Exercise 7.1: Area law - ground state versus random state

We consider a chain of an even length L of spin-$\frac{1}{2}$ degrees of freedom; the Hilbert space has thus dimension 2^L. Moreover, we consider half-chain bipartitions into regions A (left half) and B (right half).

a) Using exact diagonalization and/or the lanczos algorithm from last week(s) exercise sheets, generate the ground state of the transverse field Ising model for $L = 14, g = 1.5, J \equiv 1$. The state should be given by 2^L complex numbers ψ_i (arranged into a 1D array) such that

$$|\psi\rangle = \sum_{i=0}^{2^L-1} \psi_i |i\rangle = \sum_{j_1,\dots,j_L \in \{0,1\}} \psi_{j_1,\dots,j_L} |j_1, \dots, j_L\rangle , \tag{1}$$

where we identify the basis states with binary representations of the index i (c.f. sheet 5), e.g., for $L = 6$ we have

$$|\uparrow\downarrow\uparrow\downarrow\downarrow\uparrow\rangle \equiv |j_1 = 0, j_2 = 1, j_3 = 0, j_4 = 1, j_5 = 1, j_6 = 0\rangle = |i = 010110_2 = 22_{10}\rangle .$$

b) Convince yourself that `psi_ab = np.reshape(psi_i, (2**(L//2), 2**(L//2)))` arranges the state into the form

$$|\psi\rangle = \sum_{a=0}^{2^{\frac{L}{2}}-1} \sum_{b=0}^{2^{\frac{L}{2}}-1} \psi_{a,b} |a\rangle_A |b\rangle_B , \tag{2}$$

where $|a\rangle_A$ labels the basis states in the left half A, and $|b\rangle_B$ in the right half, respectively. (The result $\psi_{a,b}$ is a 2D array, i.e., a matrix with the same number 2^L of entries as in the original state.)

Hint: In general, $i = a \cdot 2^{\frac{L}{2}} + b$, e.g, for the above example,

$$|\uparrow\downarrow\uparrow\downarrow\downarrow\uparrow\rangle = |i = 010110_2 = 010_2 \cdot \underbrace{1000_2}_{=8_{10}=2^3} + 110_2\rangle = |a = 010_2\rangle_A |b = 110_2\rangle_B . \tag{3}$$

c) Find the Schmidt decomposition $|\psi\rangle = \sum_\alpha \lambda_\alpha |\alpha\rangle_A |\alpha\rangle_B$ by performing a singular value decomposition of $\psi_{a,b}$.

Hint: Here and in the following: if you get a `LinAlgError: SVD did not converge`, try using `scipy.linalg.svd(...., lapack_driver='gesvd')`.

d) Sort the Schmidt values descending and plot the largest 20 Schmidt values λ_α versus the index α. Use a logarithmic y-axis for the Schmidt values λ_α.

e) Generate a random state drawn from the Haar measure of the full Hilbert space. Calculate its Schmidt spectrum $\tilde{\lambda}_\alpha$ as in c) and include it into the plot of d).

   Hint: To draw a state from the Haar measure means to choose a normalized vector (uniformly) pointing in a random direction. You get such a vector if you draw the real and imaginary part of each coefficient from independent normal (gaussian) distributions and finally normalize it.

f) Write a function which calculates the entanglement entropy $S = -\sum_\alpha \lambda_\alpha^2 \log(\lambda_\alpha^2)$ (when given λ_α).

g) Calculate the entanglement entropy S of each

   - the ground state in the paramagnetic phase ($g = 1.5J$),
   - the ground state at the critical point ($g = J$)
   - the ground state in the ferromagnetic phase ($g = 0.5J$), and
   - a random state drawn from the Haar measure

   for various system sizes $L = 6, 8, 10, 12, 14$. Plot S versus L. What do you observe?

Exercise 7.2: Matrix product state (MPS) basics

An MPS is just a different representation of eq. (1),

$$|\psi\rangle = \sum_{j_1,\dots,j_L} \sum_{\alpha_2=0}^{\chi_2-1} \cdots \sum_{\alpha_L=0}^{\chi_L-1} M^{[1]j_1}_{\alpha_1\alpha_2} M^{[2]j_2}_{\alpha_2\alpha_3} \dots M^{[L]j_L}_{\alpha_L\alpha_{L+1}} |j_1, j_2, \dots, j_L\rangle . \tag{4}$$

Here, α_1 and α_{L+1} are dummy indices with a single entry only ($\chi_1 = \chi_{L+1} = 1$), such that the matrix product of the Ms for a given index combination $(j_1, \dots, j_L)$ yields a 1×1 matrix which we can identify as the coefficient $\psi_{j_1,\dots,j_L}$.

a) Get the ground state of the transverse field Ising model for $L = 14, g = 1.5, J \equiv 1$ as in Exercise 7.1 a). Make sure it is normalized to $\langle\psi|\psi\rangle = 1$.

b) Write a function `compress(psi, L, chimax)`, which takes the state, length of the chain and the maximal desired bond dimension `chimax` as input and compresses the state into MPS form using successive SVDs. It should return a list of L numpy arrays, namely the $M^{[n]}$, each with 3 indices $(\alpha_n, j_n, \alpha_{n+1})$.

   Hint: Let me define the indices $R_n = (j_n, j_{n+1}, \dots, j_L)$, such that $R_1 \equiv i$.

   First, introduce the dummy index α_1 with a reshape of psi into shape $(1, 2^L)$ for the indices α_1, R_1 Then you can perform a loop over n which generates one $M^{[n]}$ in each iteration by splitting $\psi_{\alpha_n,R_n} = M^{[n]j_n}_{\alpha_n,\alpha_{n+1}} \psi_{\alpha_{n+1},R_{n+1}}$. The necessary steps for this iteration are:

   - Reshape ψ_{α_n,R_n} into shape $(\chi_n \cdot 2, \dim(R_{n+1}))$. Note that $\dim(R_n) = 2^{L-(n-1)}$. This corresponds to a regrouping of the indices into $L_n \equiv (\alpha_n, j_n)$ and $R_{n+1} = (j_{n+1}, j_{n+2}, \dots, j_L)$.
   - Perform an SVD to split $\psi_{L_n,R_n} = \sum_{\alpha_{n+1}} M_{L_n,\alpha_{n+1}} \lambda_{\alpha_{n+1}} \tilde{\psi}_{\alpha_{n+1},R_{n+1}}$.

- If necessary, truncate to smaller dimension $\chi_{n+1} \leq \chi_{max}$. With numpy arrays, this can be done as follows:

```
keep = np.argsort(lambda_n)[::-1][:chimax]
M_n = M_n[:, keep]
lambda_ = lambda_n[keep]
psitilde = psitilde[keep, :]
```

- Reshape $M^{[n]}$ into shape $(\chi_n, 2, \chi_{n+1})$ to obtain the indices $(L_n, \alpha_{n+1}) \to (\alpha_n, j_n, \alpha_{n+1})$.
- Re-absorb the Λ_n into $\psi_{\alpha_{n+1},R_{n+1}} = \lambda_{\alpha_{n+1}} \tilde{\psi}_{\alpha_{n+1},R_{n+1}}$ using
  `psi = lambda[:, np.newaxis] * psitilde[:, :]`

The final $\psi_{\alpha_{L+1},R_{L+1}}$ is just a 1×1 matrix containing at most a phase (and overall norm of ψ, you can simply discard it.

c) What is the maximally necessary bond dimension for $L = 14$? Call `compress()` for the ground state with χ_{max} larger than that to get an *exact* MPS representation $|\psi_{ex}^{MPS}\rangle$.

d) Call `compress()` again with $\chi_{max} = 10$ to get a compressed MPS $|\psi_{compr}^{MPS}\rangle$. Compare the number of floats stored in both MPS.

Hint: The number of elements in a numpy array M are given by `M.size`.

e) Write a function to calculate the overlap between two MPS. Recall from class that there is an inefficient way (first contracting the bra and ket on top and bottom separately and finally contracting over the $j_1, \dots j_n$) and an efficient way (contracting from left to right); implement the efficient one! Check that the overlap $\langle\psi_{ex}^{MPS}|\psi_{ex}^{MPS}\rangle$ is (close to) 1 and calculate the overlap $\langle\psi_{ex}^{MPS}|\psi_{compr}^{MPS}\rangle$.

f) Write the state $|\uparrow\uparrow\cdots\uparrow\rangle$ as an MPS with bond dimension 1. Calculate the overlap of this state with the ground state (using MPS techniques).