Quantum information and Computing
Ex. 10

Andrea Nicolai - 1233407

January 5, 2021

Abstract

In this work starting from a quantum many-body system of N spin-1/2 particles in a one-dimensional lattice and in a transverse field of strength λ, we were asked to use the Real-Space Renormalization Group to compute the ground state as $N \to \infty$.

Theory

The Hamiltonian for a N spin-1/2 particles on a 1-dim lattice is described by the Hamiltonian:

$$\hat{H_N} = \lambda \sum_i^N \sigma_z^i + \sum_i^{N-1} \sigma_x^i \sigma_x^{i+1} \tag{1}$$

where:

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2}$$

are the *Pauli* matrices and λ represents a linear chain of N interacting spins 1/2 in presence of an external field of intensity λ.
RSRG (Real-Space Renormalization Group) method is an approximation method based on the hypothesis that the ground state of a system is composed of low-energy states of the system's (non-interacting) bipartitions. Consequently, it is indeed possible to introduce an algorithm that allows describing the ground state properties of many-body quantum systems with large sizes N, eventually up to the thermodynamical limit corresponding to the fixed point of the renormalization flow. This method consists in two different parts: the initialization and, later, the iterating one.

1. The initialization consists to consider a system composed of N sites that can be studied in an exact numerical way. Therefore a Hamiltonian $\mathcal{H}_N : \mathbb{C}^{2^N} \to \mathbb{C}^{2^N}$ for N particles of the kind as in eq 1 is introduced. Moreover, there is the interaction term between the left and the right part of the bipartite systems, namely $A_{left} = \left(\bigotimes_{i=1}^{N-1} \mathbb{1}_2\right) \otimes \sigma_x$ and $B_{right} = \sigma_x \otimes$

$\left(\bigotimes_{i=1}^{N-1} \mathbb{1}_2\right)$. The Hamiltonian for the bipartite system, using the matrices we have just introduced is:

$$\mathcal{H}_{2N} = \overbrace{\mathcal{H}_N \otimes \left(\bigotimes_{i=1}^{N} \mathbb{1}_2\right)}^{\text{left subsystem}} + \overbrace{\left(\bigotimes_{i=1}^{N} \mathbb{1}_2\right) \otimes \mathcal{H}_N}^{\text{right subsystem}} + \overbrace{A_{left} \otimes B_{right}}^{\text{interaction term}} \tag{3}$$

where $\mathcal{H}_{2N} : \mathbb{C}^{2^{2N}} \to \mathbb{C}^{2^{2N}}$.

2. Diagonalize $\mathcal{H}_{2N}$ and obtain its eigenvalues $\{E_i\}$ $i = 1, ..., m$ and correspondent eigenvectors $\{|E_i\rangle\}$ $i = 1, ..., m$ stored in ascending order, where it is chosen $m = 2^N$. Using these eigenvectors, the projector P onto the lowest m eigenstates is $P = \sum_{i=1}^{m} |E_i\rangle\langle E_i|$. In particular, it is a rectangular matrix $[2^{2N} \times 2^N]$ that projects the Hilbert space on the subspace spanned by the first m low-energy laying eigenstates. Therefore, the Hamiltonian $\mathcal{H}_{2N}$ is projected onto this subspace and its dimensionality is reduced back to the previous $\mathcal{H}_N$:

$$\mathcal{H}_N^{proj} = P^\dagger \mathcal{H}_{2N} P \tag{4}$$

One should keep in mind that also the interaction terms have to be projected onto the new subspace according to:

$$A_{left}^{proj} = P^\dagger [A_{left} \otimes \left(\otimes_{i=1}^{N} \mathbb{1}_2\right)] P \tag{5}$$

$$B_{right}^{proj} = P^\dagger [\left(\otimes_{i=1}^{N} \mathbb{1}_2\right) \otimes B_{right}] P \tag{6}$$

3. The Hamiltonian for $2N$ particles can be rewritten as before in the new coordinates as it follows, formally equal as before:

$$\mathcal{H}_{2N} = \overbrace{\mathcal{H}_N^{proj} \otimes \left(\bigotimes_{i=1}^{N} \mathbb{1}_2\right)}^{\text{left subsystem}} + \overbrace{\left(\bigotimes_{i=1}^{N} \mathbb{1}_2\right) \otimes \mathcal{H}_N^{proj}}^{\text{right subsystem}} + \overbrace{A_{left}^{proj} \otimes B_{right}^{proj}}^{\text{interaction term}} \tag{7}$$

4. Following exactly the same procedure stated in points $2, 3$, the system described at every iteration increases its size exponentially fast: indeed the number of particles considered at every iteration is $N \to 2N$, despite the dimension of the Hamiltonian representation is kept constant to $m = 2^N$. The total number of iterations is a parameter one may want to tune.

The prediction of Mean-Field for the energy density of the ground state of system takes the value in function of the external field λ:

$$\begin{cases} e = -1 - \lambda^2/4 & \lambda \in [-2, +2] \\ e = -|\lambda| & \lambda \notin [-2, +2] \end{cases} \tag{8}$$

Code Development

The code developed accepts as input two parameters: the maximum number of particles from which the algorithm starts from and will iterate from $N = 2$ up to this last number, and as second parameter the maximum number of iterations for the RSRG algorithm. Once these inputs are checked, for every number of particles included between $N = 2$ and the input $N = N_{max}$ that has to be less than 5 for hardware limit, the program fixes a value for $\lambda \in [0, 3]$ and starts the algorithm. In order to initialize the system (step 1), the code implemented in the last assignment was used for the Hamiltonian $\mathcal{H}_N$. A_{left} and B_{right} were initialized simply applying the definition using a chain tensor products and the Pauli matrix σ_x in the proper order. Therefore the $\mathcal{H}_2 N$ was computed and then diagonalized, according to step 2. From its eigenvectors the projector is built and used to compute the other matrices into the new basis, as stated in the aforementioned list at point 3. In the following code we present what occurs during every iteration of the RSRG algorithm, specifically at points 2 and 3, once we have proceeded to initialize the system.

```
DO jj = 1, n_iter
!Compute the first term of the hamiltonian
!After 1st iteration, Hamiltonian will be the projected one
H_left = RIGHT_APPLY_IDENTITY(Hamiltonian, d_dim, n_particles)
!Compute the second term of the hamiltonian
!After 1st iteration, Hamiltonian will be the projected one
H_right = LEFT_APPLY_IDENTITY(Hamiltonian, d_dim, n_particles)
!Compute the last term of the Hamiltonian (interaction term)
!After the 1st iteration A_left and B_right will be projected
AB_mat = TENSOR_PRODUCT(A_left, B_right)
!Create the new 2N Hamiltonian and update it
Hamiltonian_2N = INIT_CMAT_VALUE(d_dim**(2*n_particles), &
& d_dim**(2*n_particles), DCMPLX(0d0,0d0))
Hamiltonian_2N%element = H_left%element + H_right%element + &
&                        AB_mat%element
!Create a dummy matrix to be diagonalized
eigen_mat = INIT_CMAT_VALUE(Hamiltonian_2N%dims(1), &
& Hamiltonian_2N%dims(2), DCMPLX(0d0,0d0))
eigen_mat%element = Hamiltonian_2N%element
CALL HERMITIAN_EIGENVECTORS(eigen_mat%element, &
& eigen_mat%dims(1), array_eigenvalues, info)
!Build the projector
projector = INIT_CMAT_VALUE(Hamiltonian_2N%dims(1), &
& Hamiltonian%dims(2), DCMPLX(0d0,0d0))
projector%element = eigen_mat%element(:,1:d_dim**n_particles)
!Project the Hamiltonian into the new basis
Hamiltonian = CHANGE_OF_BASIS(Hamiltonian_2N, projector)
!Left interaction term into the new basis
A_left  = RIGHT_APPLY_IDENTITY(A_left, d_dim, n_particles)
A_left  = CHANGE_OF_BASIS(A_left, projector)
!Right interaction term into the new basis
```

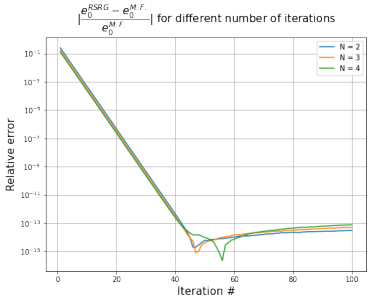
```
B_right = LEFT_APPLY_IDENTITY(B_right, d_dim, n_particles)
B_right = CHANGE_OF_BASIS(B_right, projector)
END DO
```

All debugging and dumping to screen part is neglected for a matter of space, but nevertheless is present: specially to check the Hermitianity of the matrices and their shape. Results are printed in output into two different files, one for the ground state density energy at every iteration ($\lambda = 0$), and the other for the ground state density energy for different λ, each of these files named according to the number of particles present at the beginning of the algorithm.

Results

The code was run by using a Python script which sets the maximum number of particles equal to 4 for a matter of time, and total 50 iterations. Diagonalization of the $\mathcal{H}_{2N}$ matrix would already require a long time for every iteration when starting with $N = 5$. However, this value would also have been the maximal value that my laptop would allow, because of its RAM that would saturate when storing the matrices related to $2N$ particles. Dimension for the Hilbert space was set to $d = 2$, dealing we with 1/2-spin particles.

Fig. 1: Relative error of the ground state for different number of starting particles wrt prediction of the mean field. $\lambda = 0$.



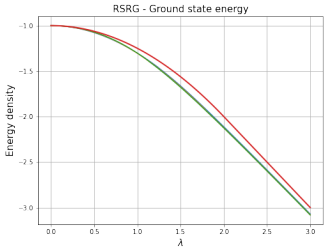
A first attempt was made to see after how many iterations the value for the ground state e_0 returned by the algorithm would converge to the MF prediction. This was done in order to tune the number of iterations and try to have a more efficient code. Result for a maximum of 200 iterations is shown in figure 1. As one can see the relative error converges to a minimum around 50 iterations, and the exact position of this optimal number varies according to the starting number of particle. Moreover precision of the prediction varies according to the starting number of particles, too, and for $N = 4$ we reach the maximum precision ($\sim 10^{-16}$) one can have working using double precision variables. When behaviour of the curve changes and reaches this minimum for the relative error, one can think that the random numbers "further that the precision digit" start playing a role, telling us that we have reached the maximal precision. However, it is a little price to pay given that a result precise up to 10^{-13} is still a good estimate. In addition, from theory we know that energy scales as the number of the particles described by the system, which in turn scales exponentially at every iteration. This is why we observe a linear behaviour for the relative error at the beginning (that is in *log* scale), since we are approaching more and more the thermodynamical limit. For

instance, at $80-th$ iteration and starting with $N = 2$, number of particles considered is larger than Avogadro's number $N_{part} \sim 2^{80} \sim 10^{24} \gtrsim N_A \sim 6 \cdot 10^{23}$. Number of iterations needed to overcome this limit is obviously less as long as the number of starting particles is increased, for instance for $N = 4$ it halves to $N_{iter} = 40$. In order to deal with these large numbers, all integers were explicitly cast to double precision.

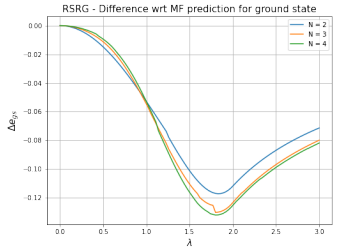
Once fixed number of iterations arbitrarily for all $N_{start} = 2, 3, 4$ to 50, we compute the ground state and compare it to the MF prediction, considering both its value (see 2a) and its difference wrt it (see fig.2b). The interval for values taken by λ, $\lambda \in [0, 3]$, is divided into 100 smaller intervals: hence its resolution is $\Delta\lambda = 0.03$.

One should note from fig. 2a that the trend followed by the RSRG results are similar to the one predicted by MF result, which we can confirm it gives us a qualitative overview on what physically happens in our system.

From fig. 2b, instead, the results of RSRG procedure are really close to the MF approximation for $\lambda \sim 0$, and as long as this parameter increases they tend to differ. In addition for $\lambda \sim 1$ all results obtained thanks to RSRG coincide regardless the number of the particles we started from. There is moreover a specific value for $\lambda \sim 1.7$ where RSRG this difference is maximal for almost all N_{start}. For even larger λ it is suggested that this difference might decrease, and in addition the larger N the smaller this difference.



(a) Ground state energy density for RSRG algorithm and MF prediction for different λ. Number of iterations is set to 50.



(b) Difference between the ground state energy density computed by RSRG algorithm and its mean field prediction for different λ. Number of iterations is set to 50.

Self Evaluation

In this work we learnt how to numerically apply the RSRG algorithm to a quantum many-body system, in order to evaluate the ground state of the system as the number of the particle increases at every iteration. As a further step one may want to increase the number of starting particles and proceed with the same analysis. Additionally one may want to use the INFINITE DMRG algorithm and see how and whether results differ.