UNIVERSITY OF PADUA
PHYSICS AND ASTRONOMY DEPARTMENT

Homework #9
Ising Model

Student name: *Beatrice Segalini* – 1234430

Course: *Quantum Information and Computing 2020* – Professor: *S. Montangero*
Due date: *December 15th, 2020*

Abstract

In this report, a quantum system formed by N spin$-1/2$ particles is considered, in a one dimensional lattice. A program to represent the $2^N \times 2^N$ matrix of the Hamiltonian of such system is written. The Hamiltonian matrix is then diagonalised and its eigenvectors are computed. Therefore, a study on the energy spectrum is performed, especially as a function of the magnitude of λ, the external field interaction strength.

Theory

Hamiltonian of the system and Pauli matrices. The Hamiltonian $\mathcal{H}$ of the considered spin system is given by:

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

where σ_x, σ_z are the Pauli matrices:

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

and i is the index of the spin considered.

As a matter of facts, this notation simplifies the one of a *tensor product* that can be fully written as:

$$\sigma_k^i = \mathbb{1}^1 \otimes \cdots \otimes \mathbb{1}^{i-1} \otimes \sigma_k^i \otimes \mathbb{1}^{i+1} \otimes \cdots \otimes \mathbb{1}^N \tag{3}$$

The properties of Pauli matrices are known and proved: in particular, Pauli matrices have eigenvalues ± 1 and their eigenstates are:

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} = |\psi_{z+}\rangle, \quad \begin{pmatrix} 0 \\ 1 \end{pmatrix} = |\psi_{z-}\rangle, \quad \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} = |\psi_{x+}\rangle, \quad \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ -1 \end{pmatrix} = |\psi_{x-}\rangle \tag{4}$$

Moreover, the following relationships hold:

$$\sigma_x|\psi_{z+}\rangle = |\psi_{z-}\rangle, \quad \sigma_x|\psi_{z-}\rangle = |\psi_{z+}\rangle, \quad \sigma_z|\psi_{x+}\rangle = |\psi_{x-}\rangle, \quad \sigma_z|\psi_{x-}\rangle = |\psi_{x+}\rangle \tag{5}$$

Relevant limits. Given the structure of $\mathcal{H}$, one can notice that there are two interesting regimes: $\lambda = 0$ and $\lambda \to \infty$.

$\lambda = 0$**.** In this case, the only relevant interactions are the ones of a spin with its nearest neighbours, because the value of the external field is set to zero.

It is simple to infer the shape of the *ground state*, that is the lowest energy state, as one of the two possible configurations with nearest spins always having opposite values, i.e. $|1/2, -1/2, 1/2, \dots\rangle$ and $|-1/2, 1/2, -1/2, \dots\rangle$. The energy E_{gs}^{nn}, i.e. the first eigenvalue of $\mathcal{H}$, is consequently equal to $-(N-1)$ for both the spin patterns.

Also the first excited state is easy to deduce: in this configuration there are two consequent spins with the same direction. Therefore, the energy is equal to $E_{1ex}^{nn} = E_{gs}^{nn} + 2 = -(N-3)$.

$\lambda \to \infty$**.** On the contrary, in the limit of an infinitely strong external field, one can deduce that the interactions between spins are negligible, and all the spins are hence aligned in the same direction, given by the external influence. The energy of the ground state, in this case, is proportional to both the magnitude of the external field and the number of spins: $E_{gs}^{\lambda} = -\lambda N$.

In this situation, the first excited state differs from the ground one only of a spin flip: this means that the energy is equal to: $E_{1ex}^{\lambda} = E_{gs}^{\lambda} + 2\lambda = -\lambda(N-2)$.

As one can deduce, being the two regimes significantly different, the two phases are separated by a *quantum phase transition*. It can be demonstrated that the transition occurs for $\lambda = 1$ in the limit of $N \to \infty$.

Code development

Computing tensor products. The first step in the code development consists of defining some fundamental support functions that allows computation of tensor (Kronecher) products. In particular, 3 functions are defined (Listing 1):

- `id_tens_mat`: performs the product between an identity matrix of given dimension and another input square matrix $\mathbb{1}_1 \otimes M$.
- `mat_tens_id`: performs the product between an input square matrix and an identity matrix of given dimension $M \otimes \mathbb{1}_2$.
- `id_tens_mat_tens_id`: combines the two previous functions to compute the tensor product $\mathbb{1}_1 \otimes M \otimes \mathbb{1}_2$.

Listing 1: Tensor product support functions.

```
function id_tens_mat(i_dim, N, matrix)result(txm)
! tensor product identity (X) matrix
integer, intent(IN)                                    :: i_dim
double complex, dimension(:,:), allocatable, intent(IN)  :: matrix
integer, intent(IN)                                    :: N
double complex, dimension(i_dim*N, N*i_dim)            :: txm

txm = (0d0, 0d0)
txm(1:N, 1:N) = matrix

do aa =2, i_dim
  bb = N*(aa-1) + 1
  txm( bb:(bb+N-1), bb:(bb+N-1) ) = matrix
end do

end function id_tens_mat

function mat_tens_id(i_dim, N, matrix)result(mxt)
! tensor product matrix (X) identity
integer, intent(IN)                          :: i_dim
double complex, dimension(:,:), intent(IN)   :: matrix
integer, intent(IN)                          :: N
double complex, dimension(:,:), allocatable  :: i_mat
double complex, dimension(i_dim*N, N*i_dim)  :: mxt

allocate(i_mat(i_dim, i_dim))
mxt = (0d0, 0d0)
i_mat = (0d0, 0d0)

do aa = 1, i_dim
  i_mat(aa, aa) = (1d0, 0d0)
end do

do aa = 1, N
  do bb = 1, N
    mxt( (i_dim*(aa - 1)+1):(i_dim*aa), &
       & (i_dim*(bb - 1)+1):(i_dim*bb) ) = matrix(aa, bb)*i_mat
  end do
end do

deallocate(i_mat)

end function mat_tens_id

```

```
function id_tens_mat_tens_id(i_dim1, N, matrix, i_dim2)result(prod)
! identity (X) matrix (X) identity
integer, intent(IN) :: i_dim1, i_dim2, N
double complex, dimension(:,:), intent(IN)                  :: matrix
double complex, dimension(i_dim1*i_dim2*N,i_dim1*i_dim2*N) :: prod
double complex, dimension(:,:), allocatable                 :: supp

! matrix (X) id2
if( i_dim2 > 0 ) then
  allocate( supp(N*i_dim2, N*i_dim2) )
  supp = mat_tens_id( i_dim2, N, matrix )
end if

! id1 (X) (matrix (X) id2)
if( i_dim1 > 0 ) then
  prod = id_tens_mat( i_dim1, N*i_dim2, supp )
  deallocate(supp)
end if

end function id_tens_mat_tens_id
```

Computing $\mathcal{H}$. Thanks to the functions defined in Listing 1, it is possible to compute the Hamiltonian $\mathcal{H}$ by applying them to the Pauli matrices σ_x, σ_z. This is done through the code in Listing 2.

The first piece of code performs the computation of the external interaction term, while the nearest neighbour interaction is calculated afterwards. The derivation of the latter has been split for each spin considered (i.e. `ii`, `ii+1`) as noted in Equation 1, then the two pieces are combined and added to the Hamiltonian.

Listing 2: Computation of the Hamiltonian.

```
! Interaction with the external field
do ii=1, N
  ! tensor product id (X) sigma_z
  temp = id_tens_mat_tens_id(d**(ii-1), d, sigma_z, d**(N-ii))
  ! multiplication for external field strength and summation
  H = H + lambda*temp
end do
temp = (0d0, 0d0)
! Interaction between nearest neighbors
do ii =1, N-1
  ! on sigma_x^ii
  temp = id_tens_mat_tens_id(d**(ii-1), d, sigma_x, d**(N-ii))
  ! on sigma_x^(ii+1)
  temp_int = id_tens_mat_tens_id(d**ii, d, sigma_x, d**(N-ii-1))
  ! combine the two interactions
  temp = MATMUL(temp, temp_int)
  ! summation
  H = H + temp
end do
```

Diagonalisation and computation of eigenvalues. The matrix $\mathcal{H}$ is now diagonalised and its first `kk`$= 4$ eigenvalues are computed thanks to the `LAPACK` subroutine `DSYEVR`. The eigenvalues are stored in order to study the energy spectrum, execution times for the computation of $\mathcal{H}$ and for solving the eigenproblem are also recorded. The final analysis is carried out via a `Python` script able to compile the `FORTRAN` code for several inputs and to manage the big number of files produced easily.

Results

Maximum number of spins. To evaluate the maximum size of the system N_{max} that the program can handle, the execution times for the two main processes, i.e. the creation of $\mathcal{H}$ and its eigenvalues computation, are collected for different system sizes N with an arbitrary fixed $\lambda = 0.5$. Results are shown in Figure 1.

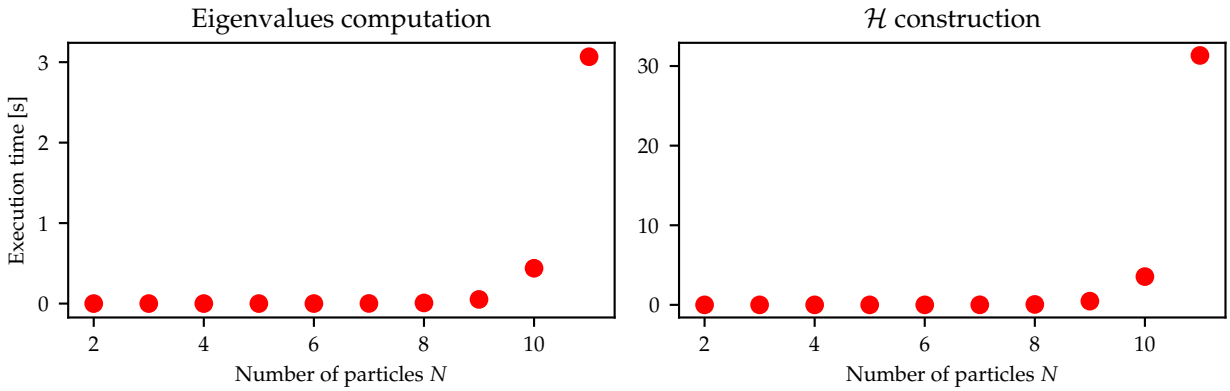

Figure 1: Execution times as a function of N, with fixed $\lambda = 0.5$.

It is immediately noticeable comparing the graphs that the process of computing the various tensor products is significantly the most time consuming (Fig. 1 on the right). However, both graphs show an exponential growth with N increasing, with a relevant raise of execution time for $N \geq 10$. As a consequence, for further analysis, only sizes of $N < 10$ are considered and N_{max} is assume to be 10.

Energy spectrum. The first 4 eigenvalues are collected for couples of $\lambda - N$, with $\lambda \in [0 : 3]$ and $N \in [3 : 9]$. In Figure 2, a comparison between the same eigenvalue as a function of λ for different N is reported. In order to properly compare different system sizes, a factor $1/(N-1)$ is multiplied for each eigenvalue.

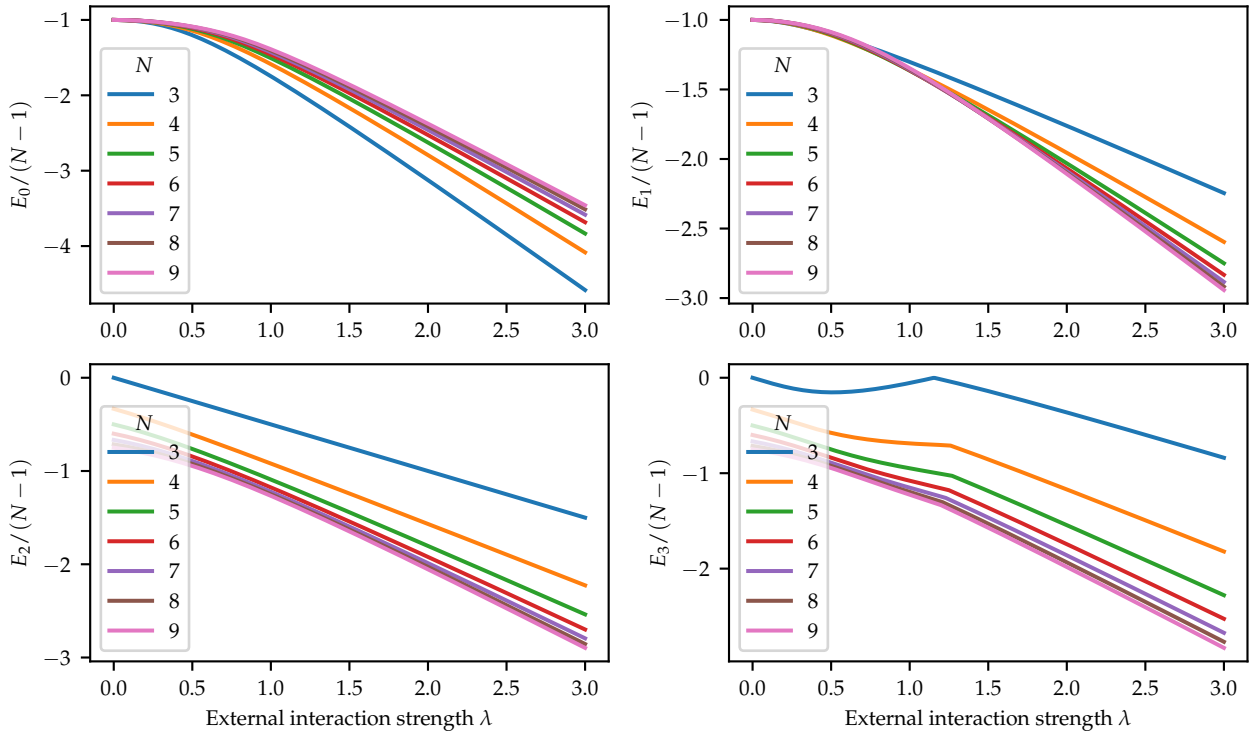

Figure 2: $E_k/(N-1)$ with $k \in [0 : 3]$, $N \in [3 : 9]$ as a function of λ.

The results obtained are consistent with the theory: in fact, when $\lambda \to 0$, the first two eigenvalues represent the energy of the two degenerate ground state configurations and indeed the graphs shows a y value very close to -1. With λ and N increasing, a diverging tendency can be observed: the degeneracy is broken by the increasing magnitude of the external field, and the studied quantities split, symbolising the phase transition expected from the theory. For higher values of λ the trend seems almost linear, for all the graphs, proving again the implementation to be coherent with the known theory.

Another striking feature proof of good implementation is that the eigenvalues increase with their order, as the ground state is the one of lowest energy.

Finally, an interesting trend can be outlined in every graphs, more evidently in the last one. In fact, there is a clear threshold ($\lambda \approx 1$) over which the behaviour of the observed quantities suddenly changes: this could be a clear sign of the quantum phase transition.

To observe this phenomenon better, some other plots are produced, both with a $\lambda-$logscale: in particular, in Figure 3 the first 4 eigenvalues are plotted vs. λ for 3 different Ns; while in Figure 4 the difference between E_2, E_1 and E_0 (energy gap) is displayed, in order to show better the break of degeneracy and the quantum phase transition.

In both the aforementioned pictures, the change of behaviour at $\lambda \approx 1$ is evident, and the turning point approaches 1 as N increases, because the theoretical model is studied ignoring the boundary conditions in the limit of $N \gg 1$.

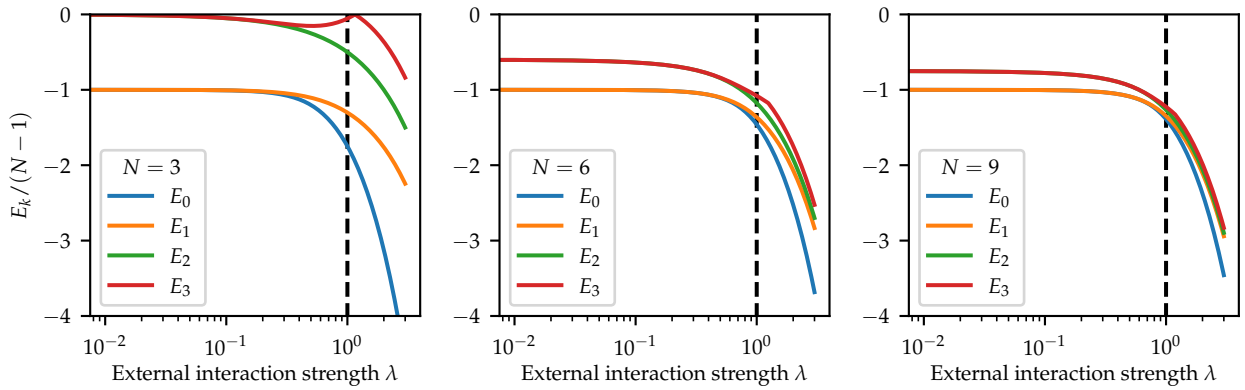

Figure 3: First 4 eigenvalues, divided by $N-1$, plotted vs. the logarithm of λ. The black line represents the threshold $\lambda = 1$.

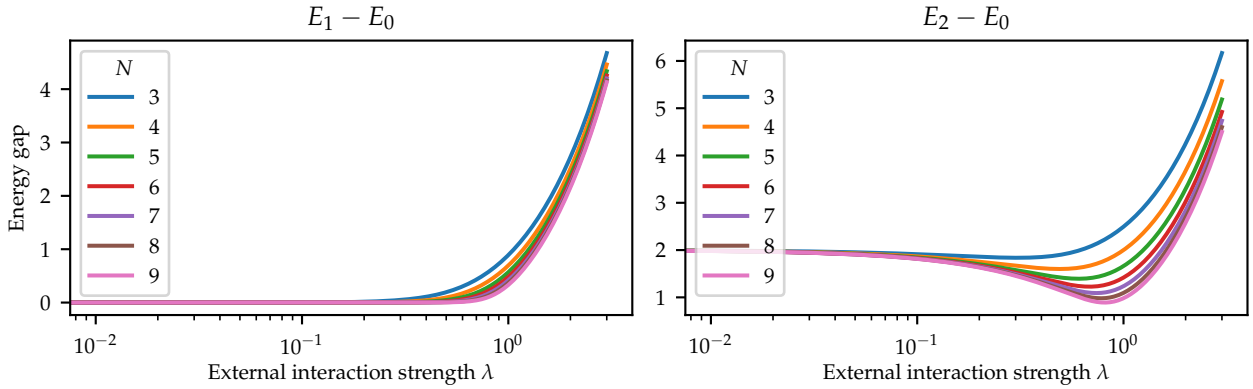

Figure 4: Energy gaps between E_1, E_2 wrt E_0, as a function of the logarithm of λ, for different sizes of the system N.

In this last picture, some other interesting theoretical results besides the already mentioned ones are displayed. Firstly, it is meaningful to remark that for $\lambda \to 0$ $E_1 \approxeq E_0$, while $E_2 - E_0 \approxeq 2$: this means that E_2 is indeed the energy of the first excited state. After the transition, one can observe that the energy gap between is roughly proportional to 2λ, as expected.

Self evaluation

All the implementation analysed produced results consistent with the theory, so one can conclude that the program worked properly. The code, however, is not particularly flexible and cannot analyse matrices with big dimensions, so this aspect can definitely be improved to have more insight about the problem. To conclude, a more deep analysis might be carried out concerning the excited states and their degeneracy, also considering a wider range of λs, but due to computational limits performing such task is considerably difficult.