Real Space Renormalization Group

Abstract

In this assignment the quantum Ising Hamiltonian, again in a 1-dimensional lattice and with nearest neighbor interaction, is considered. In particular, we are asked to compute the ground state energy, as a function of the transverse field λ, using the ***real-space Renormalization Group*** algorithm.

Theory

The system composed by N spin-1/2 particles is described by the following Hamiltonian:

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

where the σs are the Pauli matrices and λ is the interaction strength, considered $\lambda \in$ [0,3], like in the previous assignment.
Recall that the Pauli matrices (in the z-basis) are:

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

To compute the matrix representation of this Hamiltonian, we have to define the Hilbert space $\mathcal{H} = \bigotimes_{i=1}^{N} \mathcal{H}_i$ where $\mathcal{H}_i$ is the Hilbert space of the i-th site. The dimension of this $\mathcal{H}$ is 2^N. On the contrary the terms that composed our Hamiltonian are not defined on a space of dimension 2^N, since each Pauli matrix acts on a Hilbert space of dimension 2. Moreover each term acts on its respective subsystem.
The given Hamiltonian can be splitted in

- Diagonal part $\lambda \sum_{i=1}^{N} \sigma_z^i$
  This term is diagonal in the z-basis and the i-th term in the sum can be written as

$$\bigotimes_{j=1}^{i} \mathbb{1}_2 \otimes \sigma_z^i \otimes \bigotimes_{j=i+1}^{N} \mathbb{1}_2 \tag{3}$$

- Interaction term $\sum_{i=1}^{N-1} \sigma_x^i \sigma_x^{i+1}$
  Each term of the summation is a tensor product of two matrices

$$\sigma_x \otimes \sigma_x = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix} \tag{4}$$

The system is not separable. In this exercise the ***real-space RG*** algorithm is used to estimate the energies. In particular we are interested in the ground state.
The structure of such algorithm, that is an iterative process to build a system of size $2N$ from an initial size N, is the following:

1. We start with a system of size N with Hamiltonian $\hat{H}_N : \mathbb{C}^d \to \mathbb{C}^d$, where d is the dimension of the system composed by N subsystems. If each single subsystem (i.e. single particle) has dimension dim, then $d = dim^N$.

2. Next step is to build a compound system of size $2N$. To do that, we double the system described by the aforementioned Hamiltonian $\hat{H}_N$, taking two subsystems, named A and B, with Hamiltonians $\hat{H}_N^A$ and $\hat{H}_N^B$, and make them interact. In this case we consider *nearest neighbor* interaction, so the "rightmost" particle of the system A interacts with the "leftmost" particle of B. The total Hamiltonian of the system is,

$$\hat{H}_{2N} = \hat{H}_N^A + \hat{H}_N^B + \hat{H}_{int}^{AB} \tag{5}$$

where $\hat{H}_{2N} : \mathbb{C}^{d^2} \to \mathbb{C}^{d^2}$ and $\hat{H}_N^A : \mathbb{C}^d \to \mathbb{C}^d$, $\hat{H}_N^B : \mathbb{C}^d \to \mathbb{C}^d$.
One can write

$$\begin{aligned} \hat{H}_N^A &\to \hat{H}_N^A \otimes \mathbb{1}_N : \mathbb{C}^{d^2} \to \mathbb{C}^{d^2} \\ \hat{H}_N^B &\to \mathbb{1}_N \otimes \hat{H}_N^B : \mathbb{C}^{d^2} \to \mathbb{C}^{d^2} \end{aligned} \tag{6}$$

where $\mathbb{1}_N$ is the identity matrix of size d.
Regarding the interaction term, it is initialized this way

$$\hat{H}_{int}^{AB} = (\mathbb{1}^1 \otimes ... \otimes \mathbb{1}^{N-1} \otimes \sigma_x) \otimes (\sigma_x \otimes \mathbb{1}^{N-1} \otimes ... \otimes \mathbb{1}^1) \tag{7}$$

Note that $\mathbb{1}^{N-1} = ... = \mathbb{1}^1$ is simply the $2x2$ identity matrix $\mathbb{1}$.

3. Now, we have to diagonalize the matrix $\hat{H}_{2N}$, so to find the eigenvectors matrix. The latter, let us call it V, has dimension d^2xd^2. We want to truncate it, taking only the first d eigenvectors (that correspond to the first d eigenvalues, taken in ascending order), so the matrix becomes $V^{truncated}$ of dimension d^2xd. The aim of this latter procedure is to project the Hamiltonian $\hat{H}_{2N}$ in a lower dimensional space.

$$\hat{H}_{2N}^{truncated} = (\hat{V}^{truncated})^\dagger \cdot \hat{H}_{2N} \cdot \hat{V}^{truncated} \tag{8}$$

The algorithm consists in iterating the aforementioned procedure, taking into account that the interaction term transforms as follow:

$$\begin{aligned} \hat{H}_{int\ next\ step}^{AB} = (\hat{V}^{truncated})^\dagger (\mathbb{1}^1 \otimes ... \otimes \mathbb{1}^{N-1} \otimes \sigma_x \otimes \mathbb{1}^N)\hat{V}^{truncated} \otimes \\ (\hat{V}^{truncated})^\dagger (\mathbb{1}^N \otimes \sigma_x \otimes \mathbb{1}^{N-1} \otimes ... \otimes \mathbb{1}^1)\hat{V}^{truncated} \end{aligned} \tag{9}$$

From the theoretical side, we can estimate the ground state energy using a *Mean Field* approximation:

$$E[\psi_{MF}] = \langle\psi_{MF}|\hat{H}|\psi_{MF}\rangle = \lambda\sum_{j=1}^{N}\langle\psi_{MF}|\sigma_z^j|\psi_{MF}\rangle + \sum_{j=1}^{N-1}\langle\psi_{MF}|\sigma_x^j|\psi_{MF}\rangle^2 \tag{10}$$

In the thermodynamic limit $(N \to \infty)$

$$e[\psi] = \frac{E[\psi]}{N} \xrightarrow{N\to\infty} \lambda\langle\psi_{MF}|\sigma_z|\psi_{MF}\rangle + \langle\psi_{MF}|\sigma_x|\psi_{MF}\rangle^2 \tag{11}$$

This quantity has to be minimized to find the ground state of the system. The result is the following:

$$e = \begin{cases} -1 - \frac{\lambda^2}{4} & \lambda \in [-2,2] \\ -|\lambda| & \lambda \notin [-2,2] \end{cases} \tag{12}$$

Code development

Firstly two `MODULE`s are included

- The usual `MODULE 'DEBUGMOD'`, used also in previous exercises, to debug a generic code
- The `'OPER' MODULE`, that contains a function to do the tensor product of the two matrices given as input, used also in the previous exercise.

Moreover, at each run the program asks the user the number of particles N and the dimension d (indicated as *dim* in the **Theory** part) of every subsystem, assumed to be the same for all the subsystems. The aforementioned `MODULE 'DEBUGMOD'`, activated if the logical flag `'debug_flag'` in the program is set to `TRUE`, is used to check if the inserted quantities are positive, otherwise a warning message is printed.

```
CALL DEBUG(debug_flag,d,d < 0 ,'Insert a positive dimension')
CALL DEBUG(debug_flag,N,N < 0 ,'Insert a positive number of particle')
```

Listing 1: DEBUG check

In the main PROGRAM, called Ex10, first of all the 'non duplicated' Hamiltonian H_N, called H in the code, is evaluated considering separately the two parts, the diagonal part, called H_nonint in the code, and the interaction part , called H_int. Both are initialized as matrices of dimension $d^N x d^N$. The first part, since it includes a diagonal matrix, is simply evaluated as a vector, then shaped in matrix form. The interesting part concerns the computation of the interaction term, in which we have to evaluate N-1 tensor products. Recall that the first one is easy to evaluate, it is the one in the **Theory** section. As the listing 2 shows, a 'container matrix' , allocated with the same dimension of H_int is used to perform the computation, that includes the previous defined function TENSOR_PROD.

```
DO ii=1,N-1

   container = 0

   DO jj=1,N

      IF (jj.EQ.1) THEN

         IF (((N+1-jj).EQ.ii).OR.((N+1-jj).EQ.(ii+1))) THEN
            container(1:d**(jj),1:d**(jj)) = sigma_x(:,:)

         ELSE
            container(1:d**(jj),1:d**(jj)) = id_2(:,:)

         END IF
      ELSE

         IF (((N+1-jj).EQ.ii).OR.((N+1-jj).EQ.(ii+1))) THEN
            container(1:d**(jj),1:d**(jj)) = TENSOR_PROD(sigma_x, container(1:d**(
    jj-1),1:d**(jj-1)))

         ELSE
            container(1:d**(jj),1:d**(jj)) = TENSOR_PROD(id_2, container(1:d**(jj
    -1),1:d**(jj-1)))

         END IF
      END IF
   END DO

   H_int(:,:) = H_int(:,:) + container(:,:) ! Update the interaction term

END DO
```

Listing 2: Evaluation of the interaction term

I have chosen to perform 15 iterations (15 values of λ), with λ calculated according to the following rule

$$\lambda = \frac{3}{14} * i \tag{13}$$

in a loop with $i = 0, 14$. Recall that $\lambda \in$ [0,3].

At this point *real space RG* is applied. The algorithm is performed into the loop over λ, since it uses the same λ value each time. Before the algorithm starts the variables corresponding to the Hamiltonian H_N, named H, and to the yellow and light-blue parts in equation (7), named respectively H_left and H_right, are initialized.

```
! loop to evaluate lambda
DO ii=0,14

   lambda = 3.0/(14.)*ii

   WRITE(*,*) "lambda = ", lambda

   H =  H_int + lambda*H_nonint ! system Hamiltonian for every lambda

   H_left = 0  ! initialize them to zero for every lambda
   H_right = 0

   DO jj=0,N-1

```

```fortran
      IF (jj.EQ.0) THEN
         H_right(1:d**(jj+1),1:d**(jj+1)) = sigma_x(:,:)
         H_left(1:d**(jj+1),1:d**(jj+1)) = id_2(:,:)

      ELSE

         IF (jj.EQ.N-1) THEN
            H_right(1:d**(jj+1),1:d**(jj+1)) =  TENSOR_PROD(H_right(1:d**jj,1:d**jj
   ), id_2(:,:))
            H_left(1:d**(jj+1),1:d**(jj+1)) =  TENSOR_PROD(H_left(1:d**jj,1:d**jj),
    sigma_x(:,:))
         ELSE

            H_right(1:d**(jj+1),1:d**(jj+1)) =  TENSOR_PROD(H_right(1:d**jj,1:d**jj
   ), id_2(:,:))
            H_left(1:d**(jj+1),1:d**(jj+1)) =  TENSOR_PROD(H_left(1:d**jj,1:d**jj),
    id_2(:,:))
         END IF
      END IF
   END DO
```

Listing 3: Initialization of `H, H_right` and `H_left`

The *real space RG* algorithm, iterated 50 times, is shown in listing 4. The Hamiltonian H_{2N} is evaluated and then diagonalized, using the `ZHEEV` subroutine provided by `LAPACK`. The documentation can be found at the following link [1].

```fortran
! RG algorithm (50 iterations)
   DO ww = 1,50
      H_2N = TENSOR_PROD(H,identity_n) + TENSOR_PROD(identity_n,H) + TENSOR_PROD(
   H_left,H_right)
      eig_vec = H_2N
      ! DIAGONALIZATION
      ALLOCATE(wk_opt(1))
      LWORK_=-1
      ! LAPACK's subroutine to do the diagonalization
      CALL ZHEEV('V', 'U', d**(2*N), eig_vec, d**(2*N), eig_val, wk_opt, LWORK_,
   RWORK_, info_eig)
      LWORK_ = INT(wk_opt(1))
      ALLOCATE(WORK_(LWORK_))  ! optimal dimension for WORK

      !subroutine to do the diagonalization
      CALL ZHEEV('V', 'U', d**(2*N), eig_vec, d**(2*N), eig_val, WORK_, LWORK_,
   RWORK_, info_eig)
      CALL DEBUG(debug_flag,info_eig, info_eig==0,'Diagonalization performed
   successfully')
      DEALLOCATE(wk_opt)
      DEALLOCATE(WORK_)

      ! Updating H using truncation (first d**N  eigenvectors)
      H = MATMUL(TRANSPOSE(CONJG(eig_vec(:,1:d**N))),MATMUL(H_2N, eig_vec(:,1:d**N)
   ))

      ! Updating H_left and H_right
      H_left = MATMUL(TRANSPOSE(CONJG(eig_vec(:,1:d**N))),MATMUL(TENSOR_PROD(H_left
   ,identity_n),eig_vec(:,1:d**N)))
      H_right = MATMUL(TRANSPOSE(CONJG(eig_vec(:,1:d**N))),MATMUL(TENSOR_PROD(
   identity_n,H_right),eig_vec(:,1:d**N)))
   END DO
```

Listing 4: Real space RG algorithm

At the end a matrix of dimension 15x2 is printed on a file, called `resN*.txt`, where `*` stands for the chosen number for N. The file contains in the first column the values of λ and in the second column the eigenvalues corresponding to the ground state energy. These energy values, before being printed on the file, are divided by the number of particles in the last 'duplicated' system, in order to compare them with the 'theoretical values' provided by the equation (12).

```fortran
   eigval_res(ii+1,1) = lambda
   eigval_res(ii+1,2) = eig_val(1)/(N*2**50) ! ground state energy
END DO
```

Listing 5: 'normalized' Eigenvalues

[1] http://www.netlib.org/lapack/explore-html/df/d9a/group__complex16_h_eeigen_gaf23fb5b3ae38072ef4890ba43d5cfea2.html#gaf23fb5b3ae38072ef4890ba43d5cfea2

The results are plotted through the `GNUPLOT` script `'plot.gnu'`, for different values of N, as shown in the following section.

Results

In the following, the graph of the ground state energy vs the transverse field λ is shown. The dimension of every subsystem d (recall that is called *dim* in the **Theory** part), is fixed to 2 while N varies. In particular, the plots are shown for $N = 2, 3, 4, 5$. We have to consider that, for a given N, a matrix of size $2N$ has to be diagonalized at each step of the RG algorithm. Indeed, for $N > 4$ the program takes a lot of time (more or less an hour for $N = 5$), while it takes seconds/minutes for $N <= 4$. The maximum value for the number of particles N_{max} that can be reached is $N = 6$. For higher numbers the terminal gives a memory error, the system is not able to allocate the objects. This is linked to the RAM available in my laptop and is coherent with the fact that, in the previous exercise, $N_{max} = 12$.
In the plot also the theoretical values, obtained from equation (12) are shown in black.

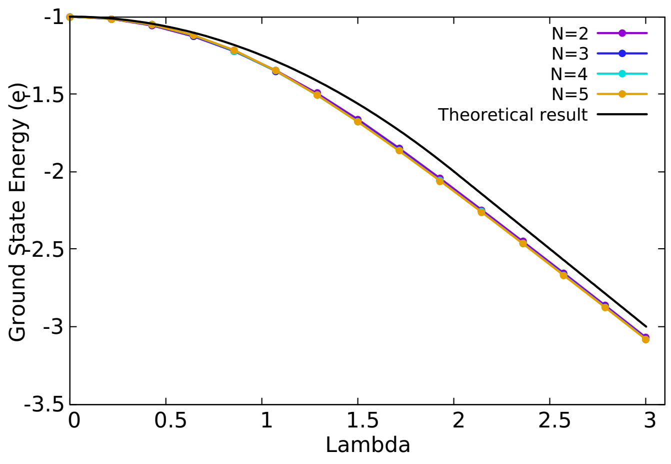

Figure 1: Plot of the ground state energy *per particle* value vs λ with the theoretical result, given by the *Mean Field* prediction, for N=2,3,4,5 .

This plot highlights that the shape of the curves for different N is very similar. Moreover, the computed ground energy is closer to the Mean Field prediction around $\lambda \sim 0$. We can explain this behaviour with the fact that the Mean Field approximation is valid when the external field is sufficiently small. However, we can see that the 'computed' curve get closer once again to the 'theoretical' one when λ becomes higher.
One can note that the eigenvalues have a negative sign. I stress once again that, like in the previous exercise, this fact has no physical meaning since energy is defined up to a constant. Moreover where we expect a quantum phase transition the black curve (*Mean Field* approximation) is more departed. We had a discrepancy in that region also in the previous assignment.

Self-Evaluation

In this exercise I deal one more time with a composite quantum system described by an Ising Hamiltonian. In this framework, I learned how to implement the *real space Renormalization Group* algorithm to study such system. The work could be improved, maybe using a subroutine or something *ad hoc* ('built in') to perform the tensor product.