Exercise 10
Real Space Renormalization Group

Michele Guadagnini - ID 1230663

January 11, 2021

Abstract

The aim of this exercise is to compute the ground state of a 1D lattice of spin particles in a transverse field by mean of the Real Space Renormalization Group method. The obtained results are compared to the analytical Mean Field solution.

1 Theory

The system to be studied is the same of the previous exercise, 1D Ising model with transverse field with nearest neighbor interaction. The Hamiltonian is:

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

In this assignment we will use the *Real Space Renormalization Group* method (RSRG) to compute the ground state of the system in the thermodynamical limit ($N \to \infty$).

RSRG is based on the physical hypothesis that the ground state of the whole system is composed only by the lowest-energy states of the system's non-interacting bipartitions. In this way, starting from a system of a size that allows us to easily computes its eigenstates, it is possible to iteratively double the number of particles represented by the Hamiltonian while keeping low the computational cost, as the Hamiltonian size remains the same through all the iterations.

Starting from a system with N particles the first step consists in building the initial Hamiltonian matrix H_{in} of size 2^N. Then the matrix is diagonalized and the algorithm proceeds repeating the following three steps:

- the matrix is projected into a reduced space by using the matrix P composed by the first m eigenvalues; the same projection is applied also to the interaction terms.

$$\begin{aligned} \widetilde{H_m} &= P^\intercal H_{in} P \\ \widetilde{H_{left}} &= P^\intercal H_{left} P \\ \widetilde{H_{right}} &= P^\intercal H_{right} P \end{aligned} \tag{2}$$

- the matrix of size 2^{2m} is created by padding the projected Hamiltonian and by computing the interaction term. In formula ($\otimes$ denotes the *kronecker product* between two matrices):

$$\widetilde{H_{2m}} = \widetilde{H_m} \otimes \mathbb{I}_m + \mathbb{I}_m \otimes \widetilde{H_m} + \widetilde{H_{left}} \otimes \widetilde{H_{right}} \tag{3}$$

- the matrix is diagonalized in order to get the new ground state and the projector for the next iteration.

Finally, we compare the obtained energy densities with the Mean Field analytical solution, that is:

$$e_{MF}(\lambda) = \begin{cases} -1 - \frac{\lambda^2}{4} & \text{if } \lambda \in [-2;2] \\ -|\lambda| & \text{if } \lambda \notin [-2;2] \end{cases} \tag{4}$$

2 Code Development

2.1 Design and Implementation

A lot of the functions and subroutines needed to solve this problem have been already implemented for the previous assignment. Only some mathematical tools has been added to the module contained in the file *Ex10-Guadagnini-RSRG-CODE.f90*. In Listing 1 the implementation of the kronecker product. The module defines also the operator interface for this operation as ***.kron.***

```
124     FUNCTION kronProduct(Mat1, Mat2) RESULT(Res)
125         DOUBLE PRECISION, DIMENSION(:,:), INTENT(IN) :: Mat1
126         DOUBLE PRECISION, DIMENSION(:,:), INTENT(IN) :: Mat2
127         DOUBLE PRECISION, DIMENSION(SIZE(Mat1,dim=1)*SIZE(Mat2,dim=1),SIZE(Mat1,dim
    =2)*SIZE(Mat2,dim=2)) :: Res
128         INTEGER Nrows1, Ncols1, Nrows2, Ncols2
129         INTEGER ii, jj, aa,bb,cc,dd
130
131         Nrows1 = SIZE(Mat1, dim=1)
132         Ncols1 = SIZE(Mat1, dim=2)
133         Nrows2 = SIZE(Mat2, dim=1)
134         Ncols2 = SIZE(Mat2, dim=2)
135
136         Res = 0d0
137         DO ii = 1,Nrows1
138             DO jj = 1,Ncols1
139                 aa = (ii-1)*Nrows2 +1
140                 bb = ii*Nrows2
141                 cc = (jj-1)*Ncols2 +1
142                 dd = jj*Ncols2
143                 Res(aa:bb,cc:dd) = Mat1(ii,jj) * Mat2
144             ENDDO
145         ENDDO
146
147         RETURN
148     END FUNCTION
```

Listing 1: Kronecker product between two matrices.

The main effort of this work has been to build up the RSRG algorithm in the main program contained in the file *Ex10-Guadagnini-CODE.f90* (see Listing 2). The loop contains the three steps explained in the Theory section and an additional portion of code that decides when to stop the iterations. A parameter representing the minimum improvement in the last step needed to continue the computation has been defined; it is called *ExitTh* and can be changed by editing the configuration file. Also a maximum number of iterations has been set.

```
99      ! number of particles in the system (to be updated at each step)
100     DescrSz = DFLOAT(Pars%NN)
101
102     DO idx = 1,Pars%MaxItr
103         WRITE(*,"(A,I4)") "STARTING ITERATION #: ", idx
104
105         !## 1) Projection ##!
```

```
106         WRITE(*,"(A)") "  Projecting the Hamiltonian and interaction operators..."
107         IF (idx .eq. 1) THEN  !to have (NN) and iterated size (TruncN) independent
108             ! padding the interaction terms
109             tempSz = Pars%NN - Pars%TruncN
110             Init_HamL = TruncHamL .kron. DIdentity(2**tempSz)
111             Init_HamR = TruncHamR .kron. DIdentity(2**tempSz)
112             ! projections
113             TruncHam  = Projection( Init_EigVectors(:,1:2**Pars%TruncN), Init_Ham )
114             TruncHamL = Projection( Init_EigVectors(:,1:2**Pars%TruncN), Init_HamL)
115             TruncHamR = Projection( Init_EigVectors(:,1:2**Pars%TruncN), Init_HamR)
116         ELSE
117             ! padding the interaction terms
118             HamL = TruncHamL .kron. DIdentity(2**Pars%TruncN)
119             HamR = TruncHamR .kron. DIdentity(2**Pars%TruncN)
120             ! projections
121             TruncHam  = Projection( EigVectors(:,1:2**Pars%TruncN), Ham )
122             TruncHamL = Projection( EigVectors(:,1:2**Pars%TruncN), HamL)
123             TruncHamR = Projection( EigVectors(:,1:2**Pars%TruncN), HamR)
124         ENDIF
125         CALL Checkpoint(Pars%Debug, "Projections completed.")
126 
127         !## 2) Double the system ##!
128         WRITE(*,"(A)") "  Building the doubled Hamiltonian..."
129         Ham = (TruncHam .kron. DIdentity(2**Pars%TruncN)) + &
130               (DIdentity(2**Pars%TruncN) .kron. TruncHam) + &
131               (TruncHamL .kron. TruncHamR)
132         CALL Checkpoint(Pars%Debug, "Doubled Hamiltonian computation completed.")
133 
134         !## 3) Diagonalization ##!
135         WRITE(*,"(A)") "  Diagonalizing the doubled Hamiltonian..."
136         EigVectors = Ham  !creating a copy in order to not overwrite it
137         CALL SymmetricEigenpairs(2**(2*Pars%TruncN), EigVectors, EigVals)
138         CALL Checkpoint(Pars%Debug, "Hamiltonian diagonalization completed.")
139 
140         OldSz   = DescrSz     ! size of the described system in the previous step
141         DescrSz = 2d0*DescrSz ! size of the present described system
142 
143         ! Check exit condition
144         IF ( AbsDiff( OldGS, EigVals(1)/(DescrSz-1d0) ) .lt. Pars%ExitTh) THEN
145             WRITE(*,"(A,ES16.8)") "  Improvement in the ground state eigenvalue: <
    ",Pars%ExitTh
146             WRITE(*,"(A,I4)")     "  Exiting at iteration: ", idx
147             EXIT    !exit the RSRG loop
148         ELSEIF (idx == Pars%MaxItr) THEN
149             WRITE(*,"(A,I4)")     "  Reached maximum number of iterations: ",idx
150             EXIT
151         ELSE
152             OldGS = EigVals(1)/(DescrSz-1d0)
153         ENDIF
154         WRITE(*,*) " "  !simple newline
155     ENDDO
```

Listing 2: RSRG loop implementation.

A Python script (*Ex10-Guadagnini-Script-CODE.py*) has been created in order to run the program over different parameters. A set of values between 0 and 3 has been used for λ, some values for N and m has been tested, while $ExitTh$ has been set to 5×10^{-16}, which is close to machine precision with DOUBLE numbers. This script has the capability to capture the output of the FORTRAN program, by setting the variable *verbosity* to $false$, in order to limit the printings on the screen. Also a Gnuplot script has been used to produce the results plots.

Finally, a second Python script (*Ex10-Guadagnini-MeanField-CODE.py*) has been created to implement the Mean Field solution and use it as a comparison.

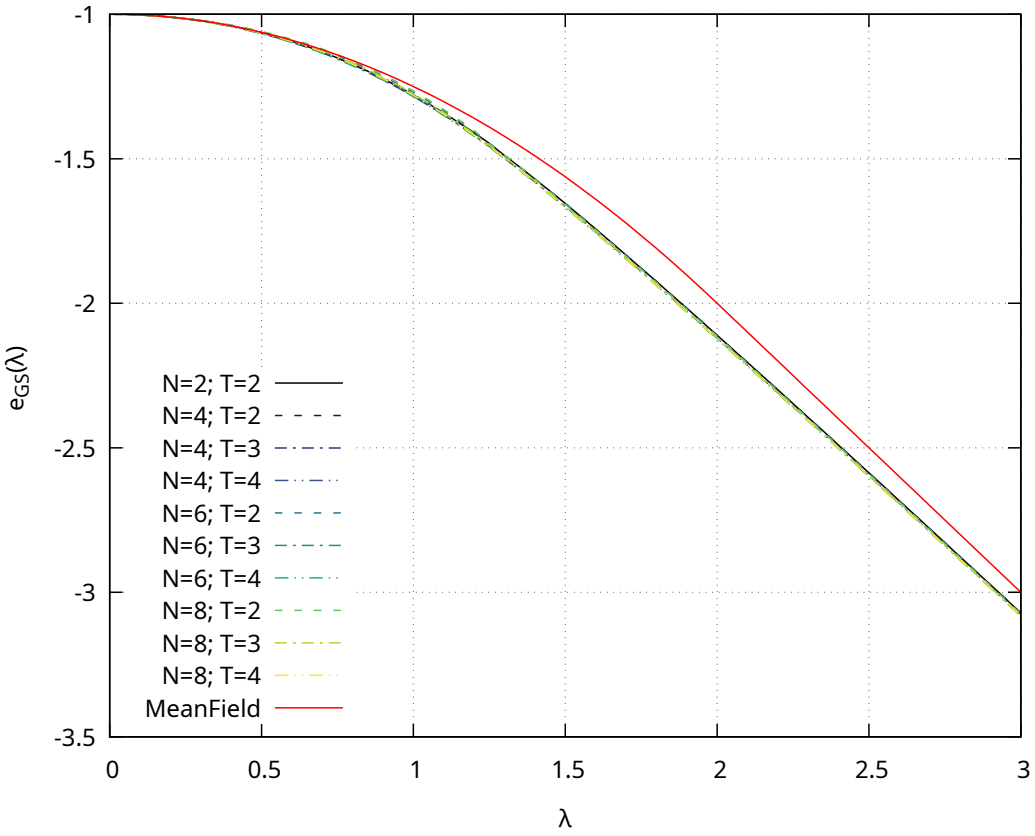


Figure 1: Energy densities of all the computed solutions. Also the Mean Field results are reported.

2.2 Debug and Test

The program has been compiled and executed with the following commands:

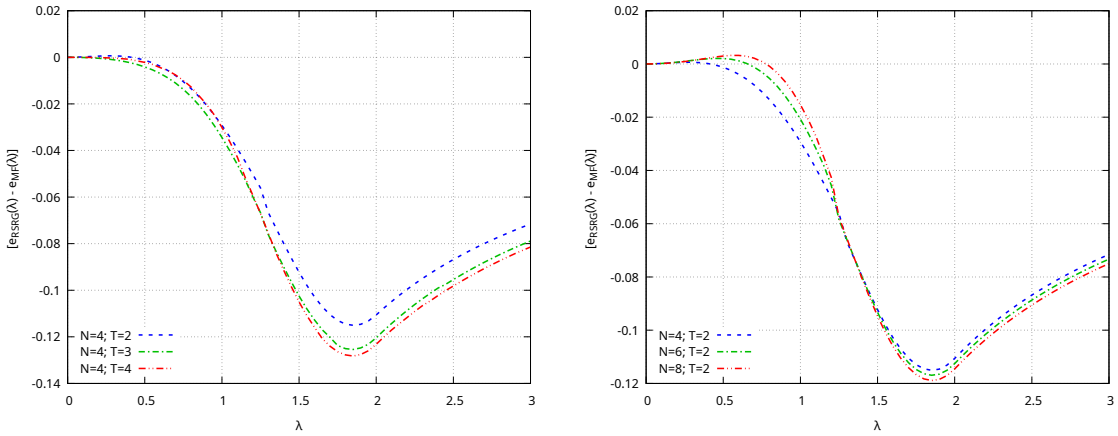
```
gfortran *CODE.f90 -o RGIsing1D.x -lblas -llapack -g -fcheck=all -Wall
./RGIsing1D.x
```

To check the correctness of the code the results of the RSRG algorithm for systems with few particles have been compared to the exact solution obtained in the previous assignment.

3 Results

The algorithm has been run with several values for the input parameters. In particular, its behavior has been explored with different sizes of the initial system (parameter "NN") and different sizes of the matrix during iteration (parameter "TruncN"). The results are reported in Figure 1. It can be seen that there are no significant differences in the energy densities obtained.

There is a small region around $\lambda = 1$, where we know that the system has a phase transition, where they differ slightly more between each other. Anyway, the curves are not well overlapping in any region of the λ space, except for $\lambda = 0$; this could be due to numerical error. Comparing the RSRG solutions with the Mean Field energy density, we can see that, while the solutions are equal for low λ values, they start to separate again



(a) Same N, different T.

(b) Same T, different N.

Figure 2: Difference between the energy densities of the RSRG results and the Mean Field solution. N represents the initial system size, while T is the iteration size.

around $\lambda = 1$. Indeed, we know that the Mean Field method predicts the phase transition at $\lambda = 2$ and the greater difference between the two methods is found to be at $\lambda \sim 1.8$, as can be seen in Figure 2.

4 Self-evaluation

The Real Space Renormalization Group is a powerful method to estimate the energy density of the ground state of a spin system in thermodynamical limit. The method presents slightly different results for different initial or iteration sizes, especially around the phase transition region. An interesting point not discussed here is the comparison with the Infinite Density Matrix Renormalization Group technique.