightarrow\rangle$ . Check the expectation values again.  
*Hint:* This state is also a product state of  $|\rightarrow\rangle = \frac{1}{\sqrt{2}}(|\uparrow\rangle + |\downarrow\rangle)$ , so the singular values remain the same and the shape of each `B` is still  $(1,2,1)$ . You should expect rounding errors of the order of machine precision  $\approx 10^{-15}$ .
- c) Read the file `b_model.py`. It defines a class representing the transverse field ising model for a given choice of coupling parameters. Calculate the energy for  $L = 14$ ,  $J = 1$  and  $g \in \{0.5, 1, 1.5\}$  for each of the above defined two product states.
- d) Read the file `c_tebd.py`, which implements the time evolving block decimation. Call the function `example_TEBD_gs_finite`, which performs an imaginary time evolution to project onto the ground state. (As we will see next week, DMRG is a better alternative to find ground states, but since we only discussed TEBD in class so far, we will use this method.)
- e) **Global quench.** Calculate the real time evolution of the spin-up state,  $|\psi(t)\rangle = e^{-iHt} |\uparrow \dots \uparrow\rangle$  for  $L = 14$ ,  $J = 1$ ,  $g = 1.5$ . As a first choice, use the parameters `chi_max = 30`, `eps=1.e-10`. Evolve up to time  $t = 10J$ . Measure and plot the total magnetization  $\sum \sigma_i^z$  and the half-chain entropy as a function of time  $t$ .  
*Hint:* Don't forget the imaginary  $i$  for the time step when calculating `U_bonds`. For the measurements, you can use the methods `MPS.site_expectation_value` and `MPS.entropy`.
- f) By plotting the same expectation values for different parameter choices, check whether (or up to which time) the results are converged in `dt` and `chi_max`, for the small chain of  $L = 14$  and for a larger chain with  $L = 50$ .

- g) Write a function replacing `c_tebd.run_TEBD` to run TEBD with a second-order (in  $dt$ ) Trotter-decomposition. Regenerate the plot of f) with the second-order TEBD.

*Hint:* E.g. for `N_steps = 3`, the first order expansion evolves with

$$e^{-iH^E dt} e^{-iH^O dt} e^{-iH^E dt} e^{-iH^O dt} e^{-iH^E dt} e^{-iH^O dt}, \quad (1)$$

while the second order expansion would read

$$e^{-iH^E \frac{dt}{2}} e^{-iH^O dt} \underbrace{e^{-iH^E \frac{dt}{2}} e^{-iH^E \frac{dt}{2}}}_{=e^{-iH^E dt}} e^{-iH^O dt} \underbrace{e^{-iH^E \frac{dt}{2}} e^{-iH^E \frac{dt}{2}}}_{=e^{-iH^E dt}} e^{-iH^O dt} e^{-iH^E \frac{dt}{2}} \quad (2)$$

Therefore, you need another argument `U_bonds_half_dt`.

- h) **Local quench.** Calculate the (approximate) ground state  $|\psi_0\rangle$  of a  $L = 50$  chain using `c_tebd.example_TEBD_gs_finite` for  $g = 1.5$ . Apply the local operator  $S_{n_0}^x$ , where  $n_0$  is the index of a site in the center of the chain, by multiplying it to the corresponding  $B$  tensor of the ground state<sup>1</sup>. Perform a real time evolution of this initial state. Measure the entropy for cuts on the different bonds. Create a color-plot showing the entropy versus time  $t$  on the  $y$ -axis and the bond of the cut  $n$  on the  $x$ -axis. You should observe a light-cone structure.

---

<sup>1</sup>Since  $S_{n_0}^x$  is unitary, the canonical form is preserved and you don't need to worry about that. Be warned that if you apply a generic operator like  $S_{n_0}^+$ , you need to restore the canonical form before starting the time evolution