Quantum Information and Computing

prof: *Simone Montangero*

Report of exercise 10

Real space and infinite density matrix Renormalization Group for the transverse field Ising model

Alberto Bassi

January 10, 2021

Abstract

The aim of this exercise is to solve the transverse field Ising model in the thermodynamic limit by means of the Real Space Renormalization Group (RSRG) algorithm and the Infinite Density Matrix Renormalization Group (IDMRG) algorithm.

Theory

Let us consider the N spins hamiltonian acting on $\mathcal{H}_N = \otimes_{k=1}^{N}\mathcal{H}$, where $\mathcal{H} \equiv \mathbb{C}^2$ is the one-body spin 1/2 Hilbert space,

$$\hat{H}_N = \lambda \sum_{i=1}^{N} \sigma_z^i + \sum_{i=1}^{N-1} \sigma_x^i \sigma_x^{i+1} \,, \tag{1}$$

where σ_x and σ_z are the Pauli matrices.
Generally, given an operator acting on a single body Hilbert space, its action on Hilbert spaces' tensor product can be obtained by tensor multiplication with identities acting on the other Hilbert spaces. Namely, we have

$$\begin{aligned} \sigma_z^i &= \mathbb{1}_{i-1} \otimes \sigma_z \otimes \mathbb{1}_{N-i} \,, \\ \sigma_x^i \sigma_x^{i+1} &= \mathbb{1}_{i-1} \otimes \sigma_x \otimes \sigma_x \otimes \mathbb{1}_{N-i-1} \,, \end{aligned} \tag{2}$$

where $\mathbb{1}_k$ is the identity acting on $\mathcal{H}_k$.
The matrix Equation 1 is exactly diagonalizable when the number of spins is low (around $N = 12$ spins with a home laptop, as seen in the previous assignment). On the other hand there are many approximates ways to infer about the ground state energy density. Theoretically, one the first approaches is a Mean Field(MF) solution, that consists in neglecting correlations among spins, with the ansatz that the total wave function is written as the tensor product of 1 spin wave functions

$$|\psi_{MF}\rangle = \otimes_{i=1}^{N} |\psi\rangle \ . \tag{3}$$

This leads to the ground state energy density

$$g_{GS} = \begin{cases} -1 - \lambda^2/4 & \lambda \in [-2:2] \\ -|\lambda| & otherwise \end{cases} \tag{4}$$

However, even if MF solution is a good approximation in the thermodynamic limit and with weak fields, as we will see, it fails to predict correctly the point of quantum phase transition, which should be at $\lambda = 1$. Better results can be obtain with two numerical approaches which allow us to enlarge the system while keeping the resources needed fixed in reaching the thermodynamic limit.

RSRG

The main idea is to start from a sufficiently small system to be diagonalized exactly. Then, we will take the lowest energy eigenvectors with the hypothesis that the ground state of the doubled system is a linear composition of only low eigenstates of the right and left subsystems. Then, the system will be doubled again in order to reach the thermodynamic limit while keeping the computational resources, such as the dimension of the matrix to be diagonalized ($2N$), constant.
The algorithm consist of the following steps:

0. Take the maximum N such that the $2N$ spins Hamiltonian $\hat{H}_{2N}$ is diagonalizable. Construct the left Hamiltonian $\hat{H}^{(0)}$ and the right Hamiltonian $\hat{H}^{(0)}$, each of them Ising Hamiltonians of N spins. At the beginning, the interaction term between the left and the right system is $\hat{H}^{(0)}_{int} = \hat{L}^{(0)} \otimes \hat{R}^{(0)}$, where $\hat{L}^{(0)} = \mathbb{1}_{N-1} \otimes \sigma_x$ and $\hat{R}^{(0)} = \sigma_x \otimes \mathbb{1}_{N-1}$.

1. Build the total $2N$ spins Hamiltonian

$$\hat{H}^{(k)}_{2N} = \hat{H}^{(k)}_{L} \otimes \mathbb{1}_N + \mathbb{1}_N \otimes \hat{H}^{(k)}_{R} + \hat{H}^{(k)}_{int} \,, \tag{5}$$

   for $k \geq 0$.

2. Diagonalize $\hat{H}^{(k)}_{2N}$ and construct the projector P on the 2^N lowest eigenvalues subspace, which in this case is a $2^{2N} \cdot 2^N$ matrix, whose columns are the normalized eigenvectors corresponding to the lowest eigenvalues taken in increasing order.

3. Project the Hamiltonian and all the operators on this subspace

$$\hat{H}^{(k+1)}_{L} = \hat{H}^{(k+1)}_{R} = P^\dagger \hat{H}^{(k)}_{2N} P \,, \quad \hat{R}^{(k+1)} = P^\dagger \big(\mathbb{1}_N \otimes \hat{R}^{(k)}\big) P \,, \quad \hat{L}^{(k+1)} = P^\dagger \big(\hat{L}^{(k)} \otimes \mathbb{1}_N\big) P \,. \tag{6}$$

4. Repeat steps 1-2-3 until the ground-state energy density $g_{GS} = E_0/(N \cdot 2^{k+1})$ converges.

IDMRG

An improvement of the previous algorithm is IDMRG, for which the projector is built by diagonalizing a density matrix and taking the most populated states, which consists in the best approximation. The cost is that now the growth is no longer exponential, but linear. The system is composed by two $N+1$ spins parts, left and right. At each iteration, we add one spin to the left system and one to the right, while keeping constant the dimension of the desity matrix to be diagonalized$(N+1)$.

0. Take the maximum integer N such that the $N+1$ spins density matrix is diagonalizable. At the beginning, take the $N+1$ spins Hamiltonians of the left and the right system, $\hat{H}^{(0)}_{L}$ and $\hat{H}^{(0)}_{R}$. The interaction term is $\hat{H}^{(0)}_{int} = \mathbb{1}_N \otimes \sigma_x \otimes \sigma_x \otimes \mathbb{1}_N$.

1. Build the total $2N+2$ spins Hamiltonian

$$\hat{H}^{(k)}_{2N+2} = \hat{H}^{(k)}_{L} \otimes \mathbb{1}_{N+1} + \mathbb{1}_{N+1} \otimes \hat{H}^{(k)}_{R} + \hat{H}^{(k)}_{int} \,, \tag{7}$$

   for $k \geq 0$.

2. Diagonalize it and take the ground state $|\psi_0\rangle$. Build its density matrix $\rho = |\psi_0\rangle \langle\psi_0|$.

3. Trace out the right part of the system and get $\rho_L = Tr_R(\rho)$. Diagonalize ρ_L and take the eigenvalues in decreasing order.

4. Build the projector P on the subspace spammed by the first 2^N eigenvectors. You will obtain a $2^{N+1} \cdot 2^N$ matrix whose columns are the eigenvectors of ρ_L corresponding to the highest eigenvalues, i.e. the most populated states.

5. Project the left Hamiltonian $\hat{H}^{(k)}_{L}$ on this subspace and get a N spins hamiltonian $\hat{H}^{(k)}_{L,reduced} = P^\dagger \hat{H}^{(k)}_{L} P$.

6. Enlarge the left part of the system by adding 1 spin to the right. The new $\hat{H}^{(k+1)}_{L}$ becomes

$$\hat{H}^{(k+1)}_{L} = \hat{H}^{(k)}_{L,reduced} \otimes \mathbb{1} + \lambda \mathbb{1}_N \otimes \sigma_z + P^\dagger \big(\mathbb{1}_N \otimes \sigma_x\big) P \otimes \sigma_x \,. \tag{8}$$

7. Repeat the same procedure for the right part of the system. We have $\hat{H}^{(k+1)}_{R} = \hat{H}^{(k+1)}_{L}$, due to symmetry.

8. Repeat step 1-2-3-4-5-6-7 until the ground-state energy density $g_{GS} = E_0/(2N + 2(k+1))$ converges.

Code Development

As always, the code is divided in many files. The main subroutines are written in the file "modules.f95", while "RSRG.f95" contains the the real space Renormalization Group, while infinite density matrix Renormalization Group is contained in the file "IDMRG.f95". As explained in the theory section, we implement the two algorithms. We report only the main loops. Both of them accept as input the parameter N, the λ's mesh grid N_{lam} and the number of iterations N_{iter}.
For the RSRG we start from step 1), with the Hamiltonians for the left and right systems already initialized (they are in the previous part of the code, here not reported due to the limited space).

```
do ii=1,N_iter
            !Double the system
            call matrix_tens_product(ham_left,DD**NN,identity,DD**NN,temp,DD**(2*NN),count)
            hamiltonian=temp
            call matrix_tens_product(identity,DD**NN,ham_right,DD**NN,temp,DD**(2*NN),count)
            hamiltonian=hamiltonian+temp
            call matrix_tens_product(int_left,DD**NN,int_right,DD**NN,temp,DD**(2*NN),count)
            hamiltonian=hamiltonian+temp
            !Diagonalize the Hamiltonian
            temp=hamiltonian
            call diagonalize(temp,DD**(2*NN),w,info)
            call check_eigen(info,debug,printer,count)
            gs=w(1)/dble(NN*(2**ii))
            !Check convergence at lambda=0
            if(ll==1) then
               write(15,*) ii,gs
            endif
            !Save first  D**N eigenvectors in projector and check normalization
            do hh=1,DD**NN
               projector(:,hh)=temp(:,hh)
               call check_norm(temp(:,hh),DD**(2*NN),0.0001d0,debug,printer,count)
            enddo
            !Project the hamiltonian in the new basis
            ham_left=matmul(transpose(conjg(projector)),matmul(hamiltonian,projector))
            ham_right=ham_left
            !Compute the new left interaction term
            call matrix_tens_product(int_left,DD**NN,identity,DD**NN,temp,DD**(2*NN),count)
            int_left=matmul(transpose(conjg(projector)),matmul(temp,projector))
            !Compute the new right interaction term
            call matrix_tens_product(identity,DD**NN,int_right,DD**NN,temp,DD**(2*NN),count)
            int_right=matmul(transpose(conjg(projector)),matmul(temp,projector))
         enddo
```

In the same way, IDMRG starts with the construction of the total Hamiltonian.

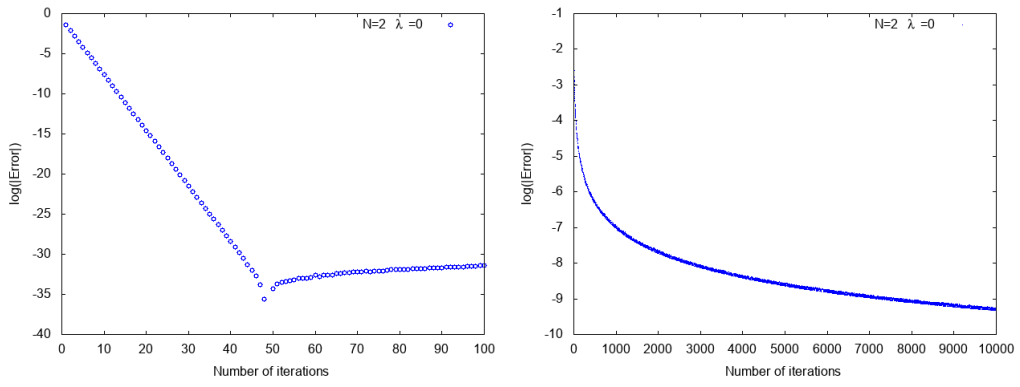
```
do ii=1,N_iter
            !Construct the total 2*NN+2 hamiltonian
            call init_identity(identity,DD,NN)
            call matrix_tens_product(identity,DD**(NN),prod,DD**2,small_temp,DD**(NN+2),count)
            call matrix_tens_product(small_temp,DD**(NN+2),identity,DD**NN,temp,DD**(2*NN+2),
    count)
            hamiltonian=temp
            deallocate(identity)
            call init_identity(identity,DD,NN+1)
            call matrix_tens_product(ham_left,DD**(NN+1),identity,DD**(NN+1),temp,DD**(2*NN+2)
    ,count)
            hamiltonian=hamiltonian+temp
            call matrix_tens_product(identity,DD**(NN+1),ham_right,DD**(NN+1),temp,DD**(2*NN
    +2),count)
            hamiltonian=hamiltonian+temp
            deallocate(identity)
            !Diagonalize the Hamiltonian
            call diagonalize(hamiltonian,DD**(2*NN+2),w,info)
            call check_eigen(info,debug,printer,count)
            gs=w(1)/dble(2*NN+2*ii)
            !Save the ground state
            groundstate=hamiltonian(:,1)
            !Check convergence at lambda=0
            if(ll==1) then
               write(15,*) ii,gs
            endif
            !Compute its density matrix
            call density_matrix(groundstate,DD**(2*NN+2),gsdensity)
            !Trace out half of the system
            call right_trace(gsdensity,DD**(2*NN+2),density_left,DD**(NN+1),DD**(NN+1),count)
```

```
            call left_trace(gsdensity,DD**(2*NN+2),density_right,DD**(NN+1),DD**(NN+1),count)
            !check if the two matrices are equal, as expected due to symmetry
            threshold=0.00001
            call check_diff(density_left,density_right,DD**(NN+1),DD**(NN+1),threshold,debug,
    printer,count)
            !Diagonalize density_left
            call diagonalize(density_left,DD**(NN+1),w2,info)
            call check_eigen(info,debug,printer,count)
            !Define the projector. We take the greatest m=DD**NN eigenvalues and check
    normalization
            do jj=1,DD**NN
               projector(:,jj)=density_left(:,DD**(NN+1)-jj+1)
               call check_norm(density_left(:,jj),DD**(NN+1),0.0001d0,debug,printer,count)
            enddo
            !Project the hamiltonian of left part
            reduced_ham=matmul(transpose(conjg(projector)),matmul(ham_left,projector))
            !Add the reduced hamiltonian
            call init_identity(identity,DD,int(1,16))
            call matrix_tens_product(reduced_ham,DD**NN,identity,DD,least_temp,DD**(NN+1),
    count)
            ham_left=least_temp
            deallocate(identity)
            !Enlarge the left system by adding 1 spin to the right
            call init_identity(identity,DD,NN)
            call matrix_tens_product(identity,DD**NN,sigmaz,DD,least_temp,DD**(NN+1),count)
            ham_left=ham_left+lambda*least_temp
            !Finally add the interaction term
            call matrix_tens_product(identity,DD**NN,sigmax,DD,least_temp,DD**(NN+1),count)
            reduced_ham=matmul(transpose(conjg(projector)),matmul(least_temp,projector)) !use
    reduced_ham as temporary
            call matrix_tens_product(reduced_ham,DD**NN,sigmax,DD,least_temp,DD**(NN+1),count)
            deallocate(identity)
            ham_left=ham_left+least_temp
            !Set the new right Hamiltonian equal due to symmetry
            ham_right=ham_left
         enddo
```

Results

Convergence

The algorithms is supposed to converge as soon as the energy density of ground state reaches a stationary point. At the beginning, we iterated the algorithms for $\lambda = 0$ in order to evaluate how many iterations were needed. In Figure 1 the plot of $\log(|g_{GS} + 1|)$ is reported in function of the number of iterations, starting from a system made of $N = 2$ spins at $\lambda = 0$, where the ground-state energy density is supposed to converge to -1.



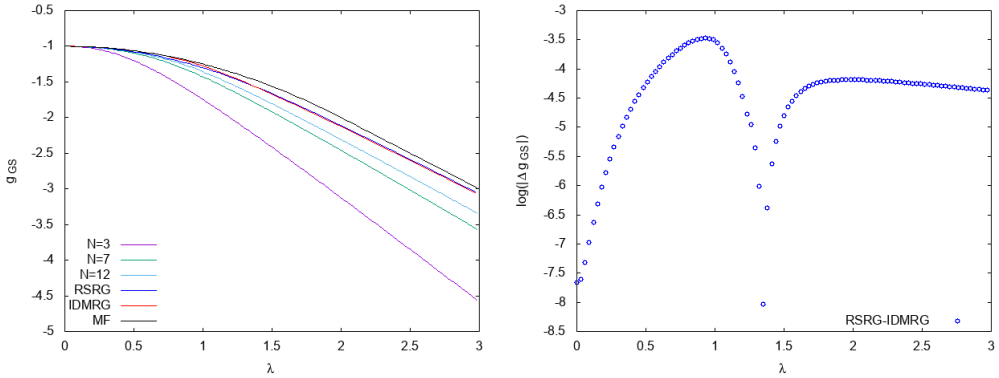
(a) (b)

Figure 1: The convergence behaviour for a) RSRG and b) IDMRG for $\lambda = 0$ and $N = 0$.

Energy density

We completed a simulation starting from $N = 2$ and completing $N_{iter} = 50$ of the RSRG and $N_{iter} = 2000$ for the IDMRG. The results are reported in Figure 2. Along with the RSRG, IDMRG and MF solutions, we report

also the corrected energy density $E_0/(N-1)$[1] for some exact solutions for $N = 3, 7, 12$ spins. We notice that the solutions increase toward the RSRG and IDRG as the number of spins approaches infinity. Moreover, the algorithms are in perfect accordance to each other, as shown in Fig. b). However, the MF solution is quite different from the results of the two algorithms. We notice that is very similar below the critical point of phase transition at $\lambda = 1$, but becomes to differ at higher λ's, maintaining the same linear decay at strong fields.



(a) (b)

Figure 2: a) The energy density for some models and b) the logarithmic difference between RSRG and IDMRG.

Self Evaluation

In this last problem, we have learned to powerful method to simulate quantum systems. At first, we encountered some problems in solving RSRG, but after a while we ended up finding what was exactly the error. In fact, we initialized the interaction term between left and right system at each step in RSRG, before projecting. But the projector goes from the basis at step k to the basis at step $k+1$, while by initializing each time the interaction term we were writing it always in the computational basis (of step 0), committing an error especially at low λ's. Debugging can be done very efficiently due to the way we have been working these months. Writing a lot of subroutines which can be sees as not very useful at first glance in the beginning can be nasty, but it worth it in the end.

However, the code can be for sure improved. In particular, it would be easier to put a condition in order to stop the iterations when the energy density converges (and in fact we first implemented them in that way). However, it is useful to see how the convergence goes. The simulation can be improved to monitor the the behaviour of the two algorithms for starting from different N, taking into account that the maximum N suggested is 4 for RSRG and 3 for IDMRG. We tried some simulations for fun, but the differences were very slight. So it is better to start with $N = 2$, so the convergence is very fast (in fact, it means to diagonalize a 16 x 16 matrix for RSRG and a 8x8 for IDMRG, which is very efficient without losing much information).

Moreover, the code is very versatile and it can be easily reused to solve other quantum problems in an efficient way.

[1] This has been done to have a nice plot at low λ's. However, in the thermodynamic limit the two definitions of energy density coincide.