Exercise 9
One-Dimensional Ising Model

Michele Guadagnini - ID 1230663

December 15, 2020

Abstract

The aim of this exercise is to solve a 1D Ising model with N spin-1/2 particles, transverse field and nearest neighbor interaction. The solution is done by computing and diagonalizing the Hamiltonian matrix.

1 Theory

We consider a system made of N spin-1/2 particles in a 1D lattice in presence of a transverse field, with nearest neighbor interaction and open boundary condition. The Hamiltonian of this system is the following:

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

where λ is a parameter that describes the field strength with respect to interaction strength. σ_z^i and $\sigma_x^j \sigma_x^{j+1}$ are the matrices of size 2^N that results from the direct (Kronecker) products of the corresponding Pauli matrices with some padding identity matrices, as reported in Equation 2.

$$\begin{aligned} \sigma_z^i &= (\bigotimes_{j=1}^{i-1} \mathbb{I}_2) \otimes \sigma_z \otimes (\bigotimes_{j=i+1}^{N} \mathbb{I}_2) \\ \sigma_x^j \sigma_x^{j+1} &= (\bigotimes_{i=1}^{j-1} \mathbb{I}_2) \otimes \sigma_x \otimes \sigma_x \otimes (\bigotimes_{i=j+2}^{N-1} \mathbb{I}_2) \end{aligned} \tag{2}$$

Since in front of the interaction term we have + sign, this is an *anti-ferromagnetic* system.

A local basis of the Hilbert space of a single spin particle can be described as $B_i = \{|0\rangle, |1\rangle\}$. From this basis we can express the one of the global system as a composition of multiple local bases by mean of Kronecker product. This will be useful when implementing the Hamiltonian matrix computation.

2 Code Development

2.1 Design and Implementation

In the implementation of this exercise some subroutines and functions coming from previous exercises have been reused:

- the subroutine that reads the configuration file with the parameters values, *InitParamsFromFile*. The used parameters are: N, number of spin particles; k, number of eigenvalues to save; λ, field strength.
- the subroutines that diagonalize a real (*DOUBLE PRECISION*) symmetric matrix, *SymmetricEigenpairs*.

The main effort in solving this problem has been to find a way to efficiently compute the Hamiltonian matrix. This computation has been split in two independent functions, one for the field term, one for the interaction one.

2.1.1 Field term

Due to the Pauli matrix σ_z, we know that the field term is diagonal and that the *j-esim* term of the matrix σ_z^i:

- is 1 if we apply $|0\rangle$ on the *j-esim* local space;
- is -1 if we apply $|1\rangle$.

This produces a mathematical trick to compute the field term, consisting in initialize all the diagonal elements to N and subtract -2 to the *j-esim* element whenever a $\langle 1_j | \sigma_z | 1_j \rangle$ appears. The implementation is reported in Listing 1.

```
65      !it computes the field term of the hamiltonian
66      FUNCTION Ham_Zsum(Pars) RESULT(Mat_Zsum)
67          TYPE(Parameters) Pars
68          INTEGER aa, ii
69          DOUBLE PRECISION, DIMENSION(2**Pars%NN, 2**Pars%NN) :: Mat_Zsum
70          DOUBLE PRECISION temp
71
72          ! Z contribution is diagonal, so it can be computed in this way
73          Mat_Zsum = 0d0
74          DO aa = 1, 2**Pars%NN
75              Mat_Zsum(aa,aa) = DFLOAT(Pars%NN)
76              DO ii = 1,Pars%NN
77                  temp = 2d0*DFLOAT( MOD((aa-1)/(2**(ii-1)), 2) )
78                  Mat_Zsum(aa,aa) = Mat_Zsum(aa,aa) - temp
79              ENDDO
80          ENDDO
81
82          RETURN
83      END FUNCTION
```

Listing 1: Computation of the Hamiltonian field term.

2.1.2 Interaction term

Regarding the interaction term, we notice that σ_x^i is always zero apart from the matrix elements where $\langle bra|$ and $|ket\rangle$ are different for the *i-esim* particle. From this fact, as before, we can build a method based on the binary representation of the index of the global basis and the *XOR* operation. The code is reported in Listing 2.

```
85      !it computes the interacting term of the hamiltonian
86      FUNCTION Ham_Xsum(Pars) RESULT(Mat_Xsum)
87          TYPE(Parameters) Pars
88          INTEGER aa, bb, ii
89          DOUBLE PRECISION, DIMENSION(2**Pars%NN, 2**Pars%NN) :: Mat_Xsum
```

```
90
91          Mat_Xsum = 0d0
92          DO aa = 1, 2**Pars%NN
93              DO bb = 1, 2**Pars%NN
94                  DO ii = 1,Pars%NN-1
95                      IF ( IEOR(aa-1,bb-1) .eq. (2**(ii-1) + 2**ii) ) THEN
96                          Mat_Xsum(aa,bb) = Mat_Xsum(aa,bb) + 1d0
97                      ENDIF
98                  ENDDO
99              ENDDO
100         ENDDO
101
102         RETURN
103     END FUNCTION
```

Listing 2: Computation of the Hamiltonian interaction term.

2.1.3 Scripting

In order to automatize the running of the program with different N and λ values, a Python script, called *Ex9-Guadagnini-Script-CODE.py*, has been created, that loops over the values of the two variables. It also saves the execution time of the program. Since the computation time does not depend on the value of λ, it computes the average time for a particular N.

Also the plotting has been automatized by creating a gnuplot script contained in the file *Ex9-Guadagnini-LevelsPlot.gnuplot*.

2.2 Debug and Test

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

```
gfortran *CODE.f90 -o Ising1D.x -lblas -llapack -g -fcheck=all -Wall
./Ising1D.x
```

Once checked that it was working correctly, it has been tested the maximum number of spin particles that is possible to simulate and the computation time by running the Python script. The plot of the timings is reported in Figure 1. The maximum N used is 12. We can, in principle, store in memory an Hamiltonian of size also 2^{14}, since, with double precision numbers, it requires $2GB$ of RAM. However, looking at Figure 1, we can see that the computation time grows very fast with N, and by exploiting the fitted parameters we can estimate that, in this machine, even with $N = 13$ it would take almost $30min$ to complete a single step in the λ grid space.

3 Results

The computed eigenvalues have been plotted using the gnuplot script. Eigenvalues are also normalized by a factor of $\frac{1}{N-1}$ to better compare them. Two types of plots have been created:

- the first type reports up to the first 8 eigenvalues in function of λ for a particular system size N. A sample of all the plots of this kind created is reported in Figure 2.
- the second type reports instead the same eigenvalue (i.e. ground state, first excited, ...) in function of λ for all the tested N values. A sample of this plots is reported in Figure 3.

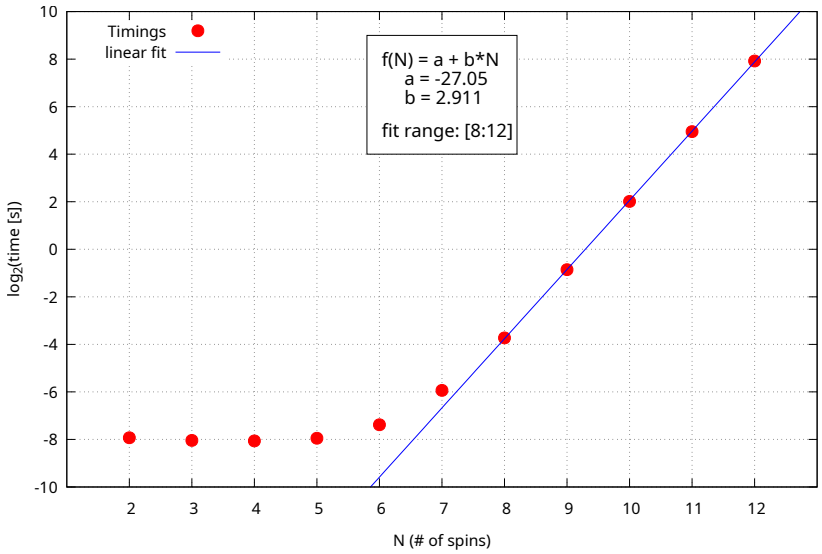
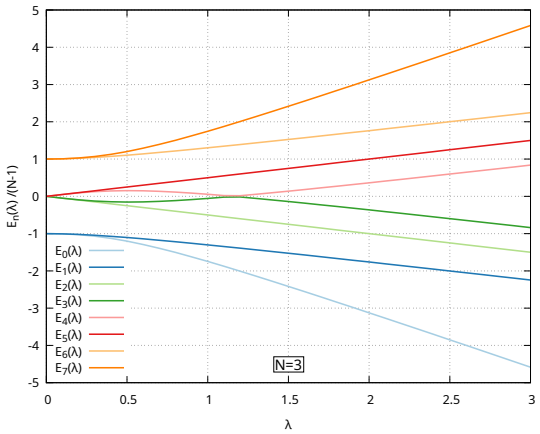


Figure 1: Log_2 of the average computation times for different value of N. Timings are averaged over the λ steps.

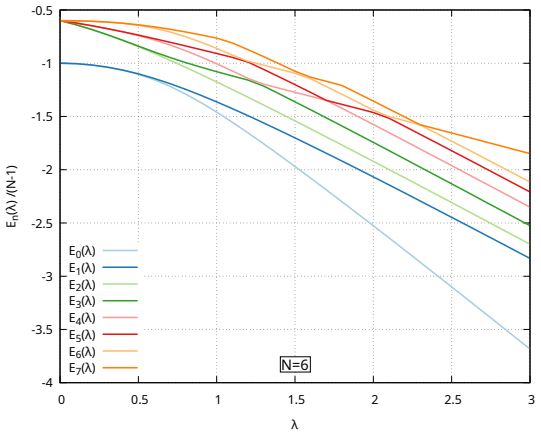
A thing that arises looking at the pictures, in particular at Figure 2a, is that for $\lambda = 0$, so without an external field, the eigenvalues show degeneracy, while with λ increasing they start to separate more and more.

4 Self-evaluation

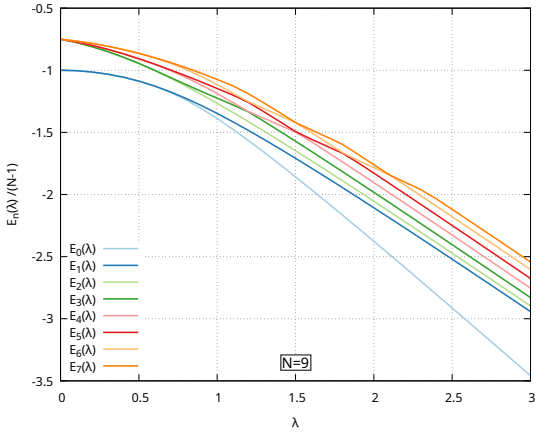
In this exercise we exploited the binary representation of the indexes to compute the Hamiltonian in a more efficient way with respect to do all the tensor products to pad the Pauli matrices. It would be interesting to evaluate the gap in terms of computational time between these two procedures. To improve the execution time estimation we could have saved also the variance of the computed means in order to improve the fit and the plot with empirical errors.



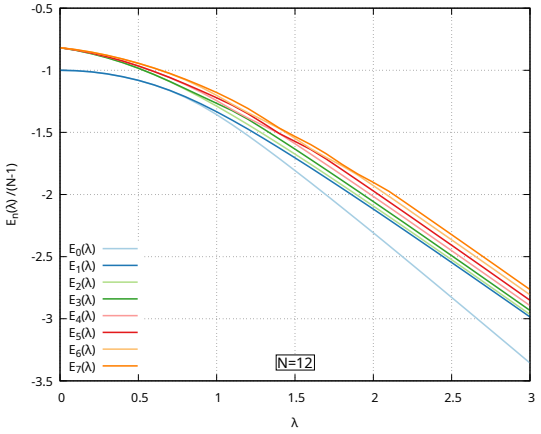
(a) N=3



(b) N=6

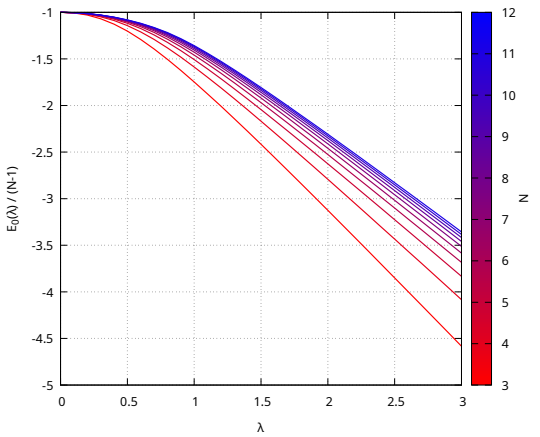


(c) N=9

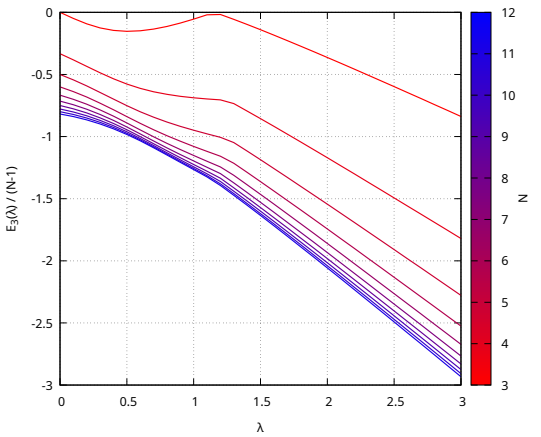


(d) N=12

Figure 2: Grid of plots of the first 8 eigenvalues corresponding to the first 8 excited states for several values of N.



(a) E_0, ground state.



(b) E_3, third excited.

Figure 3: Grid of plots of the ground state and the third excited eigenvalues in function of λ for all the values of N from 3 to 12.