[Tutorial T08] iTEBD: Ground state

Zoom session, 09.06.2020

Author: Seung-Sup Lee

Date: 01.06.2020

Here we will obtain the ground states of infinite-size spin chains, by using infinite time-evolving block decimation (iTEBD) method. There are a few differences from the last tutorial. In **the last tutorial T07**, we have obtained the ground state of **finite systems** by using the one-site DMRG in which the MPS in the **site-canonical form** is **variationally updated**. In **this tutorial**, we consider **infinite systems** by using the **imaginary time evolution** of the MPS in **Vidal's Γ-Λ notation**.

Exercise 1: Complete `iTEBD_GS_Ex.m` for Vidal's original iTEBD

There is a function `iTEBD_GS_Ex.m` which is zipped together with this document. It is designed to compute the ground state by using G. Vidal's original iTEBD algorithm [G. Vidal, Phys. Rev. Lett. **98**, 070201 (2007) or its arXiv version]. Complete this function. The parts to be completed are enclosed by comments "`TODO - Exercise 1`". Ignore the parts enclosed by "`TODO - Exercise 2`", which is the next exercise.

AKLT model

Check whether your implementation of `iTEBD_GS_Ex.m` is correct, by applying it to the exactly solvable model. The simplest example is the AKLT model, which is the chain of spin-1's that interact via nearest-neighbor interactions,

$$\hat{H} = \sum_{\ell} \left[(\hat{\vec{S}}_{\ell} \cdot \hat{\vec{S}}_{\ell+1}) + \frac{1}{3} (\hat{\vec{S}}_{\ell} \cdot \hat{\vec{S}}_{\ell+1})^2 \right].$$

```
clear

% iTEBD parameters
Nkeep = 10;
beta_init = 1; % initial imaginary time step size
beta_fin = 0.01; % final imaginary time step size
Nstep = 500; % number of imaginary time steps
betas = beta_init*((beta_fin/beta_init).^linspace(0,1,Nstep));
% discrete imaginary time steps; decays slowly but exponentially

% Local operators
[S,I] = getLocalSpace('Spin',1);
```

Construct the interaction term. Note that the second spin operator, $\hat{\vec{S}}_{\ell+1}$ in the construction of $\hat{\vec{S}}_{\ell} \cdot \hat{\vec{S}}_{\ell+1}$, should be Hermitian conjugated (i.e., complex conjugated and transposed).

```
% Heisenberg interaction as two-site gate S*S'
```

```
HSS = contract(S,3,2,permute(conj(S),[3 2 1]),3,2);
% (S*S')^2 interaction
HSS2 = contract(HSS,4,[2 4],HSS,4,[1 3],[1 3 2 4]);
%      2      4
%      ^      ^
%      |      |
%   [ HSS or HSS2 ]
%      |      |
%      ^      ^
%      1      3

% % % AKLT
H = HSS + HSS2/3;
```

As the interaction term acts on two sites, we consider a unit cell of two sites. Initialize the tensors, which would consitute the ket tensor for the unit cell, with random tensors. The conditions to be fulfilled as the ket tensor, such as the left- or right-normalization, will be automatically achieved by the imaginary time evolution.

```
% Initialize with random Lambda and Gamma
Lambda = cell(1,2);
Gamma = cell(1,2);
for itn = (1:numel(Lambda))
    Lambda{itn} = rand(Nkeep,1);
    Gamma{itn} = rand(Nkeep,size(I,2),Nkeep);
end
```

For convenience, we will call the bond associated with `Lambda{1}` as even, and the other with `Lambda{2}` as odd.

Start the iTEBD calculation.

```
% iTEBD ground state search
[Lambda,Gamma,Es] = iTEBD_GS_Ex(Lambda,Gamma,H,Nkeep,betas);
```

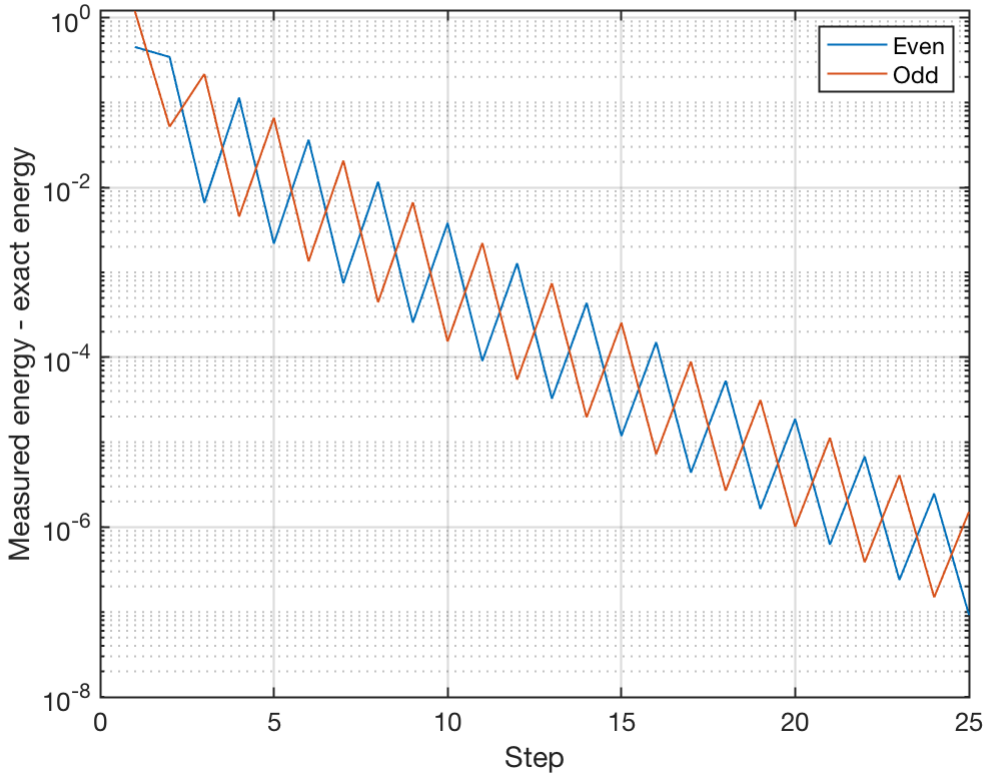
```
iTEBD ground state search: Vidal's method.
  # of sites = 2, Bond dim. Nkeep = 10, # of imag. time steps = 500
20-06-01 11:38:55 | #500/500, E = -0.66666667
Elapsed time: 0.6528s, CPU time: 0.88s, Avg # of cores: 1.348
20-06-01 11:38:55 | Memory usage : 1.90GiB
```

As we studied in the previous lecture and the previous tutorial, the ground state of the AKLT model is literally the AKLT state. We will compare the iTEBD result with the exact results.

First, let's compare the energy. The exact ground state energy per site for the AKLT state is -2/3. We observe that the energies at each bond (each column of `Es`) converge to the exact value with oscillations. Why oscillating? When we apply the imaginary time evolution to the tensors sandwiching one bond (say even bond), the time evolution operator changes the tensors in a way that the energy for the even bond decreases at the cost of increasing the energy of the odd bond.

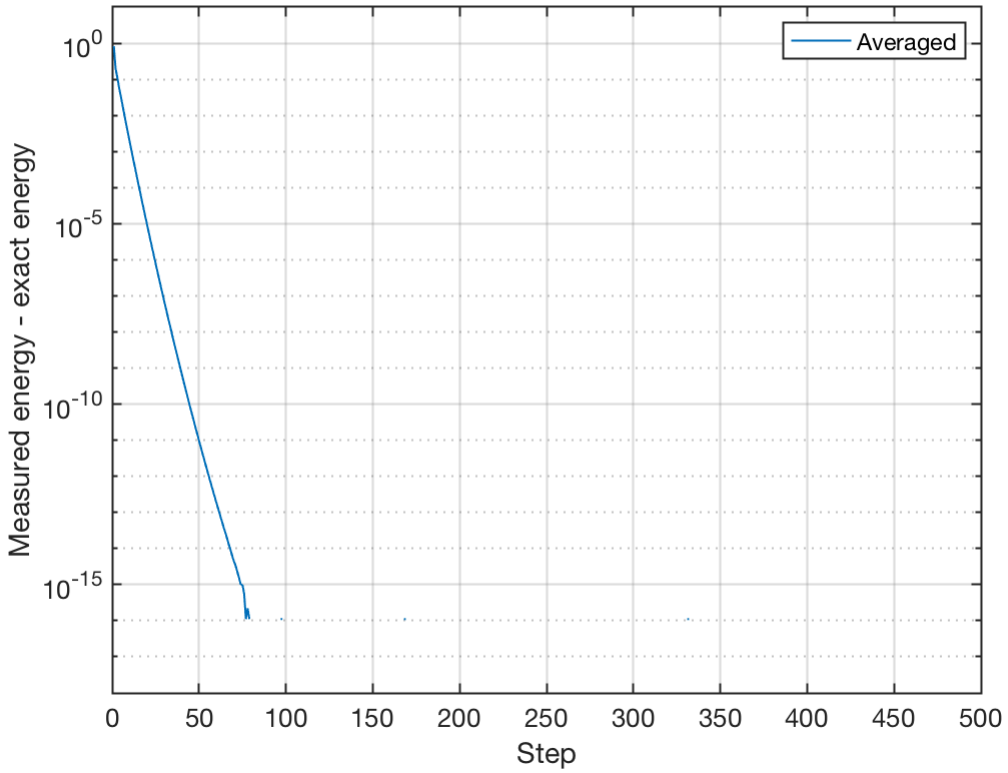
```
Eexact = -2/3;
figure;
plot((1:size(Es,1)).',Es-Eexact,'LineWidth',1);
```

```
xlim([0 25]);
set(gca,'LineWidth',1,'FontSize',13,'YScale','log');
xlabel('Step');
ylabel('Measured energy - exact energy');
legend({'Even','Odd'});
grid on;
```



To further see the convergence, we average the energies over even and odd bonds, which is equivalent to the energy per site for infinite systems. Actually, the energy values (E = ...) shown along the progress are averaged ones at those steps. We plot the error of the energy per site in log scale.

```
figure;
plot((1:size(Es,1)).',mean(Es,2)-Eexact,'LineWidth',1);
ylim([1e-18 10]);
set(gca,'YScale','log','LineWidth',1,'FontSize',13);
xlabel('Step');
ylabel('Measured energy - exact energy');
legend({'Averaged'});
grid on;
```



The energy per site well converges to the exact value -2/3, up to double precision $\sim 10^{-16}$!

Then let's compare the tensors. Obtain the left-normalized ket tensor by contracting `Lambda{2}` and `Gamma{1}`. This ket tensor is associated with one site within the unit cell of two sites.

```
A1 = contract(diag(Lambda{2}),2,2,Gamma{1},3,1);
disp(A1);
```

```
(:,:,1) =

   -0.0000    0.5774   -0.0000
   -0.8165   -0.0000   -0.0000


(:,:,2) =

    0.0000    0.0000   -0.8165
    0.0000    0.5774    0.0000
```

```
% check whether left-normalized
T = contract(conj(A1),3,[1 2],A1,3,[1 2]);
disp(T); % should be identity
```

```
    1.0000    0.0000
    0.0000    1.0000
```

While the construction of the ket tensor `A` is of `Lambda*Gamma` type, the tensor `A` is also right-normalized. (In Vidal's convention, `Lambda*Gamma` is for left-normalized and `Gamma*Lambda` is for right-normalized.) It is, indeed, the non-trivial nature of the AKLT state.

```
% check whether right-normalized
T = contract(conj(A1),3,[2 3],A1,3,[2 3]);
disp(T); % should be identity
```

```
    1.0000   -0.0000
   -0.0000    1.0000
```

```
% also for the tensor for another site
A2 = contract(diag(Lambda{1}),2,2,Gamma{2},3,1);
disp(A2);
```

```
(:,:,1) =

   -0.0000   -0.5774   -0.0000
   -0.8165   -0.0000   -0.0000


(:,:,2) =

   -0.0000    0.0000   -0.8165
    0.0000   -0.5774   -0.0000
```

```
% check whether left-normalized
T = contract(conj(A1),3,[1 2],A1,3,[1 2]);
disp(T); % should be identity
```

```
    1.0000    0.0000
    0.0000    1.0000
```

```
% check whether right-normalized
T = contract(conj(A1),3,[2 3],A1,3,[2 3]);
disp(T); % should be identity
```

```
    1.0000   -0.0000
   -0.0000    1.0000
```

There are only two finite singular values which are identical. They are the Schmidt coefficients of the valence bond state which is the singlet of two spin-1/2's.

```
% singular values
disp(Lambda{1}.');
```

```
    0.7071    0.7071
```

```
disp(Lambda{2}.');
```

```
    0.7071    0.7071
```

The tensor of the AKLT state that we used in the tutorial T03b is:

```
% AKLT tensor from the tutorial T02b
AKLT = zeros(2,3,2);
% local spin Sz = -1
```

```
AKLT(2,1,1) = -sqrt(2/3);
% local spin Sz = 0
AKLT(1,2,1) = -1/sqrt(3);
AKLT(2,2,2) = +1/sqrt(3);
% local spin Sz = +1
AKLT(1,3,2) = sqrt(2/3);
disp(AKLT);
```

```
(:,:,1) =

         0   -0.5774         0
   -0.8165         0         0


(:,:,2) =

         0         0    0.8165
         0    0.5774         0
```

Often, the tensors `AKLT` and `A1` (or `A2`) may look different, since there are arbitrary degrees of freedom associated with the unitary transformation on the Hilbert space of the left/right legs (so-called bond space), which is called gauge transformation.

Also, there is even-odd oscillation in our iTEBD result: Two ket tensors within the unit cell are not identical. This even-odd oscillation naturally comes from our consideration of the unit cell of two sites.

Another way to analyze the tensor is to study the transfer operator. The eigenvalues of the transfer operator are independent from the gauge transformation on the bond space. Due to the even-odd oscillation, we will consider the transfer operator for the whole unit cell (i.e., two sites).

```
% contract Gamma*Lambda*Gamma*Lambda
T = contract(Gamma{1},3,3,diag(Lambda{1}),2,1);
T = contract(T,3,3,Gamma{2},3,1);
T = contract(T,4,4,diag(Lambda{2}),2,1);
% transfer operator
W = contract(conj(T),4,[2 3],T,4,[2 3],[1 3 2 4]);
% reshape W as matrix
MW = reshape(W,[size(W,1)*size(W,2) size(W,3)*size(W,4)]);
[VW,DW] = eig(MW);
[DW,ids] = sort(diag(DW),'descend');
VW = VW(:,ids);
```

(Note that `eig` function computes the eigenvectors `V` and eigenvalues `D` of matrix `M` such that `M*V = V*D`, i.e., the eigenvectors are acted onto the right side of `M`.) The eigenvalues of the transfer operator per one site can be obtained as the square roots of the eigenvalues for two sites.

```
% eigenvalue of transfer operator per *one site*
disp(sqrt(DW(:)).');
```

```
    1.0000    0.3333    0.3333    0.3333
```

The largest eigenvalue of the transfer operator per one site should be one, and the eigenvector associated with that eigenvalue should be proportional to the identity matrix, after reshaping.

```
% should be proportional to the identity if tensors are "orthogonal"
VM2 = reshape(VW(:,1),[size(W,1) size(W,2)]);
disp(VM2);
```

```
   -0.7071   -0.0000
   -0.0000   -0.7071
```

Why? This "eigenvector" means that the identity matrix (rank-2) acted onto the transfer operator (rank-4) on the right side is "transferred" to the identity matrix (rank-2) on the left side. It means nothing but the right-normalization of the tensor `Gamma*Lambda*Gamma*Lambda`.

The other eigenvalues, which are three degenerate 1/3's, are identical to what we have learnt from the exercise T04.

Spin-1 Heisenberg model

Now let's move on to more complicated example. Consider the spin-1 Heisenberg model,

$$\hat{H} = \sum_{\ell} \hat{\vec{S}}_{\ell} \cdot \hat{\vec{S}}_{\ell+1}.$$

This Hamiltonian looks more complicated than one for the AKLT model, but the ground state and its energy are more complicated. Nevertheless, they share the same nature: The ground state of this model lies also in the Haldane phase which has symmetry-protected topological order, as the AKLT state does. We use the iTEBD to obtain the ground state and its energy. Here we set larger `Nkeep` and more `Nstep` than in the AKLT example. In accordance with larger `Nstep`, we also decrease the final imaginary time step size `beta_fin`.

```
clear

% iTEBD parameters
Nkeep = 30;
beta_init = 1; % initial imaginary time step size
beta_fin = 1e-6; % final imaginary time step size
Nstep = 3e3; % number of imaginary time steps
betas = beta_init*((beta_fin/beta_init).^linspace(0,1,Nstep));
% discrete imaginary time steps; decays slowly but exponentially

% Local operators
[S,I] = getLocalSpace('Spin',1);

% Heisenberg interaction as two-site gate S*S'
H = contract(S,3,2,permute(conj(S),[3 2 1]),3,2);

% Initialize with random Lambda and Gamma
Lambda = cell(1,2);
Gamma = cell(1,2);
for itn = (1:numel(Lambda))
    Lambda{itn} = rand(Nkeep,1);
    Gamma{itn} = rand(Nkeep,size(I,2),Nkeep);
end

% iTEBD ground state search
```

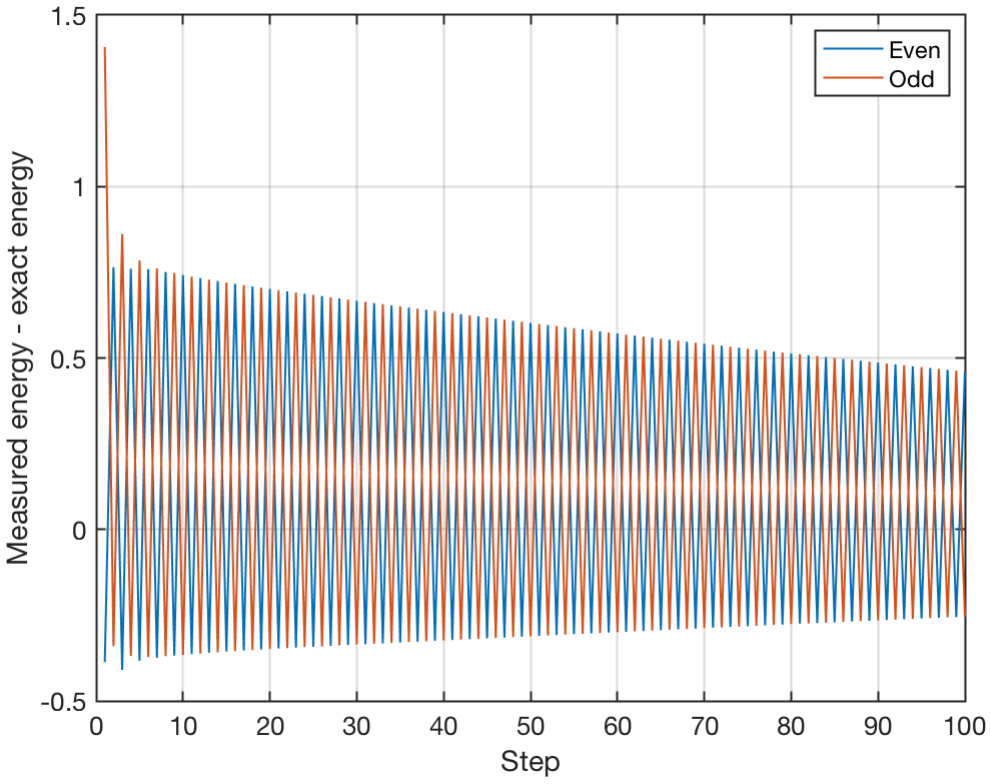
```
[Lambda,Gamma,Es] = iTEBD_GS_Ex(Lambda,Gamma,H,Nkeep,betas);
```

```
iTEBD ground state search: Vidal's method.
  # of sites = 2, Bond dim. Nkeep = 30, # of imag. time steps = 3000
20-06-01 11:38:58 | #500/3000, E = -1.3936303
20-06-01 11:39:00 | #1000/3000, E = -1.4008237
20-06-01 11:39:01 | #1500/3000, E = -1.4014188
20-06-01 11:39:03 | #2000/3000, E = -1.4014771
20-06-01 11:39:05 | #2500/3000, E = -1.4014829
20-06-01 11:39:06 | #3000/3000, E = -1.4014835
Elapsed time: 9.949s, CPU time: 46.14s, Avg # of cores: 4.638
20-06-01 11:39:06 | Memory usage : 1.91GiB
```

As mentioned in the tutorial T03b, the ground-state energy of the spin-1 Heisenberg model is -1.401484039 [S. R. White and D. A. Huse, Phys. Rev. B **48**, 3844 (1993)]. This value is obtained by extrapolating the DMRG result of finite-sized systems to the thermodynamic limit.

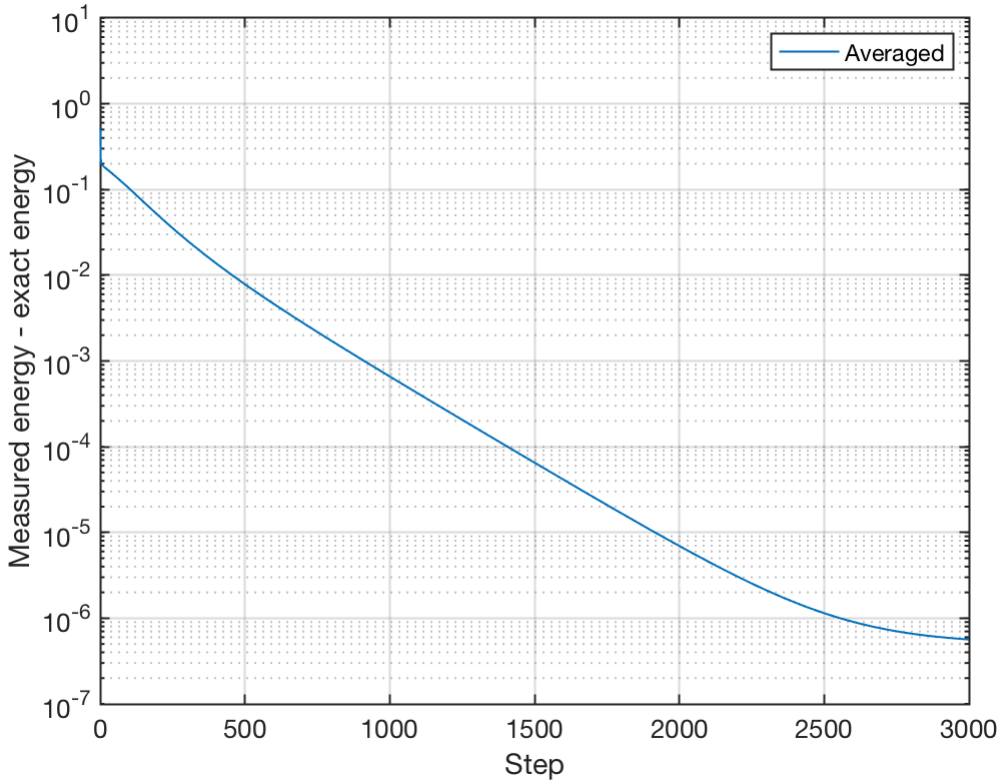
The energies for each bond converge to the exact value, with oscillation.

```
Eexact = -1.401484039;
figure;
plot((1:size(Es,1)).',Es-Eexact,'LineWidth',1);
xlim([0 100]);
set(gca,'LineWidth',1,'FontSize',13);
xlabel('Step');
ylabel('Measured energy - exact energy');
legend({'Even','Odd'});
grid on;
```



And the averaged energy, i.e., the energy per site, converges to the exact value from above; that is, the numerically obtained energy per site is larger than the the exact value.

```
figure;
plot((1:size(Es,1)).',mean(Es,2)-Eexact,'LineWidth',1);
ylim([1e-7 10]);
set(gca,'YScale','log','LineWidth',1,'FontSize',13);
xlabel('Step');
ylabel('Measured energy - exact energy');
legend({'Averaged'});
grid on;
```

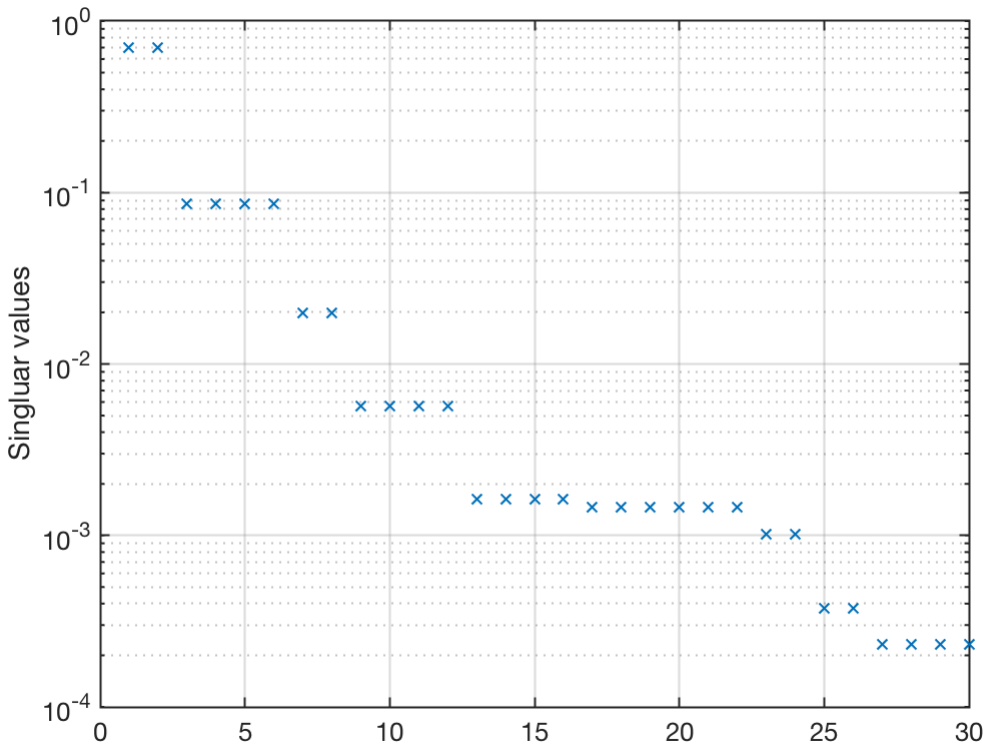


The error of energy can be further decreased by several ways: (i) decrease `beta_min`, (ii) increase `Nstep`, or (iii) increase `Nkeep`. (**Quick exercise**: try this out!)

The result singular values, stored in `Lambda`, are decaying exponentially. The exponential decay corresponds to the nature of the system which is gapped. The logarithms of the singular values, called **entanglement spectrum**, have certain structure. We see that they have even degeneracies: 2, 4, 2, 4, 4, 6, and so on. Such even degeneracies are indicators of topological nature of the system. Refer to [F. Pollmann, A. M. Turner, E. Berg, and M. Oshikawa, Phys. Rev. B **81**, 064439 (2010)] or its arXiv version for details.

```
figure;
plot(Lambda{1},'x','LineWidth',1);
set(gca,'YScale','log','LineWidth',1,'FontSize',13);
ylabel('Singluar values');
```

```
grid on;
```



As mentioned above, the eigenvector of the transfer operator associated with the largest eigenvalue needs to be proportional to the identity matrix after reshaping, since it indicates the left- or right-normalization. How about this solution of the spin-1 Heisenberg chain?

```
% contract Gamma*Lambda*Gamma*Lambda
T = contract(Gamma{1},3,3,diag(Lambda{1}),2,1);
T = contract(T,3,3,Gamma{2},3,1);
T = contract(T,4,4,diag(Lambda{2}),2,1);
% transfer operator
W = contract(conj(T),4,[2 3],T,4,[2 3],[1 3 2 4]);
% reshape W as matrix
MW = reshape(W,[size(W,1)*size(W,2) size(W,3)*size(W,4)]);
[VW,DW] = eig(MW);
[DW,ids] = sort(diag(DW),'descend');
disp(DW(1));
```

```
    1.0000
```

```
VW = VW(:,ids);
% should be proportional to the identity if tensors are "orthogonal"
VM2 = reshape(VW(:,1),[size(W,1) size(W,2)]);
% as the matrix VM2 is large, so let's compare only the differences
% deviation from the identity matrix (up to overall prefactor)
VM2diff = abs(eye(size(VM2))*(trace(VM2)/size(VM2,1)) - VM2);
disp(mean(VM2diff(:))); % average error
```

```
    1.6792e-08
```

The eigenvector of the transfer operator associated with the largest eigenvalue ("dominant eigenvector" of the transfer operator) is proportional to the identity matrix up to numerical error $\sim 10^{-8}$.

Exercise 2: Implement Hastings' method in `iTEBD_GS_Ex.m`

Further complete `iTEBD_GS_Ex.m` by implementing Hastings' method [M. Hastings, J. Math. Phys. **50**, 095207 (2009) or its arXiv version]. The parts enclosed by comments "`TODO - Exercise 2`" indicate the points to be completed.

Exercise 3: Correlation length of the spin-1 Heisenberg model

Compute the spin-spin correlation $\langle \hat{S}^z_{\ell} \hat{S}^z_{\ell+n} \rangle$. For large distance n between the sites on which the spin operators act, the correlation decays exponentially, i.e., $\langle \hat{S}^z_{\ell} \hat{S}^z_{\ell+n} \rangle \sim \exp(-n/\xi)$. Obtain the value of ξ.

(*Hint*: indeed, ξ depends on the maximum bond dimension N_{keep}. In the limit of $N_{\mathrm{keep}} \to \infty$, it will converge to $\xi \simeq 6$ given by the extrapolation of finite-size DMRG calculation as in S. R. White and D. A. Huse, Phys. Rev. B **48**, 3844 (1993)).

Exercise 4: Complete `canon_iMPS_Ex.m` for "orthogonalization"

In the above demonstration of the spin-1 Heisenberg chain, the error for the dominant eigenvector is of order of $O(10^{-8})$. It seems good enough, but we can even push further. We can make tensors to yield numerically exact left- and right-normalized ket tensors, i.e., **up to double precision**!

For this, we need to "orthogonalize" the `Lambda` and `Gamma` tensors, following the method introduced in Sec. II of R. Orus & G. Vidal, Phys. Rev. B **78**, 155117 (2008) (or its arXiv version). There is a function `canon_iMPS_Ex.m` which is zipped together with this document. **Complete the function.** The parts to be completed are enclosed by comments "`TODO - Exercise 4`".

(*Hint*: You need to add only three lines!)