

# NRG: Iterative diagonalization and energy flow

Author: [Seung-Sup Lee](#)

Numerical renormalization group (NRG) is a method for solving quantum impurity systems. [Kenneth G. Wilson](#) has invented NRG to solve the Kondo problem which was not solvable then. The invention of NRG, indeed, is a part of his Nobel prize citation. NRG is an ancestor of all numerical methods having "renormalization group" in their names.

We have already covered the iterative diagonalization and the Lanczos tridiagonalization in earlier tutorials. So one can recycle the solutions of those earlier tutorials to solve the exercises for this tutorial.

## Logarithmic discretization of bath

The starting point of the methods we covered before, such as DMRG and iTEBD, is the Hamiltonian of a system which is already discrete, such as chain. On the other hand, an NRG calculation starts from discretizing the continuous system.

Here we consider an example of quantum impurity system, where the impurity is a spinful fermionic level and the bath consists of non-interacting spinful fermions. The Hamiltonian is given by

$$H = H_{\text{imp}}[\hat{d}_s, \hat{d}_s^\dagger] + H_{\text{hyb}}[\hat{d}_s, \hat{d}_s^\dagger, \hat{c}_{ks}, \hat{c}_{ks}^\dagger] + H_{\text{bath}}[\hat{c}_{ks}, \hat{c}_{ks}^\dagger],$$

$$H_{\text{hyb}} = \sum_k \sum_{s=\uparrow, \downarrow} v_k (\hat{d}_s^\dagger \hat{c}_{ks} + \hat{c}_{ks}^\dagger \hat{d}_s),$$

$$H_{\text{bath}} = \sum_k \sum_{s=\uparrow, \downarrow} \epsilon_k \hat{c}_{ks}^\dagger \hat{c}_{ks},$$

where  $s = \uparrow, \downarrow$  is spin,  $\epsilon_k$  is the energy of bath fermion of momentum  $k$ , and  $v_k$  is the coupling amplitude between the impurity level (to which a particle of spin  $s$  is added by applying  $\hat{d}_s^\dagger$ ) and the bath level of momentum  $k$ . The coupling between the impurity and the bath is characterized by the hybridization function,

$$\Delta(\omega) = \sum_k v_k^2 \delta(\omega - \epsilon_k).$$

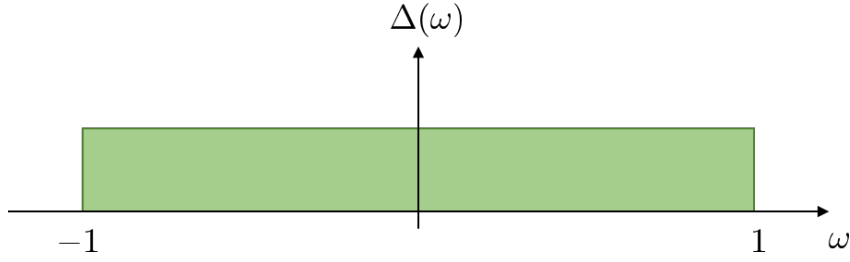
In this demonstration, we choose the Anderson impurity and the "box-shaped" hybridization function,

$$H_{\text{imp}} = U \hat{n}_{d\uparrow} \hat{n}_{d\downarrow} + \epsilon_d (\hat{n}_{d\uparrow} + \hat{n}_{d\downarrow}),$$

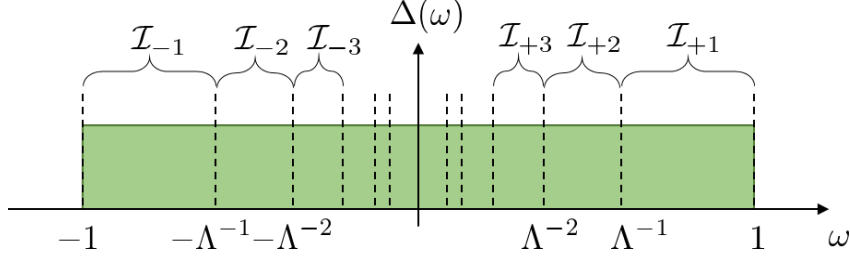
$$\Delta(\omega) = \frac{\Gamma}{\pi} \Theta(D - \omega),$$

where  $\hat{n}_{ds} = \hat{d}_s^\dagger \hat{d}_s$  is a number operator at the impurity,  $U$  is the local Coulomb interaction,  $\epsilon_d$  is the impurity energy level,  $\Gamma$  is hybridization strength, and  $D$  is the half-bandwidth of the bath. This case is called **single-impurity Anderson model (SIAM)**. Throughout this tutorial, we set  $D = 1$  as an energy unit, without loss of generality.

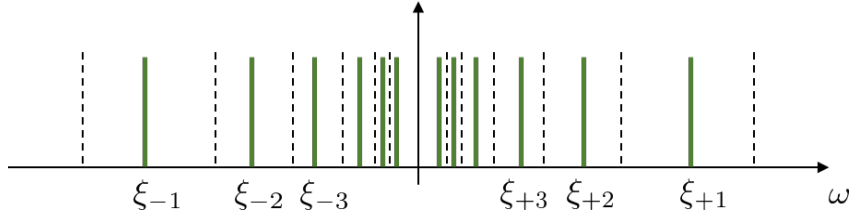
The "box-shaped" hybridization function would look like:



To discretize this, we consider a logarithmic discretization parameter  $\Lambda > 1$ . The logarithmic frequency grid  $\pm\Lambda^{-n}$  splits the whole bandwidth  $\omega \in [-1, 1]$  into the intervals  $\mathcal{I}_{\pm n}$  that are defined by  $\mathcal{I}_{+n} = [\Lambda^{-n}, \Lambda^{-n+1}]$  and  $\mathcal{I}_{-n} = [-\Lambda^{-n+1}, -\Lambda^{-n}]$ .



The part of the bath on each interval  $\mathcal{I}_{\pm n}$  is replaced by a discrete level at  $\omega = \xi_{\pm n}$  that is coupled to the impurity with amplitude  $\gamma_{\pm n}$ .



The discretized Hamiltonian is written by

$$H_{\text{bath}} \mapsto H_{\text{bath}}^{\text{star}} = \sum_{\pm n} \sum_s \xi_{\pm n} \hat{a}_{\pm n, s}^{\dagger} \hat{a}_{\pm n, s},$$

$$H_{\text{hyb}} \mapsto H_{\text{hyb}}^{\text{star}} = \sum_{\pm n} \sum_s \gamma_{\pm n} (\hat{d}_s^{\dagger} \hat{a}_{\pm n, s} + \hat{a}_{\pm n, s}^{\dagger} \hat{d}_s).$$

Each discretized bath level (to which a particle of spin  $s$  is added by applying  $\hat{a}_{\pm n, s}^{\dagger}$ ) represents the part of the bath on an interval  $\mathcal{I}_{\pm n}$ . Therefore the coupling strength of the level should be the same as the integrated hybridization strength over the interval,

$$\gamma_{\pm n}^2 = \int_{\mathcal{I}_{\pm n}} d\omega \Delta(\omega).$$

While the coupling amplitude  $\gamma_{\pm n}$  is unambiguously determined, there are several different ways to determine the discretized level position  $\xi_{\pm n}$ . Here we use the Campo–Oliveira scheme [V. L. Campo and L. N. Oliveira, *Phys. Rev. B* **72**, 104432 (2005)], which defines the level position  $\xi_{\pm n}$  as

$$\xi_{\pm n} = \frac{\int_{\mathcal{J}_{\pm n}} d\omega \Delta(\omega)}{\int_{\mathcal{J}_{\pm n}} d\omega \frac{\Delta(\omega)}{\omega}}.$$

(This way of determining  $\xi_{\pm n}$  is better than Wilson's original way. There is a more advanced scheme, but it is harder to implement. That's why we use the Campo–Oliveira scheme here.)

The discretized Hamiltonian  $H_{\text{imp}} + H_{\text{hyb}}^{\text{star}} + H_{\text{bath}}^{\text{star}}$  is so-called **star-geometry Hamiltonian**. The impurity level and the discretized bath levels (as vertices of a graph), which are coupled via hopping (as edges of the graph), can be depicted as a star graph.

## Lanczos tridiagonalization

The star-geometry Hamiltonian is  $H_{\text{imp}} + H_{\text{hyb}}^{\text{star}} + H_{\text{bath}}^{\text{star}}$  is mapped onto the **Wilson chain Hamiltonian**, via the Lanczos tridiagonalization. The Lanczos method (which is implemented, for example, in `DMRG/eigs_1site.m` in the context of DMRG) first constructs a tridiagonal matrix representation of the input matrix constrained within the Krylov space, and then diagonalizes the tridiagonal matrix to obtain the ground state. The Lanczos tridiagonalization indicates the first part of this process. Here in the mapping onto the Wilson chain, we consider the tridiagonal matrix representation of the quadratic (i.e., single-particle) terms of the bath and the hybridization, without the quartic (i.e., interacting) impurity Hamiltonian.

The Wilson chain Hamiltonian for the SIAM is given by

$$\begin{aligned} H_{\text{SIAM}}^{\text{chain}} &= H_{\text{imp}} + H_{\text{bath}}^{\text{chain}} + H_{\text{hyb}}^{\text{chain}}, \\ H_{\text{imp}} &= U \hat{n}_{d\uparrow} \hat{n}_{d\downarrow} + \epsilon_d (\hat{n}_{d\uparrow} + \hat{n}_{d\downarrow}), \\ H_{\text{bath}}^{\text{chain}} &= \sum_{\ell \geq 0} \sum_{s=\uparrow, \downarrow} t_{\ell} (\hat{f}_{\ell, s}^{\dagger} \hat{f}_{\ell+1, s} + \hat{f}_{\ell+1, s}^{\dagger} \hat{f}_{\ell, s}), \\ H_{\text{hyb}}^{\text{chain}} &= \sum_s t_{\text{imp}} (\hat{d}_s^{\dagger} \hat{f}_{0, s} + \hat{f}_{0, s}^{\dagger} \hat{d}_s). \end{aligned}$$

Note that the impurity Hamiltonian  $H_{\text{imp}}$  is not changed along the logarithmic discretization and the tridiagonalization. The Wilson chain is in principle semi-infinite, but in practice we consider a large but finite length. The length  $N$  sets in the minimum energy scale  $\sim \Lambda^{-N/2}$  to consider.

We will solve this one-dimensional system with the iterative diagonalization.

## Exercise (a): Complete doCLD\_Ex.m for the logarithmic discretization and the Lanczos tridiagonalization

Complete the function doCLD\_Ex.m included in the same .zip file with this document. Its subfunction doCLD\_1side performs the logarithmic discretization. **Complete the part which are enclosed by the comments TODO - Exercise (a)**, within the main function for the Lanczos tridiagonalization.

Once you solve Exercise 1, you can run this demonstration for verifying your implementation.

```
clear

Gamma = 8e-5*pi; % hybridization strength

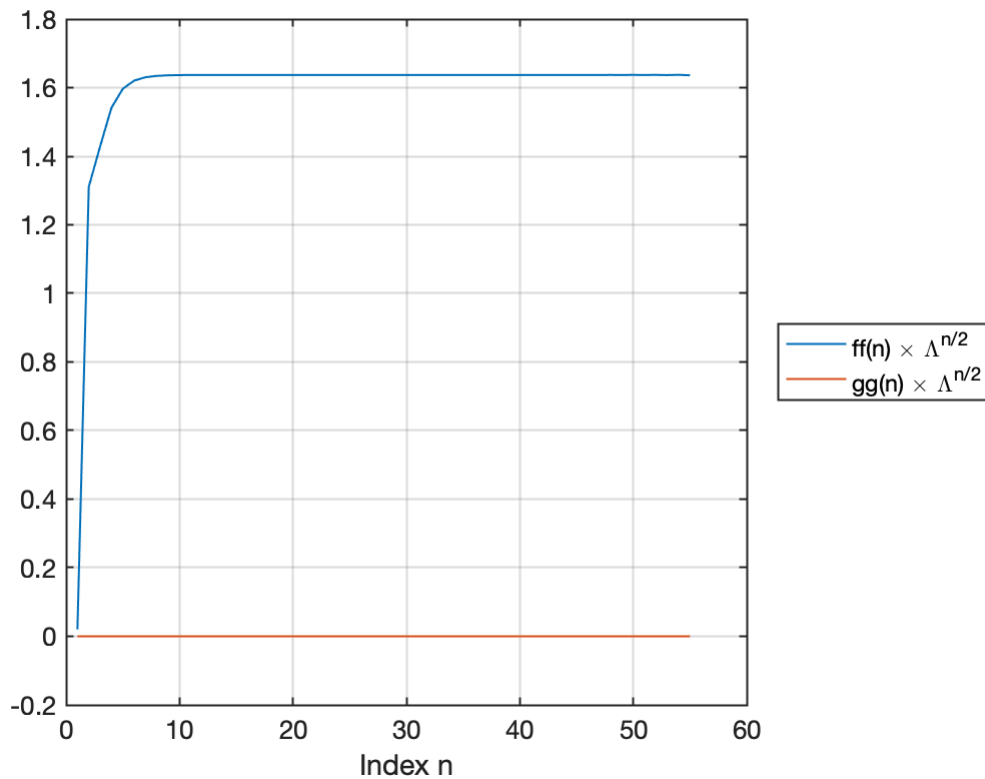
% NRG parameters
Lambda = 2.5; % discretization parameter
N = 55; % length of the Wilson chain

[ff,gg] = doCLD_Ex([-1 1],[1 1]*Gamma/pi,Lambda,N);
```

Since MATLAB indexes the array from 1, we need to shift the indices:  $ff(1)$  corresponds to  $t_{\text{imp}}$  and  $ff(n)$  for  $n > 1$  corresponds to  $t_{\ell=n-2}$ .

The hopping amplitudes  $ff$  decay exponentially, while the on-site energies  $gg$  are zeros up to double precision. To see this, we rescale the values with factors  $\Lambda^{n/2}$ . **[Quick exercise:** Why  $\Lambda^{n/2}$ , not  $\Lambda^n$ ? We started from the discretization grid of  $\pm\Lambda^{-n}$ , so it might look weird to have factor  $1/2$  in the exponent; but of course, there is a good reason.]

```
figure;
plot([ff gg].*(Lambda.^(1:numel(ff)).'/2)), ...
     'LineWidth',1);
set(gca,'FontSize',13,'LineWidth',1);
grid on;
xlabel('Index n');
legend({'ff(n) \times \Lambda^{n/2}', ...
       'gg(n) \times \Lambda^{n/2}'}, ...
       'Location','eastoutside');
```



The first elements of `ff` deviate from the exponential dependence, as we see the deviation from the horizontal line. They come from the specific details of the hybridization function. For example, the square of `ff(1)` is equivalent to the integral of the hybridization,

$$t_{\text{imp}}^2 = \int_{-D}^D d\omega \Delta(\omega) = 2\Gamma D/\pi.$$

```
ff(1)^2 - 2*Gamma/pi % D = 1 as energy unit
```

```
ans = -4.6892e-18
```

## Exercise (b): Complete `NRG_IterDiag_Ex.m` for the iterative diagonalization *a la* NRG

Complete the function `NRG_IterDiag_Ex.m` included in the same `.zip` file with this document. This NRG style of the iterative diagonalization differs from the earlier tutorial in that (i) the Hamiltonian is rescaled by the energy scale factors  $\sim \Lambda^{-n/2}$  and (ii) the energy eigenvalues are shifted so that the lowest energy eigenvalue becomes zero. Other than these, it is the same iterative diagonalization. **Complete the part which are enclosed by the comments `TODO - Exercise (b)`.**

As a demonstration of the completed iterative diagonalization, we apply it to the SIAM.

```
% Hamiltonian parameters
U = 4e-3; % Coulomb interaction at the impurity
```

```

epsd = -U/2; % impurity on-site energy

% NRG parameter
Nkeep = 300;

% Construct local operators
[F,Z,S,I] = getLocalSpace('FermionS');

% particle number operator
NF = cat(3,contract(conj(F(:,1,:)),3,[1 2],F(:,1,:),3,[1 2]), ...
        contract(conj(F(:,2,:)),3,[1 2],F(:,2,:),3,[1 2]));

% Impurity Hamiltonian
H0 = U*(NF(:, :, 1)*NF(:, :, 2)) + epsd*(NF(:, :, 1)+NF(:, :, 2));

% ket tensor for the impurity
A0 = getIdentity(1,2,I,2); % 1 for dummy leg

% iterative diagonalization
Inrg = NRG_IterDiag_Ex (H0,A0,Lambda,ff,F,gg,sum(NF,3),Z,Nkeep);

```

```

21-06-15 16:02:29 | NRG: start
21-06-15 16:02:29 | #00/55 : NK=4/4, EK=0/0
21-06-15 16:02:29 | #01/55 : NK=16/16, EK=0.04894/0.04894
21-06-15 16:02:29 | #02/55 : NK=64/64, EK=3.209/3.209
21-06-15 16:02:29 | #03/55 : NK=256/256, EK=6.225/6.225
21-06-15 16:02:29 | #04/55 : NK=328/1024, EK=5.002/13.02
21-06-15 16:02:29 | #05/55 : NK=304/1312, EK=4.685/9.131
21-06-15 16:02:30 | #06/55 : NK=328/1216, EK=5.012/10.57
21-06-15 16:02:30 | #07/55 : NK=304/1312, EK=4.745/9.188
21-06-15 16:02:30 | #08/55 : NK=328/1216, EK=5.041/10.61
21-06-15 16:02:30 | #09/55 : NK=304/1312, EK=4.855/9.274
21-06-15 16:02:31 | #10/55 : NK=328/1216, EK=5.113/10.71
21-06-15 16:02:31 | #11/55 : NK=304/1312, EK=5.052/9.439
21-06-15 16:02:31 | #12/55 : NK=328/1216, EK=5.269/10.93
21-06-15 16:02:31 | #13/55 : NK=300/1312, EK=5.31/9.79
21-06-15 16:02:32 | #14/55 : NK=328/1200, EK=5.558/11.23
21-06-15 16:02:32 | #15/55 : NK=320/1312, EK=5.716/10.55
21-06-15 16:02:32 | #16/55 : NK=306/1280, EK=5.868/11.25
21-06-15 16:02:32 | #17/55 : NK=329/1224, EK=5.991/10.94
21-06-15 16:02:33 | #18/55 : NK=320/1316, EK=6.606/12.3
21-06-15 16:02:33 | #19/55 : NK=328/1280, EK=6.669/12.2
21-06-15 16:02:33 | #20/55 : NK=302/1312, EK=6.676/13.39
21-06-15 16:02:33 | #21/55 : NK=328/1208, EK=6.772/12.35
21-06-15 16:02:34 | #22/55 : NK=310/1312, EK=6.821/13.51
21-06-15 16:02:34 | #23/55 : NK=328/1240, EK=6.808/12.73
21-06-15 16:02:34 | #24/55 : NK=310/1312, EK=6.814/13.48
21-06-15 16:02:34 | #25/55 : NK=328/1240, EK=6.858/12.74
21-06-15 16:02:35 | #26/55 : NK=322/1312, EK=7.004/13.44
21-06-15 16:02:35 | #27/55 : NK=301/1288, EK=6.748/12.99
21-06-15 16:02:35 | #28/55 : NK=304/1204, EK=6.998/12.94
21-06-15 16:02:35 | #29/55 : NK=317/1216, EK=6.983/13.11
21-06-15 16:02:36 | #30/55 : NK=300/1268, EK=7.052/13.2
21-06-15 16:02:36 | #31/55 : NK=310/1200, EK=7.426/13.27
21-06-15 16:02:36 | #32/55 : NK=304/1240, EK=7.341/13.81
21-06-15 16:02:36 | #33/55 : NK=329/1216, EK=8.095/14.03
21-06-15 16:02:36 | #34/55 : NK=304/1316, EK=7.856/14.51
21-06-15 16:02:37 | #35/55 : NK=300/1216, EK=8.507/15.27
21-06-15 16:02:37 | #36/55 : NK=318/1200, EK=8.269/15.02
21-06-15 16:02:37 | #37/55 : NK=308/1272, EK=8.824/16.17
21-06-15 16:02:37 | #38/55 : NK=328/1232, EK=8.485/15.5

```

```

21-06-15 16:02:38 | #39/55 : NK=300/1312, EK=9.06/16.66
21-06-15 16:02:38 | #40/55 : NK=328/1200, EK=8.505/15.77
21-06-15 16:02:38 | #41/55 : NK=300/1312, EK=9.168/16.74
21-06-15 16:02:38 | #42/55 : NK=328/1200, EK=8.514/15.91
21-06-15 16:02:39 | #43/55 : NK=300/1312, EK=9.213/16.77
21-06-15 16:02:39 | #44/55 : NK=328/1200, EK=8.517/15.96
21-06-15 16:02:39 | #45/55 : NK=300/1312, EK=9.231/16.78
21-06-15 16:02:39 | #46/55 : NK=328/1200, EK=8.518/15.98
21-06-15 16:02:40 | #47/55 : NK=300/1312, EK=9.238/16.79
21-06-15 16:02:40 | #48/55 : NK=328/1200, EK=8.519/15.99
21-06-15 16:02:40 | #49/55 : NK=300/1312, EK=9.241/16.79
21-06-15 16:02:40 | #50/55 : NK=328/1200, EK=8.519/16
21-06-15 16:02:41 | #51/55 : NK=300/1312, EK=9.242/16.79
21-06-15 16:02:41 | #52/55 : NK=328/1200, EK=8.519/16
21-06-15 16:02:41 | #53/55 : NK=300/1312, EK=9.242/16.79
21-06-15 16:02:41 | #54/55 : NK=328/1200, EK=8.519/16
21-06-15 16:02:42 | #55/55 : NK=0/1312, EK=0/16.78
21-06-15 16:02:42 | Memory usage : 3.18GiB
Elapsed time: 12.83s, CPU time: 86.83s, Avg # of cores: 6.769

```

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

## Energy flow diagram

NRG provides a method to analyze the spectrum obtained along the iterative diagonalization. Let's plot the lowest-lying (many-body) energy levels. We plot the results from even iterations and those from odd iterations separately.

```

% Energy flow diagram
Eshow = 3; % energy window to show (from 0 to Eshow)

% since we start from A0; consider the step for H0 as 0, i.e., even
Even = Inrg.EK(1:2:end);
Even = cellfun(@(x) x(x <= Eshow), Even, 'UniformOutput', 0);
maxEven = max(cellfun('prodoysize', Even));
Even = cellfun(@(x) [x; nan(maxEven-numel(x), 1)], Even, 'UniformOutput', 0);
Even = cell2mat(Even).';

Eodd = Inrg.EK(2:2:end);
Eodd = cellfun(@(x) x(x <= Eshow), Eodd, 'UniformOutput', 0);
maxEodd = max(cellfun('prodoysize', Eodd));
Eodd = cellfun(@(x) [x; nan(maxEodd-numel(x), 1)], Eodd, 'UniformOutput', 0);
Eodd = cell2mat(Eodd).';

```

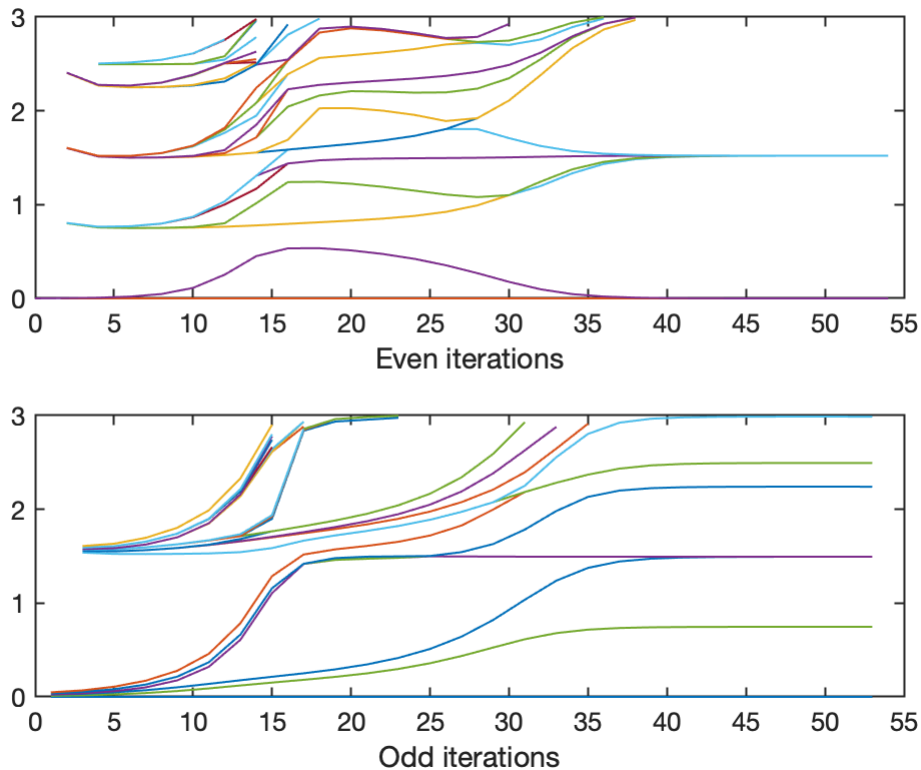
(Note that `cellfun('prodoysize', ...)` is undocumented functionality, which gives the same result as `cellfun(@numel, ...)` but much faster. Refer to [this webpage](#).)

```

figure;
% upper panel
subplot(2,1,1);
plot((1:2:numel(Inrg.EK))-1, Even, 'LineWidth', 1);
xlabel('Even iterations');
xlim([0 numel(Inrg.EK)-1]);
ylim([0, Eshow]);
set(gca, 'LineWidth', 1, 'FontSize', 13);

```

```
% lower panel
subplot(2,1,2);
plot((2:2:numel(Inrg.EK))-1,Eodd,'LineWidth',1);
xlabel('Odd iterations');
xlim([0 numel(Inrg.EK)-1]);
ylim([0, Eshow]);
set(gca,'LineWidth',1,'FontSize',13);
```



(Note that `cellfun('prodoFSIZE', ...)` is undocumented functionality, which gives the same result as `cellfun(@numel, ...)` but much faster. Refer to [this webpage](#).)

These plots are called **energy flow diagram** or **finite-size energy spectra**. The name "flow" literally comes from that the lines flow from one regime to the other. There are three regions (iterations 1–10; 17–25; 35–55) connected via two crossovers. These regions correspond to different **fixed points: free orbital, local moment, and strong coupling**. The strong-coupling fixed-point regime exhibits prominent plateau of the energy levels.

## Exercise (c): Reproduce lowest-lying energies in the strong-coupling regime by fixed-point Hamiltonians

Let's consider iteration 53 (counting from iteration 0 for the impurity only) in the strong-coupling fixed-point regime.. Their lowest-lying energies, including all degenerate levels, are:

```
fprintf([sprintf('%.4f, ',Inrg.EK{54}(1:5).'),'\n', ...
        sprintf('%.4f, ',Inrg.EK{54}(6:11).'),'...\n']);
```

```
0.0000, 0.7471, 0.7471, 0.7471, 0.7471,
```



1.4941, 1.4941, 1.4941, 1.4941, 1.4941, 1.4941, ...

We see 1-fold, 4-fold, and 6-fold degeneracies. On the other hand, the energy levels at the next iteration 54 have more degeneracies:

```
fprintf([sprintf('% .4f, ', Inrg.EK{55}(1:4) .'), '\n', ...
    sprintf('% .4f, ', Inrg.EK{55}(5:12) .'), '\n', ...
    sprintf('% .4f, ', Inrg.EK{55}(13:20) .'), '... \n']);
```

```
0.0000, 0.0000, 0.0000, 0.0000,
1.5213, 1.5213, 1.5213, 1.5213, 1.5213, 1.5213, 1.5213, 1.5213,
1.5213, 1.5213, 1.5213, 1.5213, 1.5213, 1.5213, 1.5213, 1.5213, ...
```

There are 4-fold and 16-fold degeneracies, up to numerical noise of  $O(10^{-6})$ . **Reproduce these (many-body) energy values by considering strong-coupling fixed-point Hamiltonians.**

(*Hint*. The fixed-point Hamiltonians are single-particle Hamiltonians, effectively!)

## Exercise (d): Single-impurity Kondo model

We can derive the Wilson chain Hamiltonian  $H_{\text{SIKM}}^{\text{chain}}$  for the single-impurity Kondo model (SIKM), from the chain Hamiltonian of the SIAM  $H_{\text{SIAM}}^{\text{chain}}$  shown above. By applying the Schrieffer-Wolff transformation to the impurity site (on which  $\hat{d}_s$  acts) and the first bath site (on which  $\hat{f}_{0,s}$  acts), we obtain

$$H_{\text{SIKM}}^{\text{chain}} = H_{\text{exc}} + H_{\text{bath}},$$

$$H_{\text{exc}} = 2J \hat{\vec{S}}_d \cdot \hat{\vec{S}}_0,$$

$$H_{\text{bath}} = \sum_{\ell \geq 0} \sum_{s=\uparrow, \downarrow} t_\ell (\hat{f}_{\ell,s}^\dagger \hat{f}_{\ell+1,s} + \hat{f}_{\ell+1,s}^\dagger \hat{f}_{\ell,s}).$$

Here  $\hat{\vec{S}}_d$  is the spin operator acting on the impurity site,

$$\hat{\vec{S}}_0 = \sum_{s,s'} \hat{f}_{0,s}^\dagger \frac{\vec{\sigma}_{s,s'}}{2} \hat{f}_{0,s'}$$

is the spin operator acting on the first bath site,  $\vec{\sigma} = [\sigma_x, \sigma_y, \sigma_z]$  is the vector of Pauli matrices, and

$$J = t_0^2 \left( \frac{1}{-\epsilon_d} + \frac{1}{U + \epsilon_d} \right)$$

is the Kondo coupling strength. For particle-hole symmetric case  $\epsilon_d = -U/2$  which we considered above, it becomes

$$J = \frac{4t_0^2}{U} = \frac{8\Gamma D}{\pi U}.$$

The bath term  $H_{\text{bath}}$  is the same as in the SIAM case. Note that the impurity site in the SIKM has dimension 2, while that in the SIAM has 4; the doubly occupied and the empty states are "integrated out" by the Schrieffer-Wolff transformation. Refer to [J. R. Schrieffer and P. A. Wolff, Phys. Rev. \*\*149\*\*, 491 \(1966\)](#) for the details of the Schrieffer-Wolff transformation.

**Perform the iterative diagonalization of this chain Hamiltonian for the SIKM**, with the value of  $J$  corresponding to the choice of parameters  $U$ ,  $\epsilon_d$ , and  $\Gamma$  above. (Again  $D = 1$  is the energy unit.) Compare the energy flow diagram with the SIAM result.

(*Hint*: You can do it by changing `H0`, `A0`, and `ff` only, with using the same function `NRG_IterDiag_Ex!`)