Exercise 10: *Implementation of the Real Space Renormalization Group in* FORTRAN 95

Iriarte Delfina (1231682)

Quantum Information

January 11, 2020

Abstract

In this project, the **real-space renormalization group method** is implemented in FORTRAN and apply it into the transverse Ising model in two dimensions. The **Hamiltonian** of the model is given by: $H = \lambda \sum_{i=1}^{N} \sigma_i^z + \sum_{i=1}^{N-1} \sigma_i^x \sigma_{i+1}^x$ where σ represents the Pauli matrices and λ the interaction strength. Moreover, the ground state of such Hamiltonian is found and compare it with the **mean field solution**. Lastly, the limitation about available allocate resources are qualitative study.

1 Introduction

The many interesting and relevant models in Quantum Physics encountered in these fields are usually not exactly solvable except for some privileged cases in one dimension. It is then when we resort to the **renormalization group method** to retrieve the essential features of those systems in order to have a qualitative understanding of what the physics of the model is all about.

Real-space renormalization group method (RSRG) 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**. The truncation process pertaining to the RSRG is nothing but a tensor product decomposition of representations of a symmetry group G in which the Hamiltonian commute.

The **quantum N spin 1/2 Ising model** is given with the Hamiltonian $\mathcal{H}_N : C^{d^N} \to C^{d^N}$:

$$\hat{\mathcal{H}} = \sum_{i}^{N-1} \hat{\sigma}_x^i \hat{\sigma}_x^{i+1} + \lambda \sum_{i}^{n} \hat{\sigma}_z^i \tag{1}$$

where σ are the Pauli matrices and represents a linear chain of N interacting spins 1/2 in presence of an external field of intensity λ.

Using a *mean field* approximation the ground state energy of the system can be found:

$$e = \begin{cases} -1 - \frac{\lambda^2}{4} & \lambda \in [-2; 2] \\ -|\lambda| & \lambda \notin [-2; 2] \end{cases} \tag{2}$$

In the **real-space renormalization group** context, a compound system composed by two subsystem with N spins is formed. Let $H_{2N} : \mathbb{C}^{d^{2N}} \to \mathbb{C}^{d^{2N}}$ be such system that doubles the number of spins ($2^N \times 2^N$) leading to a Hamiltonian, which is, at the first step, the original Hamiltonian:

$$H_{2N} = H_N^L + H_N^R + H^{LR} \qquad \begin{cases} H_L = \mathbb{I} \otimes \cdots \otimes \mathbb{I} \otimes \sigma_N^x \\ H_R = \sigma_{N+1}^x \otimes \mathbb{I} \otimes \cdots \otimes \mathbb{I} \end{cases} \tag{3}$$

Let P be the projector onto the lowest m eigensates. The next step is to **diagonalize** H_{2N} keeping only the first d^N dimension. This is done by **projecting** the Hilbert space on the subspace spanned by the first m low-energy laying eigenstates H_{2N} using a reduced version of P, P' leading it into a lower dimensional space:

$$H_{2N}^{trunc} = P^{\dagger} \cdot H_{2N} \cdot P \tag{4}$$

The previous steps are iterate. Recall that the projector P acts as a truncation leading to a tensor product of two transformed matrices to the original dimension. Notice that the transformation of H^{LR} is as following:

$$H_{next\ step}^{LR} = P^{\dagger} \left[\mathbb{I}^{N-1} \otimes \sigma_N \otimes \mathbb{I}^{N}\right] \cdot P \otimes P^{\dagger} \cdot \left[\mathbb{I}^{N} \otimes \sigma_1 \otimes \mathbb{I}^{N-1}\right] \cdot P \tag{5}$$

In this report, the **ground state eigenvalue** of the Ising model for a system of N spins 1/2 given by the Hamiltonian 1 are found by means of the real-space renormalization group algorithm and compare with the solution obtained by the mean field approximation (Equation 2.

2 Code development

The Hamiltonian Ising model given by Equation 1 is built by means of two subroutines named `ising_model_h` and `ising_model _J`, which computes respectively the external field component of the Hamiltonian and the spin interaction part. The complete Hamiltonian is done by adding the interaction strength $\lambda \in [0, 3]$ as the previous week assignment (which of course it is not yet duplicated).

Initially, the number of particles and the dimension are read from a file. This is automated by means of a PYTHON script, which provides such values.

Afterwards, the initialization of H_L and H_R as given in Equation 3 are implemented in the subroutine `subroutine intializing_h_AB(h_left, h_right, d, n_particles)` which takes as input the dimension, and the number of particles and returns the proper initialization of the matrices H_L and H_R. This will be helpful to build the interaction term of the Hamiltonian H_{2N} of Equation 3.

First, both matrices (H_L and H_R) are initialized in the first step with the identity or the Pauli matrix in the x-axis respectively. Notice that this is done holding the dimensionality of the matrix with d = 2. Thereafter, the matrices are enlarged by using the subroutine that was designed previously (`tensor_product`) which allows to compute the tensor product among two matrices. Listing 1 shows the implementation of such SUBROUTINE.

```
1  subroutine intializing_h_AB(h_left, h_right, d, n_particles)
2
3      h_left = 0._pr
4      h_right = 0._pr
5
6      do jj = 0, (n_particles- 1)
7
8          if (jj == 0) then
9              h_left(1:d**(jj+1), 1:d**(jj+1)) = I(:,:) !2 dimension
10             h_right(1:d**(jj+1), 1:d**(jj+1)) = sigma_x(:,:)
11         else
12             if (jj == (n_particles-1)) then
```

```
                h_left(1:d**(jj+1), 1:d**(jj+1)) = tensor_product(h_left(1:d**(jj), 1:d**(jj))
    , sigma_x(:,:))
                h_right(1:d**(jj+1), 1:d**(jj+1)) = tensor_product(h_right(1:d**(jj), 1:d**(jj
    )), I(:,:))
            else
                h_left(1:d**(jj+1), 1:d**(jj+1)) = tensor_product(h_left(1:d**(jj), 1:d**(jj))
    , I(:,:))
                h_right(1:d**(jj+1), 1:d**(jj+1)) = tensor_product(h_right(1:d**(jj), 1:d**(jj
    )), I(:,:))
            endif
        endif
    enddo
end subroutine
```

Listing 1. Generation of the matrices H_L and H_R useful to build the interactive part $H_L R$ of equation 3

Listing 2 shows the implementation of the real space renormalization group which is done for each lambda iterative 50 times. Firstly, the double Hamiltonian is done as given in equation 3. The interaction part is implemented by performing the tensor product among the two previous built matrices h_l and h_r as already explained.

The diagonalization of H_{2N} is done by calling a previously week SUBROUTINE which uses the library LAPACK to compute the eigenvalues and eigenfunctions of any complex matrix. Furthermore this is done by using a copy of the original matrix H_{2N} in a support matrix in order to no overwrite it.

Lastly, the hamiltonian of the system of size N and each matrix h_l and h_r are updated by means of Equation 5.

```
do ii = 0, 10

    lambda = 3._pr/(10._pr)*ii

    !Building total hamiltonian
    hamiltonian = 0
    hamiltonian(:,:) = hamiltonian_J(:,:) + lambda *hamiltonian_magentic(:,:)

    if (ii == 0) then
        call print_complex_matrix(hamiltonian)
        write(*,*) lambda
    endif
    call check_hermitian(hamiltonian,debug)

    call intializing_h_AB(h_l, h_r, ham_dim, n_part)

    if (ii == 0) then
        call print_complex_matrix(h_l)
    endif

    call check_hermitian(h_l,debug)
    call check_hermitian(h_r,debug)

    !Real Space Renormalization Group
    do step = 1, 50

        hamiltonian_2N = tensor_product(hamiltonian, I_n) + tensor_product(I_n, hamiltonian) +
     tensor_product(h_l, h_r)

        h_2n = hamiltonian_2N

```

```
        info =0
        !compute eigenvalues calling subroutine
        call eigenvalues(h_2n, ham_dim**(2*n_part), work, eigenvalue, rwork,  lwork, info)

        if(info.NE.0)then
            print*,'DIAGONALIZATION FAILED'
            stop
        end if

        hamiltonian = matmul(transpose(conjg(h_2n(:,1:ham_dim**n_part))),matmul(hamiltonian_2N
    , h_2n(:,1:ham_dim**n_part)))

!             if (step == 1) then
!                 call print_complex_matrix(hamiltonian)
!             endif

        ! Updating hamilt_l and hamilt_r (first N eigenvectors)
    h_l = matmul( transpose(conjg(h_2n(:,1:ham_dim**n_part))), matmul(tensor_product(h_l,I_n),
    h_2n(:,1:ham_dim**n_part)) )
    h_r = matmul( transpose(conjg(h_2n(:,1:ham_dim**n_part))), matmul(tensor_product(I_n,h_r),
    h_2n(:,1:ham_dim**n_part)) )

     enddo
```

Listing 2. Main code of the renormalization group for the different lambdas

Finally, the ground state energies values are saved for every λ and properly re-scaled. The results are then plotted using GNUPLOT.

3 Results

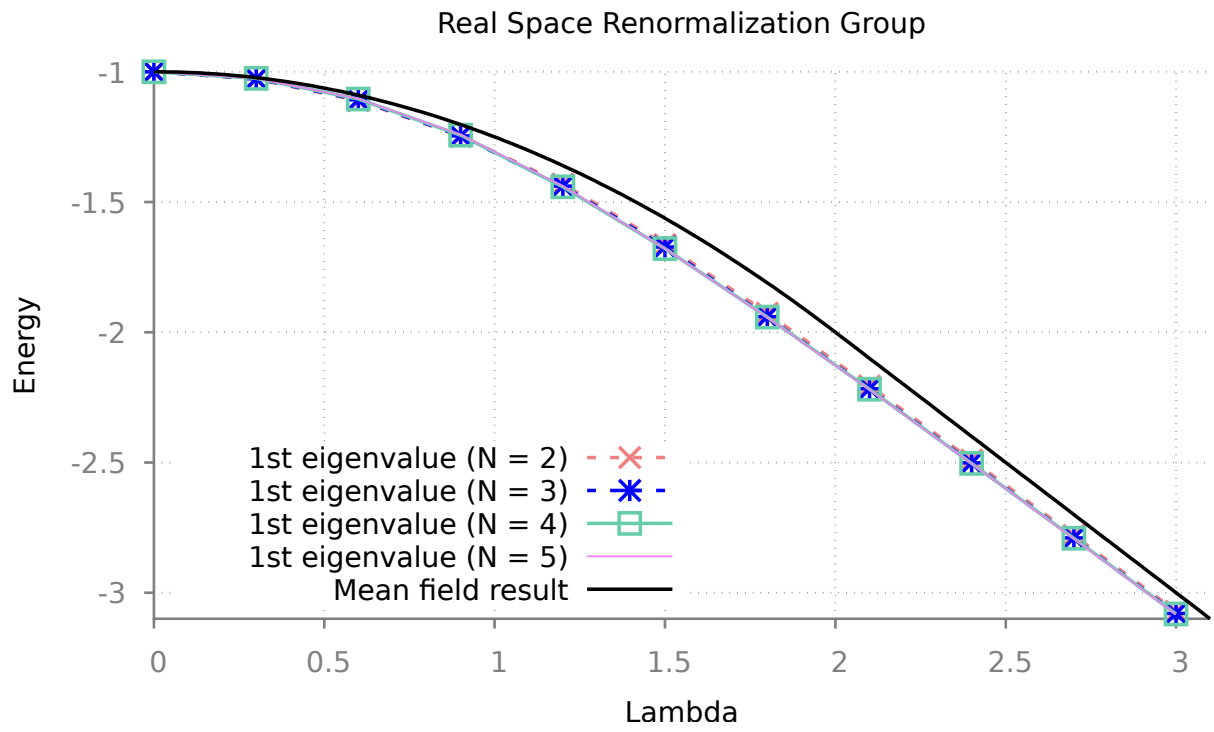

Figure 1. Ground state eigenvalues obtained by means of the real - Space renormalization groups algorithm for different number of particles.

It is important to notice that there is a big limitation regarding the size of allocatable memory available giving rise to a limitation in the number of particles. In particular the maximum number of particles adequate was found to be N = 5. Furthermore, the computational time becomes quite demanding when N = 4, 5, which is something expected since for the last week exercise the maximum number of particles was $N = 10$.

Figure 1 shows the ground states eigenvalues obtained for the several number of particles in the range of $\lambda = [0, 3]$. It can be seen that when $\lambda \simeq 0$, which correspond to the Hamiltonian with only the anti-ferromagnetic term, the MF approximation are similar to the RSRG. However, when λ increases the MF approximation tends to be more and more different and then it becomes more and more similar. This can be due to the fact that the MF approximation is good for small and large values of λ, while for critical values it is not.

4 Conclusion

In conclusion, the **real-space renormalization group method** was successfully implemented in the he transverse Ising model in two dimensions and compare with the mean field solution. It was found that the maximum number of particles that can be used was only 5 available allocate resources in FORTRAN.