Iterative diagonalization

Author: Seung-Sup Lee

Here we will study the **iterative diagonalization of a tight-binding chain of non-interacting spinless fermions**. Before starting, please review the following functions added to the code repository:

- `Tensor/getLocalSpace.m`: Generates the operators for a site of spin, spinless fermion, or spinful fermions.
- `Tensor/getIdentity.m`: Generates the identity operator for the Hilbert space of one leg, or for the product space of the spaces for two legs.
- `Tensor/updateLeft.m`: Contract the operator residing in the Hilbert space of the left part of the MPS (i.e., left of a given site) with the local operators at the given site.

Understand how to use these functions, and also how the functions generate/manipulate tensors. We will use these functions frequently in this and the following tutorials.

Tight-binding chain of spinless fermions

Consider the Hamiltonian of a tight-binding chain of non-interacting fermions of length N:

$$\hat{H} = -t \sum_{\ell=1}^{N-1} \left(\hat{c}_{\ell+1}^{\dagger} \hat{c}_{\ell} + \hat{c}_{\ell}^{\dagger} \hat{c}_{\ell+1} \right),$$

where $\hat{c}_{\ell}$ annihilates particle at site ℓ. Without loss of generality, we can set the hopping amplitude $t = 1$. Since it is non-interacting system, we can obtain the ground-state energy of the system in a simple way. First, represent the Hamiltonian in the first-quantized (i.e., single-particle) basis, $\{|1\rangle, |2\rangle, \cdots, |N\rangle\}$.

```
clear

t = 1; % hopping amplitude
N = 50; % number of sites in the chain
Hsp = diag(-t+zeros(N-1,1),1); % single-particle basis
Hsp = Hsp+Hsp';
```

The superdiagonal (accessed by `diag(Hsp,1)`) and the subdiagonal (accessed by `diag(Hsp,-1)`) of the matrix are filled by $-t = 1$. The super- and sub-diagonals represent the hopping between neighboring sites. The rest of the elements of `Hsp` are zeros.

Diagonalize the Hamiltonian, and sum over negative energy eigen values; the sum is the ground-state energy. This way is numerically exact.

```
D = eig((Hsp+Hsp')/2); % Hermitianize to avoid numerical error
fprintf('Ground-state energy per site = %.4g\n', ...
    sum(D(D<0))/N);
```

```
Ground-state energy per site = -0.6295
```

Quick tip: Hermitianize Hamiltonian

Above, we see that the Hamiltonian is "Hermitianized" before being diagonalized. As we all know, Hamiltonian should be Hermitian, in principle. But in practice, the numerical representation of the Hamiltonian is susceptible to noise, so that it becomes slightly non-Hermitian. It will lead to complex-valued eigenvalues (which should be always real originally) and eigenvectors (which should be real-valued for the real-valued Hamiltonians we consider in this tutorial). They are not only wrong, but also a possible source of unnecessary computational overhead. To avoid this numerical noise, one can Hermitianize the Hamiltonian when it is diagonalized: for example, `eig((Hnow+Hnow')/2)`.

Iterative diagonalization

We will use the iterative diagonalization for the same Hamiltonian, by using the second-quantized (i.e., many-body) basis. And we will compare the results from the iterative diagonalization and from the single-particle basis, for different chain lengths from 1 to N. First, we introduce the key parameters for the iterative diagonalization.

```
N = 100; % maximum chain length
Nkeep = 300; % maximal number of states to keep
tol = Nkeep*100*eps; % numerical tolerance for degeneracy
```

When we truncate the Hilbert space at each iteration, we need to keep all the states whose energies are close to the truncation threshold. The "closeness" is determined by the tolerance parameter `tol`, and we regard the states separated within this tolerance as being degenerate. The degeneracy often comes from physical symmetries, such as spin and particle-hole symmetry. If we keep only the part of the degenerate states, then the Hilbert space will not respect the symmetry anymore. This artificial symmetry breaking would lead to qualitatively wrong result.

Define the operators (particle annihilation `F`, fermionic sign `Z`, identity `I`) acting on each chain site.

```
[F,Z,I] = getLocalSpace('Fermion');
```

Define the Hamiltonian `H0` and the ket tensor `A0` for the first chain site. The legs of `H0` and the third leg of `A0` are associated with the same Hilbert space. The left leg (i.e., first leg) of `A0` is a dummy leg which has trivial dimension 1. To generate such dummy leg, we use a number as the first input to `getidentity`. Note that it does not need to be 1; it can be any number, as long as it is 1-by-1 object.

```
H0 = I*0; % Hamiltonian for only the 1st site
A0 = getIdentity(1,2,I,2); % 1st leg is dummy leg (vacuum)
```

Now diagonalize the Hamiltonian iteratively.

```
% lowest energies at each iteration
E0 = zeros(1,N);

for itN = (1:N)
    if itN == 1
        Hnow = H0;
        [V,D] = eig((Hnow+Hnow')/2);
```

```
        AK = contract(A0,3,3,V,2,1);
        E0(itN) = min(diag(D));
        Hprev = D;
    else
        % % add new site
        Anow = getIdentity(Hprev,2,I,2);
        Hnow = updateLeft(Hprev,2,Anow,[],[],Anow);
        % update the Hamiltonian up to the last sites
        % to the enlarged Hilbert space

        % creation operator at the new site
        Fn = permute(conj(F),[3 2 1]);
        % F'*Z; contract fermionic sign op.
        Fn = contract(Fn,3,3,Z,2,1);
        % hopping from the last site to the new site
        Hhop = updateLeft(Fprev,3,Anow,Fn,3,Anow);
        Hhop = (-t)*Hhop; % multiply hopping amplitude
        % add the hopping term from the new site
        % to the last site
        Hhop = Hhop+Hhop';

        Hnow = Hnow+Hhop;

        [V,D] = eig((Hnow+Hnow')/2);
        % sort eigenvalues and eigenvectors in the order of increasing
        % eigenvalues
        [D,ids] = sort(diag(D),'ascend');
        V = V(:,ids);

        E0(itN) = min(D);

        % truncation threshold for energy
        Etr = D(min([numel(D);Nkeep]));
        oks = (D < (Etr + tol));
        % true: to keep, false: not to keep
        % keep all degenerate states up to tolerance

        AK = contract(Anow,3,3,V(:,oks),2,1);
        Hprev = diag(D(oks));
    end

    % particle annihilation operator at the
    % current site; to generate the hopping
    % term at the next iteration
    Fprev = updateLeft([],[],AK,F,3,AK);

    disptime(['#',sprintf('%02i/%02i',[itN,N]),' : ', ...
        'NK=',sprintf('%i/%i',[size(AK,3),size(Hnow,2)])]);
end
```

```
21-04-20 22:24:09 | #01/100 : NK=2/2
21-04-20 22:24:09 | #02/100 : NK=4/4
21-04-20 22:24:09 | #03/100 : NK=8/8
21-04-20 22:24:09 | #04/100 : NK=16/16
21-04-20 22:24:09 | #05/100 : NK=32/32
```

```
21-04-20 22:24:09 | #06/100 : NK=64/64
21-04-20 22:24:09 | #07/100 : NK=128/128
21-04-20 22:24:09 | #08/100 : NK=256/256
21-04-20 22:24:09 | #09/100 : NK=302/512
21-04-20 22:24:09 | #10/100 : NK=300/604
21-04-20 22:24:09 | #11/100 : NK=300/600
21-04-20 22:24:09 | #12/100 : NK=300/600
21-04-20 22:24:09 | #13/100 : NK=300/600
21-04-20 22:24:09 | #14/100 : NK=300/600
21-04-20 22:24:09 | #15/100 : NK=300/600
21-04-20 22:24:09 | #16/100 : NK=300/600
21-04-20 22:24:09 | #17/100 : NK=300/600
21-04-20 22:24:10 | #18/100 : NK=301/600
21-04-20 22:24:10 | #19/100 : NK=300/602
21-04-20 22:24:10 | #20/100 : NK=301/600
21-04-20 22:24:10 | #21/100 : NK=300/602
21-04-20 22:24:10 | #22/100 : NK=300/600
21-04-20 22:24:10 | #23/100 : NK=300/600
21-04-20 22:24:10 | #24/100 : NK=300/600
21-04-20 22:24:10 | #25/100 : NK=300/600
21-04-20 22:24:10 | #26/100 : NK=300/600
21-04-20 22:24:10 | #27/100 : NK=300/600
21-04-20 22:24:10 | #28/100 : NK=300/600
21-04-20 22:24:10 | #29/100 : NK=300/600
21-04-20 22:24:10 | #30/100 : NK=300/600
21-04-20 22:24:10 | #31/100 : NK=300/600
21-04-20 22:24:10 | #32/100 : NK=300/600
21-04-20 22:24:10 | #33/100 : NK=300/600
21-04-20 22:24:10 | #34/100 : NK=300/600
21-04-20 22:24:10 | #35/100 : NK=300/600
21-04-20 22:24:10 | #36/100 : NK=300/600
21-04-20 22:24:10 | #37/100 : NK=300/600
21-04-20 22:24:10 | #38/100 : NK=300/600
21-04-20 22:24:10 | #39/100 : NK=300/600
21-04-20 22:24:10 | #40/100 : NK=300/600
21-04-20 22:24:10 | #41/100 : NK=300/600
21-04-20 22:24:10 | #42/100 : NK=300/600
21-04-20 22:24:10 | #43/100 : NK=300/600
21-04-20 22:24:10 | #44/100 : NK=300/600
21-04-20 22:24:10 | #45/100 : NK=300/600
21-04-20 22:24:10 | #46/100 : NK=300/600
21-04-20 22:24:11 | #47/100 : NK=300/600
21-04-20 22:24:11 | #48/100 : NK=300/600
21-04-20 22:24:11 | #49/100 : NK=300/600
21-04-20 22:24:11 | #50/100 : NK=300/600
21-04-20 22:24:11 | #51/100 : NK=300/600
21-04-20 22:24:11 | #52/100 : NK=300/600
21-04-20 22:24:11 | #53/100 : NK=300/600
21-04-20 22:24:11 | #54/100 : NK=300/600
21-04-20 22:24:11 | #55/100 : NK=300/600
21-04-20 22:24:11 | #56/100 : NK=300/600
21-04-20 22:24:11 | #57/100 : NK=300/600
21-04-20 22:24:11 | #58/100 : NK=300/600
21-04-20 22:24:11 | #59/100 : NK=300/600
21-04-20 22:24:11 | #60/100 : NK=300/600
21-04-20 22:24:11 | #61/100 : NK=300/600
21-04-20 22:24:11 | #62/100 : NK=300/600
21-04-20 22:24:11 | #63/100 : NK=300/600
21-04-20 22:24:11 | #64/100 : NK=300/600
21-04-20 22:24:11 | #65/100 : NK=300/600
21-04-20 22:24:11 | #66/100 : NK=300/600
21-04-20 22:24:11 | #67/100 : NK=300/600
21-04-20 22:24:11 | #68/100 : NK=300/600
21-04-20 22:24:11 | #69/100 : NK=300/600
21-04-20 22:24:11 | #70/100 : NK=300/600
```

```
21-04-20 22:24:11 | #71/100 : NK=300/600
21-04-20 22:24:11 | #72/100 : NK=300/600
21-04-20 22:24:12 | #73/100 : NK=300/600
21-04-20 22:24:12 | #74/100 : NK=300/600
21-04-20 22:24:12 | #75/100 : NK=300/600
21-04-20 22:24:12 | #76/100 : NK=300/600
21-04-20 22:24:12 | #77/100 : NK=300/600
21-04-20 22:24:12 | #78/100 : NK=300/600
21-04-20 22:24:12 | #79/100 : NK=300/600
21-04-20 22:24:12 | #80/100 : NK=300/600
21-04-20 22:24:12 | #81/100 : NK=300/600
21-04-20 22:24:12 | #82/100 : NK=300/600
21-04-20 22:24:12 | #83/100 : NK=300/600
21-04-20 22:24:12 | #84/100 : NK=300/600
21-04-20 22:24:12 | #85/100 : NK=300/600
21-04-20 22:24:12 | #86/100 : NK=300/600
21-04-20 22:24:12 | #87/100 : NK=300/600
21-04-20 22:24:12 | #88/100 : NK=300/600
21-04-20 22:24:12 | #89/100 : NK=300/600
21-04-20 22:24:12 | #90/100 : NK=300/600
21-04-20 22:24:12 | #91/100 : NK=300/600
21-04-20 22:24:12 | #92/100 : NK=300/600
21-04-20 22:24:12 | #93/100 : NK=300/600
21-04-20 22:24:12 | #94/100 : NK=300/600
21-04-20 22:24:12 | #95/100 : NK=300/600
21-04-20 22:24:12 | #96/100 : NK=300/600
21-04-20 22:24:12 | #97/100 : NK=300/600
21-04-20 22:24:12 | #98/100 : NK=300/600
21-04-20 22:24:12 | #99/100 : NK=300/600
21-04-20 22:24:12 | #100/100 : NK=300/600
```

Here each line indicates the information of each iteration step: time stamp, number of the kept states, and number of the total states.

Then the ground-state energy per site for different chain lengths are given by:

```
EG_iter = E0./(1:N);
```

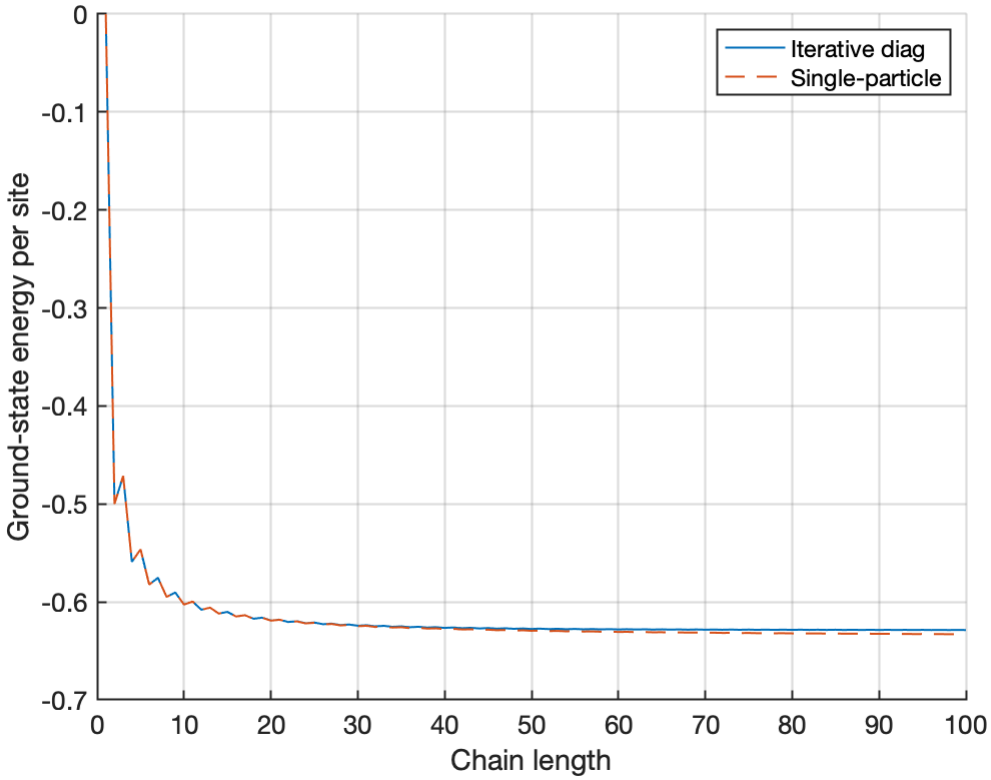
Obtain also the numerically exact result obtained by using the single-particle basis.

```
EG_single = zeros(1,N);
for itN = (1:N)
    Hsp = diag(-t+zeros(itN-1,1),1);
    Hsp = Hsp+Hsp';
    D = eig((Hsp+Hsp')/2);
    EG_single(itN) = sum(D(D<0))/itN;
end
```

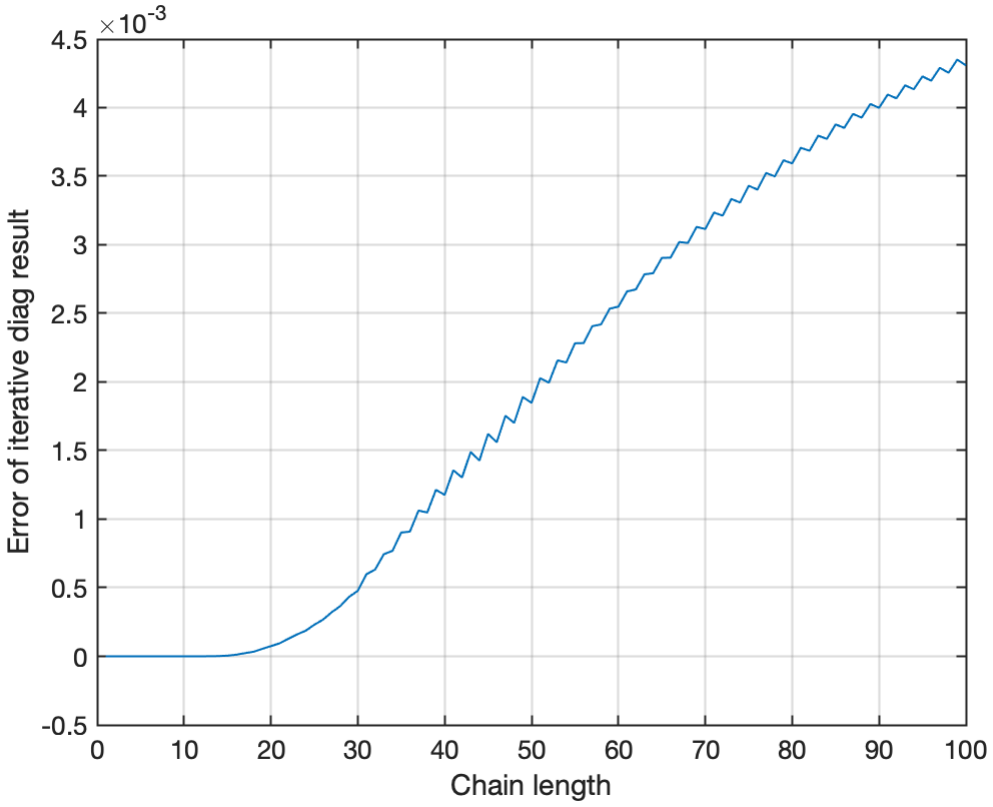
Let's compare these results.

```
figure;
hold on;
plot((1:N),EG_iter,'LineWidth',1,'LineStyle','-');
plot((1:N),EG_single,'LineWidth',1,'LineStyle','--');
hold off;
set(gca,'LineWidth',1,'FontSize',13);
xlabel('Chain length');
ylabel('Ground-state energy per site');
```

```
legend({'Iterative diag','Single-particle'}, ...
    'Location','northeast');
grid on;
```




```
figure;
plot((1:N),EG_iter-EG_single,'LineWidth',1,'LineStyle','-');
set(gca,'LineWidth',1,'FontSize',13);
xlabel('Chain length');
ylabel('Error of iterative diag result');
grid on;
```



We observe that the error increases consistently, as the chain length grows. Note that, for short chains for which we can keep all the states, the iterative diagonalization result is identical to the single-particle basis result (up to double precision error).

```
disp(EG_iter(1:floor(log(Nkeep)/log(size(I,2)))) - ...
    EG_single(1:floor(log(Nkeep)/log(size(I,2)))));
```

```
    1.0e-15 *

         0         0    0.1110         0    0.1110         0   -0.8882   -0.1110
```

Brief remark

We see that the result of this "simple-minded" approach is not that bad, at least for the given example. However, we also see the clear limitations:

- It is hard to go to the thermodynamic limit (i.e., the limit of infinite system size), since the system size under consideration increases one by one at each iteration.
- The truncation error (i.e., error induced by truncating the Hilbert space) accumulates as one proceeds with the iterations.
- It is hard to achieve higher accuracy. Larger `Nkeep` leads to better accuracy, but it also requires larger computational cost.

Moreover, this simple-minded iterative diagonalization works badly for most of the systems of interacting particles. That's why, indeed, we learn the numerical renormalization group (NRG) and the density-matrix renormalization group (DMRG) from this course!

Exercise (a): Heisenberg spin chains

Use the iterative diagonalization to obtain the ground-state energy per site for the Heisenberg spin chains, for (i) spin-1/2's and (ii) spin-1's. The Hamiltonian for the Heisenberg spin chain of length N is given by:

$$H = J \sum_{\ell=1}^{N-1} \hat{\vec{S}}_{\ell} \cdot \hat{\vec{S}}_{\ell+1}$$

Here we can set the coupling strength $J = 1$ without loss of generality. In the infinite chain limit ($N \to \infty$), the ground-state energy per site of the spin-1/2 chain is $(1/4) - \log 2$ [Bethe ansatz result; see S. R. White, Phys. Rev. Lett. **69**, 2863 (1992)], and that of the spin-1 chain is -1.401484039 [DMRG result; see S. R. White and D. A. Huse, Phys. Rev. B **48**, 3844 (1993)]. Compare your iterative diagonalization results for these infinite-limit values.

Exercise (b): Spin-1/2 fermions on a chain, with an interacting site at one end

Consider a tight-binding chain of spin-1/2 fermions of length N. Its first site has the local Coulomb interaction of strength U. That is, the spin-up and -down particles sitting together at the first chain site experience extra energy of U. The rest of the chain is non-interacting. Its Hamiltonian is given by

$$\hat{H} = -t \sum_{s=\uparrow,\downarrow} \sum_{\ell=1}^{N-1} \left(\hat{c}^{\dagger}_{\ell+1,s} \hat{c}_{\ell,s} + \hat{c}^{\dagger}_{\ell,s} \hat{c}_{\ell+1,s} \right) + U \hat{n}_{1\uparrow} \hat{n}_{1\downarrow} - \frac{U}{2} (\hat{n}_{1\uparrow} + \hat{n}_{1\downarrow}),$$

where $\hat{c}_{\ell,s}$ annihilates a particle of spin $s = \uparrow, \downarrow$ at site ℓ, and $\hat{n}_{\ell,s} = \hat{c}^{\dagger}_{\ell,s} \hat{c}_{\ell,s}$ is number operator. We set the hopping amplitude $t = 1$. The ground-state energy for this Hamiltonian can be obtained in a similar fashion as for the spinless case demonstrated above, for two limiting cases, $U = 0$ and $U \to \infty$. (i) **Derive the limiting case solutions of the ground-state energy (semi-)analytically.** (Note that the reference ground-state energy for the spinless case above is semi-analytic, as we have numerically diagonalized the single-particle Hamiltonian.) (ii) **Obtain the iterative diagonalization results, and compare them with the limiting case solutions.**