

GILT and FET

Author: [Jheng-Wei Li](#)

In this tutorial, we study two gauge-fixing schemes, namely (i) graph-independent local truncation (GILT) and (ii) full environment truncation (FET) to improve over the tensor renormalization group (TRG) approach. Either GILT or FET is applied before a TRG coarse-graining step in hope to get rid of the short-range correlations that TRG is unable to remove properly. The combined algorithm is called GILT-TNR or FET-TRG, respectively.

We benchmark the accuracy of GILT-TNR and FET-TRG against standard TRG with the classical Ising model on a square-lattice at the critical temperature, i.e., $T_c = 2/\log(1 + \sqrt{2})$.

TRG with the classical Ising model

First, we use the standard TRG approach to coarse grain the 2D classical Ising model at the critical temperature. With the bond dimension $D = 24$, we run 32 TRG steps and compute the free energy per site at each step to compare with the exact value at the thermodynamic limit. The rank-4 bulk tensor (aT) at step S approximates the partition function Z for 2^S spins up to some normalization factors. [The index ordering for tensor (aT) is left-up-down-right]

```
%% TRG
clear all;
NITER = 32; % number of iterations
DKEEP = 24; % bond dimension
BETA = log(1+sqrt(2))/2; % the inverse temperature
fexact = EXACT(BETA); % get exact solution

logN = 0;
aT = INIT_ISING(BETA); % initialize on-site tensor
for iter = 1 : NITER
    titer = tic;
    % TRG and rescale
    aT = TRG_1STEP(aT, aT, DKEEP);
    scal = max(abs(aT(:)));
    aT = aT/scal;
    logN = logN + log(scal)/2^iter;

    % compute free energy
    fene = reshape(aT, [size(aT,1)*size(aT,2), size(aT,3)*size(aT,4)]);
    fene = trace(fene);
    fene = (-log(fene)/2^NITER - logN)/BETA;
    fprintf(' *** Iter: %2d, Energy = %16.12f, Err = %6.3e >>> %6.2f sec\n', ...
        iter, fene, abs((fene-fexact)/fexact), toc(titer));
end
```

```
*** Iter: 1, Energy = -2.032917137522, Err = 3.637e-02 >>> 0.03 sec
*** Iter: 2, Energy = -2.050631594270, Err = 2.798e-02 >>> 0.00 sec
*** Iter: 3, Energy = -2.077421029758, Err = 1.528e-02 >>> 0.01 sec
*** Iter: 4, Energy = -2.090184180870, Err = 9.228e-03 >>> 0.00 sec
*** Iter: 5, Energy = -2.098604432864, Err = 5.236e-03 >>> 0.02 sec
```

```

*** Iter: 6, Energy = -2.103384757705, Err = 2.970e-03 >>> 0.04 sec
*** Iter: 7, Energy = -2.106171795800, Err = 1.649e-03 >>> 0.16 sec
*** Iter: 8, Energy = -2.107735323056, Err = 9.081e-04 >>> 0.17 sec
*** Iter: 9, Energy = -2.108607521252, Err = 4.947e-04 >>> 0.15 sec
*** Iter: 10, Energy = -2.109086425137, Err = 2.677e-04 >>> 0.13 sec
*** Iter: 11, Energy = -2.109347296866, Err = 1.440e-04 >>> 0.13 sec
*** Iter: 12, Energy = -2.109488142541, Err = 7.726e-05 >>> 0.19 sec
*** Iter: 13, Energy = -2.109563686600, Err = 4.146e-05 >>> 0.17 sec
*** Iter: 14, Energy = -2.109603953469, Err = 2.237e-05 >>> 0.17 sec
*** Iter: 15, Energy = -2.109625298799, Err = 1.225e-05 >>> 0.17 sec
*** Iter: 16, Energy = -2.109636552576, Err = 6.917e-06 >>> 0.15 sec
*** Iter: 17, Energy = -2.109642455575, Err = 4.119e-06 >>> 0.18 sec
*** Iter: 18, Energy = -2.109645534273, Err = 2.659e-06 >>> 0.19 sec
*** Iter: 19, Energy = -2.109647129906, Err = 1.903e-06 >>> 0.14 sec
*** Iter: 20, Energy = -2.109647952552, Err = 1.513e-06 >>> 0.13 sec
*** Iter: 21, Energy = -2.109648373323, Err = 1.314e-06 >>> 0.14 sec
*** Iter: 22, Energy = -2.109648586145, Err = 1.213e-06 >>> 0.13 sec
*** Iter: 23, Energy = -2.109648692941, Err = 1.162e-06 >>> 0.15 sec
*** Iter: 24, Energy = -2.109648744564, Err = 1.138e-06 >>> 0.19 sec
*** Iter: 25, Energy = -2.109648769563, Err = 1.126e-06 >>> 0.13 sec
*** Iter: 26, Energy = -2.109648781858, Err = 1.120e-06 >>> 0.18 sec
*** Iter: 27, Energy = -2.109648789377, Err = 1.116e-06 >>> 0.17 sec
*** Iter: 28, Energy = -2.109648791947, Err = 1.115e-06 >>> 0.15 sec
*** Iter: 29, Energy = -2.109648792402, Err = 1.115e-06 >>> 0.14 sec
*** Iter: 30, Energy = -2.109648793851, Err = 1.114e-06 >>> 0.17 sec
*** Iter: 31, Energy = -2.109648793733, Err = 1.114e-06 >>> 0.11 sec
*** Iter: 32, Energy = -2.109648794459, Err = 1.114e-06 >>> 0.15 sec

```

Exercise (a): Complete GILT_Ex.m

Complete the function GILT_Ex.m which is zipped together with this tutorial material.

The main function GILT corresponds to Eq. (30) in [Phys. Rev. B 97, 045111 \(2018\)](#), which truncates the upper-bond of a given plaquette.

The sub-function OPT_RP is the core of GILT. This function finds the matrix R' [see Eq. (28)] given the environmental spectrum S and the singular vectors U. This sub-function is written more or less following the original implementation [<https://github.com/GILT-TNR/GILT-TNR/blob/master/GiltTNR2D.py>]. I encourage all of you to take a look at the original source code on the Github, because the code is very well written and it's good to take a moment to appreciate it.

Complete the parts which are enclosed by the comments TODO - Exercise 1. Once you complete the function, you can follow the demonstration below.

GILT-TNR with the classical Ising model

```

%% GILT-TNR
clear all;
NITER = 32; % number of iterations
DKEEP = 24; % bond dimension
BETA = log(1+sqrt(2))/2; % the inverse temperature
GILT_EPS = 8E-7; % \epsilon defined in Eq.(31)
GILT_CONV = 1E-2; % the convergence criteria for GILT iterations
fexact = EXACT(BETA); % get exact solution

logN = 0;
aT = INIT_ISING(BETA); % initialize on-site tensor

```

```

for iter = 1 : NITER
    titer = tic;
    a1 = aT;
    a2 = aT;

    %% GILT left bond
    % rotate 90 degree counter-clockwise
    a1 = permute(a1, [3,1,4,2]);
    a2 = permute(a2, [3,1,4,2]);
    [a2, a1] = GILT_ex(a2, a1, GILT_EPS, GILT_CONV);
    % rotate back
    a1 = permute(a1, [2,4,1,3]);
    a2 = permute(a2, [2,4,1,3]);

    %% GILT up bond
    [a1, a2] = GILT_ex(a1, a2, GILT_EPS, GILT_CONV);

    %% GILT bottom bond
    % rotate 180 degree counter-clockwise
    a1 = permute(a1, [4,3,2,1]);
    a2 = permute(a2, [4,3,2,1]);
    [a1, a2] = GILT(a1, a2, GILT_EPS, GILT_CONV);
    % rotate back
    a1 = permute(a1, [4,3,2,1]);
    a2 = permute(a2, [4,3,2,1]);

    %% GILT right bond
    % rotate 270 degree counter-clockwise
    a1 = permute(a1, [2,4,1,3]);
    a2 = permute(a2, [2,4,1,3]);
    [a2, a1] = GILT(a2, a1, GILT_EPS, GILT_CONV);
    % rotate back
    a1 = permute(a1, [3,1,4,2]);
    a2 = permute(a2, [3,1,4,2]);

    % TRG and rescale
    aT = TRG_1STEP(a1, a2, DKEEP);
    scal = max(abs(aT(:)));
    aT = aT/scal;
    logN = logN + log(scal)/2^iter;

    % compute free energy
    fene = reshape(aT, [size(aT,1)*size(aT,2), size(aT,3)*size(aT,4)]);
    fene = trace(fene);
    fene = (-log(fene)/2^NITER - logN)/BETA;
    fprintf('    *** Iter: %2d, Energy = %16.12f, Err = %6.3e >>> %6.2f sec\n',...
        iter, fene, abs((fene-fexact)/fexact), toc(titer));
end

```

```

*** Iter: 1, Energy = -2.032917137516, Err = 3.637e-02 >>> 0.04 sec
*** Iter: 2, Energy = -2.050631594196, Err = 2.798e-02 >>> 0.00 sec
*** Iter: 3, Energy = -2.077421031494, Err = 1.528e-02 >>> 0.01 sec
*** Iter: 4, Energy = -2.090184182223, Err = 9.228e-03 >>> 0.00 sec
*** Iter: 5, Energy = -2.098620428780, Err = 5.229e-03 >>> 0.42 sec
*** Iter: 6, Energy = -2.103416648107, Err = 2.955e-03 >>> 0.71 sec

```

```

*** Iter: 7, Energy = -2.106214803151, Err = 1.629e-03 >>> 1.51 sec
*** Iter: 8, Energy = -2.107761350107, Err = 8.958e-04 >>> 1.31 sec
*** Iter: 9, Energy = -2.108623200235, Err = 4.873e-04 >>> 2.08 sec
*** Iter: 10, Energy = -2.109095745297, Err = 2.633e-04 >>> 1.59 sec
*** Iter: 11, Energy = -2.109355558097, Err = 1.401e-04 >>> 2.82 sec
*** Iter: 12, Energy = -2.109493436254, Err = 7.476e-05 >>> 2.42 sec
*** Iter: 13, Energy = -2.109566990032, Err = 3.989e-05 >>> 3.46 sec
*** Iter: 14, Energy = -2.109606073727, Err = 2.136e-05 >>> 4.63 sec
*** Iter: 15, Energy = -2.109627202026, Err = 1.135e-05 >>> 3.03 sec
*** Iter: 16, Energy = -2.109638284626, Err = 6.096e-06 >>> 4.33 sec
*** Iter: 17, Energy = -2.109644351865, Err = 3.220e-06 >>> 5.61 sec
*** Iter: 18, Energy = -2.109647409472, Err = 1.770e-06 >>> 6.26 sec
*** Iter: 19, Energy = -2.109649046446, Err = 9.946e-07 >>> 8.19 sec
*** Iter: 20, Energy = -2.109649905699, Err = 5.873e-07 >>> 6.35 sec
*** Iter: 21, Energy = -2.109650371137, Err = 3.666e-07 >>> 9.22 sec
*** Iter: 22, Energy = -2.109650609850, Err = 2.535e-07 >>> 9.77 sec
*** Iter: 23, Energy = -2.109650739140, Err = 1.922e-07 >>> 13.17 sec
*** Iter: 24, Energy = -2.109650805365, Err = 1.608e-07 >>> 10.20 sec
*** Iter: 25, Energy = -2.109650840455, Err = 1.442e-07 >>> 12.02 sec
*** Iter: 26, Energy = -2.109650858192, Err = 1.358e-07 >>> 21.76 sec
*** Iter: 27, Energy = -2.109650869134, Err = 1.306e-07 >>> 10.46 sec
*** Iter: 28, Energy = -2.109650873891, Err = 1.283e-07 >>> 9.80 sec
*** Iter: 29, Energy = -2.109650875695, Err = 1.275e-07 >>> 7.87 sec
*** Iter: 30, Energy = -2.109650876393, Err = 1.271e-07 >>> 7.39 sec
*** Iter: 31, Energy = -2.109650877922, Err = 1.264e-07 >>> 4.51 sec
*** Iter: 32, Energy = -2.109650878191, Err = 1.263e-07 >>> 4.84 sec

```

Next, to understand what's going on inside the iterative GILT process in the sub-function `OPT_RP`, we run one more GILT step to record the singular values of `R'` at each iteration and plot them. We can see that the tail of singular values is being truncated successively at each iteration.

```

clearvars -global sPlot;
global sPlot;
sPlot = [];
GILT(aT, aT, GILT_EPS, GILT_CONV);
figure;
hold all;
legendText = cell(1,1);
nS = size(sPlot);
clrs = flipud(parula(nS(1)));
for iter = 5 : 5 : nS(1)
    %% X-axis: the singular value index. Y-axis:
    % the corresponding singular value is plotted at every 5 iteration.
    hplot = plot(1:nS(2), sPlot(iter,1:nS(2)), 'Color', clrs(iter,:), ...
        'LineWidth', 1, 'Marker', 'x');
    legendText{iter/5} = sprintf('Iter %d', iter);
    legend(hplot, 'Location', 'west');
end
set(gca, 'YScale', 'log');
set(gca, 'LineWidth', 1, 'FontSize', 15);
grid on;
legend(legendText);

```



```

%% FET lbond
a1 = permute(a1, [3,1,4,2]);
a2 = permute(a2, [3,1,4,2]);
[a2, a1] = FET(a2, a1, DKEEP, FET_CONV);
a1 = permute(a1, [2,4,1,3]);
a2 = permute(a2, [2,4,1,3]);

%% FET ubond
[a1, a2] = FET_ex(a1, a2, DKEEP, FET_CONV);

%% FET dbond
a1 = permute(a1, [4,3,2,1]);
a2 = permute(a2, [4,3,2,1]);
[a1, a2] = FET(a1, a2, DKEEP, FET_CONV);
a1 = permute(a1, [4,3,2,1]);
a2 = permute(a2, [4,3,2,1]);

%% FET rbond
a1 = permute(a1, [2,4,1,3]);
a2 = permute(a2, [2,4,1,3]);
[a2, a1] = FET_ex(a2, a1, DKEEP, FET_CONV);
a1 = permute(a1, [3,1,4,2]);
a2 = permute(a2, [3,1,4,2]);

% TRG and rescale
%% In principle, we should not truncate here,
%% but we enforce some truncation here with D=2*DKEEP to speed it up.
aT = TRG_1STEP(a1, a2, 2*DKEEP);
scal = max(abs(aT(:)));
aT = aT/scal;
logN = logN + log(scal)/2^iter;

% free energy
fene = reshape(aT, [size(aT,1)*size(aT,2), size(aT,3)*size(aT,4)]);
fene = trace(fene);
fene = (-log(fene)/2^NITER - logN)/BETA;
fprintf('    *** Iter: %2d, Energy = %16.12f, Err = %6.3e >>> %6.2f sec\n', ...
        iter, fene, abs((fene-fexact)/fexact), toc(titer));
end

```

```

*** Iter: 1, Energy = -2.032917137522, Err = 3.637e-02 >>> 0.04 sec
*** Iter: 2, Energy = -2.050631594270, Err = 2.798e-02 >>> 0.02 sec
*** Iter: 3, Energy = -2.077421029987, Err = 1.528e-02 >>> 0.01 sec
*** Iter: 4, Energy = -2.090184181239, Err = 9.228e-03 >>> 0.01 sec
*** Iter: 5, Energy = -2.098604329292, Err = 5.236e-03 >>> 0.09 sec
*** Iter: 6, Energy = -2.103384400399, Err = 2.971e-03 >>> 0.18 sec
*** Iter: 7, Energy = -2.106176465856, Err = 1.647e-03 >>> 14.39 sec
*** Iter: 8, Energy = -2.107743253294, Err = 9.044e-04 >>> 16.18 sec
*** Iter: 9, Energy = -2.108619655643, Err = 4.889e-04 >>> 19.26 sec
*** Iter: 10, Energy = -2.109098549008, Err = 2.619e-04 >>> 15.14 sec
*** Iter: 11, Energy = -2.109357279264, Err = 1.393e-04 >>> 14.64 sec
*** Iter: 12, Energy = -2.109494165554, Err = 7.441e-05 >>> 14.25 sec
*** Iter: 13, Energy = -2.109567704508, Err = 3.955e-05 >>> 14.36 sec
*** Iter: 14, Energy = -2.109607409132, Err = 2.073e-05 >>> 14.56 sec
*** Iter: 15, Energy = -2.109628336086, Err = 1.081e-05 >>> 14.71 sec
*** Iter: 16, Energy = -2.109639005904, Err = 5.754e-06 >>> 15.56 sec

```

```

*** Iter: 17, Energy = -2.109644247311, Err = 3.269e-06 >>> 16.64 sec
*** Iter: 18, Energy = -2.109647888037, Err = 1.544e-06 >>> 21.15 sec
*** Iter: 19, Energy = -2.109647968742, Err = 1.505e-06 >>> 19.76 sec
*** Iter: 20, Energy = -2.109649913161, Err = 5.837e-07 >>> 23.36 sec
*** Iter: 21, Energy = -2.109650354994, Err = 3.743e-07 >>> 22.14 sec
*** Iter: 22, Energy = -2.109650840875, Err = 1.440e-07 >>> 20.38 sec
*** Iter: 23, Energy = -2.109651035825, Err = 5.156e-08 >>> 26.19 sec
*** Iter: 24, Energy = -2.109651074155, Err = 3.340e-08 >>> 25.09 sec
*** Iter: 25, Energy = -2.109651104023, Err = 1.924e-08 >>> 23.48 sec
*** Iter: 26, Energy = -2.109651110699, Err = 1.607e-08 >>> 21.64 sec
*** Iter: 27, Energy = -2.109651121715, Err = 1.085e-08 >>> 22.70 sec
*** Iter: 28, Energy = -2.109651125046, Err = 9.273e-09 >>> 23.87 sec
*** Iter: 29, Energy = -2.109651125542, Err = 9.037e-09 >>> 25.58 sec
*** Iter: 30, Energy = -2.109651128327, Err = 7.718e-09 >>> 23.77 sec
*** Iter: 31, Energy = -2.109651128147, Err = 7.803e-09 >>> 21.28 sec
*** Iter: 32, Energy = -2.109651126656, Err = 8.546e-09 >>> 20.84 sec

```

Conclusion

We can see that GILT and FET methods indeed improve the accuracy of standard TRG with no more than 100 lines of extra code! With bond dimension $D = 24$, the relative error in the free energy per site using standard TRG is $\delta f \sim 10^{-6}$, while GILT-TNR reduces the relative error down to $\sim 10^{-7}$, and FET-TRG reduces the relative error down to $\sim 10^{-8}$.

Besides the free energy, we would also like to see whether such gauge-fixing approaches improve other quantities. In Exercise 3, we will try to detect possible symmetry breaking order from the bulk tensors during coarse-graining steps at and away from T_c . In Exercise 4, we will extract some conformal numbers, namely the central charge and scaling dimensions from a scale-invariant tensor and compare them with exact values from the CFT prediction.

Exercis (c): Detect phase transitions

After enough coarse-graining, one would expect that the bulk tensor is going to reach some convergence, and the fixed-point tensor should characterize the phase of the system.

In this paper [Z.-C. Gu and X.-G. Wen, Phys. [Rev. B 80, 155131 \(2009\)](#)], Gu and Wen proposed a method to detect possible symmetry breaking order and phase transitions. Given the bulk tensor aT , they define the following quantity X :

$$X = \frac{(\text{Tr}(aT))^2}{\text{Tr}(aT \otimes aT)}$$

The value of X counts the number of degeneracy. If the system is at a trivial phase, the value is simply 1. If the system is symmetry broken, the value gives the number of degeneracy of the symmetry broken phase. Therefore, for the Ising model, X is 2(1) below (above) the critical temperature. At the critical temperature, one will observe a sharp jump from 1 to 2.

(a) Can TRG method confirms the above prediction away from the critical temperature, say $T_c \pm 0.1$?

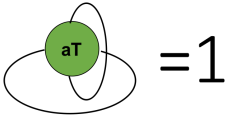
Also, how does X change during coarse-graining? Does X reach the convergence ?

(b) At the critical temperature, how does X evolve during coarse-graining using TRG, GILT-TNR and FET-TRG ?

Exercis (d): Extract conformal data

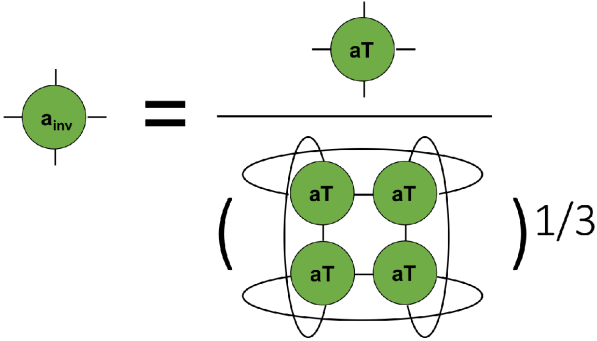
One of the hallmarks of the TRG and its related methods is the ability to obtain accurate estimates of the central charge C and scaling dimensions Δ_n at the critical temperature. The operational procedure to extract conformal data from the bulk tensor aT can be summarized as following*:

Step 1. Rescale the bulk tensor aT , i.e., $aT = \frac{aT}{\Gamma}$, such that



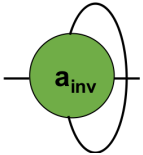
$$= 1$$

Step 2. Obtain the scale invariant tensor a_{inv} , via



$$a_{inv} = \frac{aT}{\left(\begin{array}{cc} aT & aT \\ aT & aT \end{array} \right)^{1/3}}$$

Step 3. Compute the eigenvalue spectrum λ_n of the following matrix,



The first few largest eigenvalues will have the following form $\ln \lambda_n = -2\pi \left(\Delta_n - \frac{C}{12} \right)$.

Compare your results obtained from TRG, GILT-TNR and FET-TRG with the CFT prediction, namely

$C = 0.5, \Delta_1 = 0, \Delta_2 = 0.125, \Delta_3 = 1, \Delta_{4,5} = 1.125$.

For this analysis, one would like to use fewer coarse-graining steps, e.g. $N_{\text{iter}} = 10$ (or increase the bond dimension), before the numerical artifacts set in and push the system away from criticality.

(*) For interested readers, please find the detailed explanation in Z.-C. Gu and X.-G. Wen, Phys. [Rev. B 80, 155131 \(2009\)](#) Appendix Sec. 2.