多体問題の計算科学

Computational Science for Many-body problems

2023.5.23

#7: テンソル繰り込み群

Tensor Renormalization Grope

理学系研究科　大久保毅

Graduate school of science, **Tsuyoshi Okubo**

email: t-okubo@phys.s.u-tokyo.ac.jp

- This class is from 14:55 to 16:40 (105 min.)

Today

Classical

1st: Many-body problems in physics and why they are hard to solve

2nd: Classical statistical models and numerical simulation

3rd: Classical Monte Carlo method

4th: Applications of classical Monte Carlo method

5th: Molecular dynamics simulation and its applications

6th: Extended ensemble method for Monte Carlo methods

Quantum

7th: Tensor Renormalization group

8th: Quantum lattice models and numerical simulation

9th: Quantum Monte Carlo methods

10th: Applications of quantum Monte Carlo methods

11th: Linear algebra of large and sparse matrices for quantum many-body problems

12th: Large sparse matrices and quantum statistical mechanics

13th: Advanced algorithms for quantum many-body problems

Today's contents

- Extended ensemble method
  - Extended ensemble methods using information of the density of states
  - Another extended ensemble: Replica exchange method
- Tensor Renormalization Group
- Report 1

Contents

- ~~Back ground~~
  - ~~Density of states and the histogram method~~
- Extended ensemble methods using information of the density of states
  - Multi Canonical Method
  - Wang-Landau method
- Another extended ensemble: Replica exchange method

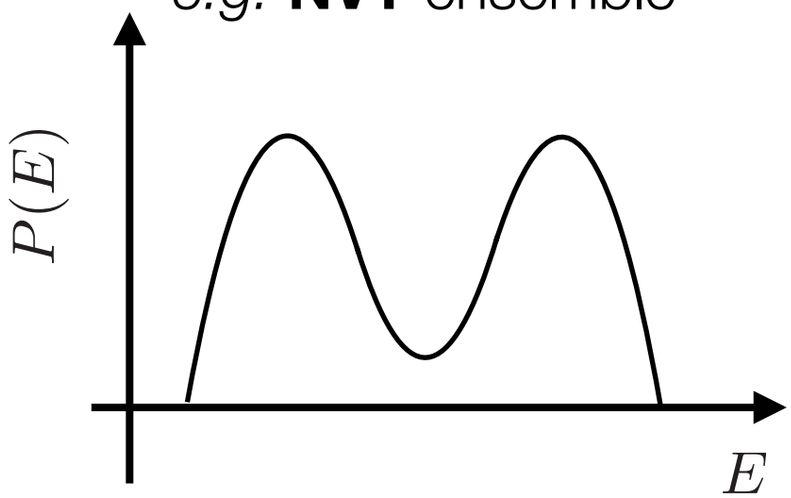
Multi Canonical methods

Idea of Multi-Canonical method

B.A. Berg and T. Neuhaus (1992)

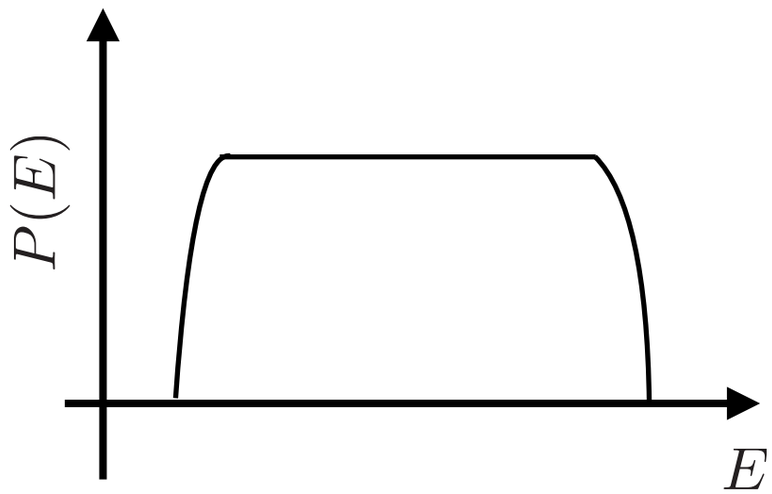
If we can prepare a special ensemble
where the energy distribution is “flat”,
we can efficiently sample all relevant states.

e.g. **NVT** ensemble



$$P(E) \propto \rho(E) e^{-\beta E}$$

“special” ensemble



$$\tilde{P}(E) \propto \rho(E) e^{-S(E)} = \text{const.}$$

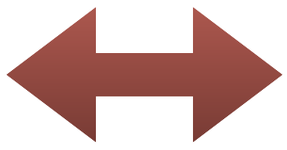
Special ensemble is related to log of DOS!

$$S(E) = \log \rho(E)$$

How to obtain the special ensemble?

Special ensemble is log of DOS!

$$S(E) = \log \rho(E)$$



DOS is unknown!

We can obtain $S(E)$ approximately by iterative calculations.

“Sketch” of an iterative algorithm:

1. Run MC simulation on a high temperature and calculate energy histogram.

$$h(E) \sim \rho(E) e^{-\beta E}$$

2. Based on the energy histogram, extract approximate $S(E)$.

$$S^0(E) = \beta E + \log h(E)$$

3. **Loop** n
    1. Run MC simulation under $S^{(n)}(E)$ and calculate histogram $h^{(n)}(E)$
    2. Calculate next $S^{(n+1)}(E)$ as

$$S^{(n+1)}(E) = S^{(n)}(E) + \log h^{(n)}(E)$$

How to obtain the special ensemble?

3. **Loop** n
   1. Run MC simulation under $S^{(n)}(E)$ and calculate histogram $h^{(n)}(E)$
   2. Calculate next $S^{(n+1)}(E)$ as

$$S^{(n+1)}(E) = S^{(n)}(E) + \log h^{(n)}(E)$$

- The histogram $h^{(n)}(E)$ is expected to be $h^{(n)}(E) \sim \rho(E)e^{-S^{(n)}(E)}$
- When $S^{(n)}(E)$ becomes close to $log\, \rho(E)$, the histogram becomes almost flat.

→ We can efficiently sample the histogram and DOS.

- By using accurate $S^{(n)}(E)$ we can calculate expectations values under the canonical ensemble by using reweighting technique.

$$\langle O \rangle_\beta = \frac{\int dE O(E)\rho(E)e^{-S(E)}e^{-\beta E+S(E)}}{\int dE \rho(E)e^{-S(E)}e^{-\beta E+S(E)}} = \frac{\langle Oe^{-\beta E+S(E)}\rangle_S}{\langle e^{-\beta E+S(E)}\rangle_S}$$

Detailed algorithm (will be skipped)

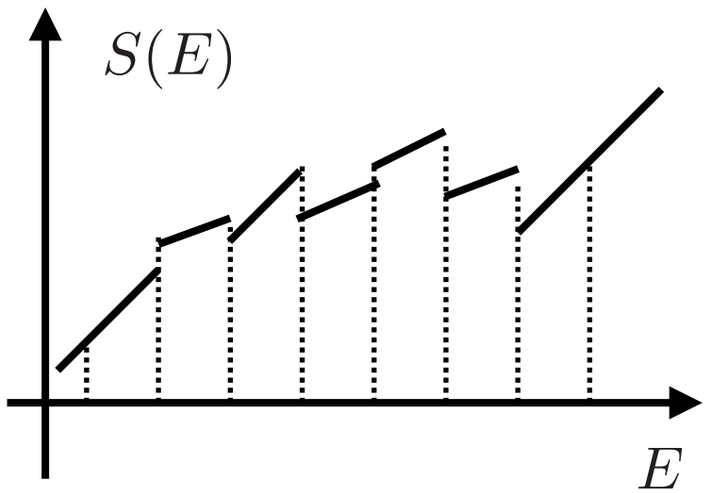
B.A. Berg, Nucl. Phys. B (Proc. Suple.) **63**A-C, 982 (1998)

Suppose $S(E)$ looks like: $S(E) = \beta(E)E - \alpha(E)$

(Energy dependent temperature)

$$S(E) \simeq \beta_i E - \alpha_i$$

for $E_i - \Delta E/2 \leq E \leq E_i + \Delta E/2$



In a specific interval, we want to optimize β and α,
i.e. $P(E)$ becomes flat.

By defining $\beta_i \equiv \dfrac{S(E_{i+1}) - S(E_i)}{\Delta E}$

$$\alpha_{i-1} = \alpha_i + (\beta_{i-1} - \beta_i)E_i$$

We fix $\alpha_{i_{max}} = 0$

Detailed algorithm (will be skipped)

B.A. Berg, Nucl. Phys. B (Proc. Suple.) **63**A-C, 982 (1998)

Iteration :how to determine next β and α

In order to make the histogram flat, $S^{(n+1)}(E) = S^{(n)}(E) + \log h^{(n)}(E)$

$$\tilde{\beta}_i^{(n+1)} = \beta_i^{(n)} + \log \frac{h_{i+1}^{(n)}}{h_i^{(n)}}$$

This estimator could be suffered from large statistical error

Gradual change from the previous β

$$\beta_i^{(n+1)} = (1 - c_i)\beta_i^{(n)} + c_i \tilde{\beta}_i^{(n+1)}$$

*For optimal c_i , see the reference

α is calculated from β

$$\alpha_{i-1}^{(n+1)} = \alpha_i^{(n+1)} + (\beta_{i-1}^{(n+1)} - \beta_i^{(n+1)})E_i$$

Example of application

q-state Potts model on the square lattice

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} \delta_{S_i, S_j} \qquad S_i = 0, 1, 2, \cdots, q-1$$

Phase transition at

$$T_c/J = \frac{1}{\log(1+\sqrt{q})}$$

$q = 2$: Equivalent to Ising model

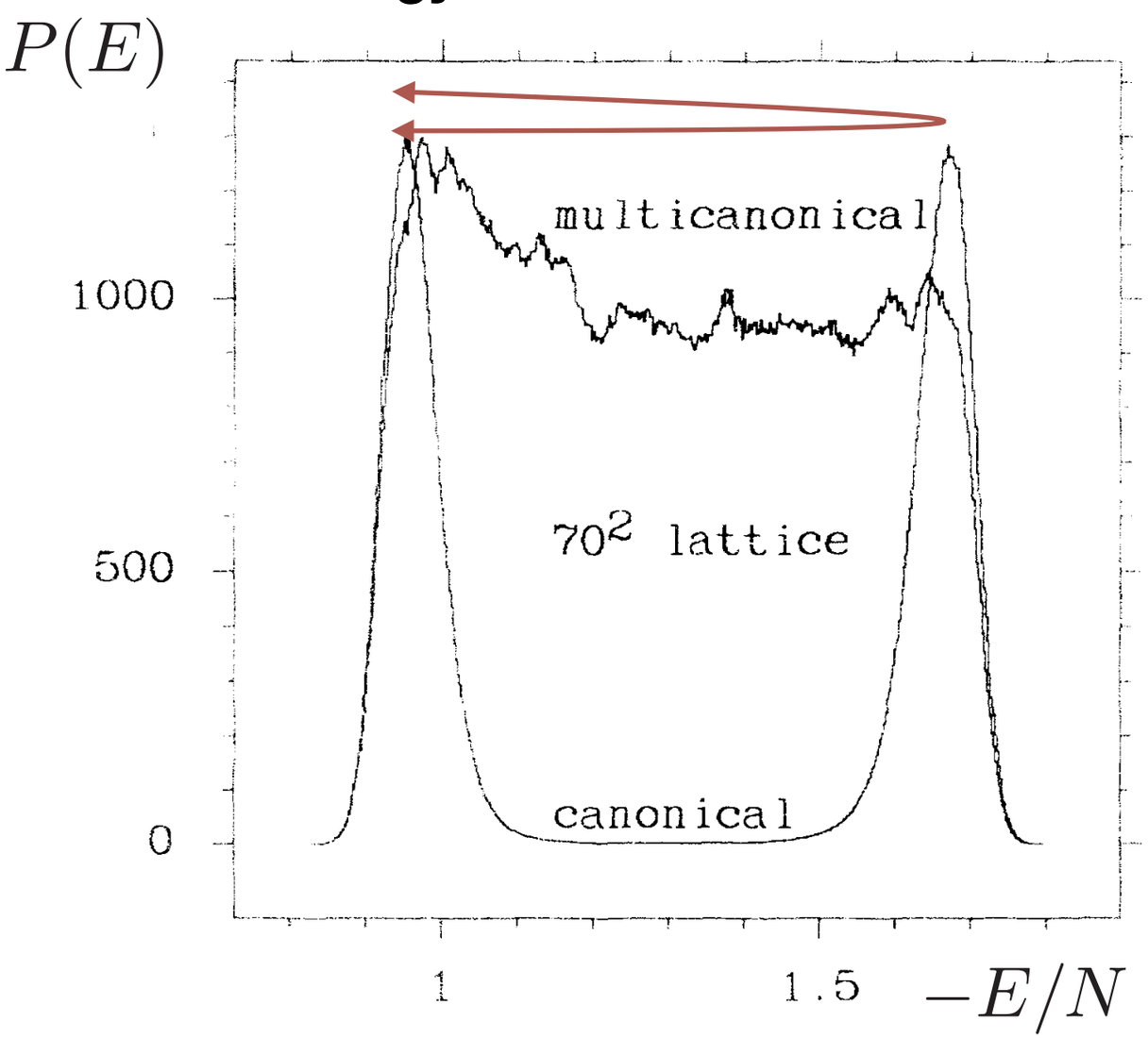
$q \leq 4$: Continuous phase transition

$q > 5$: 1st order phase transition

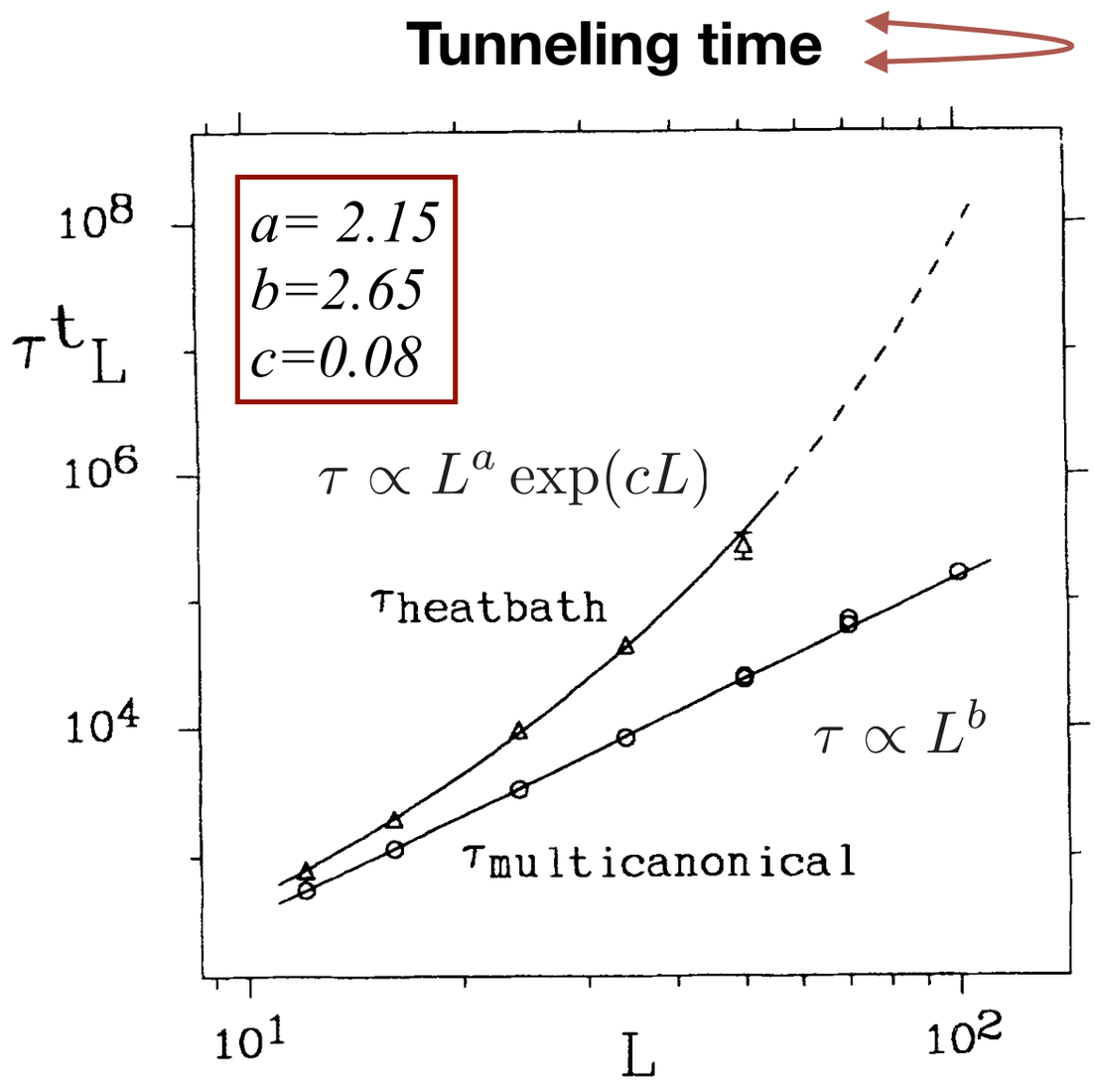
Multi Canonical method for q=10 Potts model

B.A. Berg and T. Neuhaus, Phys. Rev. Lett. **68**, 9 (1992)

Energy distribution around Tc



Tunneling time



By Multi canonical method, the tunneling time is reduced to the power of L!

Wang-Landau method

Wang-Landau method

F. Wang and D. P. Landau (2001)

Another method to obtain the density of state:

Random walk on the energy space

Markov Chain Monte Carlo with the transition probability

$$W_{\Gamma\to\Gamma'} = \min\left(\frac{g(E(\Gamma))}{g(E(\Gamma'))}, 1\right)$$

$g(E)$:estimate of DOS

→ Its steady state is

$$P(\Gamma) \propto \frac{1}{g(E(\Gamma))}$$

The energy distribution (histogram) :

$$P(E)dE = P(\Gamma)d\Gamma = P(\Gamma)\rho(E)dE \propto \frac{\rho(E)}{g(E)}dE$$

if $g(E) = \rho(E)$ → This MCMC gives us a completely flat histogram!

Wang-Landau method:update of g(E)

F. Wang and D. P. Landau (2001)

Initially, we don't know DOS. → Set an initial guess, e.g. $g(E) = 1$

Along MCMC, we update $g(E)$ of the $E(\Gamma)$ as

$$g_{new}(E) = g(E) \times f \qquad (\log g_{new}(E) = \log g(E) + \log f)$$

*Note after N step, g(E) changes like

$$g_{new}(E) \sim g(E) f^{N \frac{\rho(E)}{g(E)}}$$

g(E) increases when $g(E) < \rho(E)$ for $f > 1$.

If the multiplication factor is "gradually" reduced to $f = 1$,

$g(E)$ eventually converges to $\rho(E)$.

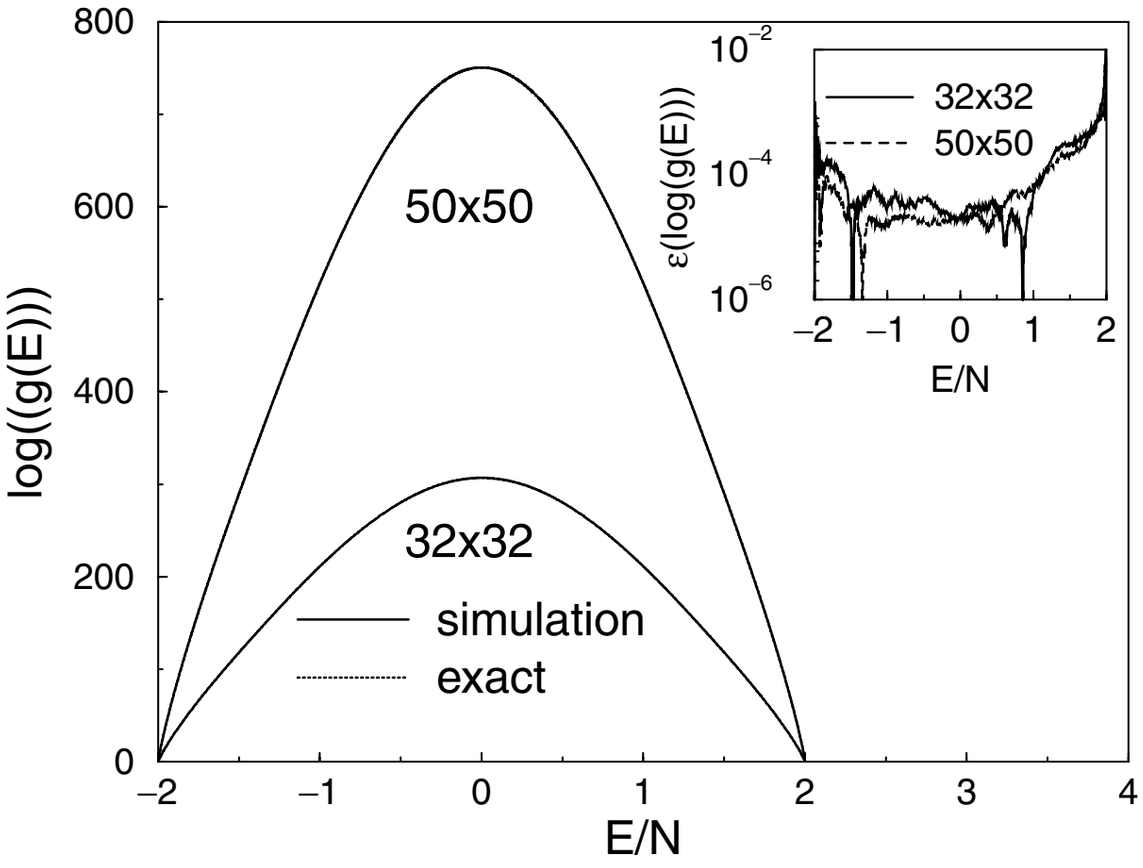
"gradual" change of f :

1. Initially $f = f_0$ (e.g. $f_0 = e^1$)
2. Loop i
   - If (the histogram $h(E)$ becomes "flat"?)
     - Then, we decrease f_i as
       $f_{i+1} = (f_i)^x$ (e.g. $x = 1/2$),
       and reset the histogram.
3. Repeat until f_i becomes enough small (e.g. $f \sim \exp(10^{-8})$)

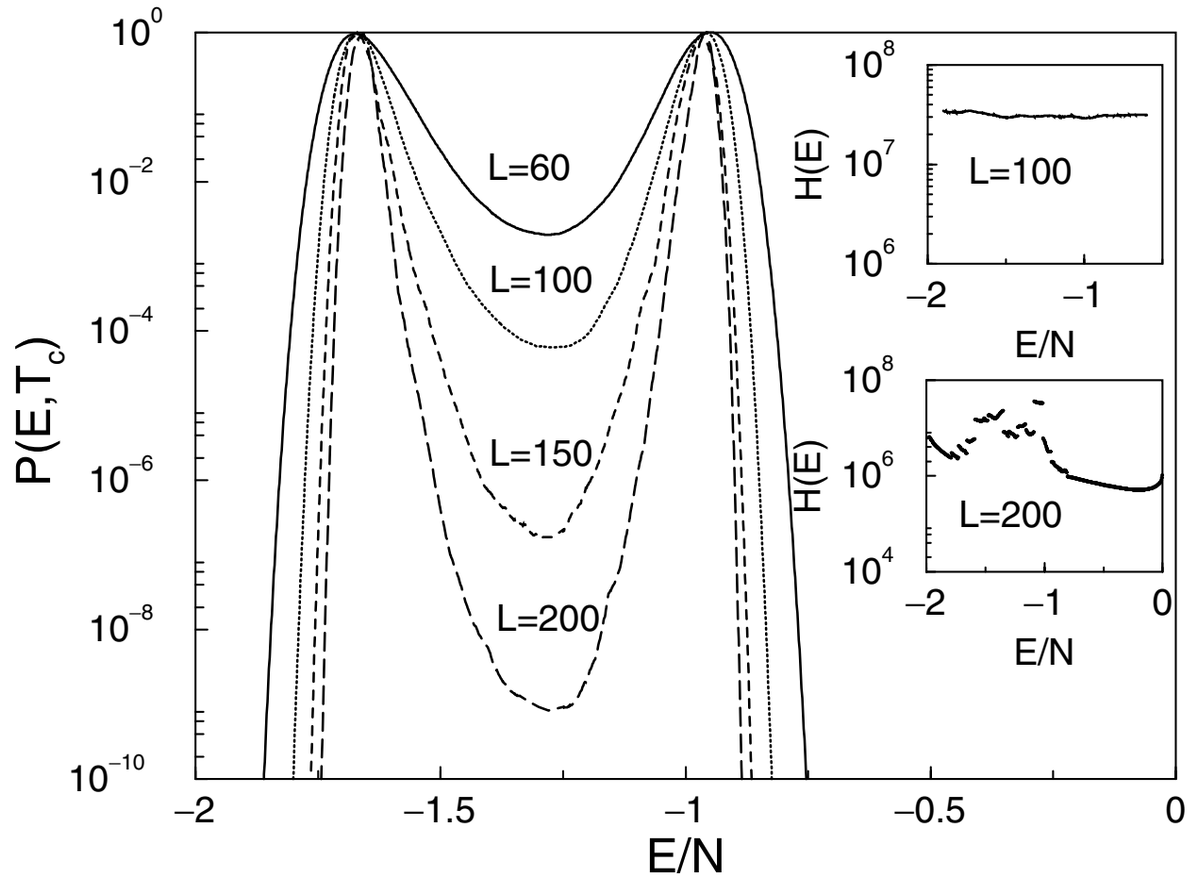
Power of Wang-Landau method

F. Wang and D. P. Landau, Phys. Rev. Lett. **86**, 2050 (2001)

Density of state of 2D-Ising model



Density of state of q=10 Potts model



We can obtain very accurate density of state by Wang-Landau method!

Replica Exchange method

K.Hukushim and K. Nemoto, J. Phys. Soc. Jpn. **65**, 1604 (1996).

Replica exchange (parallel tempering)

A different type of extended ensemble:

Usual MC or MD considers one parameter and one realization:

$$T,\ \Gamma = \{S_i\}, \{\boldsymbol{q}_i, \boldsymbol{p}_i\}$$

Replica exchange method considers
multiple parameters together with multiple realizations:

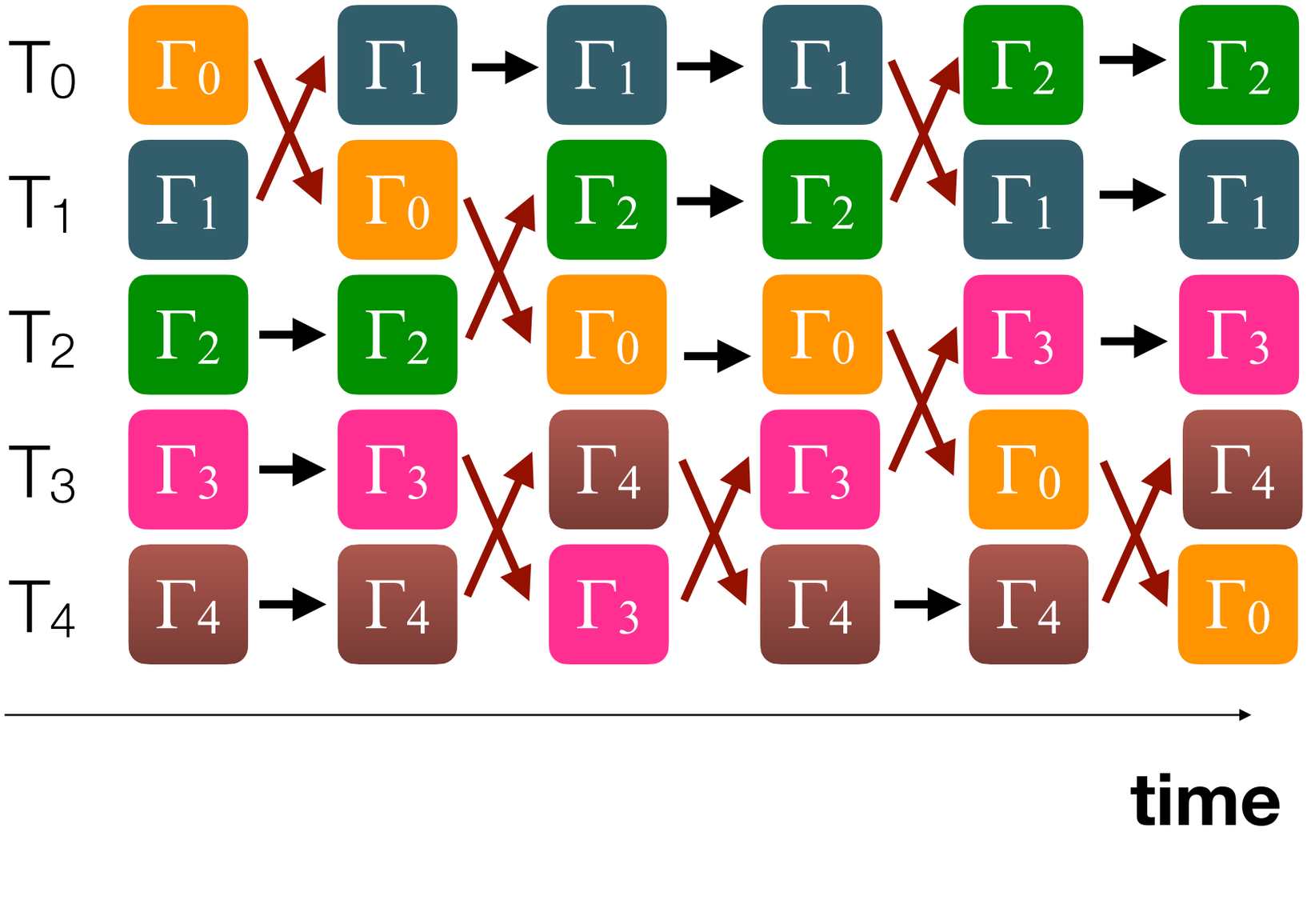
$$\{T_0, T_1, \cdots, T_M\},\ \{\Gamma_0, \Gamma_1, \cdots, \Gamma_M\},$$

Try to sample "(M+1)-dimensional" joint-distribution

$$P(\Gamma_0, \Gamma_1, \cdots, \Gamma_M; T_0, T_1, \cdots, T_M)$$

“Replica exchange”

Along simulation, we “exchange” the relationship between parameter and realization

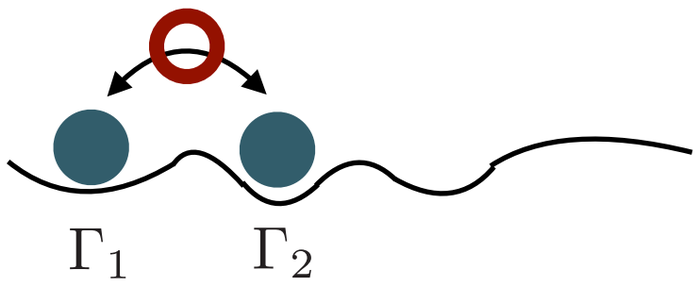


Purpose of replica exchange

Free energy landscape depends on the parameter

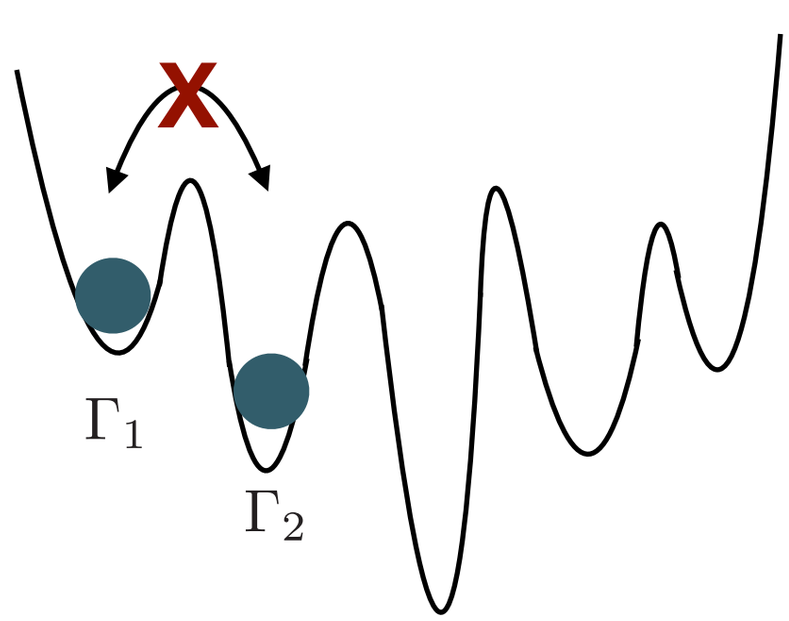
High temperature:T_h

Γ easily moves to other points!



Low temperature:T_l

Γ hardly moves to other minima!



Make a pass like:

$$\{\Gamma_1, T_l\} \rightarrow \{\Gamma_1, T_h\} \rightarrow \{\Gamma_2, T_h\} \rightarrow \{\Gamma_2, T_l\}$$

low **high** **high** **low**

* Parameter is not necessary a temperature.

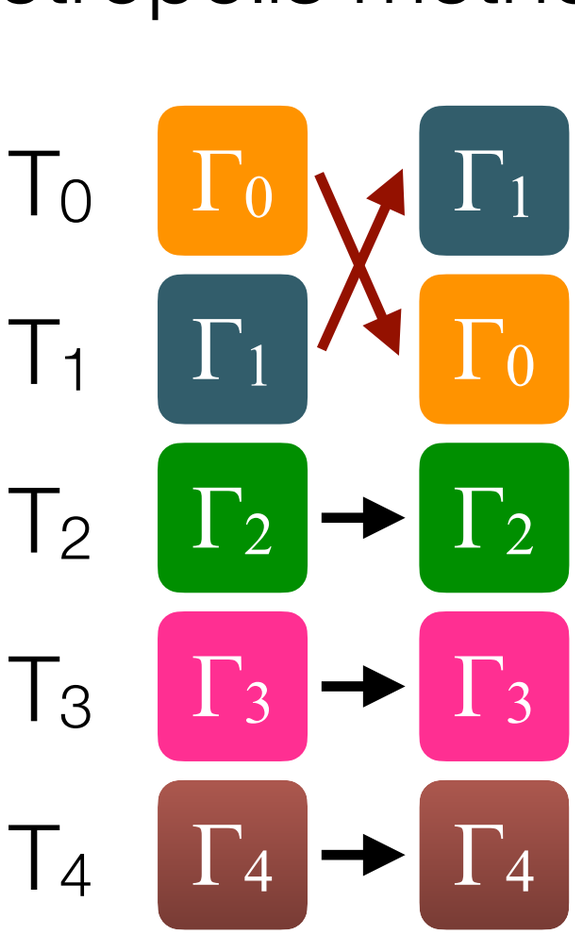
Markov Chain Monte Carlo for Replica Exchange

Target steady state distribution:

$$P(\Gamma_0, \Gamma_1, \cdots, \Gamma_M; T_0, T_1, \cdots, T_M) \propto e^{-\sum_i^M \beta_i E_i}$$

$$E_i \equiv \mathcal{H}(\Gamma_i)$$

Metropolis method:



$\mathcal{T}$ **:sequence of temperatures**

$$\mathcal{T} = \{T_1, T_0, T_2, \cdots\}$$

$$\{T_0, \Gamma_0\}, \{T_1, \Gamma_1\} \to \{T_1, \Gamma_0\}, \{T_0, \Gamma_1\}$$

$$\mathcal{T}_{01} \qquad\qquad \mathcal{T}_{10}$$

Transition probability

$$W_{\mathcal{T}_{01} \to \mathcal{T}_{10}} = \min\left(1, \frac{P(\{\Gamma_i\}; \mathcal{T}_{10})}{P(\{\Gamma_i\}; \mathcal{T}_{01})}\right)$$

$$\frac{P(\{\Gamma_i\}; \mathcal{T}_{10})}{P(\{\Gamma_i\}; \mathcal{T}_{01})} = \frac{e^{-\beta_1 E_0 - \beta_0 E_1}}{e^{-\beta_0 E_0 - \beta_1 E_1}}$$

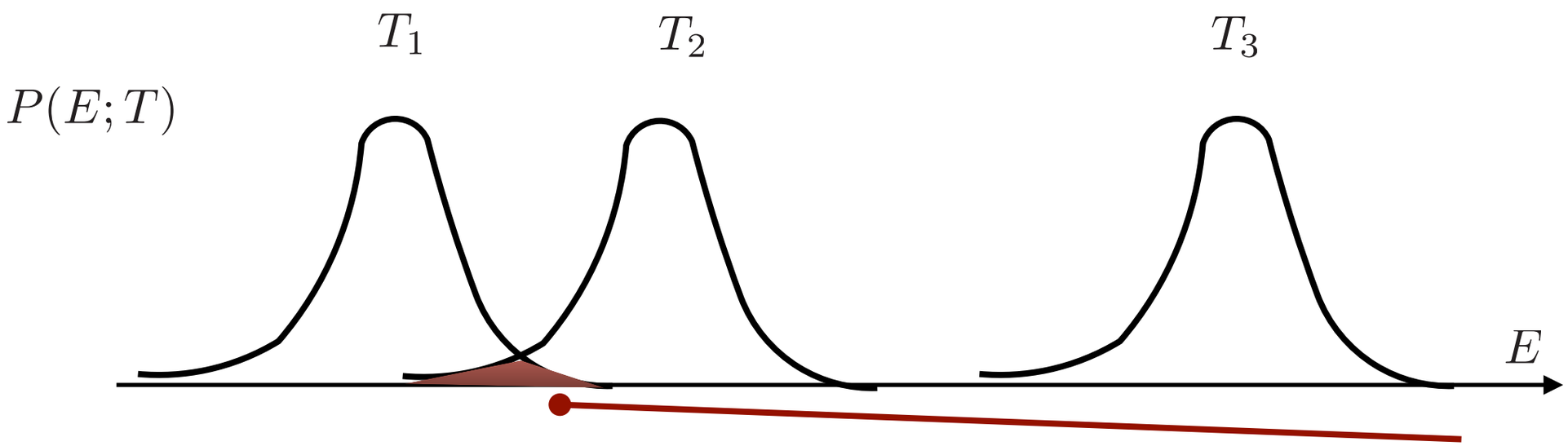
$$= e^{(\beta_0 - \beta_1)(E_0 - E_1)}$$

Select of temperature sequence

$$\frac{P(\{\Gamma_i\};\mathcal{T}_{10})}{P(\{\Gamma_i\};\mathcal{T}_{01})} = e^{(\beta_0-\beta_1)(E_0-E_1)} = \frac{P(E_1;T_0)P(E_0;T_1)}{P(E_0;T_0)P(E_1;T_1)}$$

Energy distribution at T

$P(E;T)$



Almost all exchange occurs the energy region of “overlap”.

$\{\Gamma_1, T_1\}, \{\Gamma_2, T_2\} \rightarrow \{\Gamma_1, T_2\}, \{\Gamma_2, T_1\}$:acceptable!

$\{\Gamma_2, T_2\}, \{\Gamma_3, T_3\} \rightarrow \{\Gamma_2, T_3\}, \{\Gamma_3, T_2\}$:almost rejected!

For efficient exchange, we have to choose a sequence of temperatures so that the energy distributions have finite overlap!

Usually we only exchange the nearest neighbor pairs of temperatures

Select of temperature sequence: Example

Suppose $C = \dfrac{dE}{dT} = \text{const.}$

$$\frac{P(\{\Gamma_i\}; \mathcal{T}_{10})}{P(\{\Gamma_i\}; \mathcal{T}_{01})} = e^{(\beta_0 - \beta_1)(E_0 - E_1)}$$

➡ Temperature sequence satisfying almost "flat" transition probability

$$(\beta_i - \beta_{i+1})(E_i - E_{i+1}) = \text{const.}$$

$$\Leftrightarrow \quad C\frac{(T_{i+1} - T_i)^2}{T_{i+1}T_i} = \text{const.}$$

$$\Rightarrow \quad T_{i+1} = \alpha T_i \quad \textbf{:Temperatures are geometric sequence!}$$

$$\alpha \sim 1 + O(1/\sqrt{C})$$

Important notice:

Heat capacity C is an extensive quantity: $C \sim O(N)$

➡ In order to keep finite overwrap, we need to increase temperature point M as

$$M \propto \sqrt{N} \qquad (T_{max} = T_M = \alpha^M T_{min})$$

Relaxation time of the replica exchange

In order to confirm the equilibration of the whole system,
usually we need two criterions.

1. The correlation time at the highest temperature is sufficiently short,
   e.g. $\tau=O(1)$

   → If a replica visits the highest temperature, it can easily change its state Γ.

2. All replicas make several (~O(10)) round trips between
   the lowest and the highest temperatures

   → The ensemble at the lower temperature is in the equilibrium.

The second part determines the relaxation time of the method.

$$\tau_{\mathrm{RE}} \sim \text{round trip time}$$

* If the replica exchange is an random walk:

$$\text{round trip time} \propto M^2$$

Summary of replica exchange

Algorithm:

1. Make a temperature set $\{T_1, T_2, \ldots, T_M\}$
2. Loop n
   (1) Do MC or MD for M replicas: $\{\Gamma_1, \Gamma_2, \ldots \Gamma_M; T_1, T_2, \ldots, T_M\}$
   (2) Calculate the energies of replicas
   (3) Try replica exchange based on, e.g. Metropolis method
   - Usually we alternatively try replica exchange such as
     even n; $\{1\leftrightarrow2\},\{3\leftrightarrow4\},\{5\leftrightarrow6\},\ldots$
     odd n; $\{2\leftrightarrow3\},\{4\leftrightarrow5\},\{6\leftrightarrow7\},\ldots$
     Note: each exchange trial is independent
   (4) Observe the quantities for $\{\Gamma_1, \Gamma_2, \ldots \Gamma_M; T_1, T_2, \ldots, T_M\}$

If we already have a MC or MD programs,
it is very easy to introduce the replica exchange method!

Tensor Renormalization Group

Outline

- Tensor renormalization group
  - Transfer matrix method
  - Tensor networks
    - Tensor network representation of a partition function
    - Other tensor networks
- Tensor renormalization group

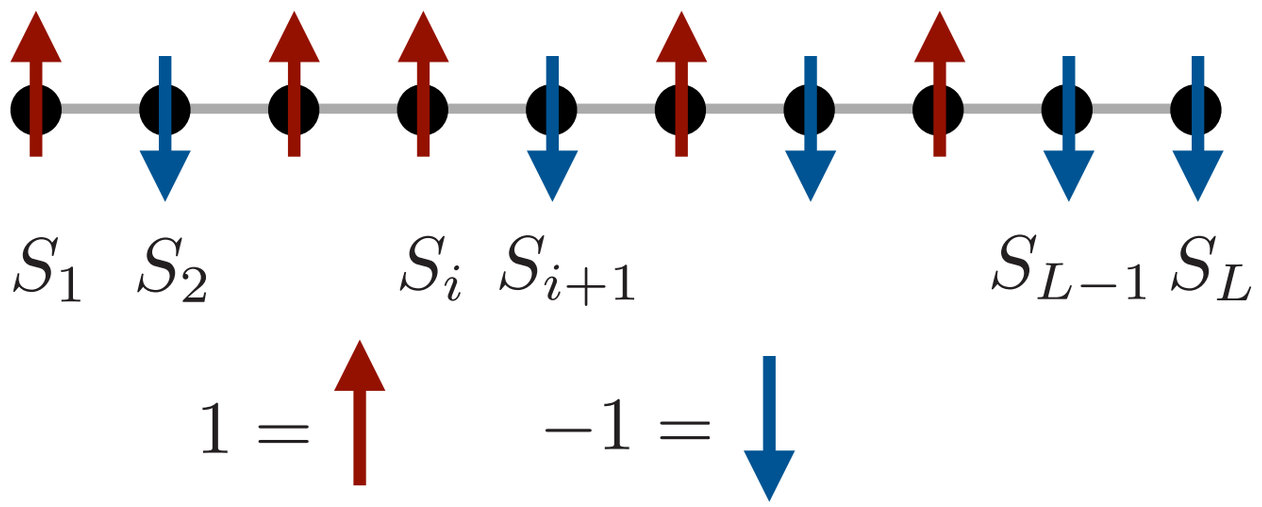
Transfer matrix method

Transfer matrix

Classical Ising model on a chain

$$\mathcal{H} = -J \sum_{i=1}^{L-1} S_i S_{i+1}$$

$$S_i = 1, -1$$



Partition function:

$$Z = \sum_{\{S_i = \pm 1\}} e^{\beta J \sum_i S_i S_{i+1}}$$

$$= \sum_{\{S_i = \pm 1\}} \prod_{i=1}^{L-1} e^{\beta J S_i S_{i+1}}$$

$$= \sum_{S_1 = \pm 1, S_L = \pm 1} \left(T^{L-1}\right)_{S_1, S_L}$$

Transfer matrix (転送行列)

$$T_{S_i, S_{i+1}} = e^{\beta J S_i S_{i+1}}$$

$$T = \begin{array}{c} \\ +1 \\ -1 \end{array} \begin{array}{c} \begin{array}{cc} +1 & -1 \end{array} \\ \begin{pmatrix} e^{\beta J} & e^{-\beta J} \\ e^{-\beta J} & e^{\beta J} \end{pmatrix} \end{array}$$

Diagonalization of the trasfer matrix

Transfer matrix

$$T = \begin{pmatrix} e^{\beta J} & e^{-\beta J} \\ e^{-\beta J} & e^{\beta J} \end{pmatrix}$$

Real symmetric matrix

→ Eigenvalues are real, and T is diagonalized by a orthogonal matrix

$$T = P^t \begin{pmatrix} \lambda_+ & 0 \\ 0 & \lambda_- \end{pmatrix} P$$

$$\lambda_+ = 2\cosh\beta J$$

$$\lambda_- = 2\sinh\beta J$$

$$|\lambda_+| > |\lambda_-|$$

$$P^t P = P P^t = I$$

Partition function

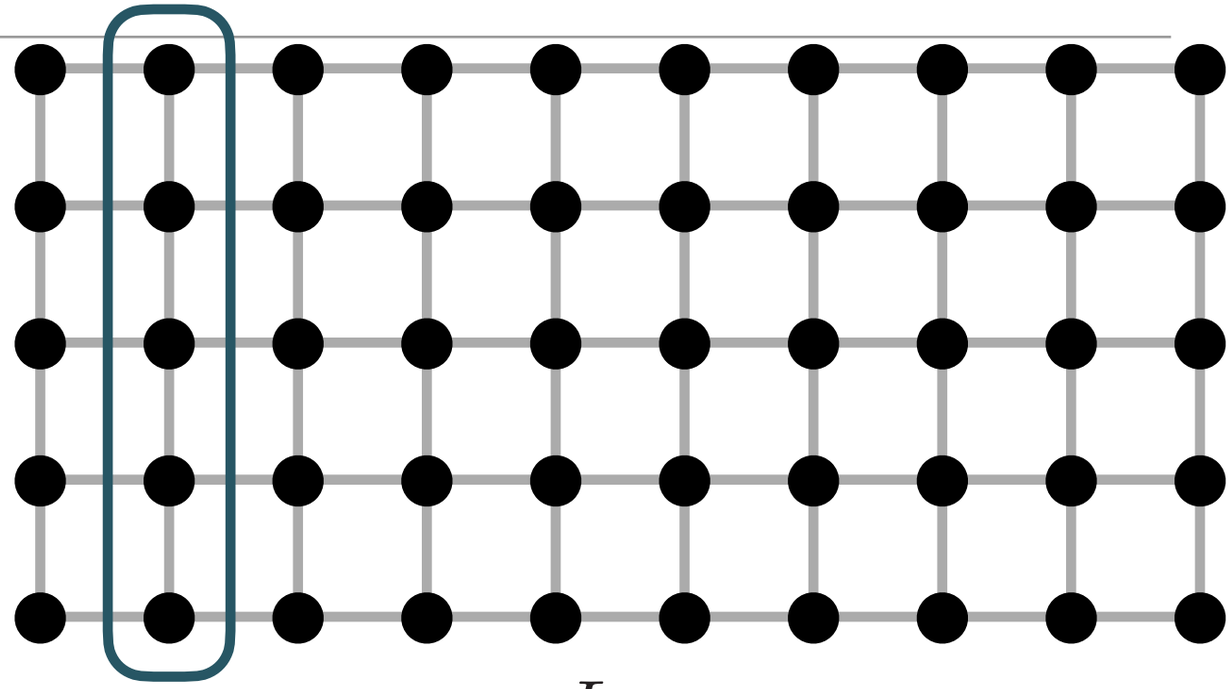
$$Z = \sum_{S_1=\pm1, S_L=\pm1} \left[P^t \begin{pmatrix} \lambda_+^{L-1} & 0 \\ 0 & \lambda_-^{L-1} \end{pmatrix} P \right]_{S_1, S_L}$$

Calculation of the partition function
= Diagonalization of the transfer matrix

Transfer matrix in 2d

$L \times M$ rectangular lattice

If we consider a set of M spin as one unit, the system is equivalent to a 1d.



Size of the transfer matrix

1d: 2×2

$L \times M$ 2d: $2^M \times 2^M$ (or $2^L \times 2^L$)

For higher dimensional systems, the size of the transfer matrix is exponentially large.

Exact calculation is limit to smaller systems.

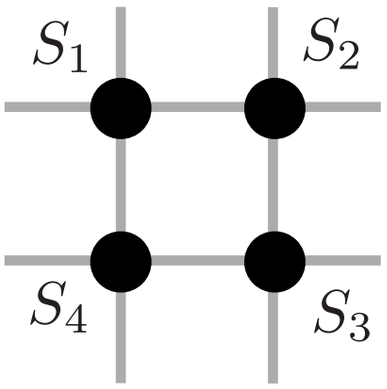
2d Ising model: $M < 50$.

Approximate calculation of the products of transfer matrices

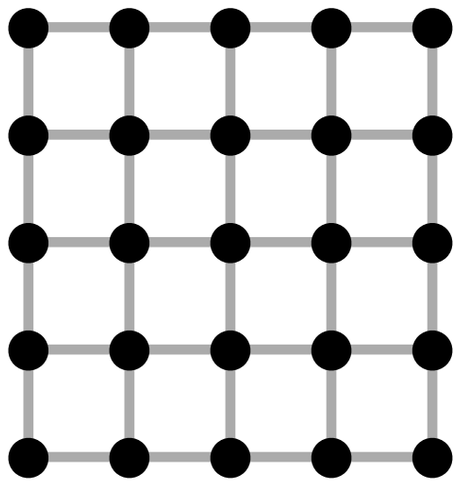
Generalization of the transfer matrix

Partition function

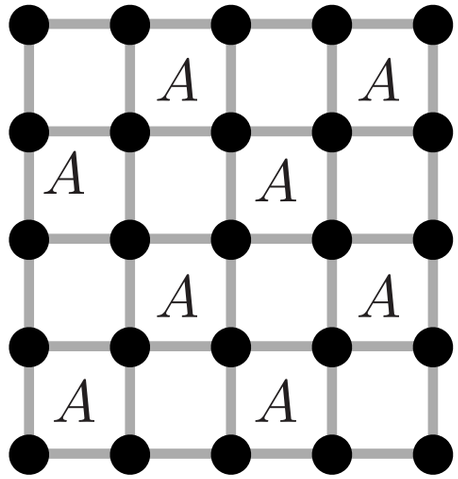
Considering a square unit



Square lattice



Arrangement of *A*



Product of weights on four edges: 4 -leg *tensor*

$$A_{S_1,S_2,S_3,S_4} = e^{\beta J(S_1S_2+S_2S_3+S_3S_4+S_4S_1)}$$

Ising model

A is a 2×2×2×2 tensor

Partition function is a “product” of tensors.

$$Z = \sum_{\{S_i=\pm 1\}} A_{S_1,S_2,S_3,S_4} A_{S_2,S_5,S_6,S_7} \cdots A_{S_i,S_j,S_k,S_l} \cdots$$

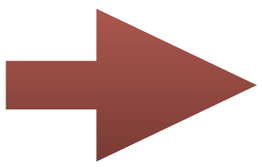
Tensor?

Scalar: c Number

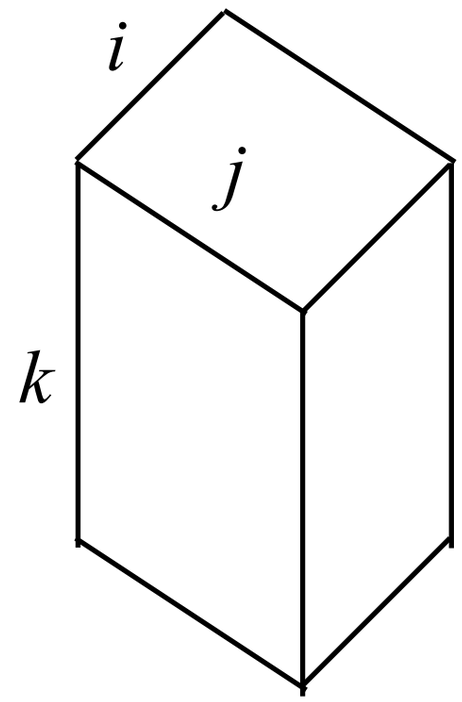
Vector: v_i One dimensional array of numbers

Matrix: M_{ij} Two dimensional array of numbers

Tensor: $T_{ijk\ldots}$ Higher dimensional array of numbers



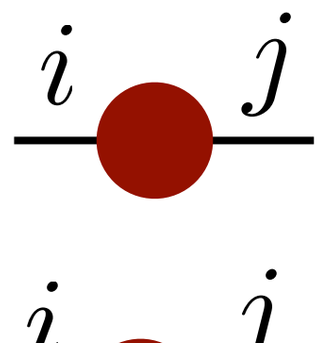
Scalar: 0-dim. tensor
Vector: 1-dim. tensor
Matrix: 2-dim. tensor



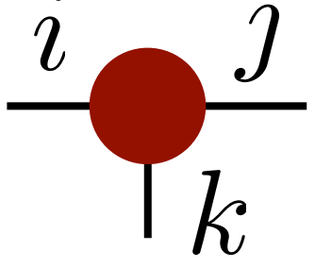
Graphical representations for tensor network

- Vector $\vec{v} : v_i$

- Matrix $M : M_{i,j}$

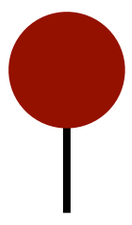


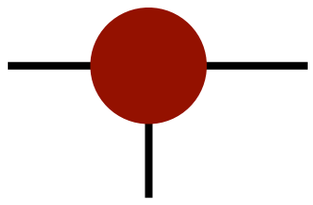
- Tensor $T : T_{i,j,k}$



*** n-rank tensor = n-leg object**

When indices are not presented in a graph, it represent a tensor itself.

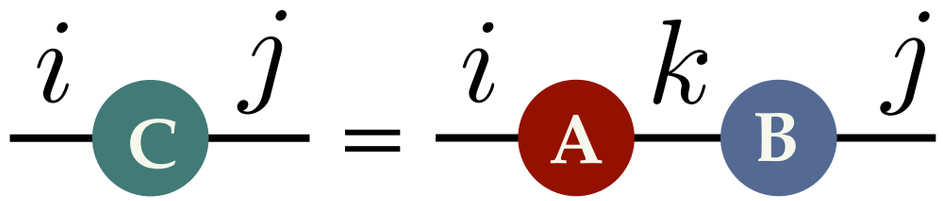
$\vec{v} =$ 

$T =$ 

Graphical representations for tensor network

Matrix product

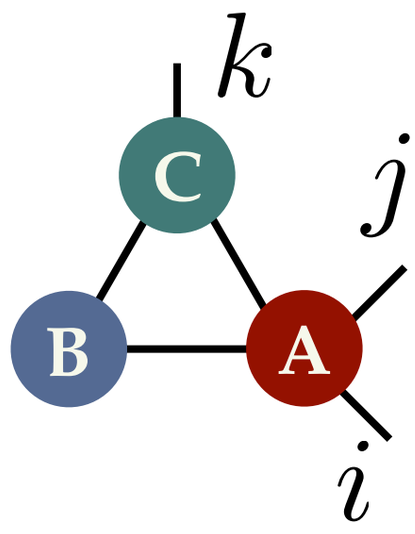
$$C_{i,j} = (AB)_{i,j} = \sum_k A_{i,k} B_{k,j}$$



$$C = AB$$

Generalization to tensors

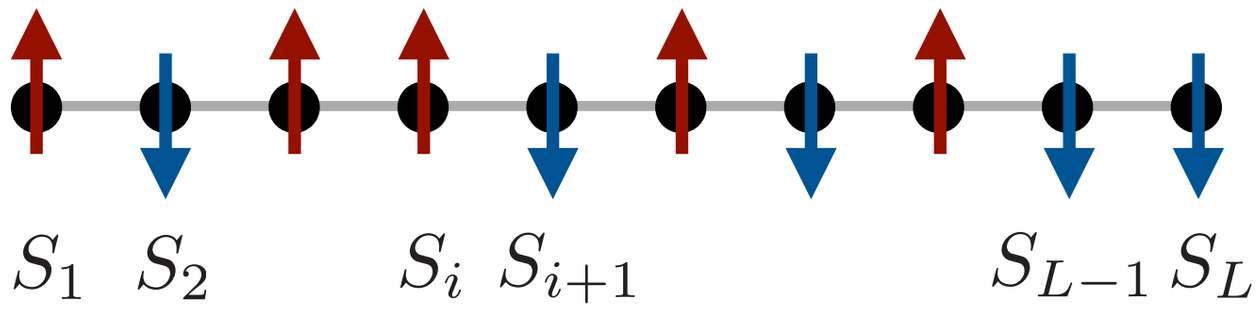
$$\sum_{\alpha,\beta,\gamma} A_{i,j,\alpha,\beta} B_{\beta,\gamma} C_{\gamma,k,\alpha}$$



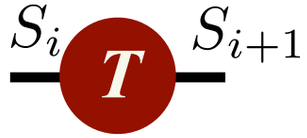
Contraction of a network = Calculation of a lot of multiplications

Diagram for the partition function

1d Ising model

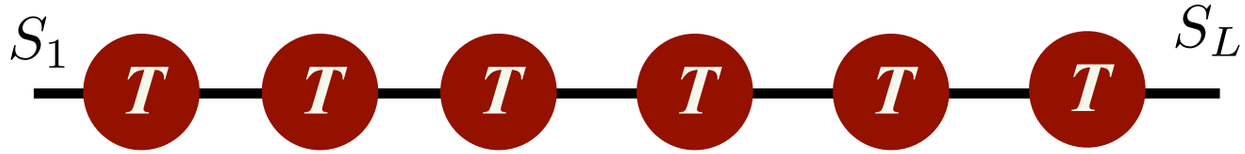


$$T_{S_i,S_{i+1}} = e^{\beta J S_i S_{i+1}}$$



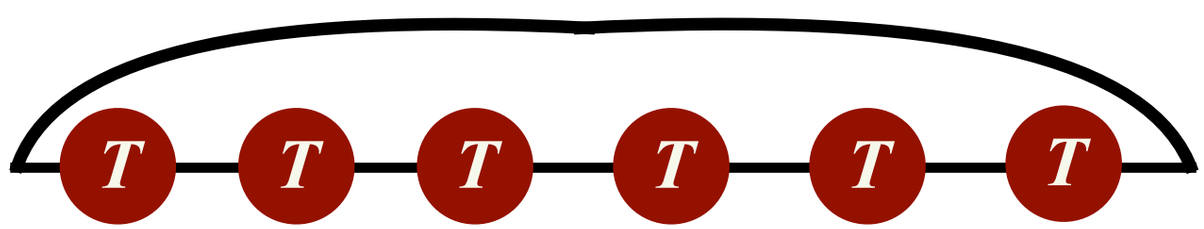
$$Z = \sum_{S_1=\pm1, S_L=\pm1} \left(T^{L-1}\right)_{S_1,S_L}$$

$$= \sum_{S_1=\pm1, S_L=\pm1}$$



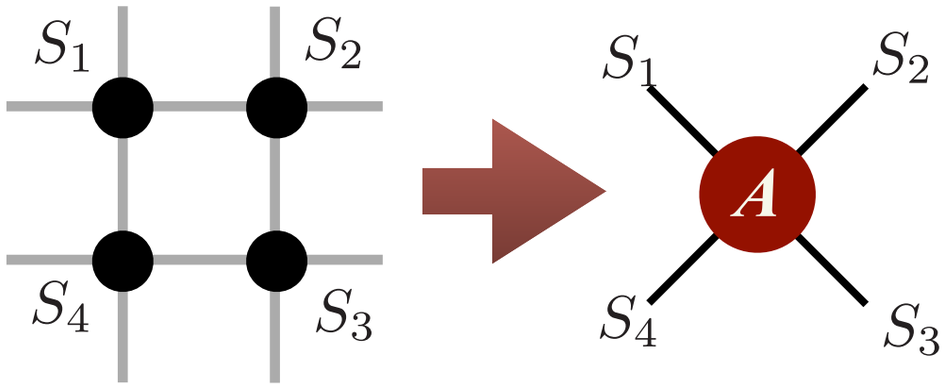
* Periodic boundary condition

$$Z = \mathrm{Tr}\ T^L =$$

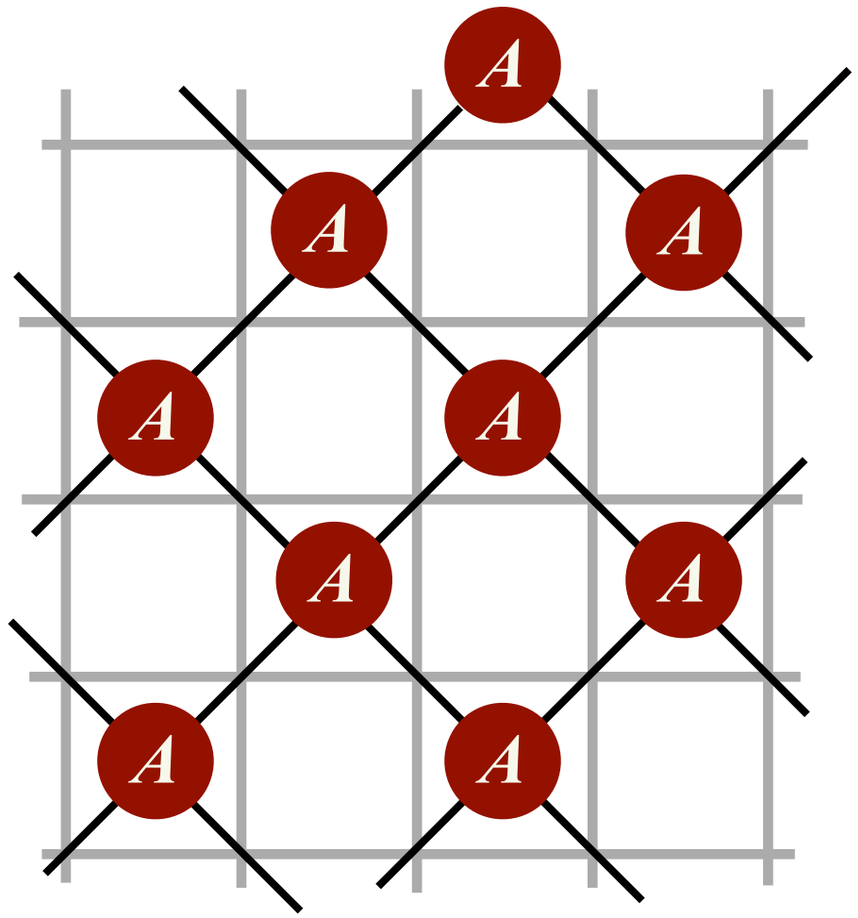


Tensor network representation in two dimension

$$e^{\beta J(S_1S_2+S_2S_3+S_3S_4+S_4S_1)} = A_{S_1S_2S_3S_4}$$



$$Z =$$



Partition function = Tensor network of tensor *A*

Square lattice Ising model→Square lattice tensor network rotating 45 degrees.

*We can construct a tensor network where tensors are on the nodes of original lattice.

Other tensor networks

Another tensor network: quantum cirquit

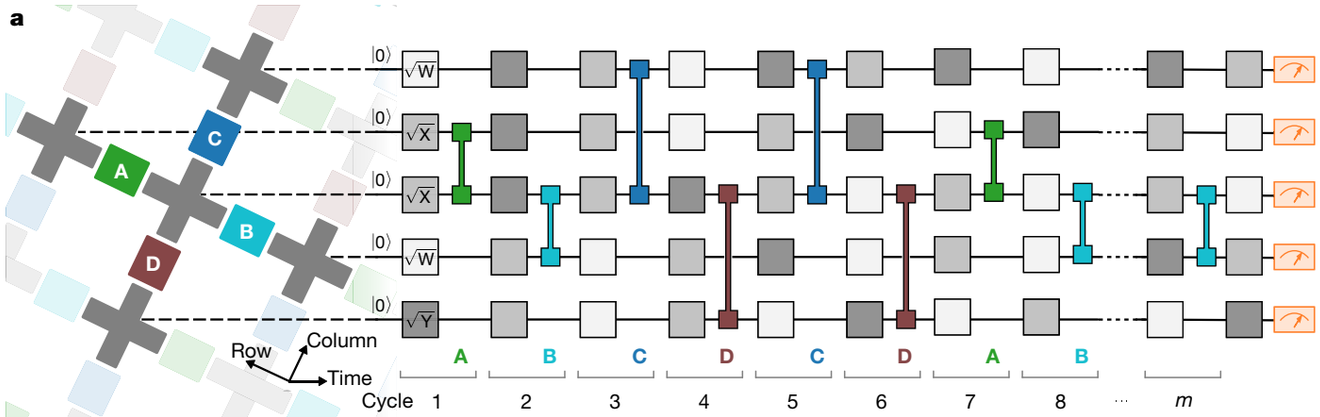
Quantum circuit:

Circuit based on gate operations on quantum bits

Unitary matrices (tensors)

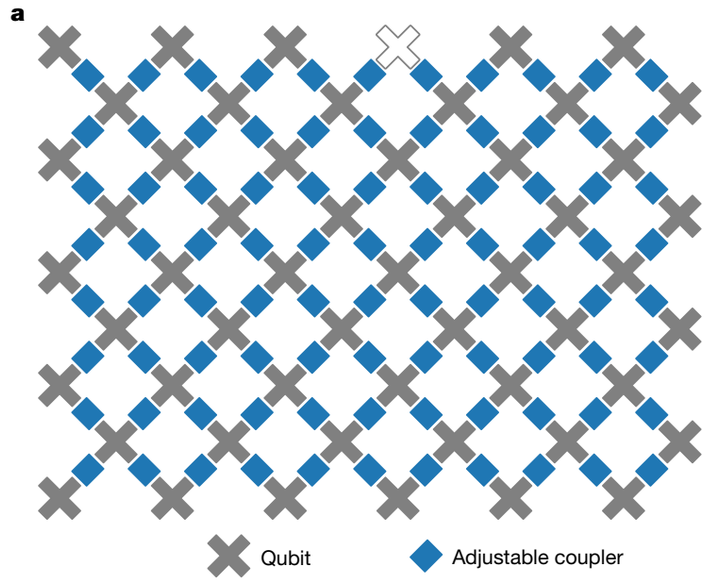
2-dim complex vectors

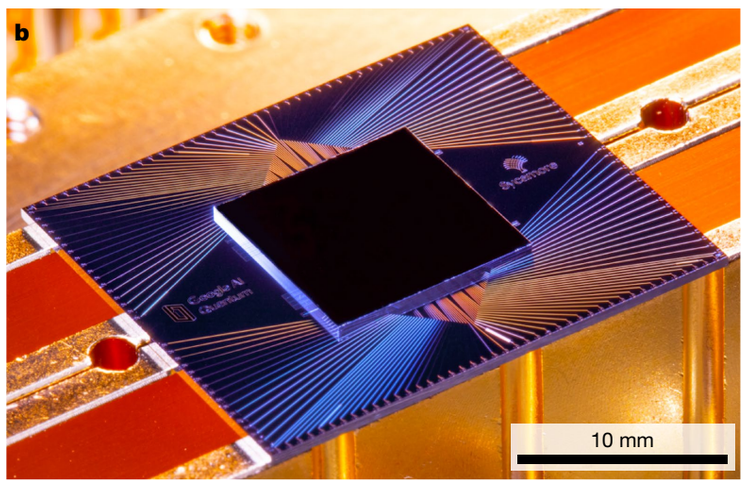
Google's "quantum supremacy" circuit



"quantum supremacy" circuit

F. Arute, *et al.*, Nature 574, 505 (2019)





Quantum circuit = tensor network

Classical simulation of quantum circuits
= contraction of tensor networks

Original estimation
10,000 years

TN based simulation
304 second !

Y. A. Liu, *et al.*, Gordon bell Prize in SC21 (2021),
(cf. quantum computer =200s)

Another tensor network: quantum state

quantum many-body state

$$|\Psi\rangle = \sum_{\{i_1, i_2, \ldots i_N\}} \Psi_{i_1 i_2 \ldots i_N} |i_1 i_2 \ldots i_N\rangle$$

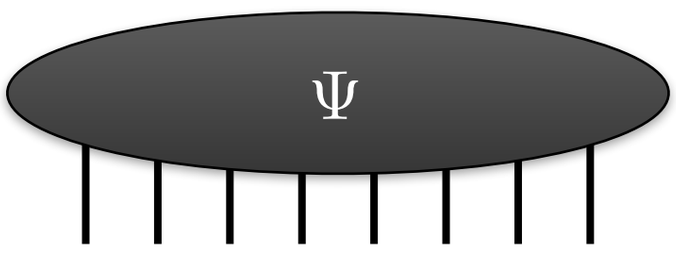
Basis

Quantum spin, qubits $i = \uparrow, \downarrow = |0\rangle, |1\rangle$

$$|010100\cdots 0\rangle = |0\rangle \otimes |1\rangle \otimes \cdots$$

Coefficient is a tensor!

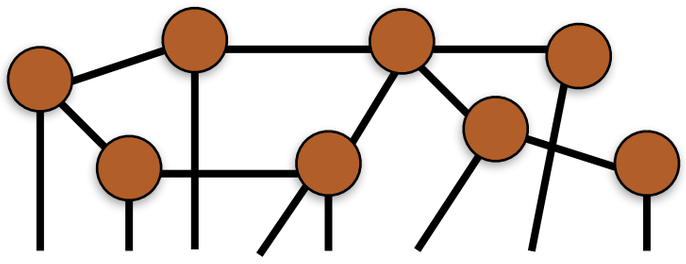
quantum state



$\sim e^N$ independent elements

Approximation based on the entanglement

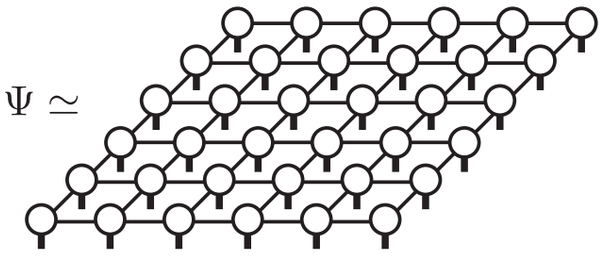
Tensor network decomposition



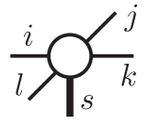
$\sim O(N)$ independent elements

PEPS (for 2d system)

$\Psi \simeq$



$T_{ijkl}[s] =$



Low energy quantum states:

- Lower quantum correlation (entanglement) than general (random) state
  - c.f. Area law of the entanglement entropy

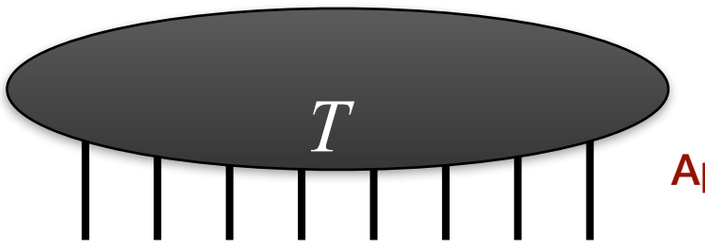
→ Accurate approximation by tensor network

Another tensor network: tensor data

Arbirary tensor data

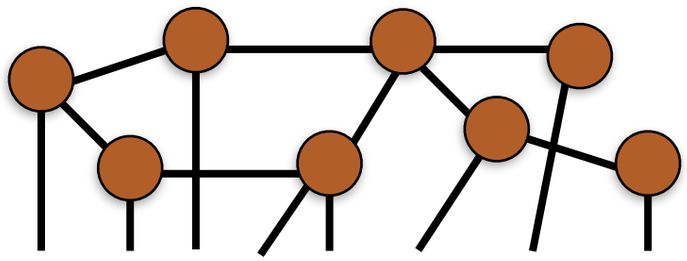
$T_{i_1,i_2,\ldots,i_N}$: decomposition similar to quantum states

Tensor data



Approximation based on information correlation

tensor network decomposition



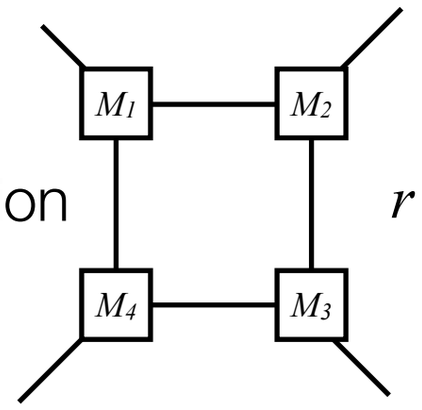
Ex1. dataset of images

(Q. Zhao, et al arXiv:1606.05535)

COIL-100 dataset = 32 x 32 x 3 x 7200 tensor

pixel color # of images

Tensor ring decomposition



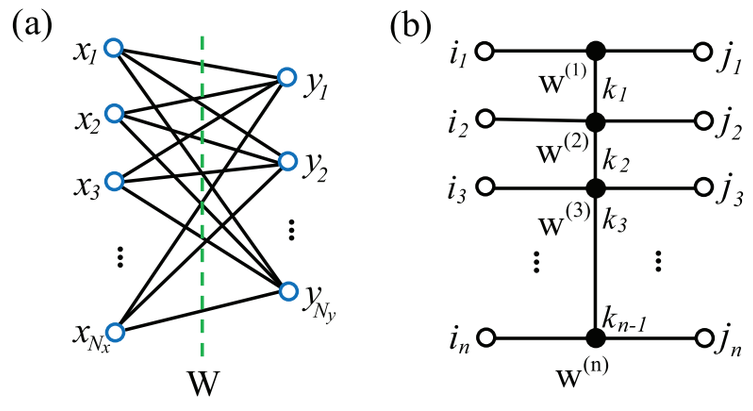
Ex2. Weight matrix in neural network

(Z.-F. Gao et al, Phys. Rev. Research **2**, 023300 (2020).)

x_i: input neuron (pixel)

y_i: output neuron

W_{ij}: weight matrix connecting x and y



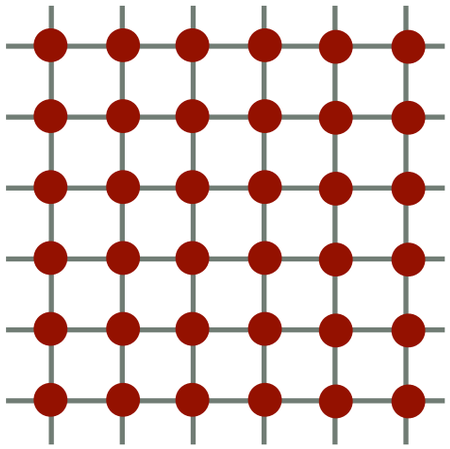
Tensor renormalization group

Tensor renormalization group
（テンソル繰り込み）

- Approximate calculation of a tensor network contraction by using "coarse graining" （粗視化）of the network
    - Coarse graining ⟷ Real space renormalization
    - （粗視化）⟷（実空間繰り込み）
- It can be applicable to (basically) any lattices, and the idea (algorithm) is independent on "models" represented by tensor networks.
    - Potential application to wide range of the science.

Outline of tensor renormalization

Scaler represented
by $L \times L$ tensors

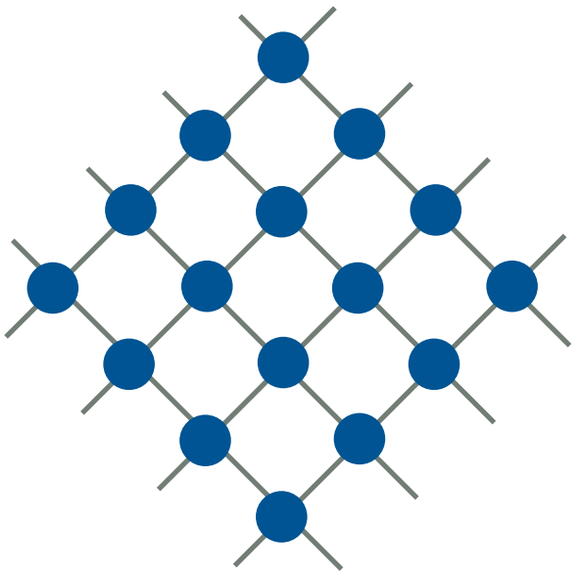


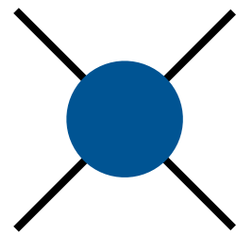
A: $D \times D \times D \times D$

Corse graining (Renormalization)
into $\sqrt{2}$ times longer scale.

Approximation

$(L \times L)/2$ tensors





$\tilde{A}$: $D \times D \times D \times D$

Reduce the number of tensors
keeping their size constant

Preliminary: low-rank approximation of a matix

Rank of a matrix

Maximum # of linearly independent row (or column) vectors in A

A: *N*×*M* → $\mathrm{rank}(A) \leq \min(N, M)$

Low-rank approximation:

Approximation of A by a lower-rank matrix $\mathrm{rank}(\tilde{A}) = R < \mathrm{rank}(A)$

Keep only the important information → data compression

Accuracy of the approximation

$$\epsilon = ||A - \tilde{A}|| \qquad ||X|| \equiv \sqrt{\sum_{i,j} X_{ij}^2}$$

→ $\min_{\tilde{A}_{ij};\mathrm{rank}\tilde{A}=R} ||A - \tilde{A}||$

Optimal!

An optimal low-rank approximation can be obtained by Singular Value Decomposition (SVD)

Preliminary: singular value decomposition

Singular value decomposition (SVD)

Any matrices can be decomposed as

$$A_{i,j} = \sum_{k=1}^{\min(N,M)} U_{ik} \lambda_k V^*_{jk}$$

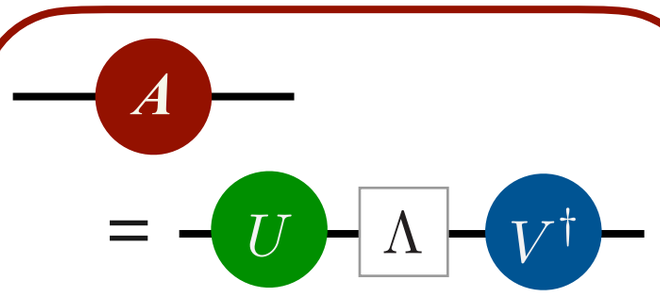
λ_k: real, non-negative $\lambda_k \geq 0$

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \cdots$$

rank(A) = # of non-zero singular values

U_{ik}, V^*_{jk} (half) Unitary matrices

$$\sum_i U_{ik} U^*_{il} = \delta_{kl} \quad \sum_i V_{jk} V^*_{jl} = \delta_{kl}$$



$$A = U \Lambda V^\dagger$$

Λ Diagonal matrix with singular values

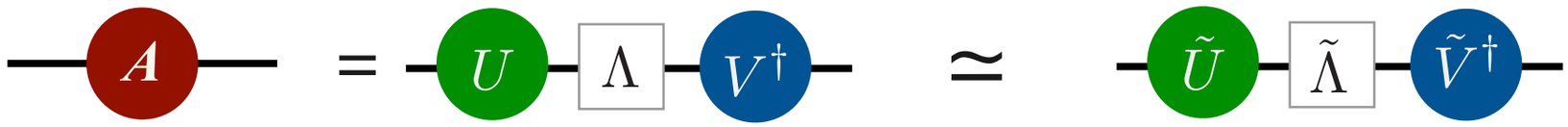
$$U^\dagger U = I$$

$$V V^\dagger = I$$

Optimal *R*-rank approximation of A → Keep the largest *R* singular values, and neglect (replace to zero) the remaining ones.

Key technique: low rank approximation by SVD

Best low-rank approximation of a matrix = SVD



$A : M \times N$ (M ≦ N)

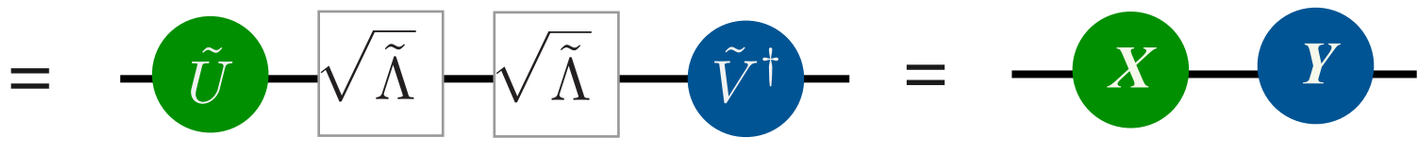
$\Lambda : M \times M$ (Diagonal matrix)

$U, V : (\mathrm{M},\mathrm{N}) \times M$

$\tilde{\Lambda} : R \times R$ (Keeping the R largest singular values)

$\tilde{U}, \tilde{V} : (\mathrm{M},\mathrm{N}) \times R$

In addition,



$\sqrt{\tilde{\Lambda}}$:Diagonal matrix those elements are $\sqrt{\lambda}$

$X = \tilde{U}\sqrt{\tilde{\Lambda}} \quad : \mathrm{M} \times R$

$Y = \sqrt{\tilde{\Lambda}}\tilde{V}^{\dagger} \quad : \mathrm{R} \times M$

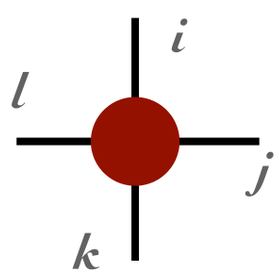
By SVD, we can decompose a matrix into a product of "small" matrices.

Recipe of Tensor Renormalization Group (TRG)

M. Levin and C. P. Nave, Phys. Rev. Lett. **99**, 120601 (2007)
Z.-C. Gu, M. Levin and X.-G. Wen, Phys. Rev. B **78**, 205116 (2008)

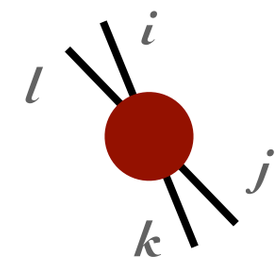
1. Decomposition

Regard a tensor as a matrix



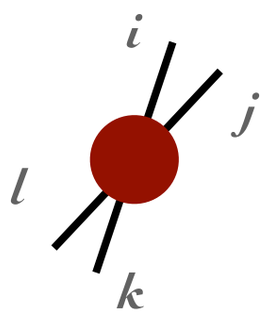
$A_{i,j,k,l}$

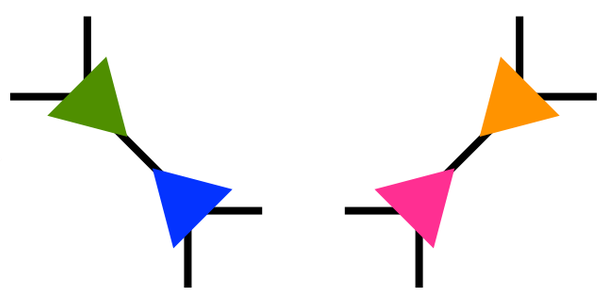
A: $D \times D \times D \times D$



$A_{(i,l),(j,k)}$

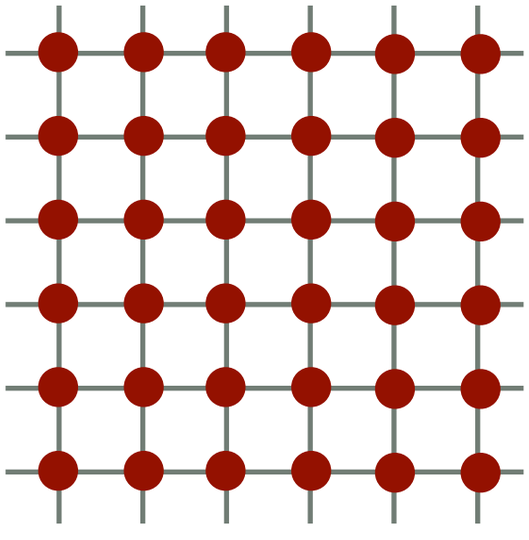
D-rank approximation by SVD



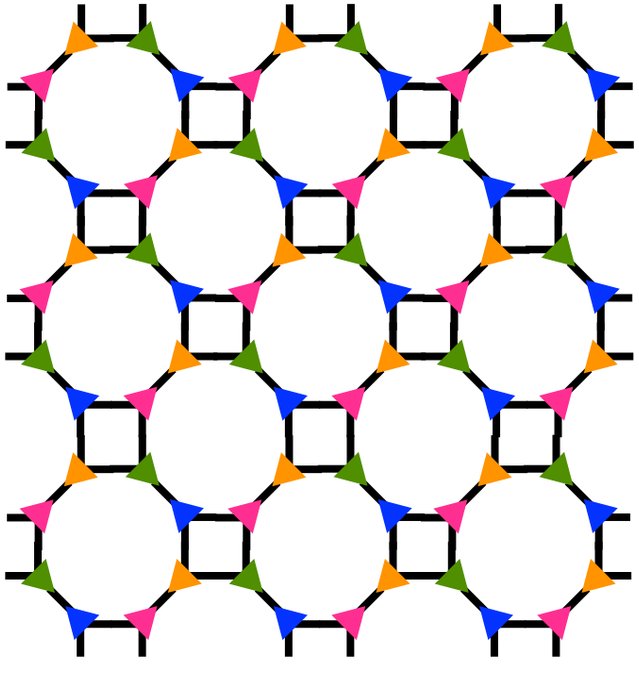


$A_{(i,j),(k,l)}$

$A : D^2 \times D^2$



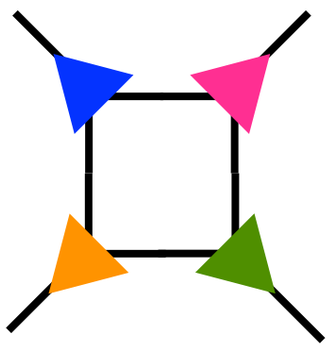
Approximation



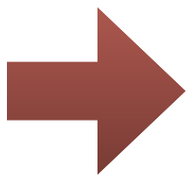
Recipe of Tensor Renormalization Group (TRG)

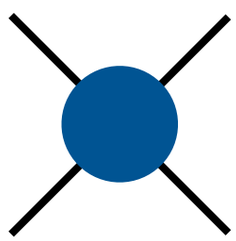
M. Levin and C. P. Nave, Phys. Rev. Lett. **99**, 120601 (2007)
Z.-C. Gu, M. Levin and X.-G. Wen, Phys. Rev. B **78**, 205116 (2008)

2. Coarse graining



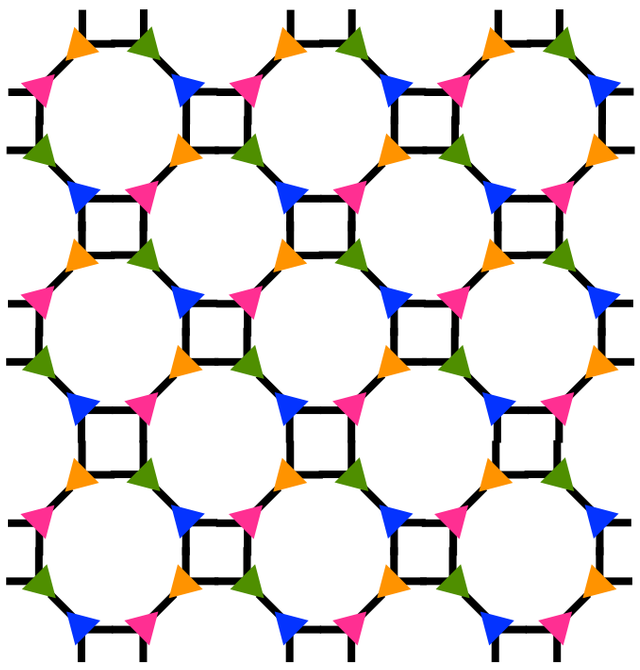
Contraction of inner indices

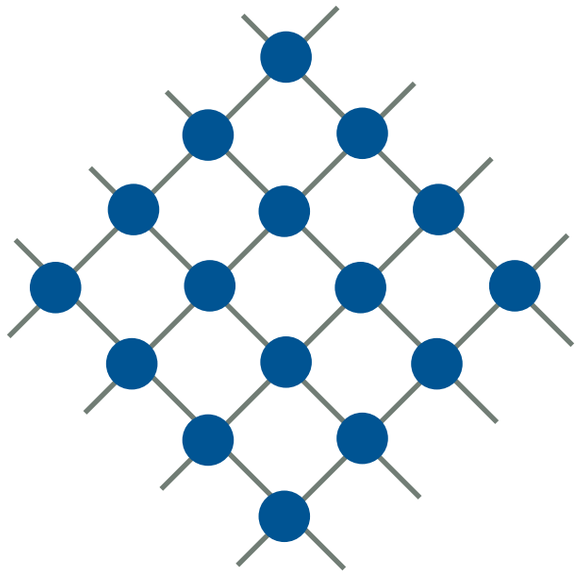




$$\tilde{A} : D \times D \times D \times D$$

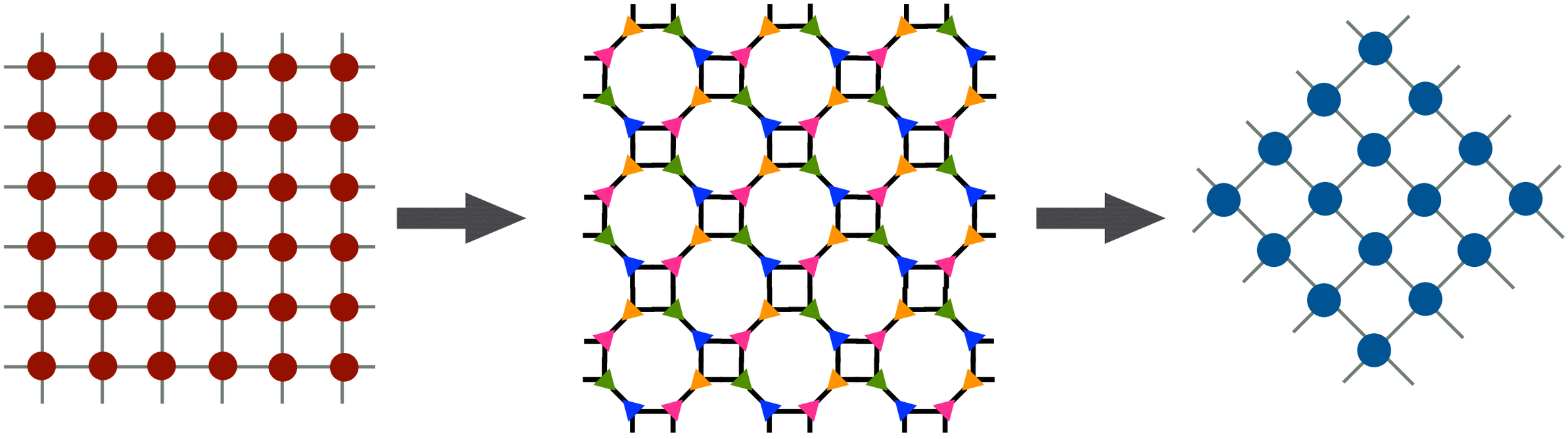
In total, two original tensors are coarse grained into a new tensor.





Recipe of Tensor Renormalization Group (TRG)

M. Levin and C. P. Nave, Phys. Rev. Lett. **99**, 120601 (2007)
Z.-C. Gu, M. Levin and X.-G. Wen, Phys. Rev. B **78**, 205116 (2008)



Calculation cost: (per tensor)

SVD= $O(D^6)$

Contraction= $O(D^6)$

*Indeed, they can be reduced to $O(D^5)$ if we do not calculate all singular values.

*By one TRG step, # of tensors is reduced by 1/2.

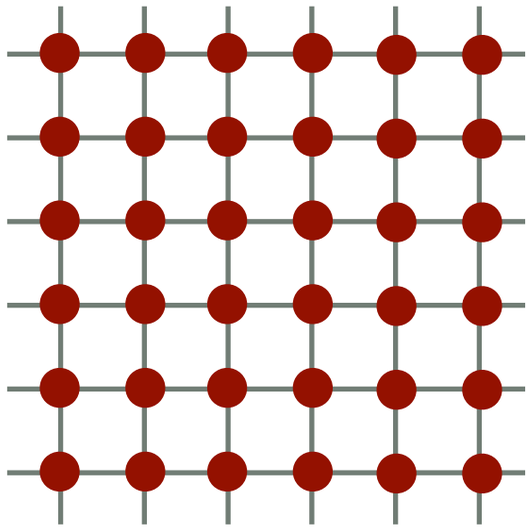
We can calculate the contraction in polynomial cost!

Application to a classical partition function

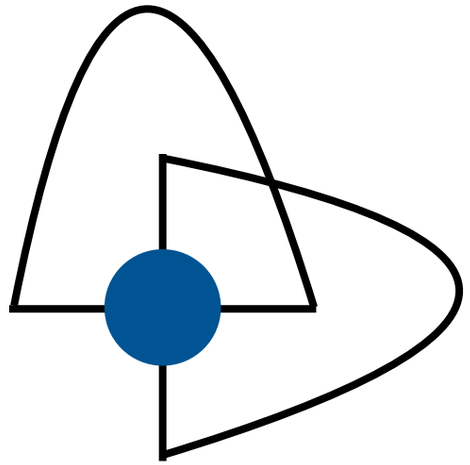
Partition function

(Periodic boundary condition)

$Z =$



Repeat TRG step until only a few tensors remain.



We can easily calculate physical quantities from Z.

Free energy: $F = -k_B T \ln Z$

Energy: $E = -\dfrac{\partial \ln Z}{\partial \beta}$

Specific heat: $C = \dfrac{1}{k_B T^2} \dfrac{\partial^2 \ln Z}{\partial \beta^2}$

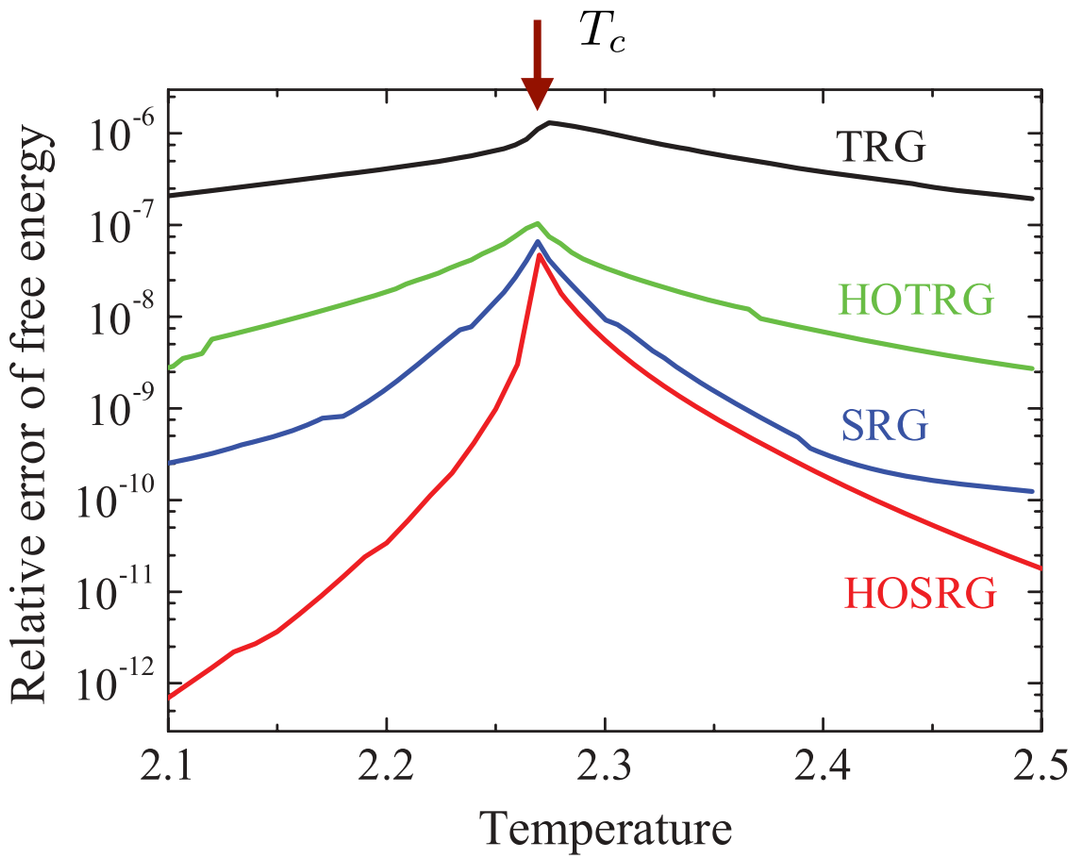
Example of calculation

Ising model in infinite size

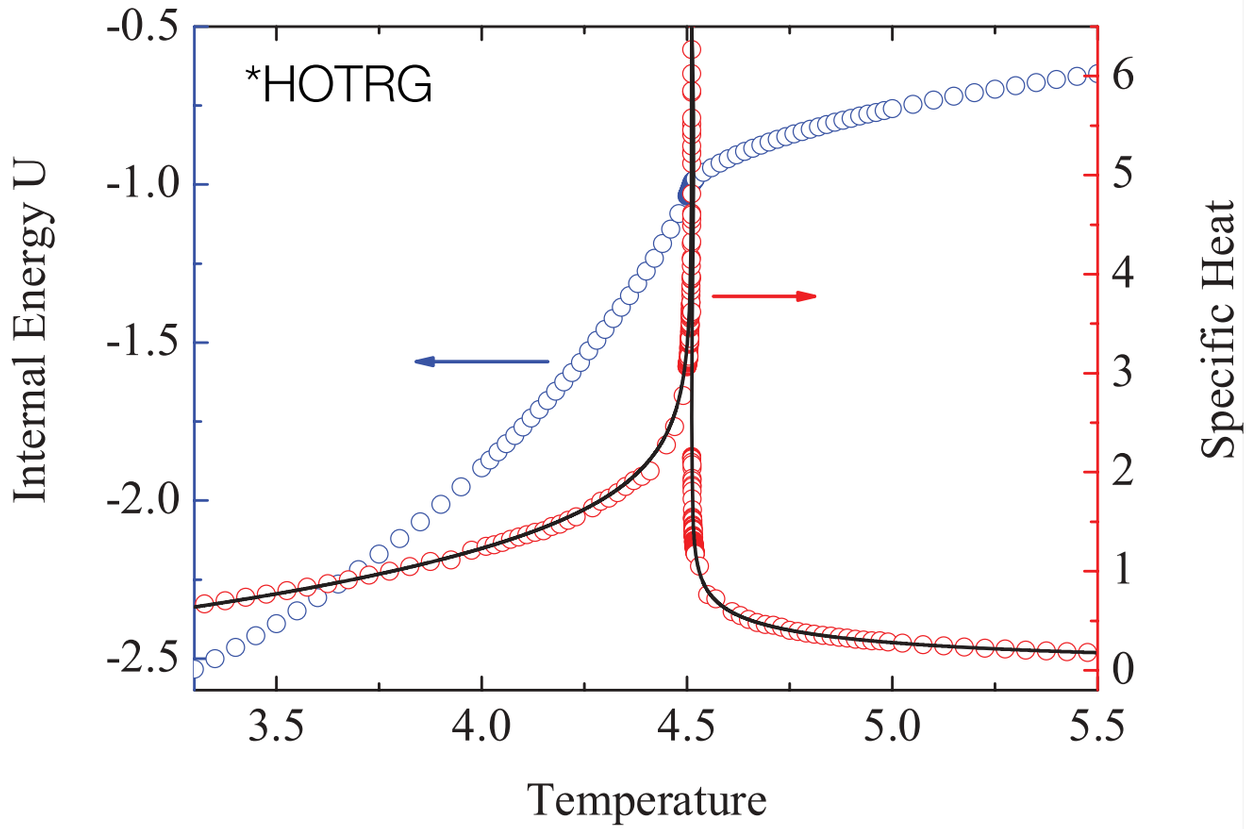
$$\mathcal{H} = -J \sum_{\langle i,j \rangle} S_i S_j$$

Z. Y. Xie *et al*, Phys. Rev. B **86**, 045139 (2012)

Error of free energy for 2D Ising model



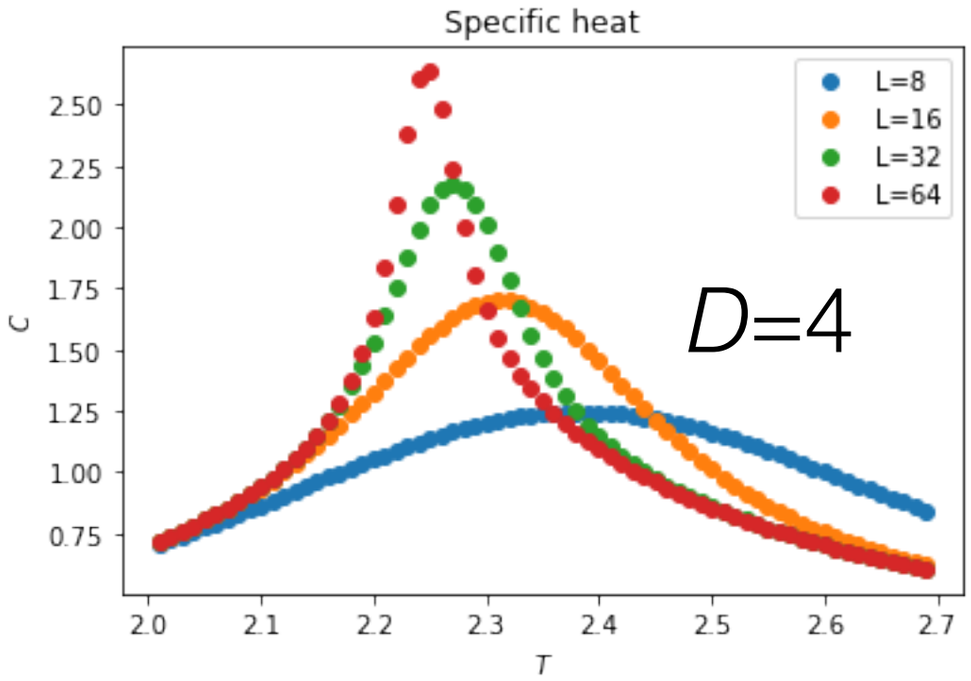
Energy and specific heat of 3D Ising model

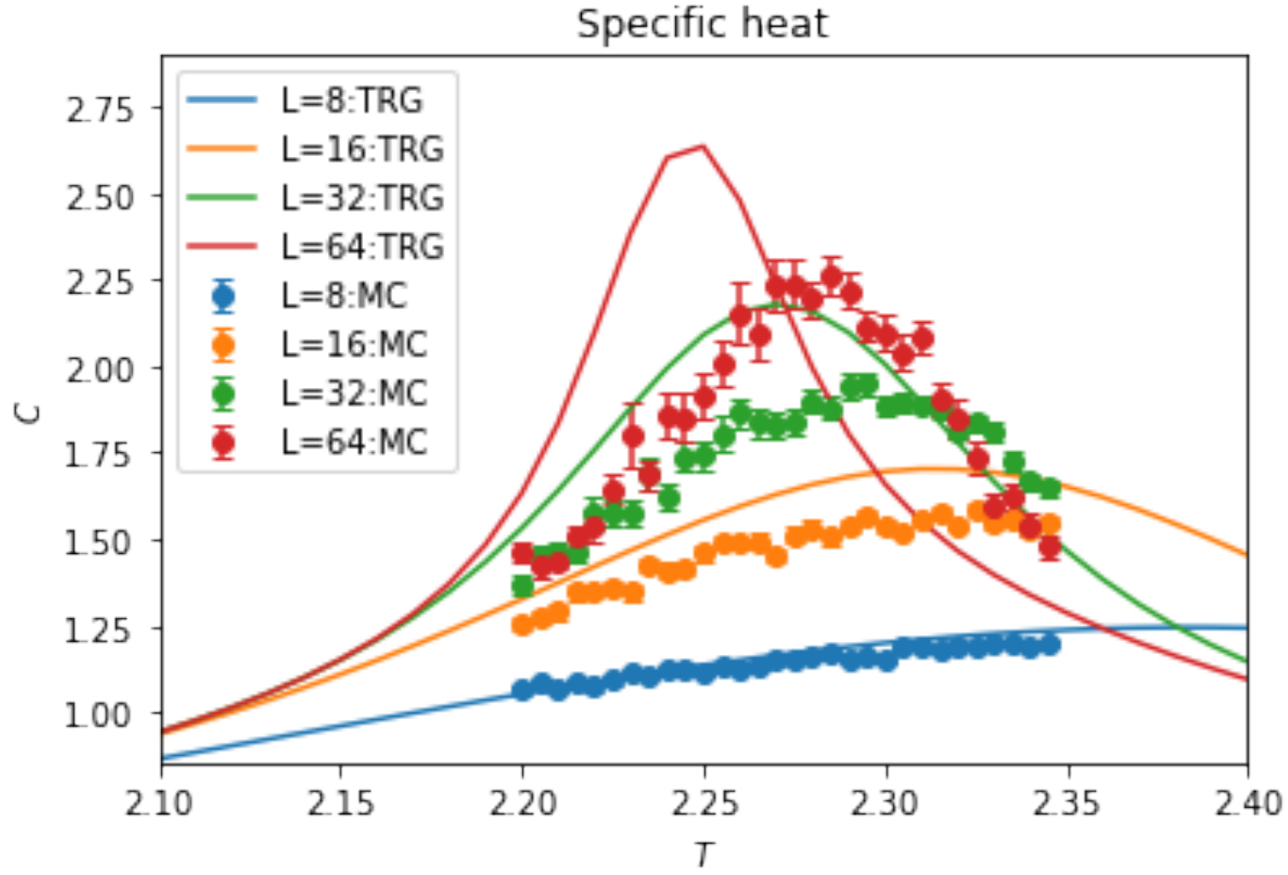


$$T_c/J = \frac{2}{\ln(1+\sqrt{2})} \simeq 2.269$$

Comparison with MCMC in finite sizes

Finite-size 2d Ising model



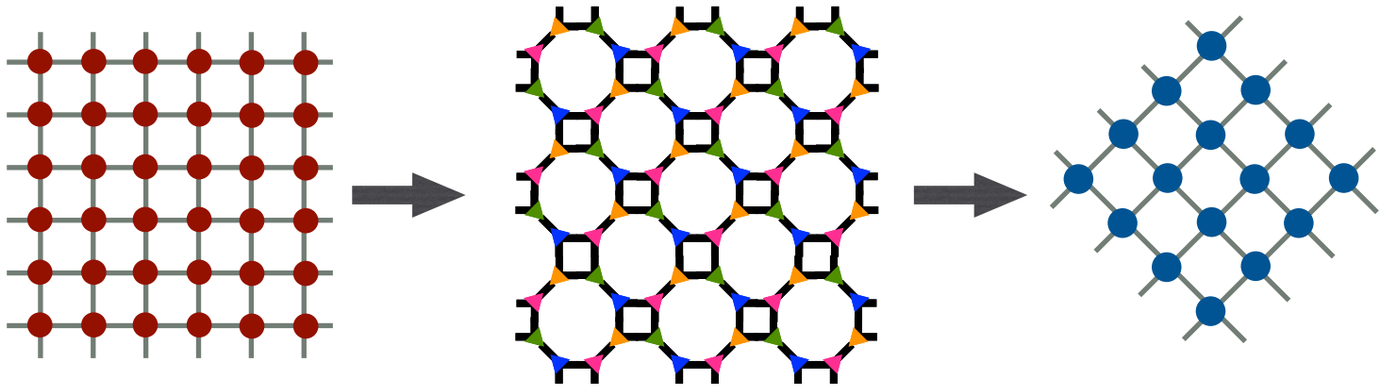


Relatively smooth data even D=4

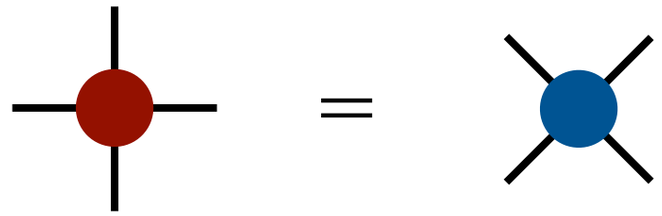
There is a strong systematic error.

Critical phenomena and tensor renormalization

By TRG-type algorithms, we can also calculate critical exponents.



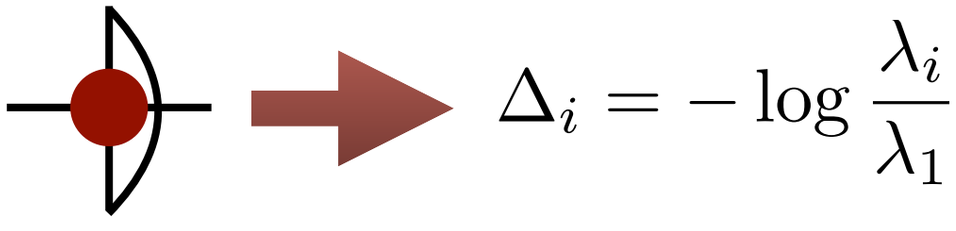
Fixed point tensor:



Unchanged after the TRG procedures

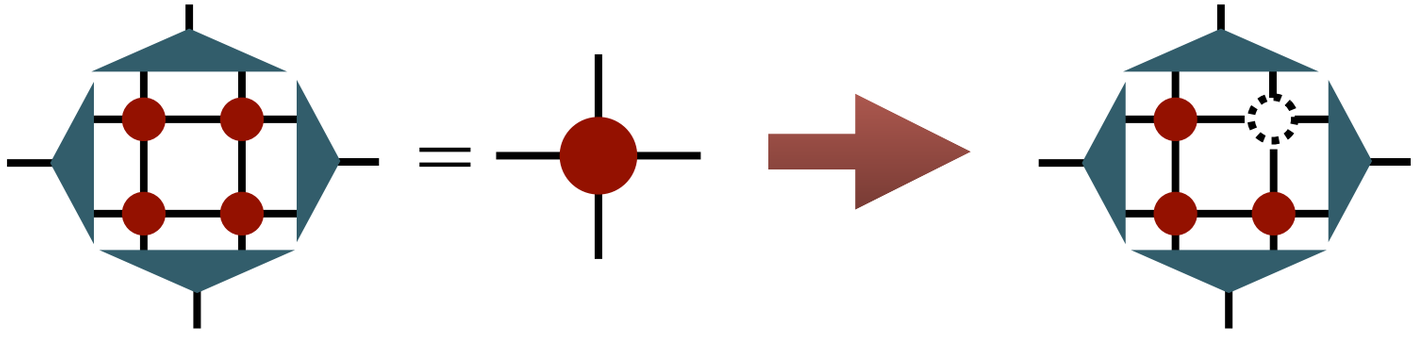
Fixed point tensor and critical exponents

- From the eigenvalues of the transfer matrix (Z. C. Gu and X. G. Wen, Phys. Rev. B **80**, 155131 (2009).)



$$\Delta_i = -\log \frac{\lambda_i}{\lambda_1}$$

- From the renormalization procedure (cf. X. Lyu, R. G. Xu, and N. Kawashima, Phys. Rev. Research 3, 023048 (2021))



eigenvalue = scaling dimension

Interesting topics in tensor network renormalization

- Try to find efficient algorithm to remove "short range" entanglement
  - TNR, Loop-TNR, GILT, Gauge fixing

TNR: G. Evenbly and G. Vidal, Phys. Rev. Lett. **115**, 180405 (2015)
Loop-TNR: S. Yang, Z.-C. Gu and , X.-G. Wen, Phys. Rev. Lett. **118**, 110504 (2017)
GILT: M. Hauru, C. Delcamp. S. Mizera Phys. Rev. B **97**, 045111 (2018)
Gauge fixing: G. Evenbly, Phys. Rev. B **98**, 085155 (2018)

- Application to lattice QCD
  - TRG with Grassmann algebra

Z.-C. Gu, F. Verstraete, and X.-G. Wen, arXiv:1004.2563
S. Takeda, and Y. Yoshimura PTEP **2015**, 043B1 (2015).

- Property at the criticality
  - Relation between TNR and MERA
  - Relation to Conformal invariance

G. Evenbly and G. Vidal,Phys. Rev. Lett. **115**, 200401 (2015)
G. Evenbly, Phys. Rev. B **95**, 045117 (2017)

Report 1

Report 1: MCMC simulation of square lattice Ising model

General information

- Please solve both 1-1 and 1-2.
    - (optional) means it is not mandatory. So, even if you do not solve the task, in principle, you can get the perfect score. When you include it, you may get additional points.
- Please include your name and student id in your report.
- Please submit it through **ITC-LMS.**
    - In case you cannot use ITC-LMS, please submit it by email.
        - I will send a reply when I receive your reports.
        - If you do not receive any response, please contact me.
- **Deadline: 2023 June 13th**

Report problem 1-1: Auto-correlation functions

- Discuss the auto-correlation function and its relaxation time as follows

1. Perform MCMC simulation of the square lattice Ising model.
   - Please try at least two algorithms: Metropolis and Swendsen-Wang (cluster)
   - If you prefer, you may simulate another model, such as MCMC simulation of cubic lattice Ising model, MD simulation of LJ particle, …
     - In those cases, please do not forget to mention the algorithms you used.
2. Calculate (and plot) auto-correlation functions of magnetization (<M>) and squared magnetization (<M^2>).
   - When you select a different model, please choose your observables accordingly.
3. **Discuss** behaviors of auto-correlation functions and correlation time by varying temperatures and system sizes (# of spins).
   - Please discuss in what cases the correlation time becomes longer.
     - Note that when the system size is too small, you may not observe explicit parameter dependences.
   - Please discuss the differences between <M> and <M^2> correlation times.
   - Please compare correlation times between Metropolis and Swendsen-Wang.

Report problem 1-1: Auto-correlation functions

4. Based on the previous discussion, consider how we should set MC steps (simulation time) to obtain reliable data when we change temperature and system size.
    - Remember the relationship between the correlation time and the statistical error.

Note:

To discuss the correlation time, it might be useful to use the integrated correlation time defined as

$$\tau = \int_0^\infty \frac{C(t)}{C(0)} dt \sim \sum_{t=0}^{T} \frac{C(t)}{C(0)}.$$

Report problem 1-1: Tips

- For MCMC simulation, you can use
  - Your own code
  - Open-source software
  - My sample codes (python) (Ising-auto.ipynb or .py in Report1.zip).
    - In order to run the sample codes, you need numpy, matplotlib, and numba modules in addition to the python3.
      - See also the header of the codes.
- In order to obtain correct auto-correlation function, we need to take care about the “thermalization" (initial state dependence) mentioned in the lecture.
  - In the case of the sample codes of Ising model,
    When you increase the system size
    or
    when you change the temperature,
    you may need to increase the "thermalization" parameter, which sets
    MC steps discarded before calculating auto-correlation functions.
  - It is also important to sample sufficiently long time, to evaluate the correlation time.

Report problem 1-2: Phase transition

- Investigate the phase transition of the $L \times L$ square lattice Ising model as follows.
  1. Perform MCMC for at least three system sizes, such as L=16, 32, and 64. If your computer environment allows, it is recommended to include larger than L=32 (e.g., L=48, 64, …).
     - Please explicitly write down the algorithm you used, e.g., Metropolis, Heatbath, Swendsen-Wang, ...
  2. Plot the figures of the squared magnetization and the specific heat as functions of temperature. (You can also include figures of other quantities.)
  3. Try to (roughly) estimate the transition temperature, T_c, from the behaviors of specific heat and the square magnetization.
     - Please note the expected behaviors in the thermodynamic limit (infinite L limit).
       - The specific heat diverges at T_c.
       - The magnetization is zero and finite when $T > T_c$ and $T < T_c$, respectively.

Report problem 1-2: Phase transition

4. Next, plot the "binder ratio of magnetization" and estimate the crossing temperature.
   - As explained in the lecture, this crossing temperature becomes an accurate estimate of $T_{c.}$
5. **Discuss** the above estimations of T_c.
   - Please compare them to the exact value of T_c. (You can find it in the lecture slide).
   - Please don't forget to mention the statistical errors.
6. (optional) Try the finite-size scaling of the binder ratio by using your estimate of T_c.
   - You may see a data collapse independent of L by adequately changing the horizontal axis. (As explained in the lecture, we need the critical exponent ν. Here you can use its exact value $\nu=1$, or you may estimate it from the data collapse.)
7. (optional) If you are interested, please compare TRG and MCMC.
   - Please note that MCMC contains statistical errors, while there are systematic errors in TRG.

Report problem 1-2: Tips

- For MCMC simulation, you can use
  - Your own code
  - Open source softwares
  - My sample codes (python) (Ising-obs.ipynb or .py in Report1.zip).
    - In order to run the sample codes, you need numpy, matplotlib, and numba modules in addition to python3.
      - See also the header of the codes.
- In order to obtain accurate results corresponding to the thermal equilibrium, MC steps are so important.
  - As I mentioned in the lecture, e.g., when the thermalization is too short compared with the correlation time, your result may be biased by the initial condition.
    - Note that even if the thermalization is sufficient, the statistical error also depends on the correlation time.
  - As you will experience in the report problem 1-1, the correlation time depends on the temperatures and the system sizes.

Report problem 1-2: Tips

- For TRG, you can use
  - Your own code
  - My sample codes (python) (Ising-trg.ipynb in Report1.zip).
    - In order to run the sample codes, you need numpy, matplotlib, and numba modules in addition to python3.
      - See also the header of the codes.
- To compare the results obtained by MCMC and TRG, Ising-comp.ipynb migght be useful.

* It might be helpful to see

  https://github.com/TsuyoshiOkubo/Introduction-to-Tensor-Network

  There are lecture materials related to TRG.

  (Unfortunately, they are presented in Japanese)

Important notice

- Please check that you can perform the report problem in your environment as soon as possible.
  - I recommend google colab to run my sample codes.
    - Please see a short instruction in the No.4 (or No.5) slide.
- If you have any troubles or questions, please ask me
  - by email: t-okubo@phys.s.u-tokyo.ac.jp
  - or come to my office **Sci. Bldg. #1 950.**
    (It is better to get an appointment by email.)
    If you have any troubles, please send us an email:
    t-okubo@phys.s.u-tokyo.ac.jp
- If a report contains only figures, you will lose lots of points.
  - Please include *explanations* and *discussions* about your results. These are essential!
- In case you use ChatGPT or other similar systems, please mention them in the report.

Next (6/7)

*Notice:

- No class on **5/30**.
- From the 8th week, the lecture will be given in **full online** by **Yamaji-sensei.**

Classical

1st: Many-body problems in physics and why they are hard to solve

2nd: Classical statistical models and numerical simulation

3rd: Classical Monte Carlo method

4th: Applications of classical Monte Carlo method

5th: Molecular dynamics simulation and its applications

Quantum

6th: Extended ensemble method for Monte Carlo methods

7th: Tensor Renormalization group

8th: Quantum lattice models and numerical simulation

9th: Quantum Monte Carlo methods

10th: Applications of quantum Monte Carlo methods

11th: Linear algebra of large and sparse matrices for quantum many-body problems

12th: Large sparse matrices and quantum statistical mechanics

13th: Advanced algorithms for quantum many-body problems