Renormalization Group

Davide Maniscalco

January 7, 2020

Abstract

In this paper the usage of the RG algorithm for computing the ground state energy for the transverse field Ising model in the thermodynamic limit is studied.

SUGGESTED COMPILATION:

- Fortran: `gfortran Maniscalco-MATRIX.f90 Ex10-Maniscalco-CODE.f90 -llapack -o Ex10-Maniscalco.out`
- Python: `python3 Ex10-Maniscalco-SCRIPT.py`

1 Theory

Let's consider a one-dimensional lattice of dimension N of $1/2$ spin particles, with external field and nearest neighbour interaction:

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

As it was seen in the previous exercise, the eigenvalues states of energy can't be computed for $N > 12$, because of computing limits of the calculator. Thank to the RG algorithm, it is possible to calculate the ground state energy for huge systems. The algorithm works as follows: let's consider a lattice composed of N sites and its Hamiltonian, H_N (its dimension is 2^N). The system so composed is doubled, and is therefore described by the Hamiltonian:

$$H_{2N} = H_N \otimes \mathbb{1}_N + \mathbb{1}_N \otimes H_N + H_{\text{int},2N}$$

where the interaction Hamiltonian can be written as $H_{\text{int}} = L \otimes R$, where

$$L = \mathbb{1}_{N-1} \otimes \sigma_x$$

$$R = \sigma_x \otimes \mathbb{1}_{N-1} \, .$$

The Hamiltonian of the doubled system H_{2N} (of dimension 2^{2N}) is now diagonalized and truncated: assuming that the ground state dependence from the higher eigenvalues is negligible, only the first 2^N eigenvectors are kept and stored in the rectangular matrix P of dimensions $(2^{2N} \times 2^N)$. At this point the Hamiltonian matrix of the doubled state is projected, and the resulting matrix has the same dimensions of the original one, but describes the first 2^Nstates of a system of doubled size:

$$\widetilde{H}_N = P^\dagger H_{2N} P$$

At this point the procedure can be iterated the desired number of times n, in order to describe the ground state of a system composed of N^{n+1} particles. For example, at the first iteration the system is doubled again:

$$\widetilde{\widetilde{H}}_{2N} = \widetilde{H}_N \otimes \mathbb{1}_N + \mathbb{1}_N \otimes \widetilde{H}_N + \widetilde{H}_{\text{int},2N}$$

where $\widetilde{H}_{\text{int},2N} = \widetilde{L} \otimes \widetilde{R}$, where $\widetilde{L}$ and $\widetilde{R}$ can be calculated as:

$$\widetilde{L} = P^{\dagger}(\mathbb{1}_{N-1} \otimes L)P$$

$$\widetilde{R} = P^{\dagger}(R \otimes \mathbb{1}_{N-1})P$$

2 Code development

The subroutine for the tensor product between tensors of any dimensions, the subroutine for creating identity matrices of any dimensions, the module for the matrix type, are retrieved from the previous code. A simple subroutine that makes the adjoint of a matrix is retrieved from code of Ex2. The heart of the code is the do-loop of the RG-subroutine, that is reported here below:

```
DO ii=1, n_iter

      ham_loop%elem = matmul(adjpp%elem,matmul(ham_doubled%elem,pp%elem))

      DEALLOCATE(ham_int%elem,temp1%elem,temp2%elem)
      DEALLOCATE(temp3%elem,temp4%elem)

      IF(ii == n_iter) THEN
        exit
      END IF

      temp1 = TENSOR_PRODUCT(ham_loop, id_2_n)
      temp2 = TENSOR_PRODUCT(id_2_n, ham_loop)
      temp3 = TENSOR_PRODUCT(id_2_n, left)
      temp4 = TENSOR_PRODUCT(right, id_2_n)

      IF(debug.eqv..true.) THEN
        OPEN(unit=10,file='dimension_debug.dat',status='replace')
        WRITE(10,*) 'adjoint pp: ', adjpp%N(1), adjpp%N(2), char(10)
        WRITE(10,*) 'temp3: ', temp3%N(1), temp3%N(2)
        WRITE(10,*) 'temp4: ', temp4%N(1), temp4%N(2)
        write(10,*) 'pp: ',pp%N(1), pp%N(2), char(10)
        write(10,*) 'left: ', left%N(1), left%N(2)
        write(10,*) 'right: ', right%N(1), right%N(2)
        CLOSE(10)
      END IF

      left%elem = matmul(adjpp%elem,matmul(temp3%elem,pp%elem))
      right%elem = matmul(adjpp%elem, matmul(temp4%elem,pp%elem))

      IF(debug.eqv..true.) THEN
        print*, 'loop: ',ii
      END IF

      ham_int = TENSOR_PRODUCT(left, right)

      ham_doubled%elem = temp1%elem + temp2%elem + ham_int%elem
      copy%elem = ham_doubled%elem

      CALL DIAGONALIZE_ONLYVEC(copy)

      pp%elem(1:,1:) = copy%elem(1:,1:2**NN)
      DEALLOCATE(adjpp%elem)
      adjpp = MATADJOINT(pp)

 END DO
```

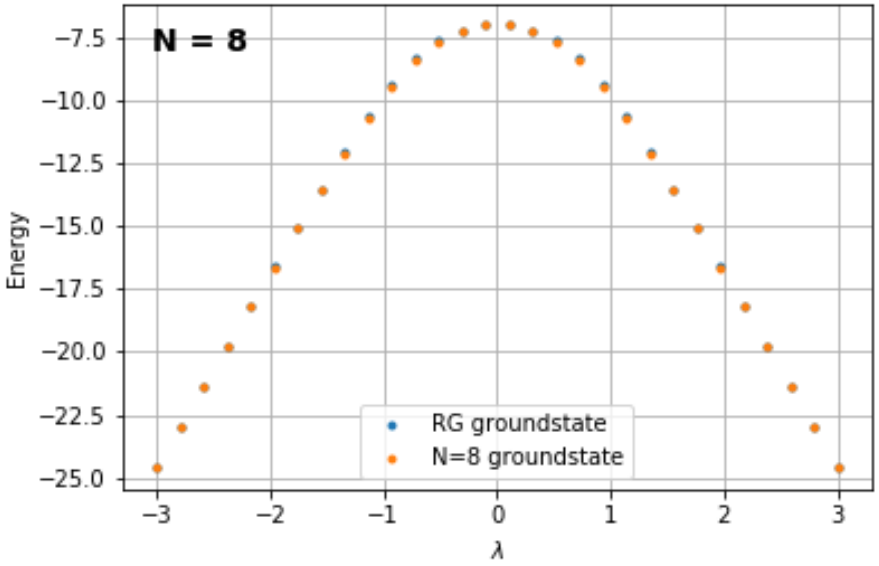

Figure 1: Ground state energy for N=8 computed with the RG algorithm and in the "standard" way.

The `ham_loop` Hamiltonian is the Hamiltonian of dimension 2^N that at the iteration `n_iter` represents the truncated system made of N^{n_iter+1} particles. The first iteration is made out of the loop, and as soon as the last iteration is reached, the program exits from the loop. At each iteration the Hamiltonian `ham_doubled` of the doubled system is calculated thank to some auxiliary matrices and to the matrices `L` and `R`. Then a copy of it is diagonalized through the `zheev` subroutine: from the diagonalized matrix the matrices `pp` and `adjpp` are extracted. At this point the loop restarts with the calculation of the truncated Hamiltonian `ham_loop`.

Once the program exits from the loop, it diagonalizes the `ham_loop` Hamiltonian, whose lower eigenvalue is the ground state energy.

For the purpose of plotting the ground state for a given dimension as a function of several values of λ, the subroutine `several lambdas` is retrieved from the previous exercise: it takes the λ values from the file `lambdas.dat` and prints in these values with the relative ground state energies in the file `for_graphics_eigens.dat`.

The python script can create the `lambdas.dat` file, run the fortran code, make the graphics.

3 Results

As first, the program was run with 2 iterations, in order to study a system of dimension 8, with the purpose of comparing the ground state computed with this code with the one computed in the "standard" way, with the code of the previous exercise. The result is shown in Fig.1.

At this point, the program was run for 99 iterations, in order to study a system of 2^{100} particles, i.e. of size $\simeq 1.27 \cdot 10^{30}$. The result is shown in Fig.2.

The result is reasonable: as expected the figure is symmetric, at $\lambda = 0$ (no external field) the energy is between $-1.5 \cdot 10^{30}$ and $-1.0 \cdot 10^{30}$. The size that was reached with 99 iterations is plenty satisfactory for studying the thermodynamic limit; the execution of the program is also very fast (about one second).

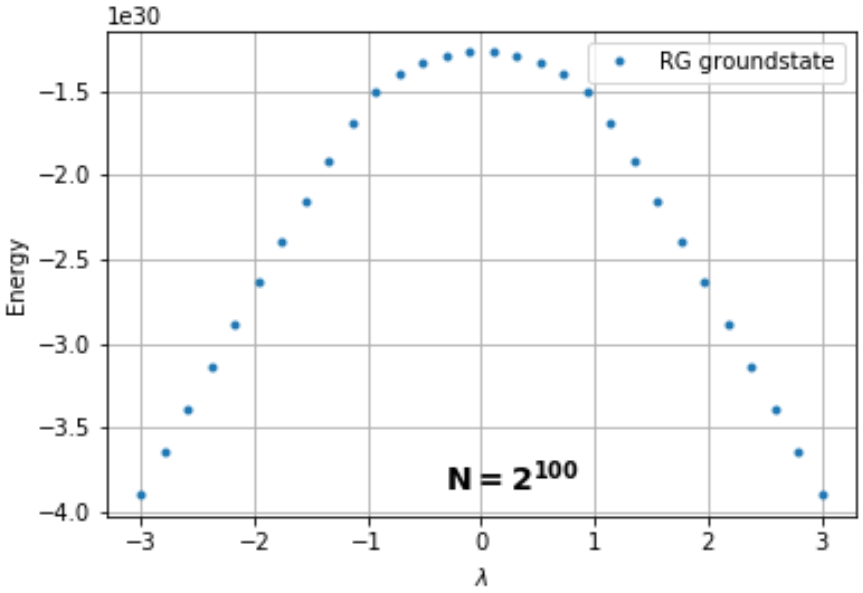

Figure 2: Ground state energy computed for $N = 2^{100}$.

4 Self evaluation

The purpose of writing a correct code for the RG algorithm for the Ising model with transverse field was achieved.