Assignment 8

https://github.com/jchryssanthacopoulos/quantum_information/tree/main/assignment_8

Quantum Information and Computing
AA 2022–23

James Chryssanthacopoulos
9 January 2023

Università degli Studi di Padova

Real-Space Renormalization Group

- **Real-space renormalization group** algorithm can be used to compute ground-state energy E_0 of quantum Ising model with traverse field in one dimension
- Following steps are repeated until $|E_0^{(i+1)} - E_0^{(i)}| < \epsilon$, where i is iteration number:
  1. Starting with Hamiltonian of system with N sites, $\hat{H}_N$, construct Hamiltonian by replicating system:

     $$\hat{H}_{2N} = \hat{H}_N \otimes \mathbb{1}_N + \mathbb{1}_N \otimes \hat{H}_N + \hat{H}_{\text{int}}$$

     where $\hat{H}_{\text{int}} = \hat{A}_N \otimes \hat{B}_N$ is interaction between left and right bipartitions, initialized with $\hat{A}_N = \mathbb{1}_{N-1} \otimes \sigma^x$, $\hat{B}_N = \sigma^x \otimes \mathbb{1}_{N-1}$. $\hat{H}_N$ is initialized to Hamiltonian of Ising model with N sites
  2. Diagonalize $\hat{H}_{2N}$, obtaining ground-state energy $E_0^{(i)}$. Construct projector using 2^N eigenvectors with lowest energy, $P = \sum_{j=i}^{2^N} |E_j\rangle \langle E_j|$
  3. Project operators into subspace spanned by chosen eigenvectors:

     $$\hat{H}_N = P^\dagger \hat{H}_{2N} P, \quad \hat{A}_N = P^\dagger (\hat{A}_N \otimes \mathbb{1}_N) P, \quad \hat{B}_N = P^\dagger (\mathbb{1}_N \otimes \hat{B}_N) P$$

Implementation

Program computes ground-state energy given number of sites N, interaction strength λ, termination threshold ϵ, and maximum number of iterations

```
! compute initial Hamiltonian
H = lambda * non_interacting_hamiltonian(N) + interacting_hamiltonian(N)

! compute operators in interaction Hamiltonian
sigma_x = get_sigma_x()
A = tensor_product(identity(N - 1), sigma_x)
B = tensor_product(sigma_x, identity(N - 1))

iter = 1
gs_energy = -1
prev_gs_energy = 0

do while ((iter .le. max_iter) .and. abs(gs_energy - prev_gs_energy) > thres)
    prev_gs_energy = gs_energy

    H = H * 0.5
    A = 1 / sqrt(2.) * A
    B = 1 / sqrt(2.) * B

    call iter_real_space_rg(N, H, A, B, eigenvalues, diag_method, debug)

    ! compute energy density
    gs_energy = eigenvalues(1) / dble(N)

    if (debug) then
        print "('Iteration = ', i3, ', ground state energy = ', f9.4)", iter, gs_energy
    end if

    iter = iter + 1
end do
```

$\hat{H}_N$, $\hat{A}_N$, $\hat{B}_N$ initialized

Operators scaled to account for increased system size

System replicated to produce $\hat{H}_{2N}$

```
H_2N = tensor_product(H_N, identity(N)) + tensor_product(identity(N), H_N) + tensor_product(A, B)
A_2N = tensor_product(A, identity(N))
B_2N = tensor_product(identity(N), B)
```

Eigenvalues and eigenvectors computed, P computed by truncating

```
! compute using zheev
call find_eigenvalues_using_zheev(H_2N, eigenvalues, full_P)

! truncate
do ii = 1, ndim
    full_P_trunc(:, ii) = full_P(:, ii)
end do
```

Operators projected into subspace

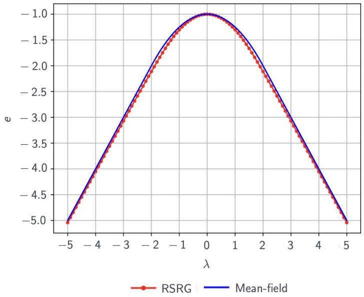
```
full_P_trunc_transpose = transpose(conjg(full_P_trunc))

H_N = matmul(matmul(full_P_trunc_transpose, H_2N), full_P_trunc)
A = matmul(matmul(full_P_trunc_transpose, A_2N), full_P_trunc)
B = matmul(matmul(full_P_trunc_transpose, B_2N), full_P_trunc)
```

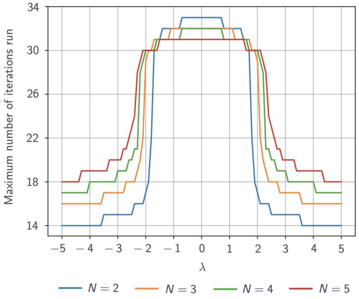
Results

- Energy density, $e \equiv E_0/N$, computed using RSRG compared to mean-field result:

$$e_{\mathrm{MF}} = \begin{cases} -1 - \lambda^2/4, & |\lambda| \leq 2 \\ -|\lambda|, & |\lambda| > 2 \end{cases}$$

- Energy densities are very similar, but deviate the most when $0 < |\lambda| < 2$, when external field is present but not strong enough to coordinate all spins
- Experiments showed that energy density doesn't change with N, as expected
- Number of iterations increases with N, except when field is weak





Density Matrix Renormalization Group

Density matrix renormalization group algorithm improves truncation rule, resulting in more accurate final state while increasing computational complexity. Below are the steps:

1. Starting with Hamiltonian of system with N sites, $\hat{H}_1$, add one site to right to get Hamiltonian of enlarged block:

$$\hat{H}_B = \hat{H}_1 \otimes \mathbb{1}_1 + \mathbb{1}_N \otimes \hat{H}_{\text{single}} + \hat{H}_{12} \otimes \sigma^x$$

   where $\hat{H}_{\text{single}}$ is Ising Hamiltonian with single site (i.e., $\lambda\sigma^z$) and $\hat{H}_{12}$ is Hamiltonian representing rightmost spin of $\hat{H}_1$ (i.e., $\mathbb{1}_{N-1} \otimes \sigma^x$)

2. Replicate system to produce right enlarged block. Total Hamiltonian is:

$$\hat{H} = \hat{H}_B \otimes \mathbb{1}_{N+1} + \mathbb{1}_{N+1} \otimes \hat{H}_B + \hat{H}_{\text{int}}$$

   where $\hat{H}_{\text{int}}$ is interaction between left and right blocks (i.e., $\mathbb{1}_N \otimes \sigma^x \otimes \sigma^x \otimes \mathbb{1}_N$)

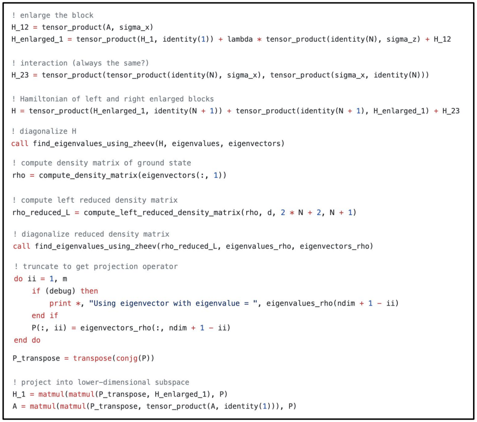
3. Diagonalize $\hat{H}$, obtaining $E_0^{(i)}$ with wavefunction $|\psi_0\rangle$. Construct density matrix $\rho = |\psi_0\rangle\langle\psi_0|$, and reduced density matrix of left bipartition, $\rho_L = \text{Tr}_R\rho$. Construct projector using 2^N eigenvectors with highest eigenvalues, $P = \sum_{j=i}^{2^N} |w_j\rangle\langle w_j|$

4. Project operators into subspace spanned by chosen eigenvectors:

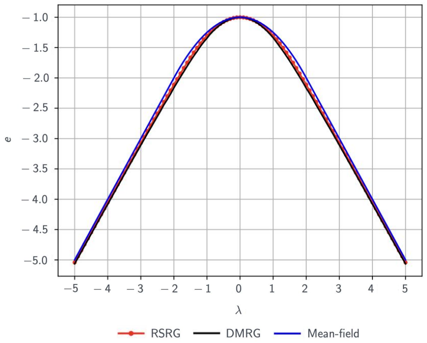
$$\hat{H}_1 = P^\dagger \hat{H}_B P, \quad \hat{H}_{12} = P^\dagger(\hat{H}_{12} \otimes \mathbb{1}_1)P$$

Implementation and Results

Ground-state energy with DMRG similar to that with RSRG, with percent difference between 1–4%



```
! enlarge the block
H_12 = tensor_product(A, sigma_x)
H_enlarged_1 = tensor_product(H_1, identity(1)) + lambda * tensor_product(identity(N), sigma_z) + H_12

! interaction (always the same?)
H_23 = tensor_product(tensor_product(identity(N), sigma_x), tensor_product(sigma_x, identity(N)))

! Hamiltonian of left and right enlarged blocks
H = tensor_product(H_enlarged_1, identity(N + 1)) + tensor_product(identity(N + 1), H_enlarged_1) + H_23

! diagonalize H
call find_eigenvalues_using_zheev(H, eigenvalues, eigenvectors)

! compute density matrix of ground state
rho = compute_density_matrix(eigenvectors(:, 1))

! compute left reduced density matrix
rho_reduced_L = compute_left_reduced_density_matrix(rho, d, 2 * N + 2, N + 1)

! diagonalize reduced density matrix
call find_eigenvalues_using_zheev(rho_reduced_L, eigenvalues_rho, eigenvectors_rho)

! truncate to get projection operator
do ii = 1, m
    if (debug) then
        print *, "Using eigenvector with eigenvalue = ", eigenvalues_rho(ndim + 1 - ii)
    end if
    P(:, ii) = eigenvectors_rho(:, ndim + 1 - ii)
end do

P_transpose = transpose(conjg(P))

! project into lower-dimensional subspace
H_1 = matmul(matmul(P_transpose, H_enlarged_1), P)
A = matmul(matmul(P_transpose, tensor_product(A, identity(1))), P)
```


e
λ
RSRG
DMRG
Mean-field