Ecole Normale Supérieure de Lyon Université de Lyon
Master: Sciences de la Matière, A. A. 2024-2025

Computational Quantum Physics (M2) *T. Roscilde, F. Mezzacapo, F. Caleca, S. Bocini*

General rule for the TD sessions: the TD sessions are fully *hands-on* – namely, in every TD session you are supposed to *write computer codes* to learn about the phenomenology and efficiency of important algorithms for problems in quantum physics. You should choose a programming platform (Python, Matlab, Mathematica, C, Fortran, etc.), and you should be able to plot your results in the form of two-dimensional functions $y = f(x)$ (using matplotlib in Python, the plotting utilities of Matlab and Mathematica, Gnuplot, etc.), or occasionally in a more complicated form. We assume that you have some familiarity with at least one programming platform; if this is not the case, you should be able to familiarize yourself rapidly *e.g.* by attending online tutorials.

TD3: Matrix product states

In this exercise sheet, we shall get familiarized with the manipulation of matrix product states, which are an efficient Ansatz to studying weakly entangled systems in one dimension (and beyond). We will focus on an ensemble of $S = 1/2$ spins.

1 Performing the singular-value decomposition of a matrix

1.1

We consider a rank-3 tensor $M^{(\sigma)}_{\alpha,\beta}$ with $\sigma = +1, -1$, $\alpha = 1, ..., D_1$, $\beta = 1, ..., D_2$. Choose two bond dimensions D_1 and D_2, and build a random tensor M with *e.g.* random real entries.

1.2

Define a function to reshape the tensor into a $2D_1 \times D_2$ rectangular matrix $M_{(\sigma,\alpha),\beta}$. Representing the tensor as a matrix is necessary to perform the SVD (see next point). Define also a function to perform the inverse step, i.e. to reshape a $2D_1 \times D_2$ matrix into a $D_1 \times 2 \times D_2$.

1.3

Perform numerically the singular-value decomposition (SVD) of the matrix, so as to build the following matrices:

- a unitary matrix U with shape $2D_1 \times D_{\min}$,
- the $D_{\min} \times D_{\min}$ singular-value matrix S,
- a $D_{\min} \times D_2$ unitary matrix $V^\dagger$,

where $D_{\min} = \min(2D_1, D_2)$. Convert the matrix U into a tensor A in such a way to obtain the representation $M^{(\sigma)}_{\sigma,\beta} = \sum_\gamma A^{(\sigma)}_{\alpha\gamma} S_{\gamma\gamma} (V^\dagger)_{\gamma\beta}$. (You can check that it reproduces the original matrix.)

2 Bringing a matrix product state to the left-canonical form

2.1

Choose a system size N (not too big!), and a bond dimension D. Build $N-2$ random tensors $M_2, M_3, ...M_{N-1}$ of size $D \times 2 \times D$; a $1 \times 2 \times D$ matrix M_1; and a $D \times 2 \times 1$ matrix M_N. The matrices define then a matrix product state (MPS):

$$|\Psi\rangle = \sum_{\sigma_1\sigma_2...\sigma_N} M_1^{(\sigma_1)} M_2^{(\sigma_2)} ... M_N^{(\sigma_N)} |\sigma_1\sigma_2... \sigma_N\rangle \tag{1}$$

where $|\sigma_1\sigma_2...\sigma_N\rangle$ is a spin state in the computational basis.
Do you understand why the above construction is in fact redundant for a lattice of N spins with open boundary conditions?

2.2

Bring the MPS into a left canonical form by successive SVDs of the matrices, following the sequence:

- via SVD bring M_1 to the form $A_1 S_1 V_1^\dagger$; store A_1;
- build $\tilde{M}_2 = S_1 V_1^\dagger M_2$;
- via SVD bring $\tilde{M}_2$ to the form $\tilde{M}_2 = A_2 S_2 V_2^\dagger$; store A_2;
- ...
- continue until you get to $\tilde{M}_N = S_{N-1} V_{N-1}^\dagger M_N$. When applying the SVD to it (and storing A_N), you should be able to read out the *norm* of the state, $\langle\Psi|\Psi\rangle$.

2.3

The $A_1, A_2, ..., A_N$ tensors that you stored give you the left-canonical form of the MPS. (You can check that.)

3 Bonus: bringing a matrix product state to the right canonical form

Now we want to do the same "canonicalization" operation, but starting from the right! Such a process is analogous to the canonicalization to the left and could be performed on any MPS (the starting MPS does not need to be in left canonical form). However, we will apply the right canonicalization to an MPS that is already in its left canonical form. This allows to easily extract the entropy of the system across any possible partition.

3.1

The reduction to right-canonical form works as follows (keep in mind that this time the physical index should be *fused* with the right index, so you should adapt the functions of point 1.2 appropriately):

- via SVD bring A_N to the form $U_{N-1} S'_{N-1} B_N$; store S'_{N-1};
- build $\tilde{A}_{N-1} = A_{N-1} U_{N-1} S'_{N-1}$;
- via SVD bring $\tilde{A}_{N-1}$ to the form $\tilde{A}_{N-1} = U_{N-2} S'_{N-2} B_{N-1}$; store S'_{N-2};

- ...
- continue until you get to the SVD of $\tilde{A}_2$, giving you the S'_1 matrix, that you will store.

3.2

Take the singular value matrices $S'_1, S'_2 ... S'_{N-1}$ that you stored, and examine their diagonal elements, $(S'_n)_{\alpha\alpha}$. Out of them build the probabilities $p_\alpha^{(n)} = (S'_n)^2_{\alpha\alpha}$, and check whether $\sum_\alpha p_\alpha^{(n)} = 1$. If not, then normalize them.
From the probabilities $p_\alpha^{(n)}$, calculate the entanglement entropy:

$$\mathcal{S}_n = -\sum_\alpha p_\alpha^{(n)} \log p_\alpha^{(n)} \tag{2}$$

for the subsystem of size n (or for its complement $N - n$).
Plot $\mathcal{S}_n$ vs. n – what do you observe?

4 Bonus: scaling properties of entanglement in random MPS

This last part wants you to look at what happens when you increase the system size – in case you made it until this point!

4.1

Repeat what you did in the part 3 for a sequence of system sizes – e.g. $N = 4, 6, 8, ...$
For each of them store the function $\mathcal{S}_n^{(N)}$. Make a plot of the various functions $\mathcal{S}_n^{(N)}$. What can you observe? Can the entropies of the state you generated reach their maximum value, given by $\log D$?