UNIVERSITÀ DEGLI STUDI DI PADOVA

Quantum Information and Computing

2022 - 2023

Nguyen Xuan Tung

26/12/2022

Exercise #08

Physics of Data

Theory

We consider the quantum system formed by N spin-1/2 particles in presence of an external field of intensity λ. The problem given in an Hamiltonian represented by H:

H = $\lambda \sum_{i=1}^{N} \sigma_i^z - \sum_{i=1}^{N-1} \sigma_i^x \sigma_{i+1}^x$

Where we set J = 1. The notation simplifies the one of a tensor product that:

$\sigma_i^z = \mathbb{1}_1 \otimes \dots \mathbb{1}_{i-1} \otimes \sigma_i^z \otimes \mathbb{1}_{i+1} \otimes \dots \mathbb{1}_N$

$\sigma_i^x \sigma_{i+1}^x = \mathbb{1}_1 \otimes \dots \mathbb{1}_{i-1} \otimes \sigma_i^x \otimes \sigma_{i+1}^x \otimes \mathbb{1}_{i+2} \otimes \dots \mathbb{1}_N$

λ is the interaction strength parameter and the σs are the Pauli matrices.

In order to get the ground state of H, we exploit the real space renormalization group algorithm.

The latter is based on the hypothesis that the ground state of a system is composed of the low-energy states of its non-interacting bipartitions.

Renormalization group algorithms

The algorithm consists of the following steps:

1. Initialize Ising's Hamiltonian for a given N: H_N

2. Double the system size: $H_{2N} = H_N \otimes \mathbb{1}_N + \mathbb{1}_N \otimes H_N + H_{int}$ where $H_{int} = H^L + H^R$

and $\begin{cases} H^L = \bigotimes_{j=1}^{N-1} \mathbb{1} \otimes \sigma_i^x \\ H^R = \sigma_i^x \otimes \bigotimes_{j=1}^{N-1} \mathbb{1} \end{cases}$

3. Diagonalize H_{2N} and build the projector P using the first 2^N eigenvalues.

4. Reduce the 2N-Hamiltonian: $\widetilde{H_{2N}} = P^\dagger H_{2N} P$ where dim $[\widetilde{H_{2N}}] = 2^N$

5. Iterate step 1 and 4 for n_{inter} times.

Code development

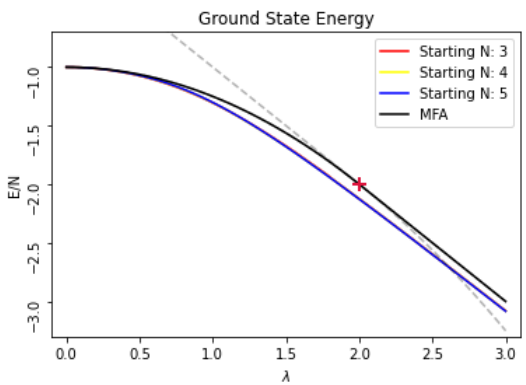
- Before looping: use function “ising_init_H” to initialize the first Hamiltonian (system size = N).
- Generate function to compute the product of 2 tensors.
- At each iteration, H2N = mat_tensor_I(HN) + I_tensor_mat(HN) + tens_prod(HL,HR)

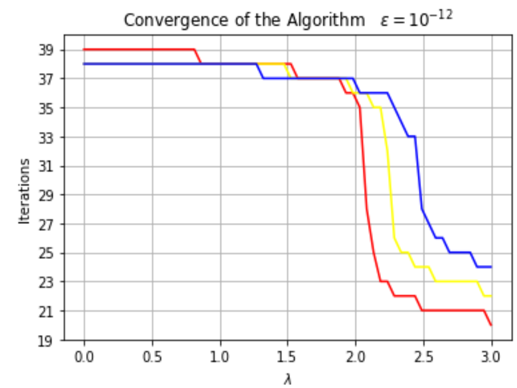
```fortran
function ising_init_H(N,lambda) result(H)
  integer   :: N
  double precision :: lambda
  double complex, dimension(:,:), allocatable :: H, int_A, int_B

  integer :: ii,jj,kk,ll

  allocate(H(2**N,2**N))
  H = 0.0 * H

  ! External field part: \lambda \sum_i^N \sigma_z^i
  do ii = 1, N, 1
    do jj = 1, 2**N, 1
      H(jj,jj) = H(jj,jj) + -2*(modulo( (jj-1)/int(2**(N-ii)),2) ) +1
    end do
  end do
  H = lambda * H ! Adding the magnetization field factor

  ! Interaction part -\sum_i^{N-1}\sigma_x^{i+1}\sigma_x^i
  do ii = 1, N-1, 1
    allocate(int_A(2**N,2**N))
    allocate(int_B(2**N,2**N))
    int_A = int_A * 0.0
    int_B = int_B * 0.0
    do kk = 0,2**(ii-1)-1,1
      do jj=1,2**(N-ii),1
        int_A(kk*(2**(N-ii+1))  + 2**(N-ii)+jj, kk*(2**(N-ii+1))  + jj) = 1
        int_A(kk*(2**(N-ii+1))  + jj, kk*(2**(N-ii+1))  + 2**(N-ii)+jj) = 1
      end do
    end do
    do kk = 0,2**(ii)-1,1
      do jj=1,2**(N-ii-1),1
        int_B(kk*(2**(N-ii)) + 2**(N-ii-1)+jj, kk*(2**(N-ii)) + jj) = 1
        int_B(kk*(2**(N-ii)) + jj, kk*(2**(N-ii)) + 2**(N-ii-1)+jj) = 1
      end do
    end do
    if(.False. .eqv. .True.) then
    print*, "mata"
    do jj = 1, ubound(int_A, 1)
      print*, "|", real(int_A(jj, :)), "|"
```

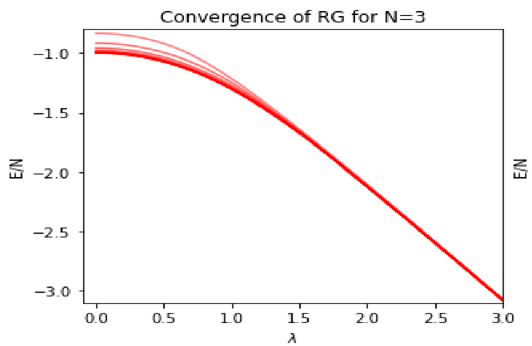
```fortran
function tens_prod(A, B) result(AoB)
  ! Computes A (X) B
  double precision, dimension(:,:) :: A, B
  double precision, dimension(:,:), allocatable :: AoB
  integer :: aa, bb, dimA, dimB

  dimA = size(A, 1)
  dimB = size(B, 1)
  allocate(AoB(dimA*dimB, dimA*dimB))

  do aa = 1, dimA, 1
    do bb = 1, dimB, 1
      AoB(dimB*(aa-1)+1:dimB*aa, dimB*(bb-1)+1:dimB*bb ) = A(aa, bb)*B
    end do
  end do
end function tens_prod
nd module tensor_prod

odule rg
use tensor_prod
contains
subroutine init_interaction_H(N,HL,HR)
  double precision, dimension(:,:), allocatable :: HL, HR
  integer :: kk, N

  allocate(HL(2**(N),2**(N)),HR(2**(N),2**(N)))
  HL = 0.d0
  HR = 0.d0

  do kk = 0, 2**(N-1)-1, 1
    HL(2+(2*kk),1+(2*kk)) = 1
    HL(1+(2*kk),2+(2*kk)) = 1

    HR(2**(N-1)+1+kk,1+kk) = 1
    HR(1+kk,2**(N-1)+1+kk) = 1
  end do
end subroutine
```

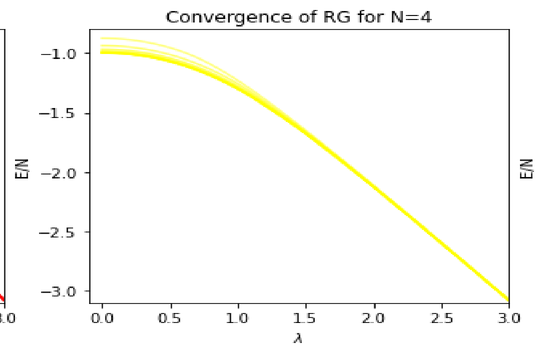
```fortran
! iterations
do it = 1, nit, 1
  if(modulo(it*10,nit)==0) then
    write(*,'(A,I6,A,I6,A)',advance='yes') " + ", it, "  /", nit, "
  end if

  allocate(H2N(2**(2*N),2**(2*N)))

  sizeofspace = 2*sizeofspace

  H2N = mat_tensor_I(HN) + I_tensor_mat(HN) + tens_prod(HL,HR)
  HLred = tens_prod(HL, identity(2**N))
  HRred = tens_prod(identity(2**N), HR)

  call diagonalize_H(H2N, evls, 2**N, P)

  call project(P, H2N, HN)
  call project(P, HLred, HL)
  call project(P, HRred, HR)
```

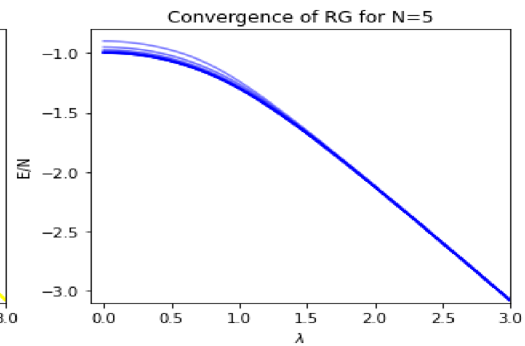
Result


Ground State Energy
Starting N: 3
Starting N: 4
Starting N: 5
MFA
E/N
λ


Convergence of the Algorithm $\varepsilon = 10^{-12}$
Iterations
λ


Convergence of RG for N=3
E/N
λ


Convergence of RG for N=4
E/N
λ


Convergence of RG for N=5
E/N
λ

1222 • 2022
800 ANNI

UNIVERSITÀ DEGLI STUDI DI PADOVA

Thanks for the attention