[credit for course materials: Prof. Jan von Delft]

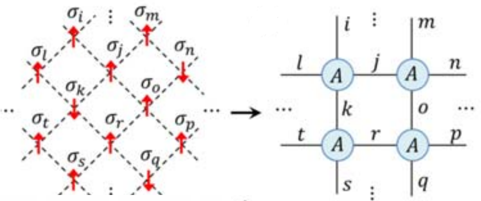
Tensor renormalization group (TRG)

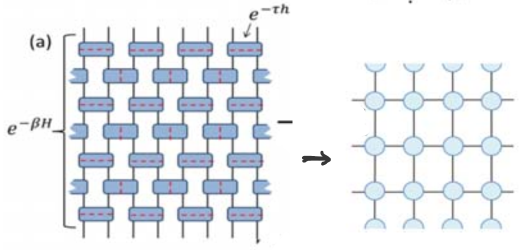
Goal: compute 2D contractions by coarse-graining RG schemes (instead of transfer matrix schemes)

Applications:

Partition functions of 2D classical models

Imaginary time evolution of 1D, 2D quantum models.





[Levin2007] Levin, Nave: proposed original idea for TRG for classical lattice models. Local approach: truncation error is minimized only locally.

[Jiang2008] Jiang, Weng, Xiang: adapted Levin-Nave idea to 2D quantum ground state projection via imaginary time evolution. Local approach: truncation is done via 'simple update'. TRG is used to compute expectation values.

[Xie2009] Jiang, Chen, Weng, Xiang and [Zhao2010] propose 'second renormalization' (SRG), a global approach taking account renormalization of environmental tensor ('full update'). Reduced truncation error significantly.

[Xie2012] Different coarse-graining scheme, using higher-order SVD, employing both local and global optimization schemes.

[Zhao2016] coarse graining on finite lattices.

[Evenbly2019] propose core tensor renormalization group (CTRG) which rescales lattice size linearly (not exponentially), but at much lower cost, (rather than).

1. TRG for 2D classical lattice models

[Levin2007]

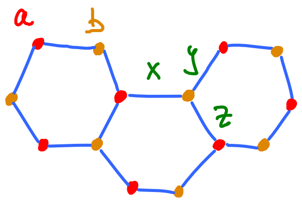
Goal: compute partition function of 2D classical model

Strategy: Express partition function as 2D tensor network, contract it by coarse graining procedure.

Example: 2D classical Ising model on honeycomb lattice [Zhao2010, Sec IIB]

Honeycomb lattice

is bipartite:

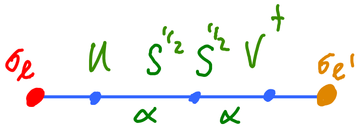


[unit cell contains two sites, labeled , and three bond directions:]

Hamiltonian:

Partition function:

‘Factorize’ the dependence on by performing an SVD:

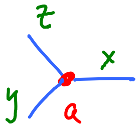


[Note: in following we will not distinguish upper/lower indices]

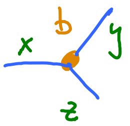
Advantage of this representation: spin dependence has been factorized.

Price to pay: additional 2D bond index, has been introduced.

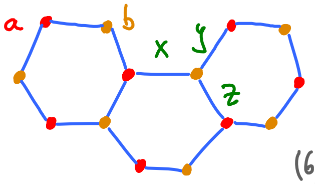
Partition function now contains products of Q matrices on all sites. Because of SVD, a given Q matrix is localized to a given site. Hence, the sum over all spin values can be taken before the product and summed over for each site. To accomplish that, we group all Q’s connected to some site on lattice, and sum over , for given



Same for site on lattice, sum over



Then partition function takes the form



(6)

sum over virtual indices on all nearest-neighbor bonds

All statistical physics models with short-range interactions can be expressed as tensor network models, i.e.

[more examples in [Zhao2010, Sec II]

Contract out the tensor network by coarse-graining

‘rewire’: switch from vertices with external leg pairings (i,j), (l,k) to vertices with pairings (i,l), (j,k)

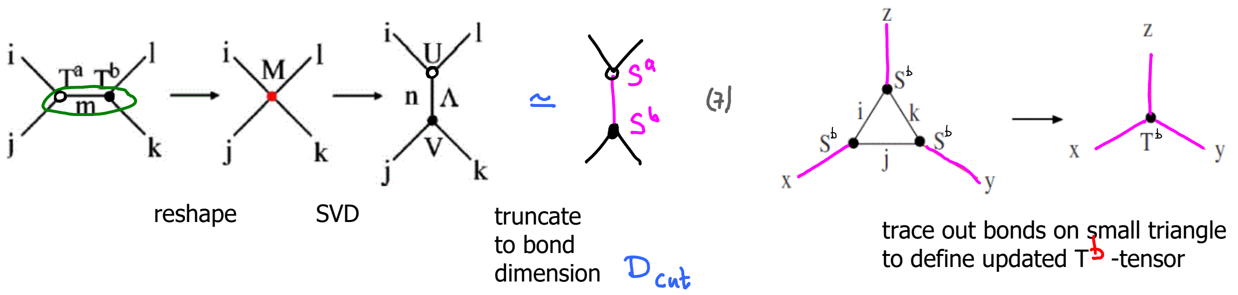
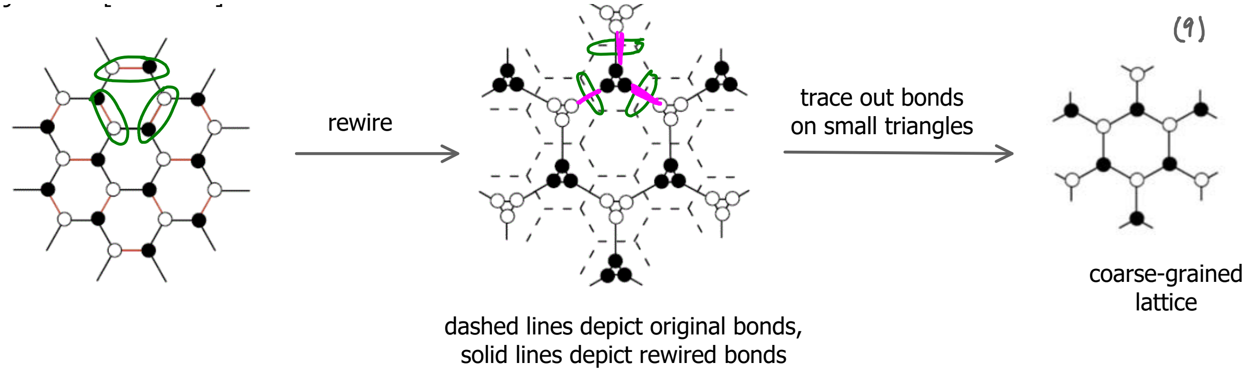
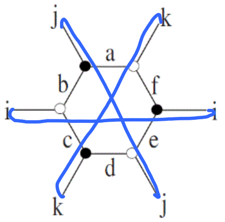

i
l
j
k
T^a
T^b
m
M
U
n
Λ
V
≃
S^a
S^b
(7)
z
S^b
i
k
j
x
y
T^b
reshape
SVD
truncate to bond dimension
D_{cut}
trace out bonds on small triangle to define updated T^b -tensor

Fig 3 from [Zhao2010]


(9)
rewire
trace out bonds on small triangles
dashed lines depict original bonds, solid lines depict rewired bonds
coarse-grained lattice

Iterate this procedure, thereby coarse-graining lattice step by step, until reach fixed point values, . Use these to compute partition function via

$Z =$



and from there the free energy per spin, and the magnetization, etc.

2. TRG for quantum lattice models

[Jiang2008]

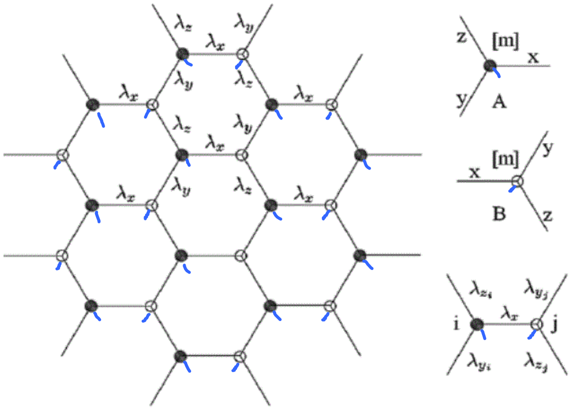
Goal: compute ground state of 2D quantum lattice model

Strategy: iterative projection via , compress by ‘simple update’;

compute and using TRG of Levin & Nave.

Model: Heisenberg on honeycomb lattice.

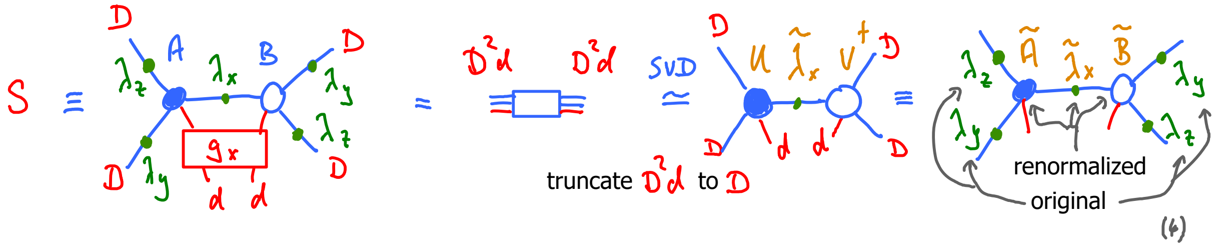
Tensor network ansatz:



Ground state projection via simple update

Suzuki-Trotter:

Sequentially update x, y, z bond using these three gates.



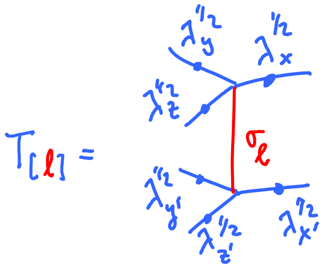
‘simple update’: outer legs of contain , which account for the ‘environment’ of in mean-field fashion. Without including these factors in definition of S, procedure does not converge.

-Similarly update y and z bonds. This concludes one iteration.

-Iterate simple update many times.

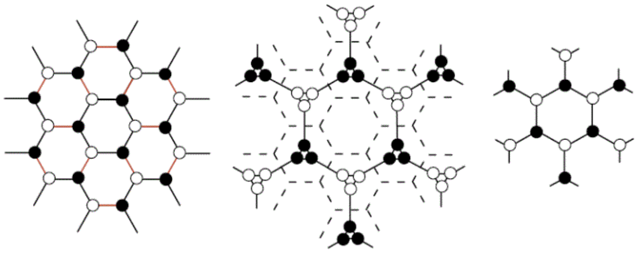
-Start with , gradually reduce it to

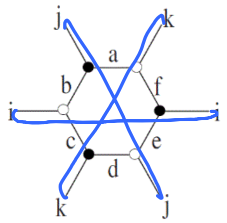
-Number of iterations needed until convergence:



is a double-layer tensor network.

Use TRG (Levin & Nave) to contract bond indices of double-layer network:





Start with a finite system, and iterate until only six sites are left, then trace out final bond indices:

Results

[Jiang2008]

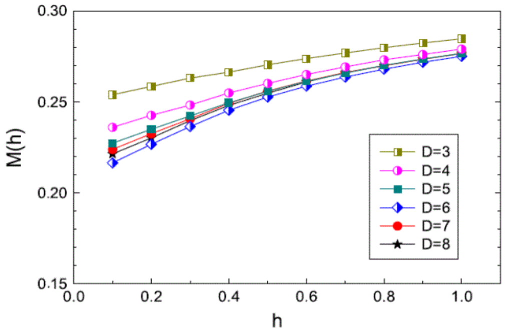


FIG. 5 (color online). The staggered magnetization $M(h)$ as a function of the staggered magnetic field, at different D.

TABLE II. Comparison of our results with those obtained by other approaches for the ground state energy per site E and the staggered magnetization M of the Heisenberg model with $h = 0$.

Method	E	M
Spin wave [12]	−0.5489	0.24
Series expansion [13]	−0.5443	0.27
Monte Carlo [14]	−0.5450	0.22
Ours $D = 8$	−0.5506	0.21 ± 0.01

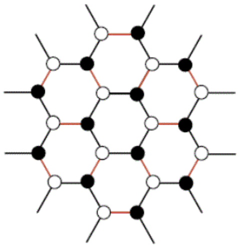
3. Second renormalization (SRG) of tensor networks

[Xie2009, Zhao2010]

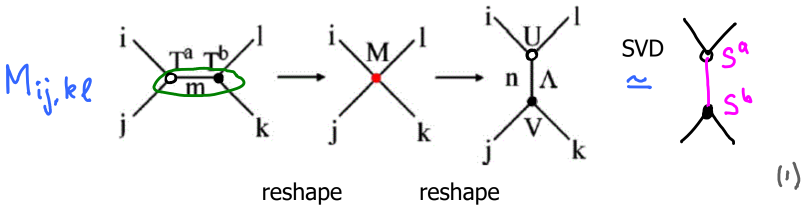
Goal: include influence of environment when doing update -> ‘global optimization’, ‘full update’

Two applications: (i) partition function of classical 2D models; (ii) 2D quantum ground states.

(i) Classical tensor network model



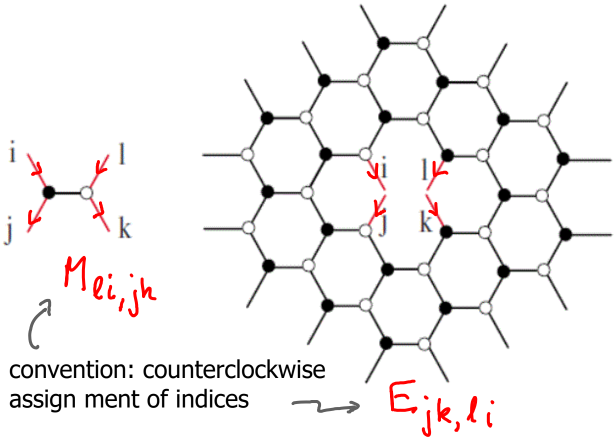
rewire:



SVD minimizes truncation error for rewiring . However, we should minimize truncation error of partition function Z.

Renormalize environment

Partition function:



convention: counterclockwise assign ment of indices

Goal: minimize truncation error of Z.

Strategy:

(i) Compute (a) cheap mean-field approach (‘single update’)

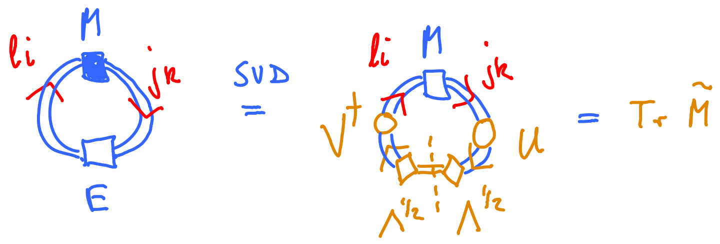
(b) on finite lattices

(c) more expensive forward/backward TRG (‘full update’)

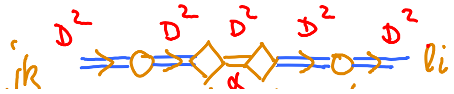
(ii) Do SVD on .

Let’s discuss (ii) first.

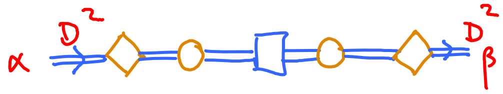
<u>Minimize truncation error of ME</u> [Zhao2010, Sec IIIB]

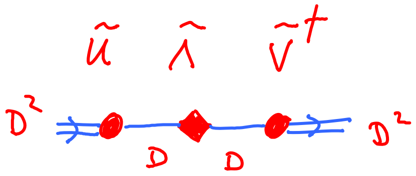


with



and





Since , this truncation directly controls error in partition function!

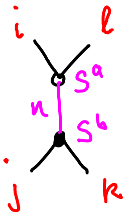
It knows about environment via

Now express in terms of truncated objects, To this end, first invert relation between and , using

then insert truncated version of

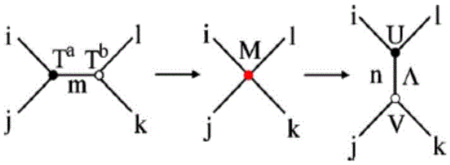
and write as product of two vertices:

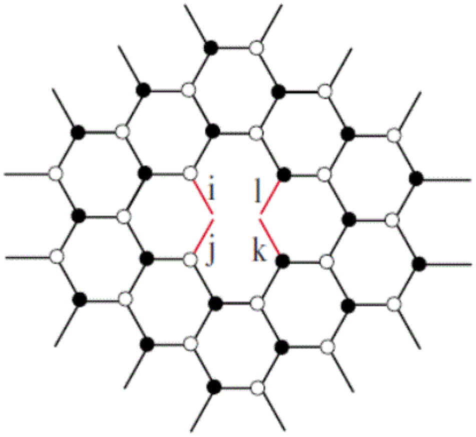
with indices:



Now we return to (i): actually computing the environment

Computing environment tensor using simple update (mean-field approach)





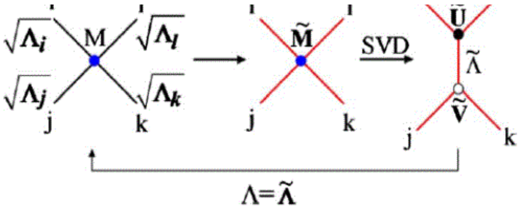
defines the

‘singular bond vector’ , which measures

entanglement between two sites. It can be used directly

to obtain a cheap, mean-field approximation of environment (‘simple update’)

-Take



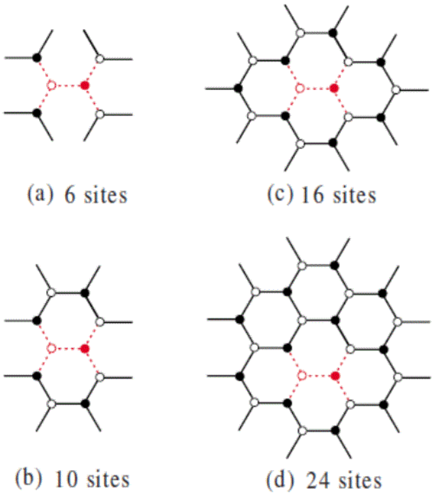
- Compute , then do SVD:

- Use new to recalculate , etc.

-Iterate until convergence (typically 2-3 iterations suffice; near critical point, more are needed).

(ii) Computing environment tensor using finite lattices

The next best approach that improves on the “cheap environment” above is to make a finite lattice approximation for the environment.


(a) 6 sites

(c) 16 sites

(b) 10 sites

(d) 24 sites

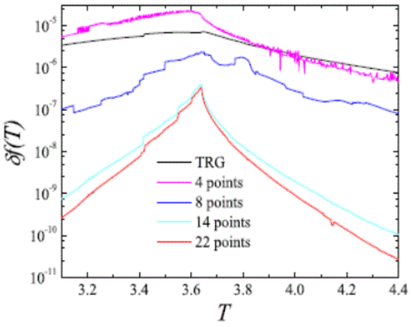


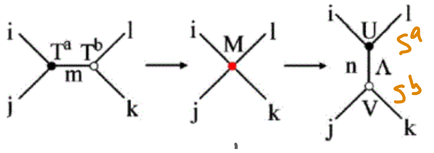
FIG. 10. (Color online) Relative errors of the free energy for the Ising model on a triangular lattice obtained by considering the second renormalization effect from four finite environment lattices which contains 4, 8, 14, and 22 sites, respectively. The configurations of these environments are shown in Fig. 9. The TRG result is also shown for comparison.

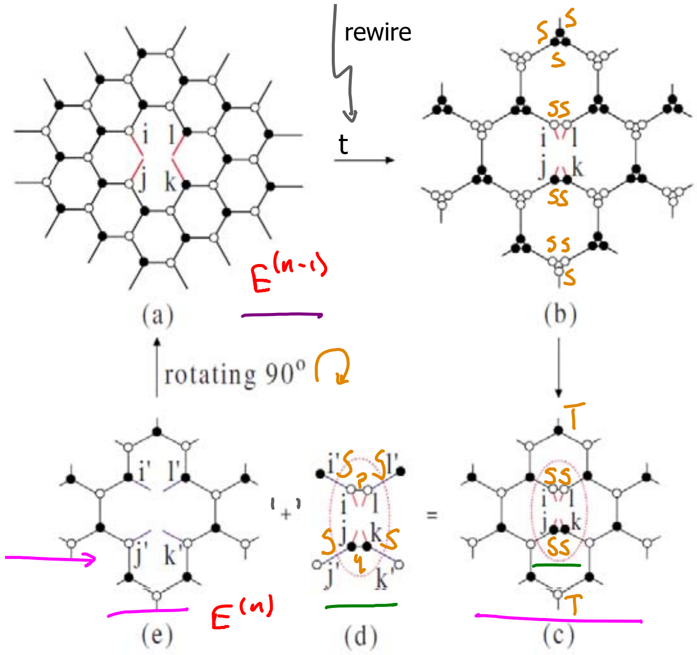
Including even just a few environmental sites already leads to big improvements!

(iii) Computing environment tensor using TRG [Zhao2010]

'Forward iteration':

(a) -> (b): Rewire environment using data at iteration n:





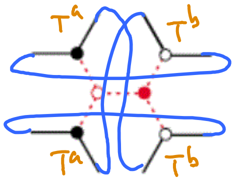
(b) -> (c): Trace out small triangles

four are left over

(c) -> (d) + (e): Identify new environment

(e) looks same as (a), only rotated by 90 degrees, and rescaled.

Iteration relation:



-Start with a very large but finite number of sites.

-Iterate until only 4 environmental sites are left.

-Compute final environment, , by tracing out open indices.

‘Backward iteration’:

-Start from current values of tensors and bond vectors .

-Use them to compute etc, all the way back to = desired result.

This completes step (i). Now go to step (ii), compute , and iterate, until have converged.

<u>Results for SRG (2nd renormalization) for classical 2D systems</u>

Ising model on triangular lattice:

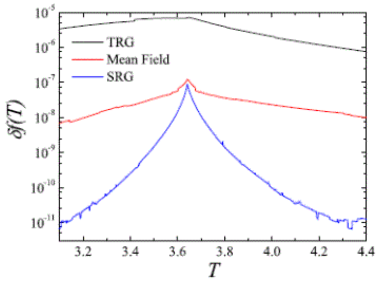
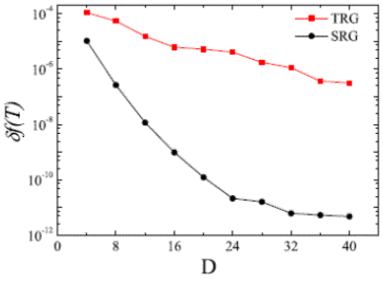


FIG. 12. (Color online) Comparison of the relative error of the free energy for the Ising model on triangular lattices obtained using TRG (red), the mean-field approximated SRG (blue), and the SRG (black) methods with $D_{cut}=24$, respectively. The critical temperature is $T_c=4/\ln 3$.



mean field

SRG

FIG. 13. (Color online) The relative error of the free energy as a function of the truncation dimension D_{cut} for the Ising model on triangular lattices obtained using the TRG (black) and SRG (blue), respectively. $T=3.2$.

critical state is hardest to simulate

error drops with increasing D faster for SRG

Results for SRG (2nd renormalization) for quantum ground state search

Optimize by imaginary time evolution; contractions performed using SRG.

Compute expectation values such as using SRG too.

SRG yields more stable results than TRG!

[Xie2009] : Heisenberg on honeycomb

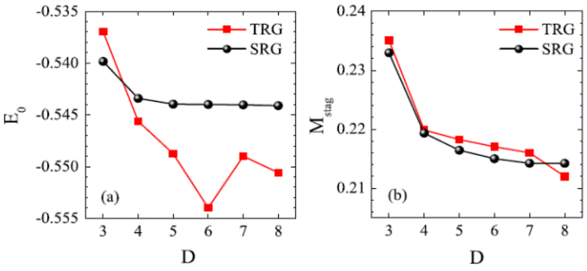


FIG. 5 (color online). (a) The ground state energy per site E_0 and (b) the staggered magnetization M_{stag} as functions of the bond degrees of freedom D on honeycomb lattices.

[Zhao2010]

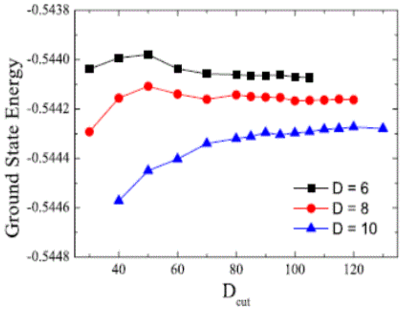


FIG. 19. (Color online) The SRG result of the ground-state energy as a function of the truncation dimension D_{cut} for the Heisenberg model on a honeycomb lattice. D is the bond dimension of the wave function.

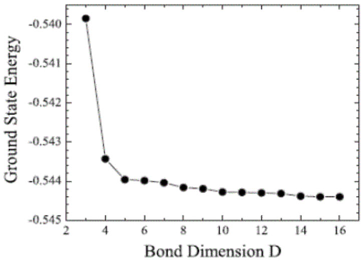


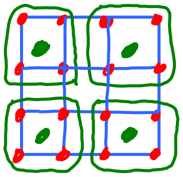
FIG. 20. The ground-state energy of the Heisenberg model on a honeycomb lattice as a function of the bond dimension D obtained by the SRG with D_{cut}=130.

$E^{SRG} = -0.54440$ $E^{QMC} = -0.54455(20)$

Energy does not decrease with D_cut, because imaginary-time evolution/SRG is not variational!

For reference: very recent results [Lan2019, core tensor renormalization group] reduces cost of TRG from

Graph-independent local truncation (GILT)



Goal: improve TRG by fully removing local correlations, including those in local loops.

Strategy: devise truncation scheme based on 'environment spectrum' of local tensors.

1. Motivation

TRG is a concrete, quantitative implementation, for lattice models, of Wilson's RG idea:

This generalizes Kadanoff's block-spin RG, , which approximates coarse-grained system by same Hamiltonian, but parametrized by rescaled coupling.

TRG instead allows the form of the Hamiltonian (or corresponding tensors) to change.

BUT: TRG, as formulated by Levin and Nave, does not fully remove all local correlations.

Reason: it is based on SVD of local tensors, so removes local correlations only for 'tree tensor networks'. Effect of environment is not included (SRG/full update is an attempt to do that). As a result, fixed point tensors still include some information from short-range physics. Hence, TRG does not yield 'proper RG flow' (which should eliminate all short-range physics).

Needed: schemes that fully remove local correlations at each length scale.

Some key players in this quest:

Levin himself pointed out that TRG fails for ‘corner double-line tensors’ (CDL). (Public talks, 2007)

[GU2009] Gu & Wen: clearly identify above problem, proposed ‘tensor entanglement filtering renormalization’ (TERG) to remedy it. This led to discovery of ‘symmetry protected topological order’, and a classification thereof via structure of fixed point tensors.

[Evenbly2015] Evenbly & Vidal: propose ‘tensor network renormalization’ (TNR), which goes beyond TRG by including ‘disentanglers’, allowing removal of all short-ranged correlations at each length scale.

[Evenbly2015a] Evenbly & Vidal: show that TNR generates a MERA (multiscale entanglement renormalization ansatz) structure. (MERA was proposed in [Vidal2007, 2007a, Evenbly 2009]

[Evenbly 2017] Detailed description of TNR, including strategies for optimizing disentanglers (carried over from MERA, as described in [Evenbly 2009]). High cost.

[Yang2017], Yang, Gu, Wen: propose loop optimization for TNR (loop-TNR), which is more effective than TERG. Also more effective and cheaper than TNR.

[Ying2017] Proposes ‘tensor network sceletonization’ (TNS). Separate steps for coarse-graining and removal of local correlations. Needs costly iterative optimization. Blind to nature of local correlations.

[Bal2017] Bal, Marien, Haegeman, Verstraete: propose TNR+, similar in spirit to [Yang2017], but using element-wise purely positive tensors.

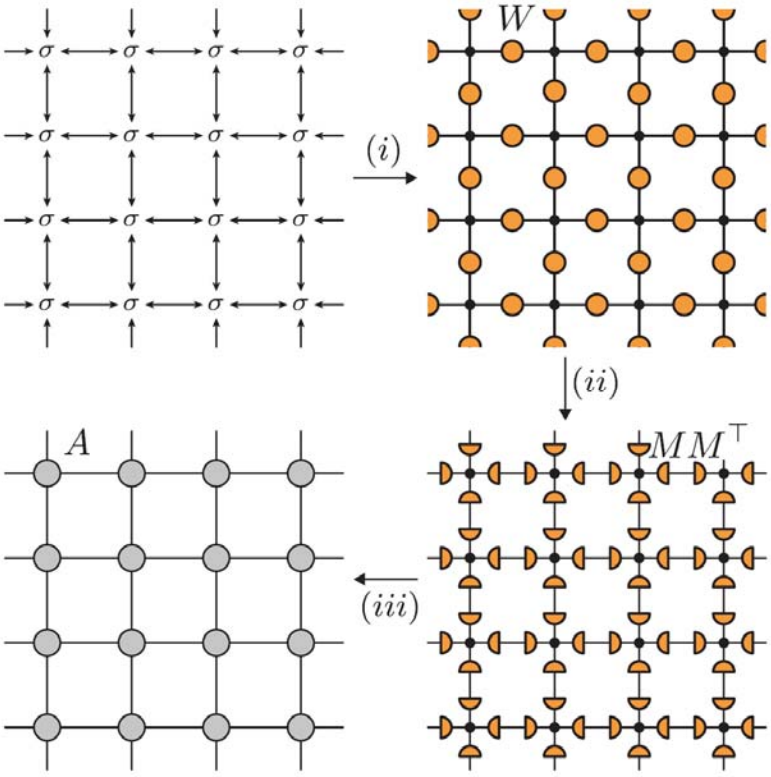
[Evenbly2018] Proposes ‘canonical form’ and ‘optimal truncations’ for tensor networks with closed loops. Optimization scheme include environment information. Simpler than optimization scheme of MERA/TNR. Performance: better than TNR, loop TNR, comparable to GILT.

[Hauru2018] Hauru, Delcamp, Mizera: propose ‘graph-independent local truncations’ (GILT) of individual bonds of network, based on analysis of environment spectrum. Very simple, clear scheme. Outperforms all previous 2D approaches. Even applied to 3D systems!

2. Why is TRG insufficient?

[Hauru2018, Sec 2]

Representing partition function as classical model as a tensor network (graphical argument)



-each site hosts a classical configuration variables

-(i) each bond is characterized by Boltzmann weight,

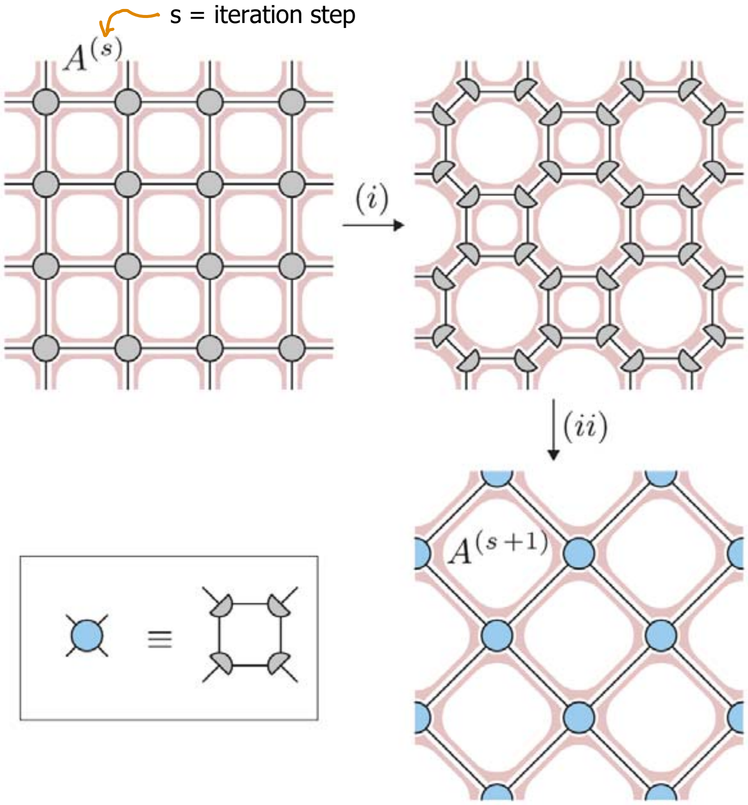
-(ii) SVD

-(iii) at each vertex, contract bonds to obtain 4-leg tensor

Single iteration of TRG for square lattice:

(i) Use truncated SVD to split 4-leg tensor along different diagonals

(ii) Contract sets of 3-leg tensors to obtain new 4-leg tensors.

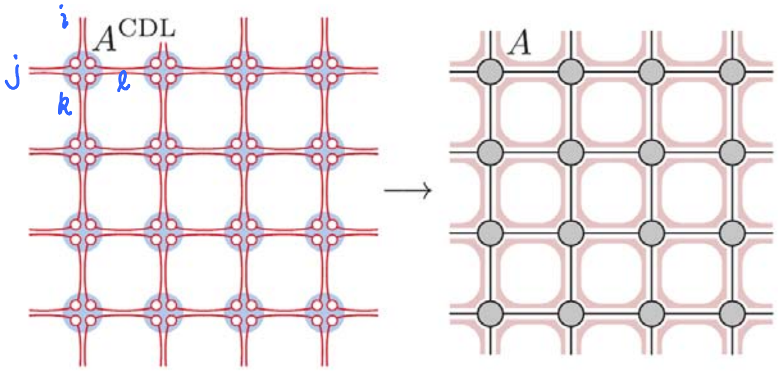


Shaded red loops represent short-range loop correlations. Step (ii) ‘captures’ half of red loops; effect is encoded in . But other half of the red loops remain, and become nearest-neighbor correlations of coarse-grained tensors.

This violates principle of RG that coarse-grained description should not include short-ranged (UV) details.

As a result, fixed-point tensors depend on non-universal details, such as exact value of temperature. (Only T < Tc or T > Tc should matter).

Corner double-line (CDL) tensors: an extreme example where TRG fails completely



Each index is two-fold composite:

$$A^{\mathrm{CDL}}_{ijkl} \equiv A^{\mathrm{CDL}}_{(i_1 i_2)(j_1 j_2)(k_1 k_2)(l_1 l_2)}$$
$$= M_{i_1 j_2} M_{j_1 k_2} M_{k_1 l_2} M_{l_1 i_2}$$

Two observables on same plaquette are strongly correlated. Observables on different plaquettes are totally uncorrelated.

CDL model is a toy model for purely short-ranged physics: it encodes no correlations at length scales larger than a lattice spacing. Under a proper RG, it should flow to a trivial fixed point.

However, TRG leaves CDL tensors invariant, i.e. CDL model is fixed point of RG transformation.

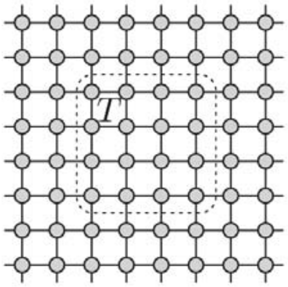
This example illustrates: TRG fails to remove loop correlations!

Terminology for this problem: ‘accumulation of local, or short-range, correlations’.

This problem gets worse in higher dimensions….

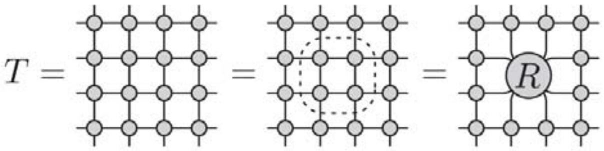
3. Environment spectrum

[Hauru2018, Sec 3]

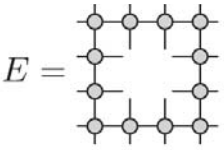


Consider a local neighborhood, T, of a global network.

Goal: make changes (typically truncations) to a local subnetwork, R, of T, without affecting T.



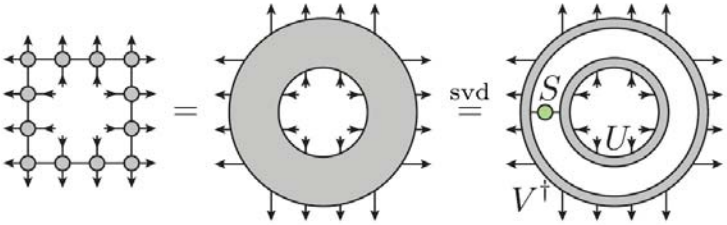
(1)



(2)

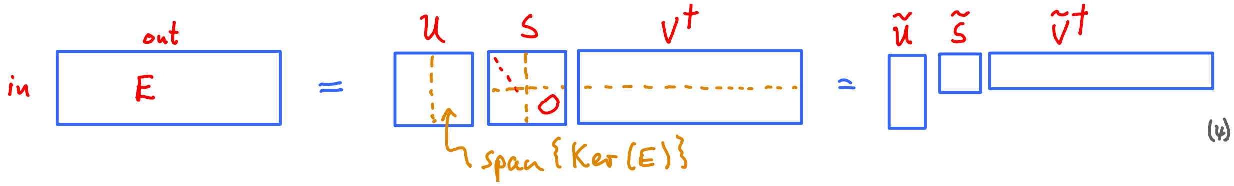
E = T / R = environment that is left upon removing R from T.

Do SVD, between incoming legs (coming from R) and outgoing legs (going to outside):



Singular values = ‘environment spectrum’ (quantifies how much R affects outside world)

The kernel of E (the incoming subspace that is mapped to 0) is irrelevant to outside and may be discarded.

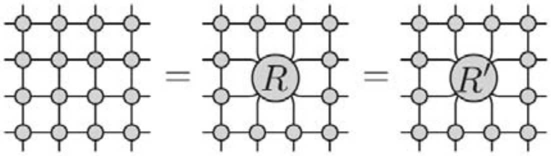


(4)

If

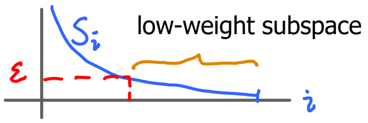
then replacement

does not affect outside world.



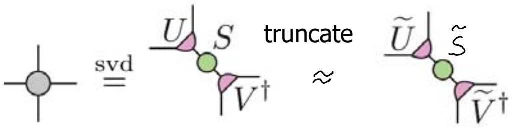
If ‘low weight subspace of E’ (where, say,)

then affects outside world only weakly.



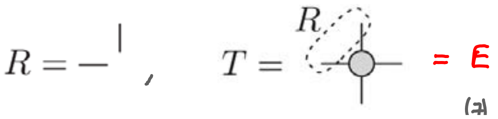
<u>Simple example: splitting 4-leg tensor via truncated SVD</u>

Consider first step of TRG:



This can be formulated as truncation of environment spectrum:

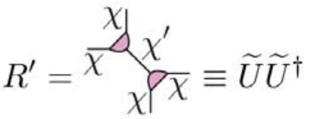
Choose R = two legs = product of two identity matrices:



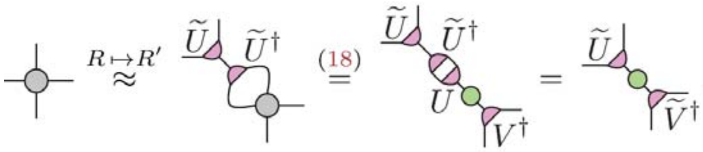
E = T, since cutting identity matrices from outer legs does nothing.

If R is replaced by R’ = projector removing low-weight subspace of E,

(with intermediate bond dimension)

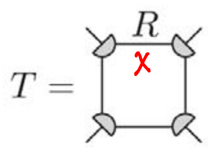


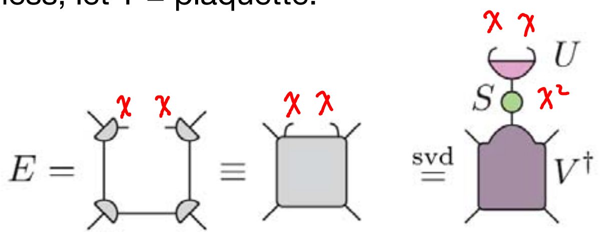
then we recover SVD-truncation:



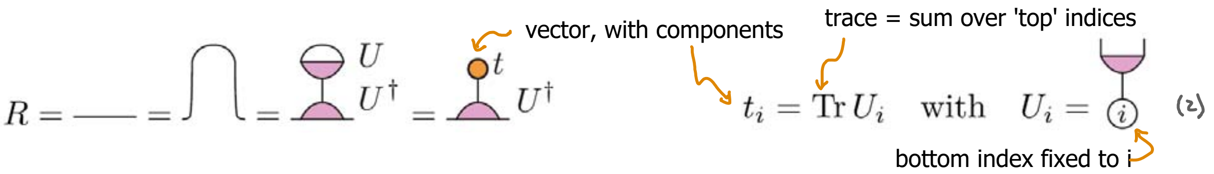
4. GILT: Graph-independent local truncation

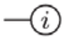
Let R = leg in network T. For concreteness, let T = plaquette:





Environment spectrum S quantifies which part of R-space matters only for physics internal to plaquette. To exploit this information, make basis change on leg R:

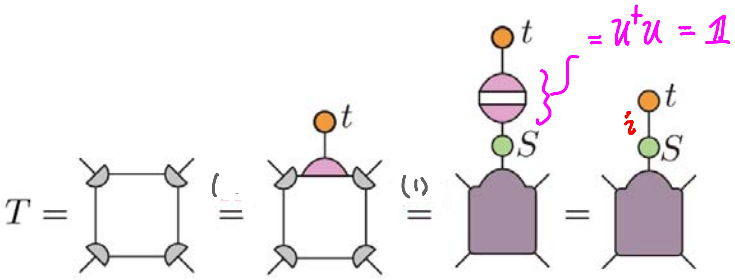




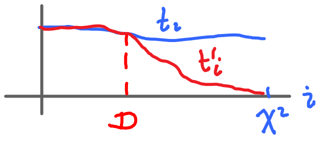
$= (0, 0, \cdots, 1, \cdots, 0)$ = vector with i-th element = 1, all others = 0

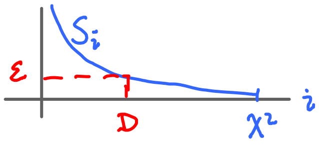
Inserting environment definition into above, we see that environment matrix S, which is diagonal, multiplies elements

of ,



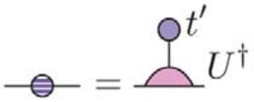
So, if we replace by some other value, , differing from only for low-weight-subspace of E, then changes only by

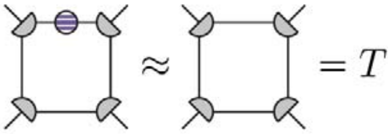




Correspondingly, we can replace leg,

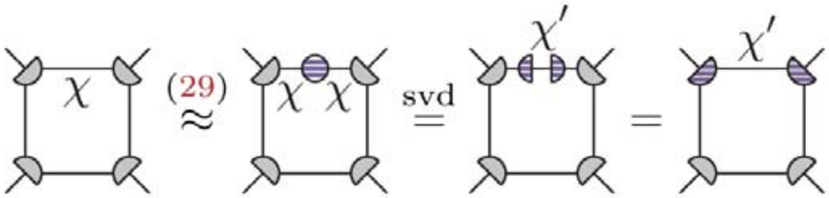
by a matrix





while T hardly changes:

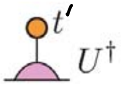
This freedom can be exploited to make matrix R’ have as low rank, , as possible. Low is desirable, since SVD of R’, followed by redefinition of neighboring sites, brings down bond dimension to .



Optimizing the choice of t’

[Hauru2018, App B]

The rank of as a function of



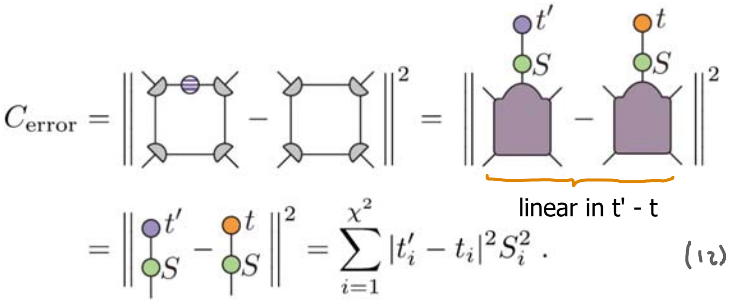
is a complicated cost function to minimize. Simpler alternative: minimize

Rationale: minimizing the cost function requires reducing the individual and the more of them come close to zero, the smaller the rank of R’.

Moreover,

So, those elements that we are free to choose (associated with low-weight subspace of E) should be chosen as small as possible.

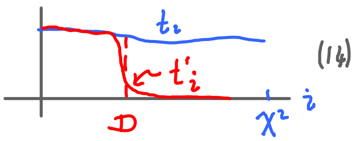
On the other hand, the replacement of R by R’ causes an error.



$$= \sum_{i=1}^{\chi^2} |t'_i - t_i|^2 S_i^2 \, . \qquad (12)$$

This error should remain . Thus minimize combined cost function:

This is minimized by choosing



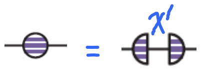
Summary of GILT algorithm for truncating leg R in environment E:

(i) SVD the environment E to obtain the unitary U and environment spectrum S (*)

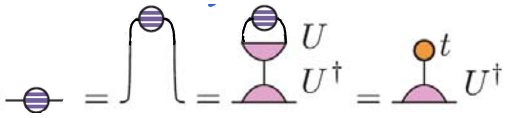
(ii) Compute the traces

(iii) Choose the vector t' as above, and compute matrix

(iv) SVD R' as



Repeat (ii)-(iv) on same bond, now with R' as input:



(v) Once singular values of R' have converged, multiply into neighboring tensors.

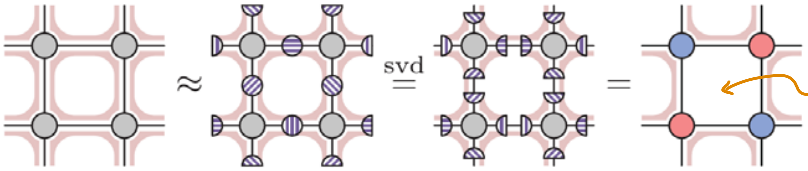
(*) We only need U and S, so instead of full SVD of , it suffices to compute eigenvalue decomposition of the Hermitian matrix

This is computationally much cheaper, and for square plaquette reduces cost to

5. GILT Tensor network renormalization (Gilt-TNR)

Problem with TNR was: local correlations are dealt with properly only around every other plaquette. Remedy: before each TRG step, apply Gilt to all four bonds of problematic plaquettes.

The four matrices R’ must be created in serial, not parallel, since each one modifies environment of the others. Together, they truncate away any details internal to plaquette, by modifying tensors at corners.



'no loop correlations inside' !

A single iteration of Gilt-TNR:

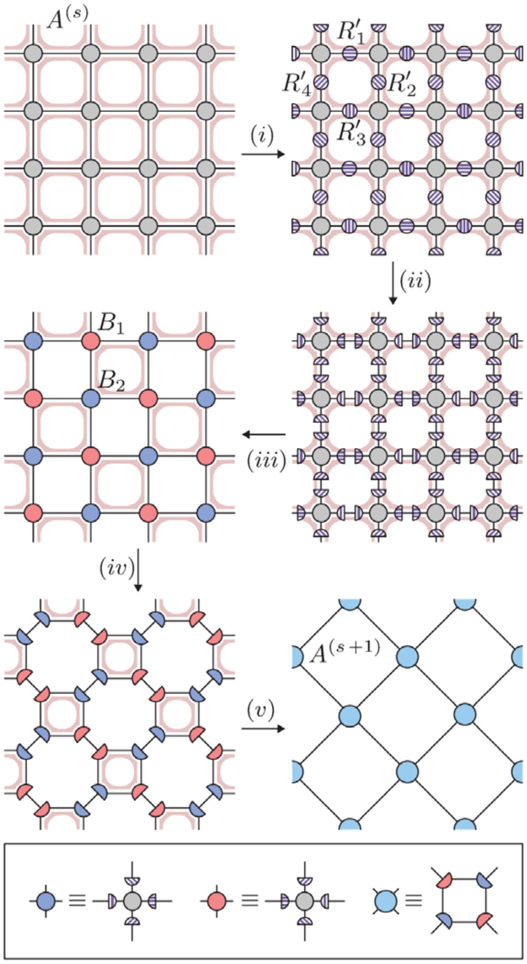
(i) Insert R’ tensors on bonds

(ii) SVD R’ tensors

(iii) Contract to compute B1, B2. This removes internal correlations for half of the plaquettes.

(iv) Split B1, B2 tensors via standard TRG.

(v) Contract to compute new A. This removes correlations from other half of the plaquettes.



Main advantage of Gilt-TNR over rivals:

-‘simplicity and generality’ (minimal working implementation takes ~100 lines of code)

-No iterative optimization of truncation.

-Graph does not change.

Applying Gilt to non-square lattices requires simply changing the neighborhood T used for Gilt step.

-So efficient it has already been applied to 3D Ising model.

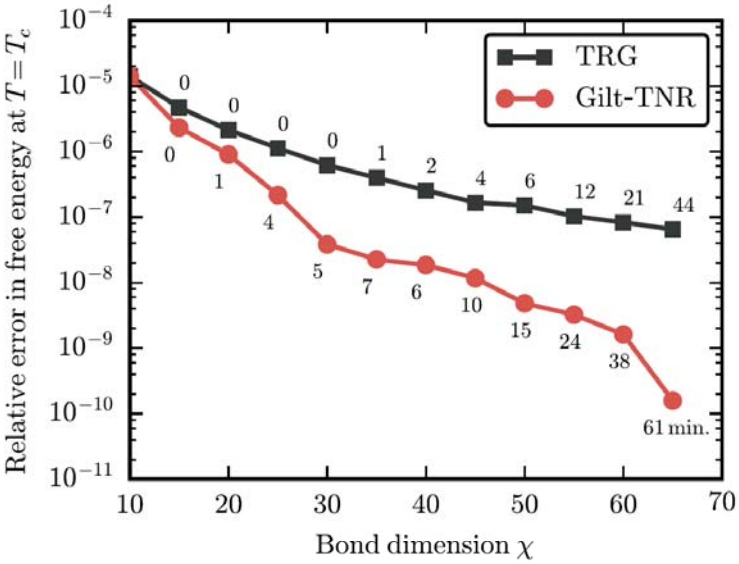
6. Benchmark results

[2D classical Ising model]

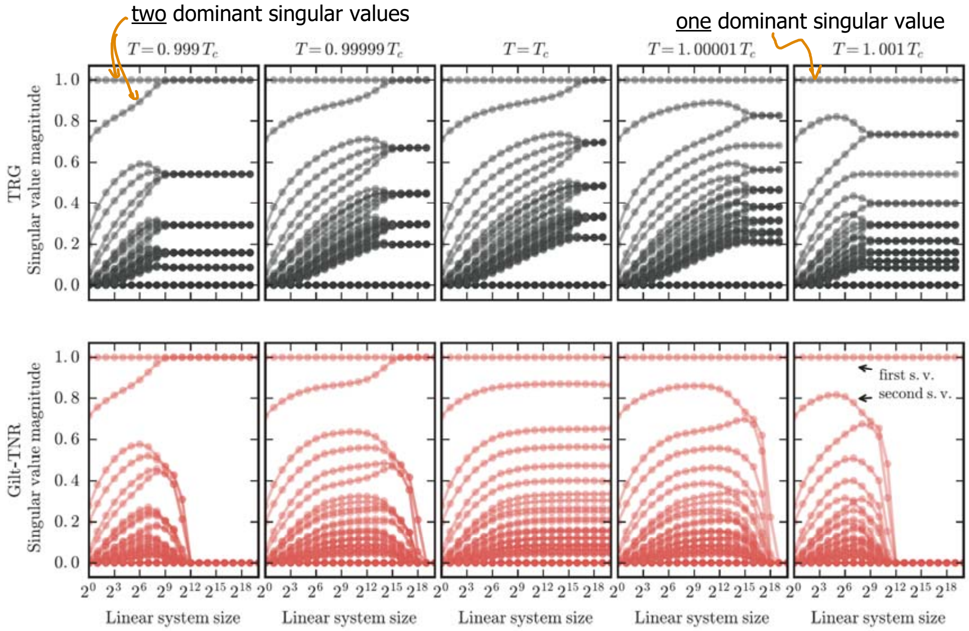
-TRG and Gilt-TNR obtained with same code; for TRG, Gilt was turned off.

-Gilt-TNR orders of magnitude more accurate than TRG

-at only moderate increase in run-time.



RG flow of TNR and Gilt-TNR



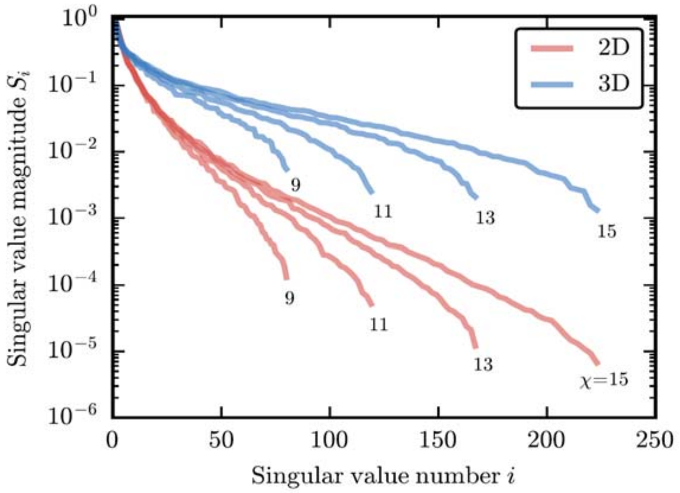


FIG. 7. Typical environment spectra of a single leg with respect to a square plaquette in 2D (bottom four spectra) and a cube in 3D (top four spectra), each labeled with the corresponding bond dimension χ. Recall that in each spectrum, the large values correspond to parts of the vector space of the leg that are relevant for physics outside the plaquette or the cube, whereas small values signify contributions relevant only for short-range details. The spectra in 3D can be seen to decay much more slowly, indicating that larger bond dimensions are necessary, before truncations with a small error are possible. The behavior of the spectra as χ is increased, is also somewhat different in 2D and 3D. In 2D the longer spectra have more values mainly at the bottom end, whereas in 3D new values appear almost throughout the whole spectrum. The example spectra shown here are for the Ising model, from systems that have been coarse grained thrice. Many other choices of system sizes would yield qualitatively similar results, and the same overall difference between 2D and 3D can also be seen with

internal correlation: