Introduction to Matrix Product State

Department of Physics, **Tsuyoshi Okubo**

理学系研究科　物理学専攻　大久保　毅

Background of lecturer

大久保　毅（OKUBO Tsuyoshi)

Projecto Lecturer,
Department of Physics,
Sci. Bldg. #1 940

Research:

Statistical Physics, Condensed matter physics, Magnetism,
(Computational Physics)

- Random packing of disks
- Mean-filed analysis of hierarchical society
- Ordering of (classical) frustrated spin system
  - Z_2-vortex, skyrmion, multiple-Q states, …
- Deconfined quantum criticality
- Tensor network methods
- ….

Outline

- Entanglement
  - Schmidt decomposition
  - Entanglement entropy and its area low
  - **Exercise 1: Calculation of entanglement entropy by python**
- Tensor network states and Matrix product state
  - Matrix product state (MPS)
  - Canonical form
  - (**Exercise 2: Making MPS by python**) May be skipped
- Optimization of MPS: TEBD algorithm
  - Szuki-Trotter decomposition and TEBD algorithm
  - iTEBD for infinite MPS
  - **Exercise 3: (TEBD and) iTEBD for S=1 Heisenberg chain by python**

Entanglement: Schmidt decomposition

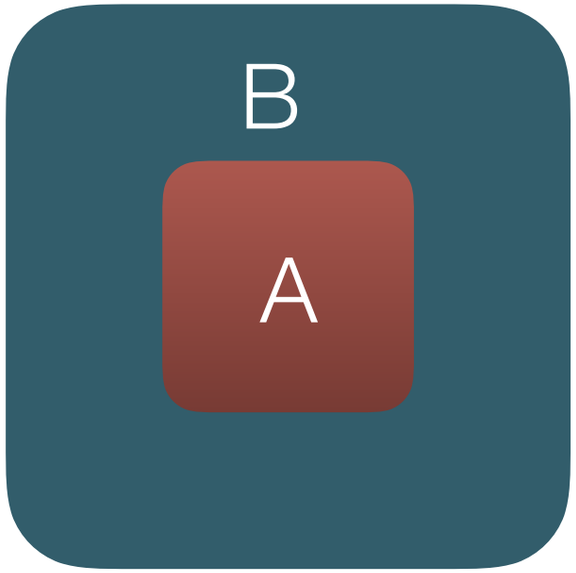
General quantum state:

$$|\Psi\rangle = \sum_{\{m_i=\uparrow\downarrow\}} T_{m_1,m_2,\cdots,m_N}|m_1, m_2, \cdots, m_N\rangle$$

2^L (or generally m^L) dimensional Hilbert space

Schmidt decomposition

Divide system into two parts, A and B:



General wave function can be represented by a superposition of orthonormal basis set.

$$|\Psi\rangle = \sum_{i,j} M_{i,j}|A_i\rangle \otimes |B_j\rangle = \sum_i \lambda_i |\alpha_i\rangle \otimes |\beta_i\rangle$$

Orthonormal basis: $\langle\alpha_i|\alpha_j\rangle = \langle\beta_i|\beta_j\rangle = \delta_{i,j}$

Schmidt coefficient: $\lambda_i \geq 0$

Schmidt decomposition is unique.

Entanglement entropy

Entanglement entropy:

Reduced density matrix of the sub system:

$$\rho_A = \mathrm{Tr}_B |\Psi\rangle\langle\Psi|$$

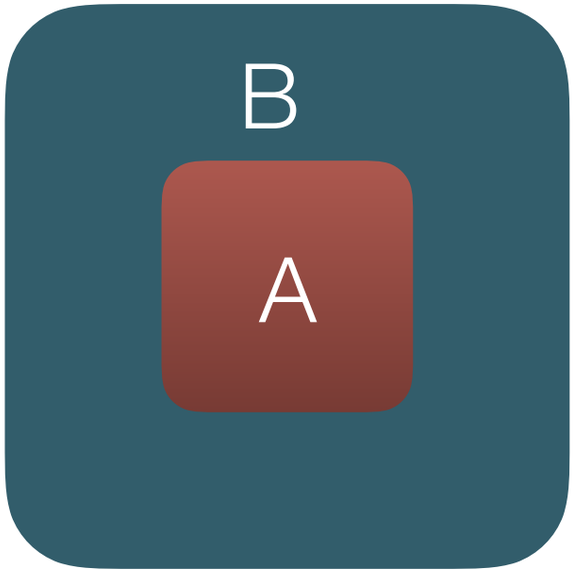
Entanglement entropy:

$$S = -\mathrm{Tr}\left(\rho_A \log \rho_A\right)$$

Schmidt decomposition $|\Psi\rangle = \sum_i \lambda_i |\alpha_i\rangle \otimes |\beta_i\rangle$

$$\rho_A = \sum_i \lambda_i^2 |\alpha_i\rangle\langle\alpha_i|$$

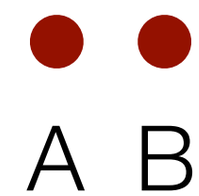
$$S = -\sum_i \lambda_i^2 \log \lambda_i^2$$



Entanglement entropy is calculated through
the spectrum of Schmidt coefficients

Intuition for EE

Example: system consists of two s=1/2 spins



1. $|\Psi\rangle = |\uparrow\rangle \otimes |\downarrow\rangle$

   A product state → $\lambda = 1,\ S = 0$

2. $|\Psi\rangle = \frac{1}{2}\left(|\uparrow\rangle - |\downarrow\rangle\right) \otimes \left(|\uparrow\rangle - |\downarrow\rangle\right)$

   Another product state → $\lambda = 1\ ,\ S = 0$

Product state : S=0

3. $|\Psi\rangle = \frac{1}{\sqrt{2}}\left(|\uparrow\rangle \otimes |\downarrow\rangle - |\downarrow\rangle \otimes |\uparrow\rangle\right)$

   Spin singlet → $\lambda_1 = \lambda_2 = \frac{1}{\sqrt{2}}\ ,\quad S = \log 2$

Maximally entangled State

4. $|\Psi\rangle = \left(x|\uparrow\rangle \otimes |\downarrow\rangle + \sqrt{1-x^2}|\downarrow\rangle \otimes |\uparrow\rangle\right)$

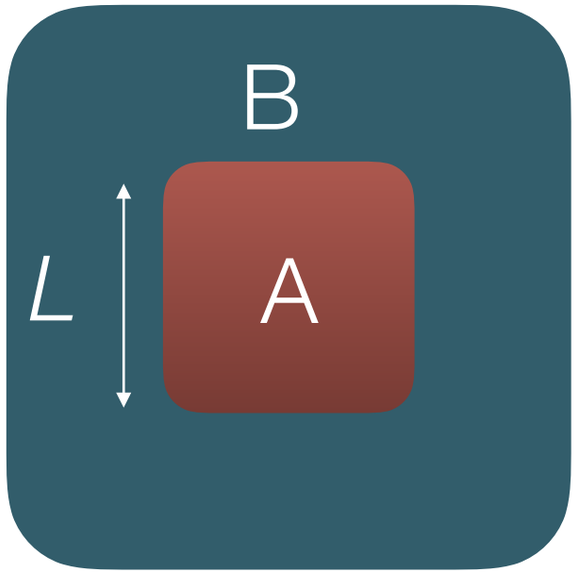
   Complicated state → $\lambda_1 = |x|, \lambda_2 = \sqrt{1-x^2}$ $\quad S = x^2 \log x^2 + \sqrt{1-x^2}\log(1 -$

Area law of the entanglement entropy

General wave functions:

EE is proportional to its volume.

$$S = -\mathrm{Tr}\left(\rho_A \log \rho_A\right) \propto L^d$$



Ground state wave functions:

For a lot of ground states, EE is proportional to its area.

J. Eisert, M. Cramer, and M. B. Plenio, Rev. Mod. Phys, 277, **82** (2010)

$$S = -\mathrm{Tr}\left(\rho_A \log \rho_A\right) \propto L^{d-1}$$

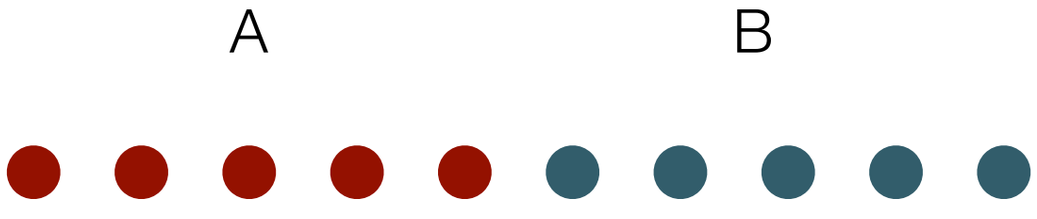
In the case of one-dimension:

Gapped ground state for local Hamiltonian

M.B. Hastings, J. Stat. Mech.: Theory Exp. P08024 (2007)

$$S = O(1)$$

Ground state are in a small part of the huge Hilbert space



Singular value decomposition

Singular value decomposition (SVD):

For a K × L matrix M,

$$M_{i,j} = \sum_m U_{i,m} \lambda_m V^\dagger_{m,j}$$

Singular values: $\lambda_m \geq 0$

Singular vectors:

$$\sum_i U_{i,m} U^\dagger_{m,j} = \delta_i, j$$

$$\sum_i V_{i,m} V^\dagger_{m,j} = \delta_i, j$$

Relation to the Schmidt decomposition:

$$|\Psi\rangle = \sum_{i,j} M_{i,j} |A_i\rangle \otimes |B_j\rangle = \sum_m \lambda_m |\alpha_m\rangle \otimes |\beta_m\rangle$$

$$|\alpha_m\rangle = \sum_i U_{i,m} |A_i\rangle$$

$$|\beta_m\rangle = \sum_j V^\dagger_{m,j} |B_j\rangle$$

$$\Rightarrow \quad \langle \alpha_i | \alpha_j \rangle = \langle \beta_i | \beta_j \rangle = \delta_{i,j}$$

By using SVD, we can calculate EE.

Exercise 1: Schmidt decomposition using python

0: Test

python ED.py

```
S=1 N-site open Heisenberg chain
N= 10
Ground state energy per bond= -1.13877810647
Excited states= 1 -1.12825125502
Excited states= 2 -1.12825125502
Excited states= 3 -1.0943321077
Excited states= 4 -1.0943321077
```

1-1: Make random wave function

SVD it and see singular value spectrum and EE

Sample code: Ex1-1.py

python Ex1-1.py

1-2: Calculate GS of S=1 Heisenberg chain

$$\mathcal{H} = \sum_i \vec{S}_i \cdot \vec{S}_{i+1}$$

SVD it and see singular value spectrum and EE

Sample code: Ex1-2.py

python Ex1-2.py

*** Try to simulate different system size "N"**

*** You can simulate other S by changing "m"**

1-1: Ex1-1.py

```
import numpy as np
import scipy.linalg as linalg
from matplotlib import pyplot

N=6## Chain length
m = 3          ## m = 2S + 1, e.g. m=3 for S=1
vec = (np.random.rand(m**N)-0.5) + 1.0j * (np.random.rand(m**N)-0.5)

## Make matrix from wave function
Mat = vec[:].reshape(m**(N/2),m**(N-N/2))

## SVD
U,s,VT = linalg.svd(Mat,full_matrices=False)

## Entanglement entropy
EE = -np.sum(s**2*np.log(s**2))
print "normalization=",np.sum(s**2)
s /=np.sqrt(np.sum(s**2))

EE = -np.sum(s**2*np.log(s**2))
print "EE=",EE

## plot singular values
pyplot.title("N sites random vector")
pyplot.plot(np.arange(m**(N/2)),s,"o")
pyplot.xlabel("index")
pyplot.ylabel("sigular value")
pyplot.yscale("log")
pyplot.show()
```

Make random vector corresponds to N-site S=1 spin chain

Singular value decomposition by "scipy.linalg.svd"

Output entanglement entropy

Plot singular values by matplotlib

1-2: Ex1-2.py

```
import numpy as np
import scipy.linalg as linalg
import ED
from matplotlib import pyplot

N=6             ## Chain length
m = 3           ## m = 2S + 1, e.g. m=3 for S=1
Delta = 1.0     ## Delta for XXZ
hx = 0.0        ## external field along x direction
D = 0.0         ## single ion anisotropy

eig_val,eig_vec = ED.Calc_GS(m,Delta,hx,D,N,k=1)


print "S=1 N-site open Heisenberg chain"
print "N=",N
print "Ground state energy per bond=", eig_val[0]/(N-1)

## Make matrix from wave function
Mat = eig_vec[:,0].reshape(m**(N/2),m**(N-N/2))

## SVD
U,s,VT = linalg.svd(Mat,full_matrices=False)

## Entanglement entropy
print "normalization=",np.sum(s**2)
s /= np.sqrt(np.sum(s**2))
EE = -np.sum(s**2*np.log(s**2))
print "EE=",EE

## plot singular values
```

import "ED.py" for exact diagonalization

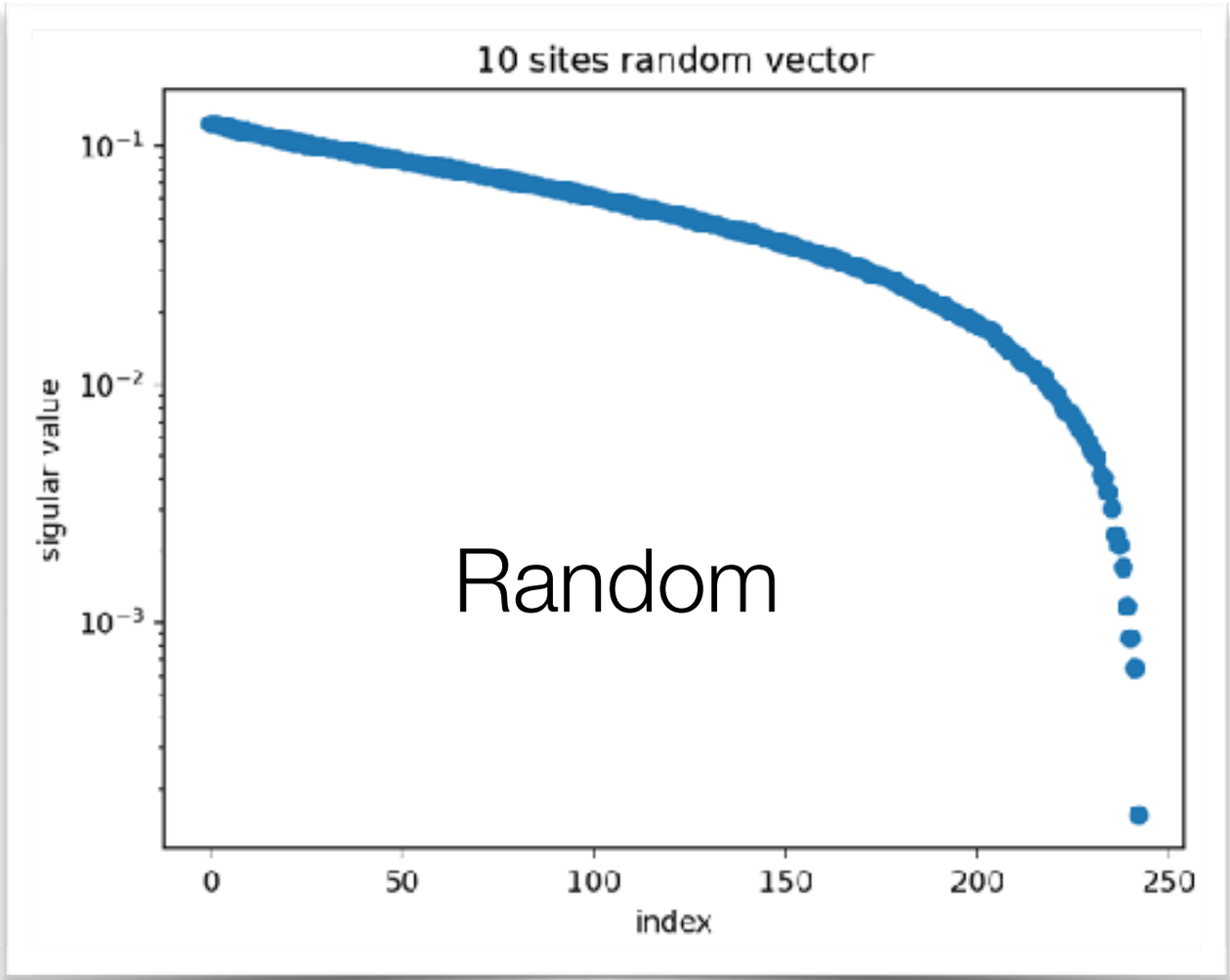
Obtain ground state of S=1 Heisenberg chain
by exact diagoanlization

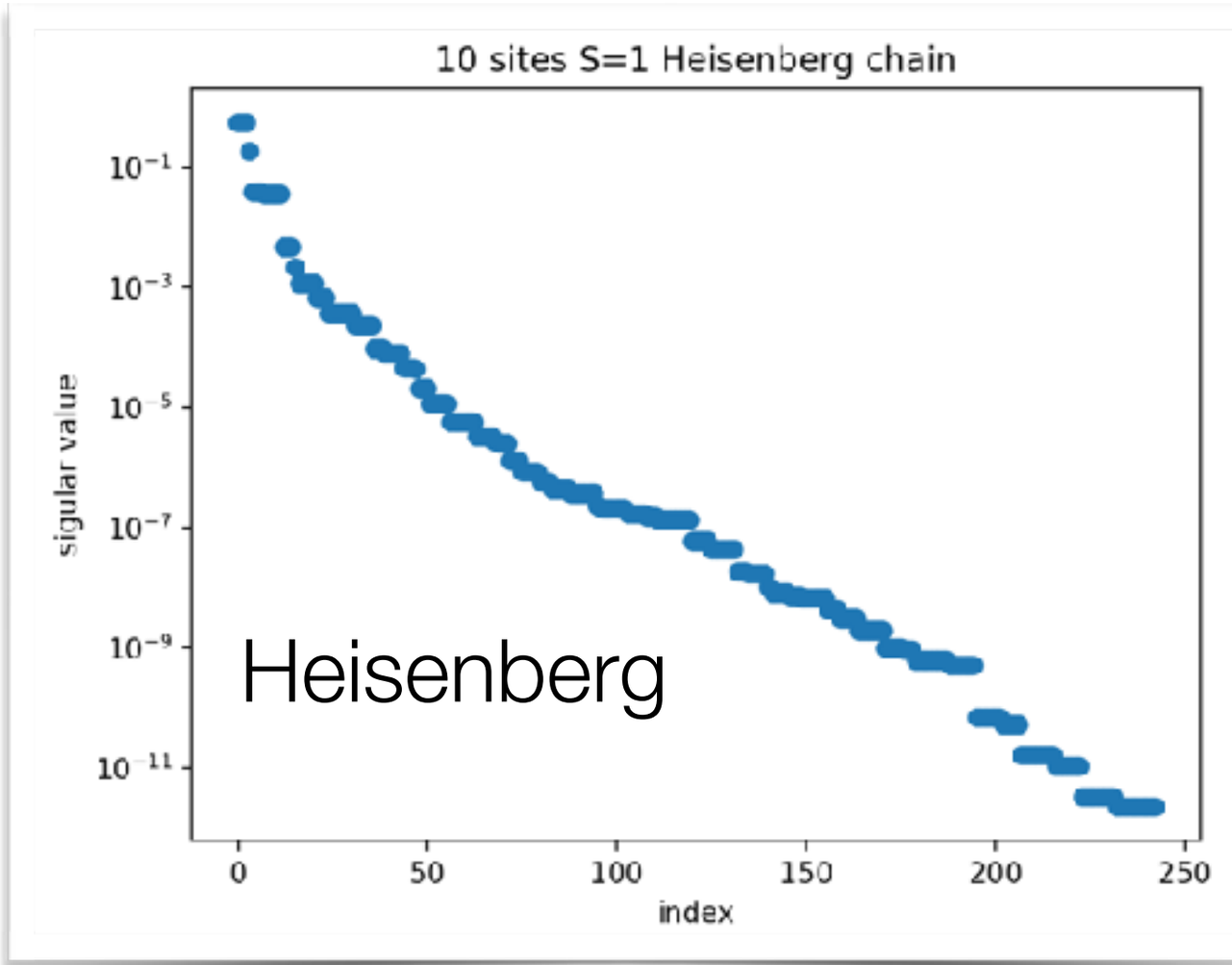
Singular value decomposition
by "scipy.linalg.svd"

Output entanglement entropy

Plot singular values
by matplotlib

Result: N=10 spectrum





Data compression of wave functions

General wave function:

$$|\Psi\rangle = \sum_{\{m_i=\uparrow\downarrow\}} T_{m_1,m_2,\cdots,m_N} |m_1, m_2, \cdots, m_N\rangle$$

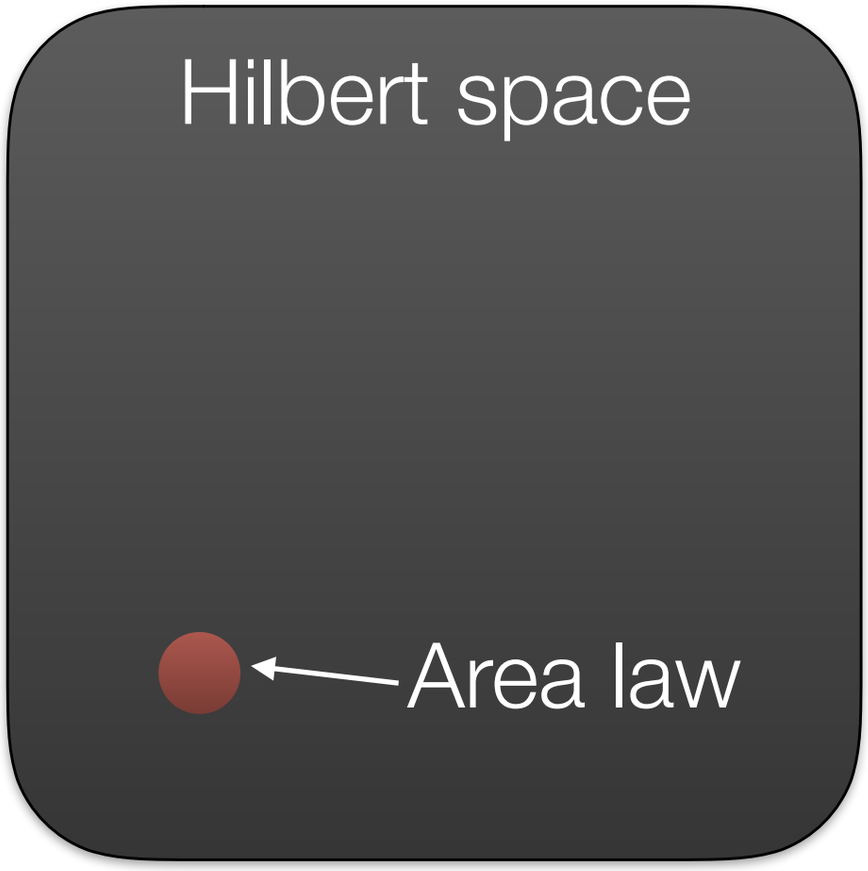
Coefficient tensor *T* can represent any points in the Hilbert space.

Ground states satisfy the area law.

In order to represent the ground state, we do not need all of 2^N elements of T.

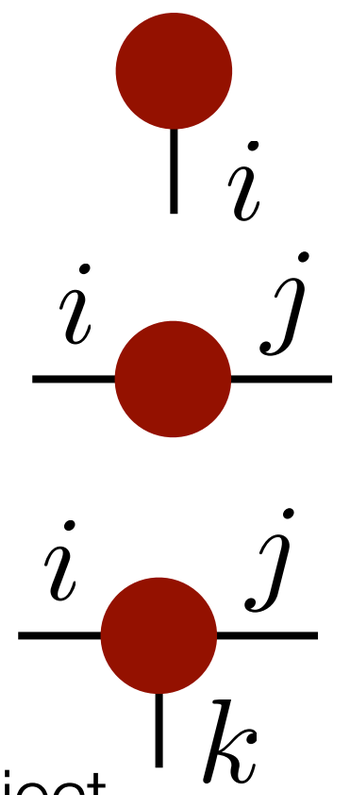
Data compression by tensor decomposition:

Tensor network states



Graphical representations for tensor network

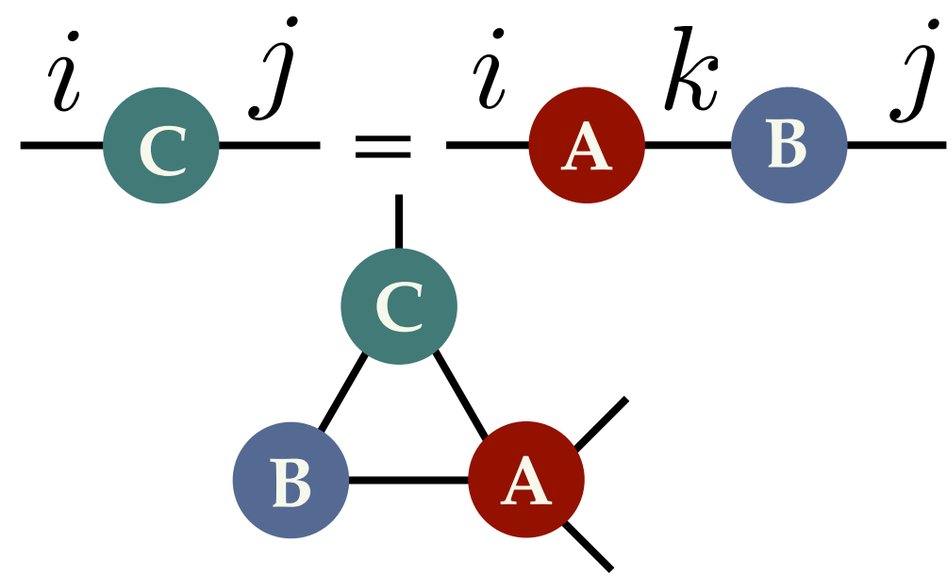
- Vector $\vec{v} : v_i$
- Matrix $M : M_{i,j}$
- Tensor $T : T_{i,j,k}$



* n-rank tensor = n-leg object

Contraction:

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



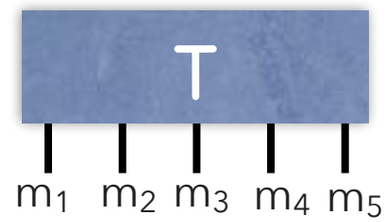
$$\sum_{\alpha,\beta,\gamma} A_{i,j,\alpha,\beta} B_{\beta,\gamma} C_{\gamma,k,\alpha}$$

Tensor network state

G.S. wave function: $|\Psi\rangle = \sum_{\{m_i=\uparrow\downarrow\}} T_{m_1,m_2,\cdots,m_N}|m_1,m_2,\cdots,m_N\rangle$

T : N-rank tensor

$T_{m_1,m_2,\cdots,m_N}$ =

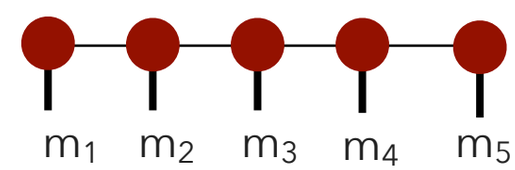


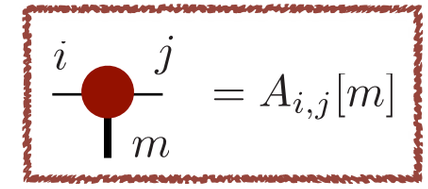
# of Elements=2^N

``Tensor network" decomposition

- Matrix Product State (MPS)

$A_1[m_1]A_2[m_2]\cdots A_N[m_N]$ =





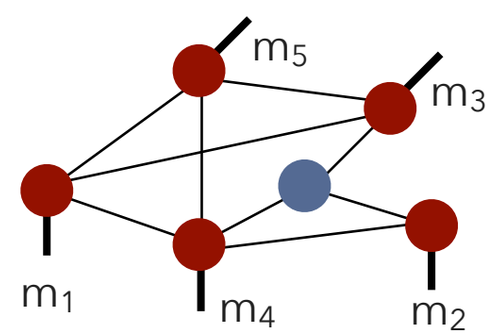
$A[m]$: Matrix for state m

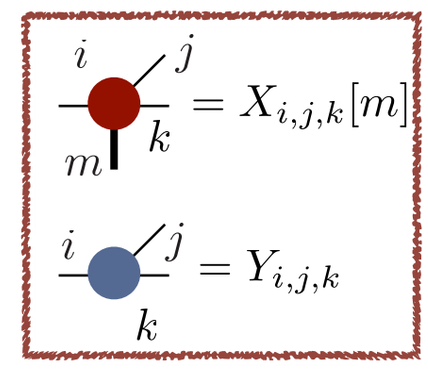
- General network

$\mathrm{Tr} X_1[m_1]X_2[m_2]X_3[m_3]X_4[m_4]X_5[m_5]Y$

X,Y : Tensors

Tr : Tensor network contraction





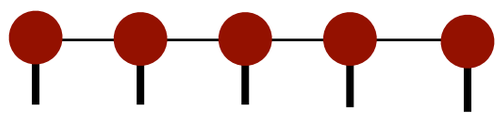
By choosing a ``good" network, we can express G.S. wave function efficiently.

ex. MPS: # of elements =$2ND^2$ D: dimension of the matrix A

Exponential→ Linear *If D does not depend on N…

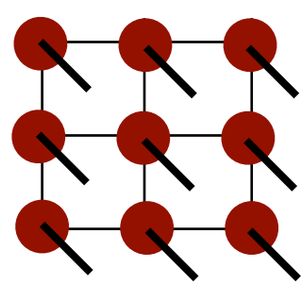
Examples of TNS

MPS:



Good for 1-d gapped systems

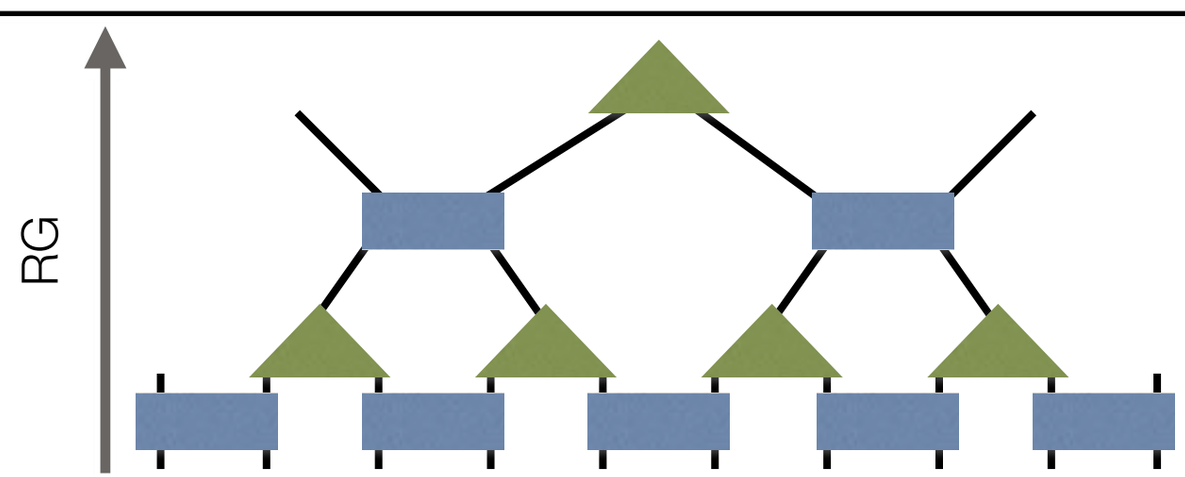
PEPS, TPS:



For higher dimensional systems

Extension of MPS

MERA:



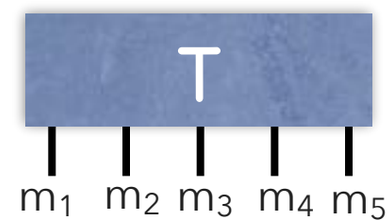
Scale invariant systems

Matrix product state (MPS)

$$|\Psi\rangle = \sum_{\{m_i=\uparrow\downarrow\}} T_{m_1,m_2,\cdots,m_N}|m_1, m_2, \cdots, m_N\rangle$$

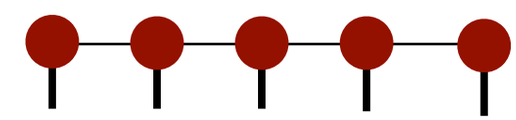
$$T_{m_1,m_2,\cdots,m_N} \simeq A_1[m_1]A_2[m_2]\cdots A_N[m_N]$$

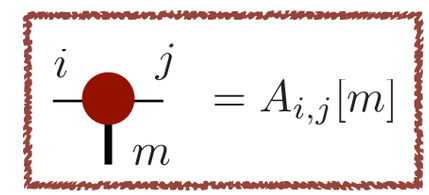
$A[m]$: Matrix for state m



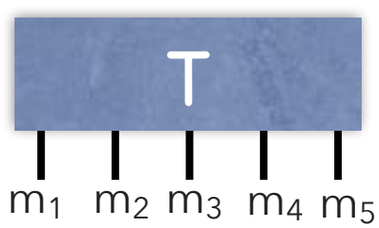
MPS

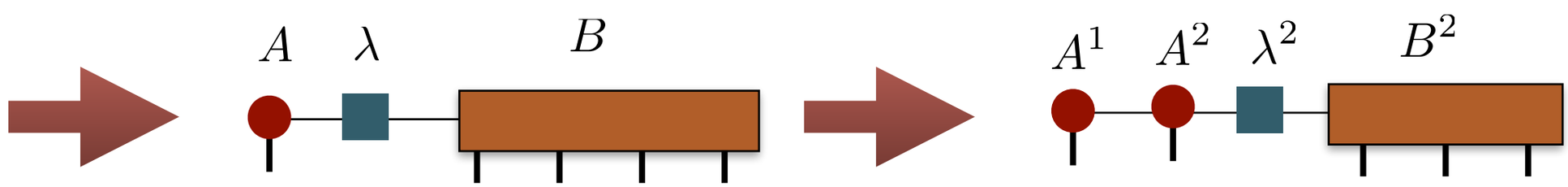
$\simeq$





General wave function can be represented by MPS through successive Schmidt decompositions





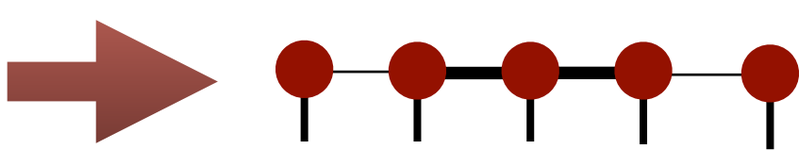
SVD of λB

In this construction, the sizes of matrices depend on the position.

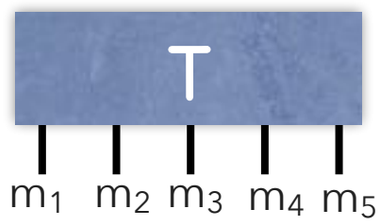
Maximum bond dimension = $m^{L/2}$

At this stage, no data compression.

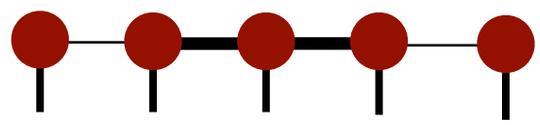
…



Matrix product state: Low rank approximation



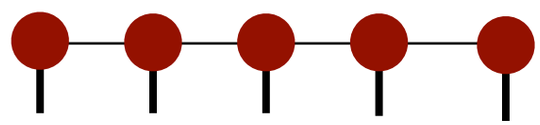
$=$



Maximum bond dimension = $2^{L/2}$

$$m^L$$

$$m^L \simeq$$



Constant bond dimension = χ

$$\propto Lm\chi^2$$

If the entanglement entropy of the system is **O(1)** (independent of L), matrix size "χ" can be small for accurate approximation.

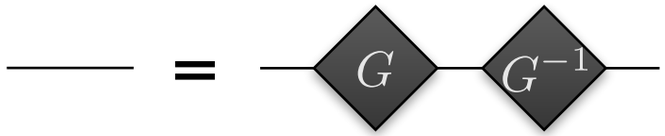
➡ MPS is good for gapped 1d systems.

On the other hand, if the EE increases as increase, "χ" must be increased to keep the same accuracy.

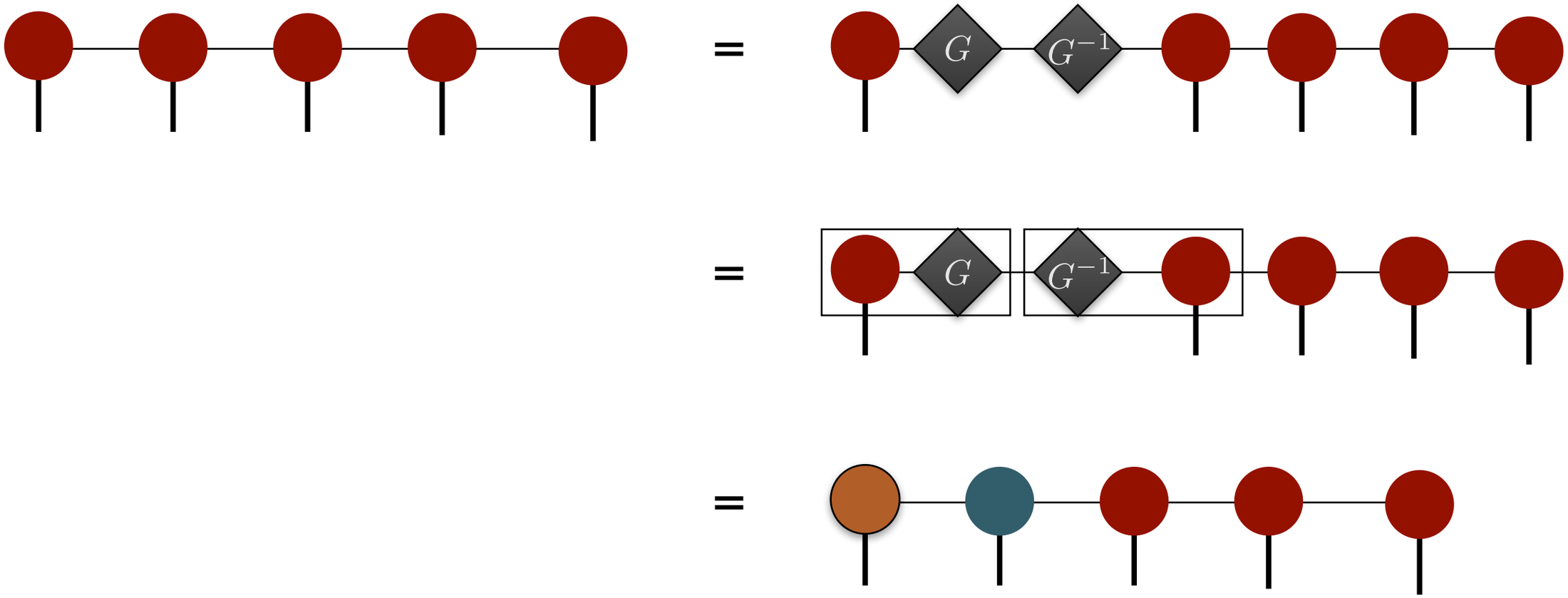
Gauge redundancy of MPS

MPS is not unique: gauge degree of freedom

$$I = GG^{-1}$$

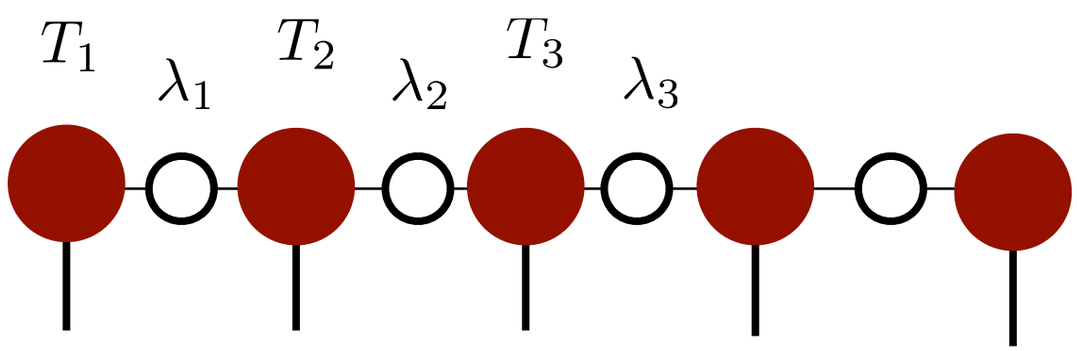


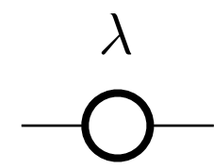
We can insert a pair of matrices GG^{-1} to MPS



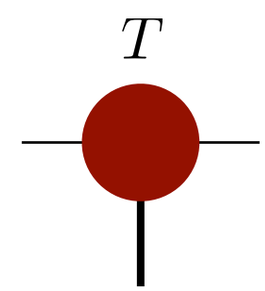
Gauge fix: Canonical form of MPS

Canonical form of MPS: (Convenient for TEBD (Vidal ...)



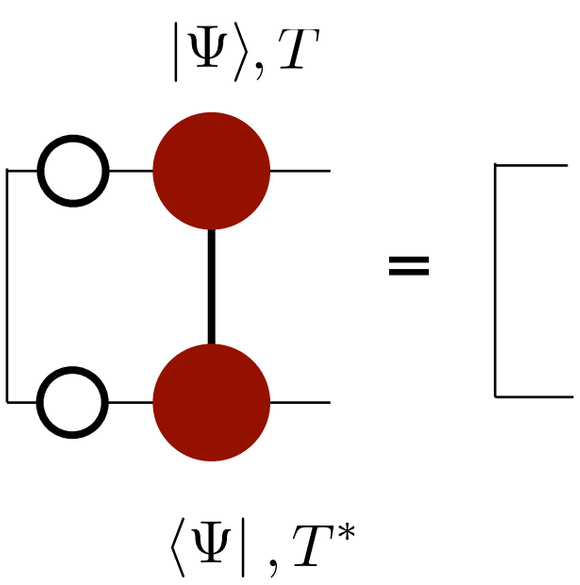


:Diagonal matrix corresponds to Schmidt coefficient

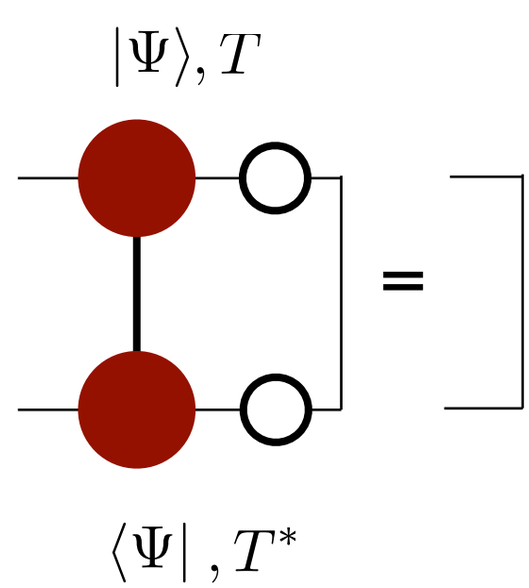


:Virtual indices corresponds to Schmidt basis

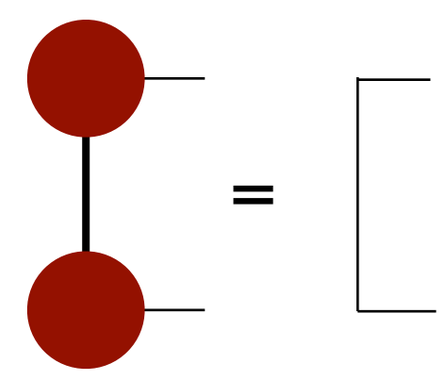
Left canonical condition:



Right canonical condition:

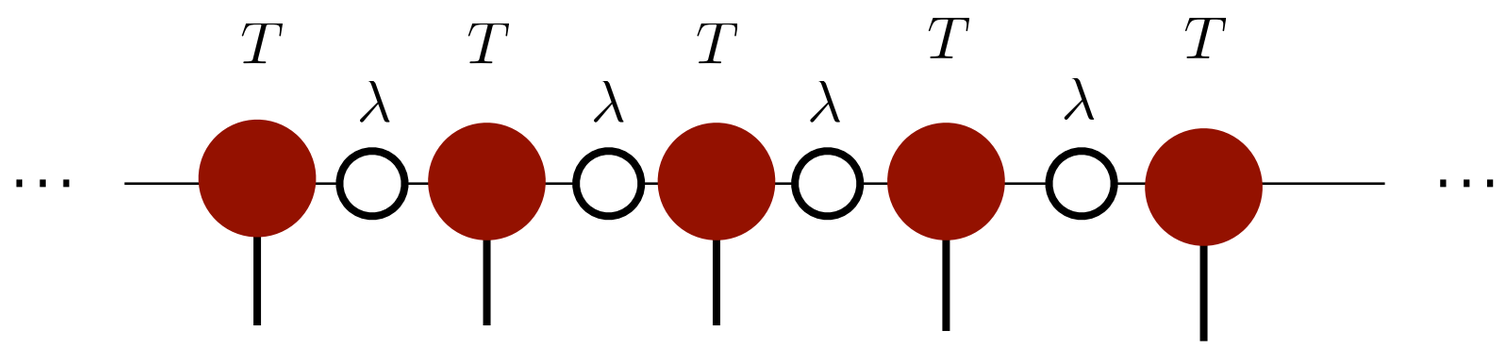


(Boundary)



MPS for infinite chains

By using MPS, we can write the wave function of
a translationally invariant **infinite chain**



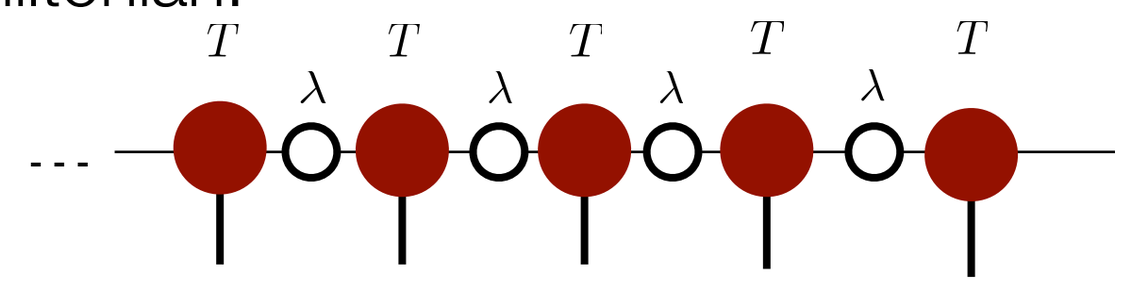
Infinite MPS (iMPS) is made by repeating T and λ infinitely.

Translationally invariant system → T and λ are independent of positions!

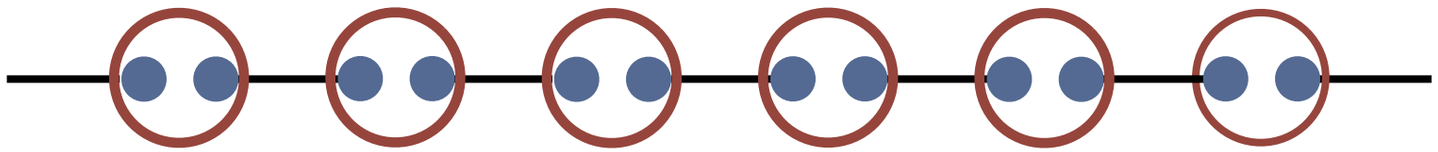
Example of iMPS: AKLT state

S=1 Affleck-Kennedy-Lieb-Tasaki (AKLT) Hamiltonian:

$$\mathcal{H} = J \sum_{\langle i,j \rangle} \vec{S}_i \cdot \vec{S}_j + \frac{J}{3} \sum_{\langle i,j \rangle} \left(\vec{S}_i \cdot \vec{S}_j \right)^2$$



The ground state of AKLT model:



S=1 spin:

S=1/2 spin

χ=2 iMPS: (U. Schollwock, Annals. of Physics **326**, 96 (2011))

Spin singlet

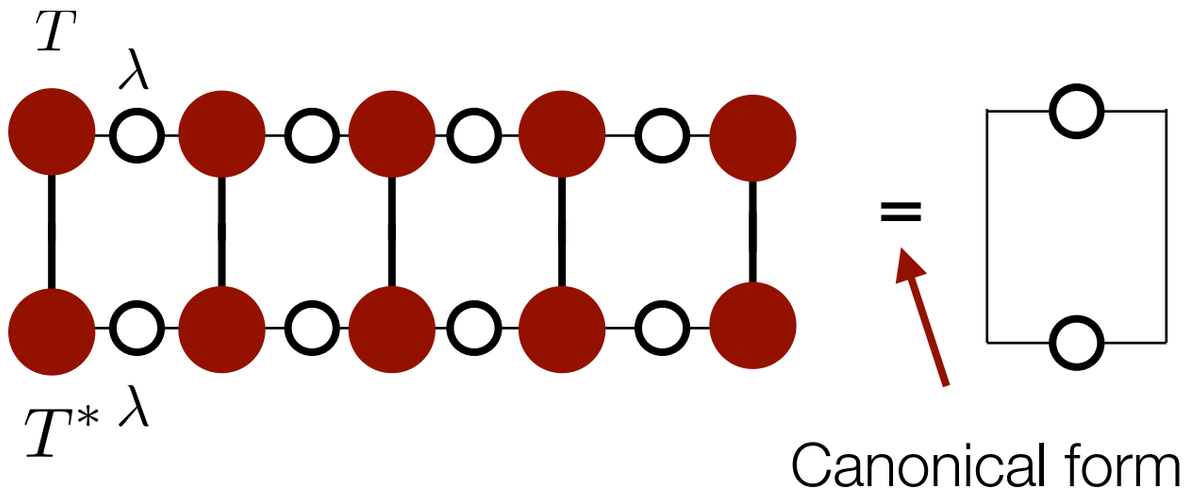
$$T[S_z = 1] = \sqrt{\frac{4}{3}} \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}$$

$$T[S_z = 0] = \sqrt{\frac{2}{3}} \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \quad , \lambda = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

$$T[S_z = -1] = \sqrt{\frac{4}{3}} \begin{pmatrix} 0 & 0 \\ -1 & 0 \end{pmatrix}$$

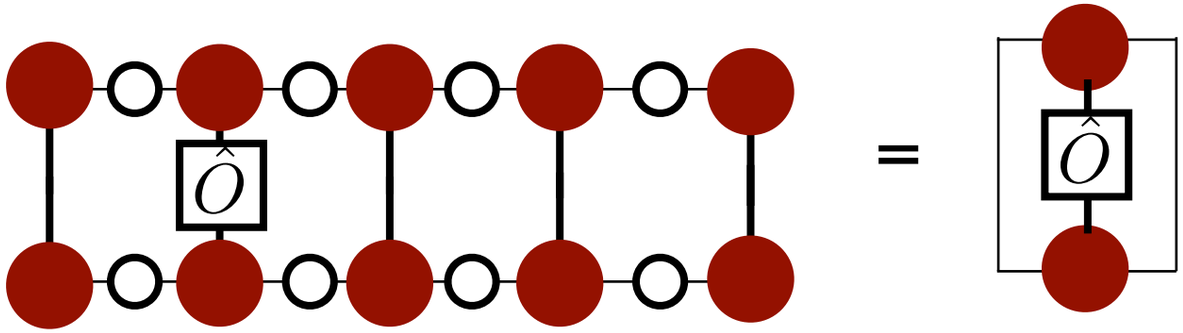
Calculation of expectation value

$\langle \Psi | \Psi \rangle =$



$= \sum_i \lambda_i^2 = 1$

$\langle \Psi | \hat{O} | \Psi \rangle =$



For iMPS, if it is in the canonical form,
the final graph is identical to the above finite system.

Exercise 2: Make MPS and approximate it

2-1: Make exact MPS from GS wave function obtained by ED

(We can easily check that the MPS obtained by successive SVD satisfy the canonical condition.)

Sample code: Ex2-1.py

python Ex2-1.py

2-2: Approximate the MPS by truncating singular values

- Calculate approximate GS energy and compare it with ED
- Change *chi_max* and see energies

Sample code: Ex2-2.py

python Ex2-2.py

2-2: Ex2-2.py

```
import numpy as np
import scipy.linalg as linalg
import ED
import TEBD
from matplotlib import pyplot

N=6             ## Chain length
m = 3           ## m = 2S + 1, e.g. m=3 for S=1
Delta = 1.0     ## Delta for XXZ
hx = 0.0        ## external field along x direction
D = 0.0         ## single ion anisotropy

chi_max = 10       ## maxmum bond dimension at truncation


eig_val,eig_vec = ED.Calc_GS(m,Delta,hx,D,N,k=1)
```

import "TEBD.py" for energy calculation of MPS

Obtain ground state of S=1 Heisenberg chain by exact diagoanlization

```
## Make exact MPS (from "left")
Tn = []
lam = [np.ones((1,))]
lam_inv = 1.0/lam[0]
R_mat = eig_vec[:,0].reshape(m,m**(N-1))

chi_l=1
for i in range(N-1):
    U,s,VT = linalg.svd(R_mat,full_matrices=False)
    chi_r = s.size

    Tn.append(np.tensordot(np.diag(lam_inv),U.reshape(chi_l,m,chi_r),(1,0)).transpose(1,0,2))
    lam.append(s)
    lam_inv = 1.0/s
    R_mat = np.dot(np.diag(s),VT).reshape(chi_r*m,m**(N-i-2))
    chi_l = chi_r
Tn.append(VT.reshape(m,m,1).transpose(1,0,2))
lam.append(np.ones((1,)))

## Truncation
for i in range(N-1):
    chi = min(chi_max,lam[i+1].shape[0])
    lam[i+1]=lam[i+1][:chi]
    Tn[i]=Tn[i][:,:,:chi]
    Tn[i+1]=Tn[i+1][:,:chi,:]
```

Successive SVD to make MPS

Truncate singular values (Data compression)

Optimization of MPS: How to obtain GS

Method to optimize MPS for GS of a specific Hamiltonian

1. Variational optimization

Change matrix elements to reduce the energy: $\min_{T,\lambda} \frac{\langle \Psi | \mathcal{H} | \Psi \rangle}{\langle \Psi | \Psi \rangle}$

➡ Density matrix renormalization group (DMRG)

(S. R. White, Phys. Rev. Lett. **69**, 2863 (1992))

2. Imaginary time evolution

Simulate imaginary time evolution: $|\Psi_{\mathrm{GS}}\rangle \propto \lim_{\beta \to \infty} e^{-\beta \mathcal{H}} |\Psi_0\rangle$

For a initial state $\langle \Psi_{\mathrm{GS}} | \Psi_0 \rangle \neq 0$

➡ Time evolving block decimation (TEBD) algorithm

(G. Vidal, Phys. Rev. Lett. **91**, 147902 (2003))

TEBD algorithm:

Suppose the Hamiltonian can be decomposed
into the sum of two-body local terms

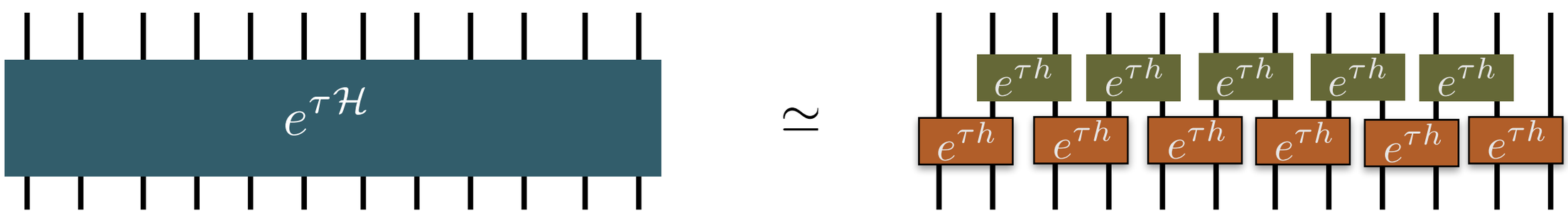
$$\mathcal{H} = \sum_i h_{[i,i+1]} = \sum_{i\in\text{even}} h_{[i,i+1]} + \sum_{i\in\text{odd}} h_{[i,i+1]}$$

$$= \mathcal{F} + \mathcal{G} \qquad\qquad [\mathcal{F},\mathcal{G}] \neq 0$$

Suzuki-Trotter decomposition of ITE operator

$$e^{-\tau\mathcal{H}} = e^{-\tau\mathcal{F}} e^{-\tau\mathcal{G}} + O(\tau^2) \qquad \text{(1st order)}$$

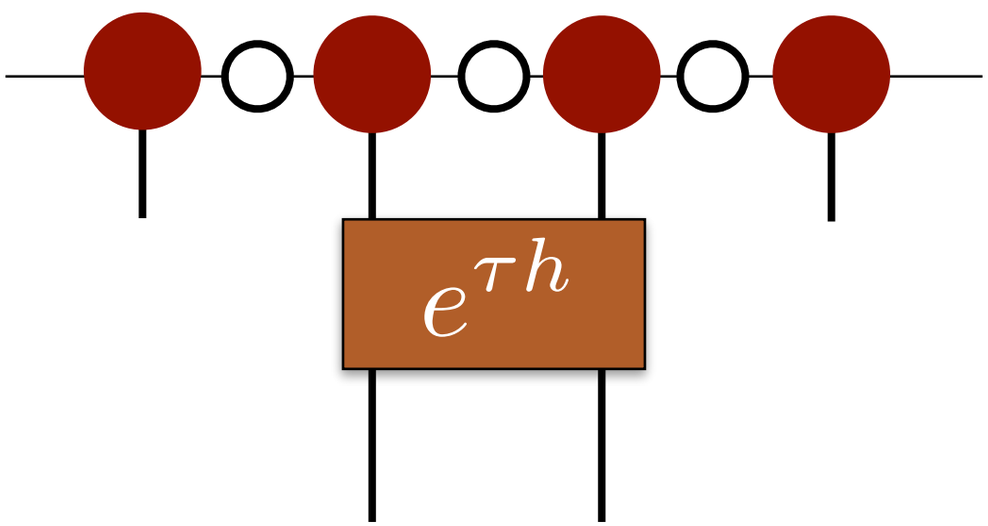
$$= e^{-\tau/2\mathcal{F}} e^{-\tau\mathcal{G}} e^{-\tau/2\mathcal{F}} + O(\tau^3) \qquad \text{(2nd order)}$$



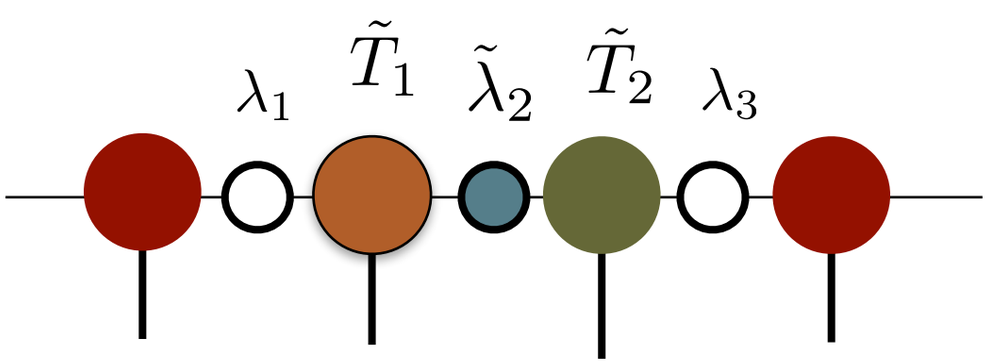
TEBD algorithm:

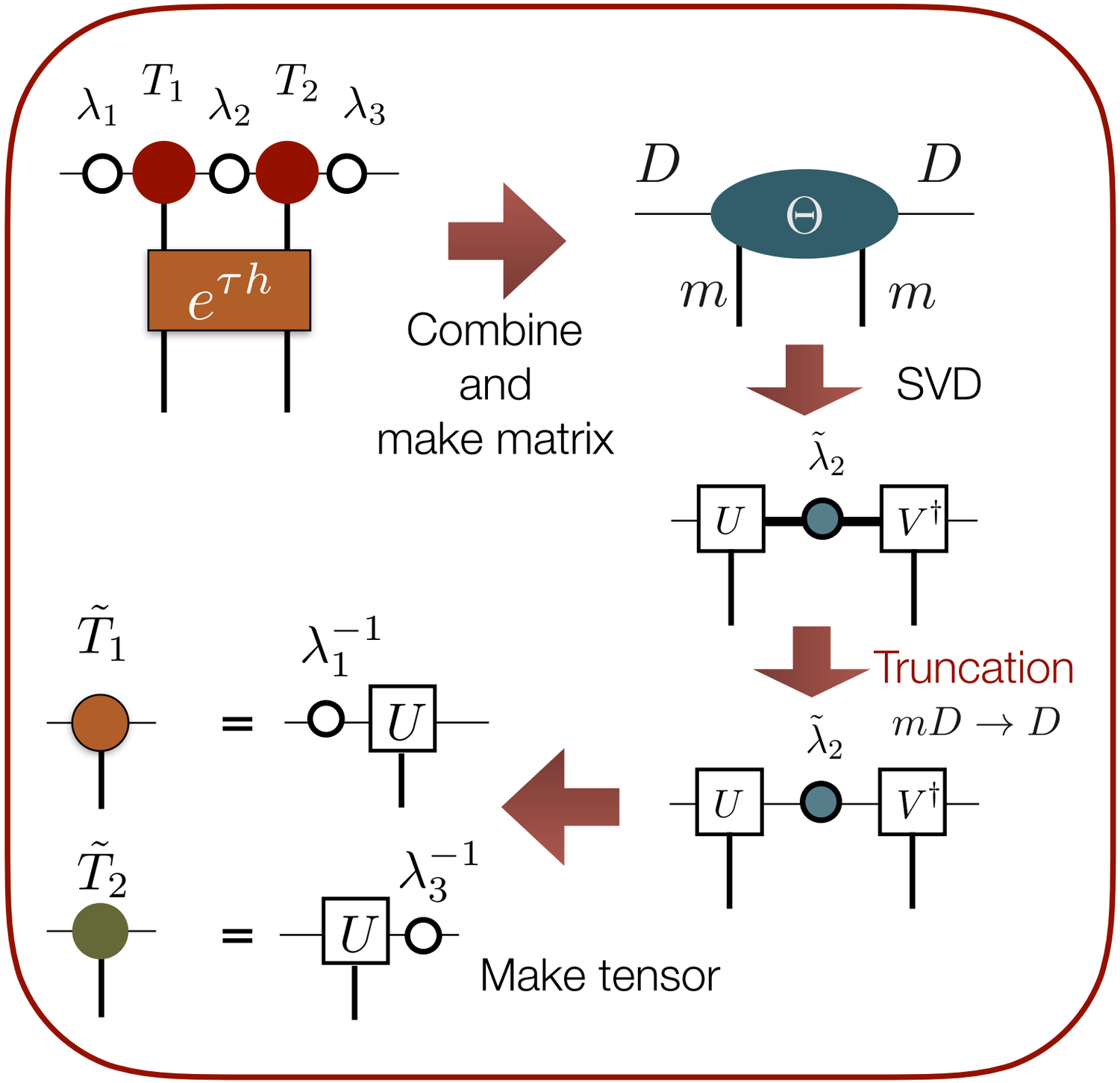
(G. Vidal, Phys. Rev. Lett. **91**, 147902 (2003))

Apply ITE operator



Approximation





iTEBD algorithm:

(G. Vidal, Phys. Rev. Lett. **98**, 070201 (2007))

(R. Orús and G. Vidal, Phys. Rev. B **78**, 155117 (2008))

Finite system: TEBD

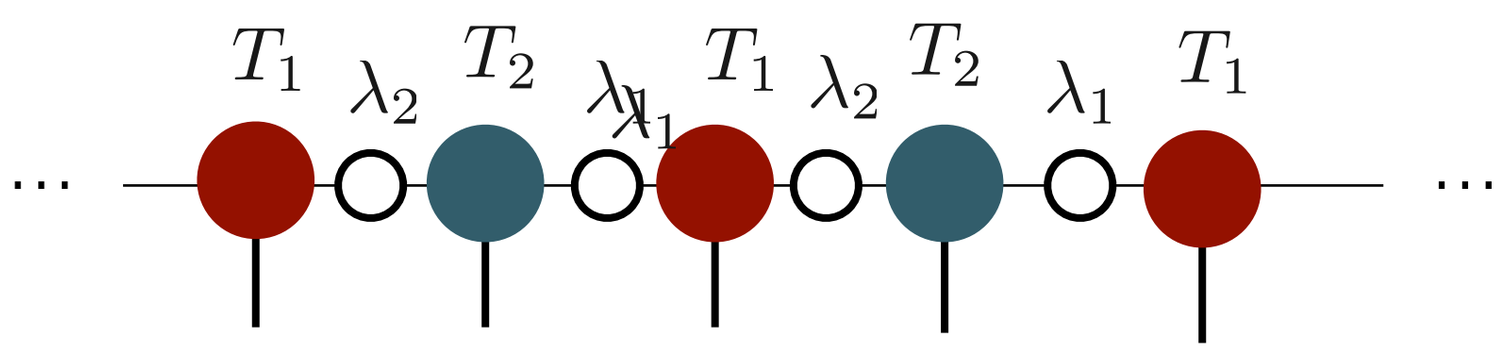
Sequentially apply ITE operators → $O(N)$ SVD for each step

Finite system: iTEBD

Due to the translational invariance, all SVD are equivalent. → $O(1)$ SVD for each step

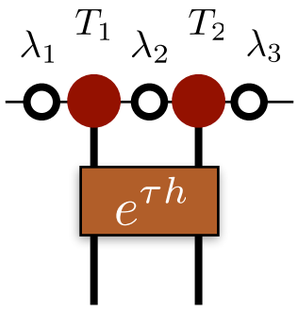
*Note

Because of SVD in iTEBD algorithm, we need at least two independent tensors even in translationally invariant system



Remarks for (i)TEBD calculation

1. For accurate calculation, the canonical form is important.



If λ is equal to the Schmidt coefficient, it contains all information of the remaining part of the system.

→ Truncation based on local SVD can be globally optimal.

2. If the operator is unitary, MPS keeps canonical form within truncation error

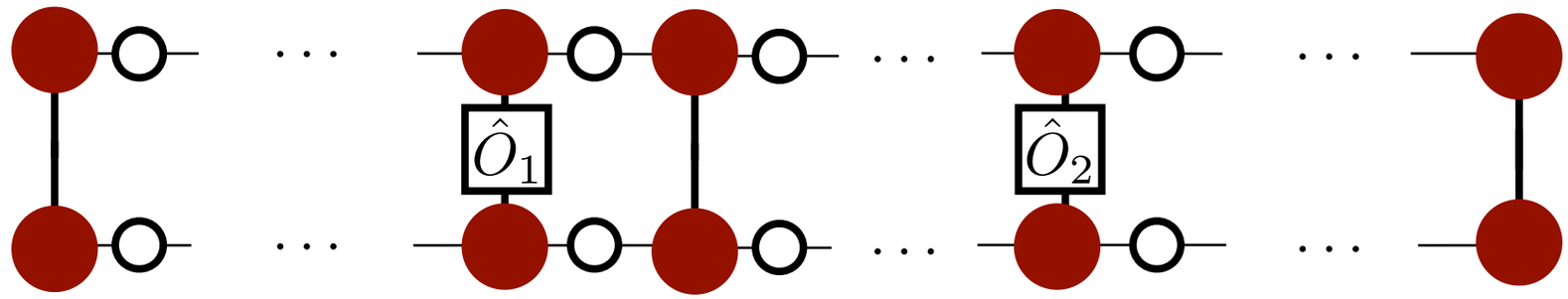
In the case of ITE, the operator is not unitary.

However, when τ is small the operator is almost unitary.

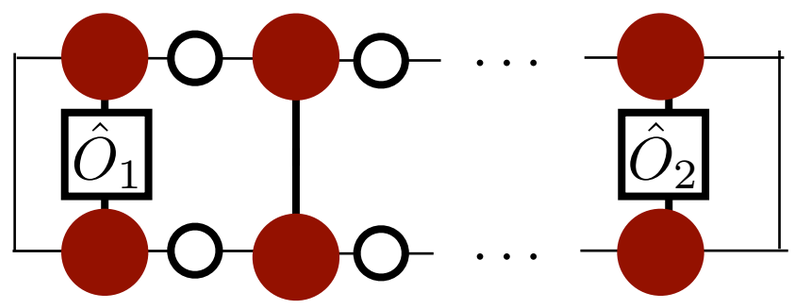
→ If we chose the initial MPS as the canonical form, TEBD algorithm almost keep it.
(So, TEBD is almost "globally optimal")

Calculation of correlation function

$\langle \Psi | \hat{O}_1(r_1) \hat{O}_1(r_2) | \Psi \rangle =$



=



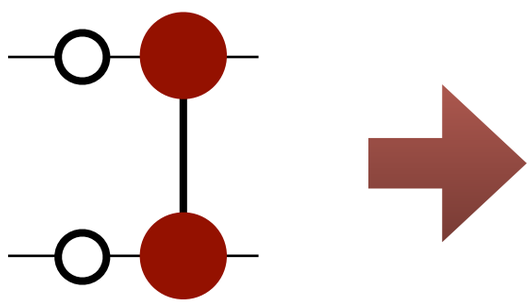
Typically, correlation functions decays exponentially.

$$\langle \Psi | \hat{O}_1(r_1) \hat{O}_1(r_2) | \Psi \rangle \propto \exp\left[-r/\xi\right] \quad (r \to \infty)$$

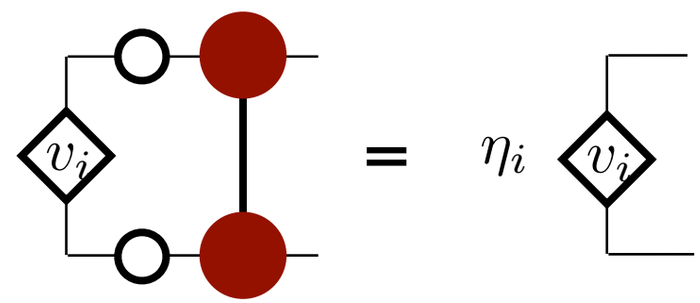
ξ **:Correlation length**

Calculation of correlation length and gap

Transfer matrix:



eigen value problem



$$|\eta_1| \geq |\eta_2| \geq \cdots$$

Correlation length: $\xi = -\log\left(\frac{|\eta_2|}{|\eta_1|}\right) \propto \frac{1}{\Delta E}$

From correlation length, we can estimate the gap.

Exercise 3: (TEBD and) iTEBD simulation

3-1: TEBD simulation

Simulate small finite size system and compare energy with ED

Sample code: Ex3-1.py

python Ex3-1.py

3-2: iTEBD simulation

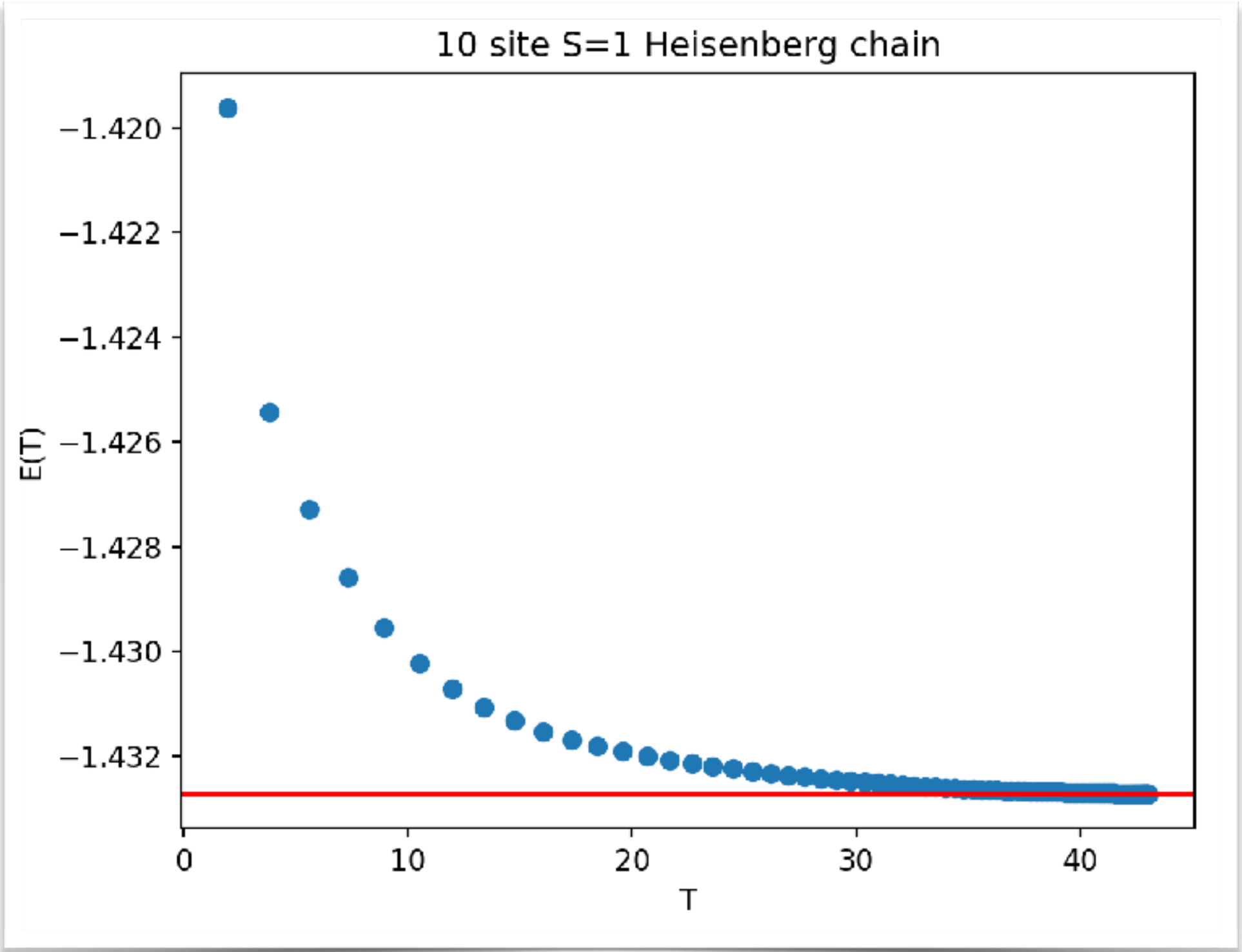
Simulate infinite system and calculate energy, correlation function

Sample code: Ex3-2.py

python Ex3-2.py

*** Try simulation with different "chi_max", "T_step"**

3-1: Energy dynamics in TEBD



3-2: Ex3-2.py

```
import numpy as np
import TEBD
import iTEBD
from matplotlib import pyplot

m = 3          ## m = 2S + 1, e.g. m=3 for S=1
Delta = 1.0    ## Delta for XXZ
hx = 0.0       ## external field along x direction
D = 0.0        ## single ion anisotropy

chi_max = 20  ## maxmum bond dimension at truncation

## parameter for TEBD
####
## tau decreases from tau_max to tau_min gradually
####
tau_max = 0.1      ## start imaginary time tau
tau_min = 0.001    ## final imaginary time tau
T_step= 2000        ## ITE steps

Tn, lam = iTEBD.iTEBD_Simulation(m,Delta,hx,D,chi_max,tau_max,tau_min,T_step)

## Calculate Energy
Env_left,Env_right = iTEBD.Calc_Environment_infinite(Tn,lam,canonical=False)

print "iTEBD simulation with chi_max =",chi_max
print "Energy of iTEBD",iTEBD.Calc_Energy_infinite(Env_left,Env_right,Tn,lam,Delta,hx,D)

## Calcuate correlation function
Sz = np.zeros((m,m))
for i in range(m):
    Sz[i,i] = 0.5 * (m - 1.0) - i

max_distance=50
step=1
Corr = iTEBD.Contract_correlation_infinite(Env_left,Env_right,Tn,lam,Sz,Sz,max_distance,s

print np.arange(2,max_distance*2,2*step)
print len(Corr)
## plot correlation function
```

import "TEBD.py" and "iTEBD.py" for iTEBD simulation

Input for ITE (scheduling of tau)

iTEBD simulation

Output energy

Calculate correlation function

Plot correlation function

Exercise 4: Correlation length of S=1 spin model

Hamiltonian:

$$\mathcal{H} = \sum_i \vec{S}_i \cdot \vec{S}_{i+1} + D \sum_i S_{z,i}^2$$

Calculate the correlation length varying D by iTEBD

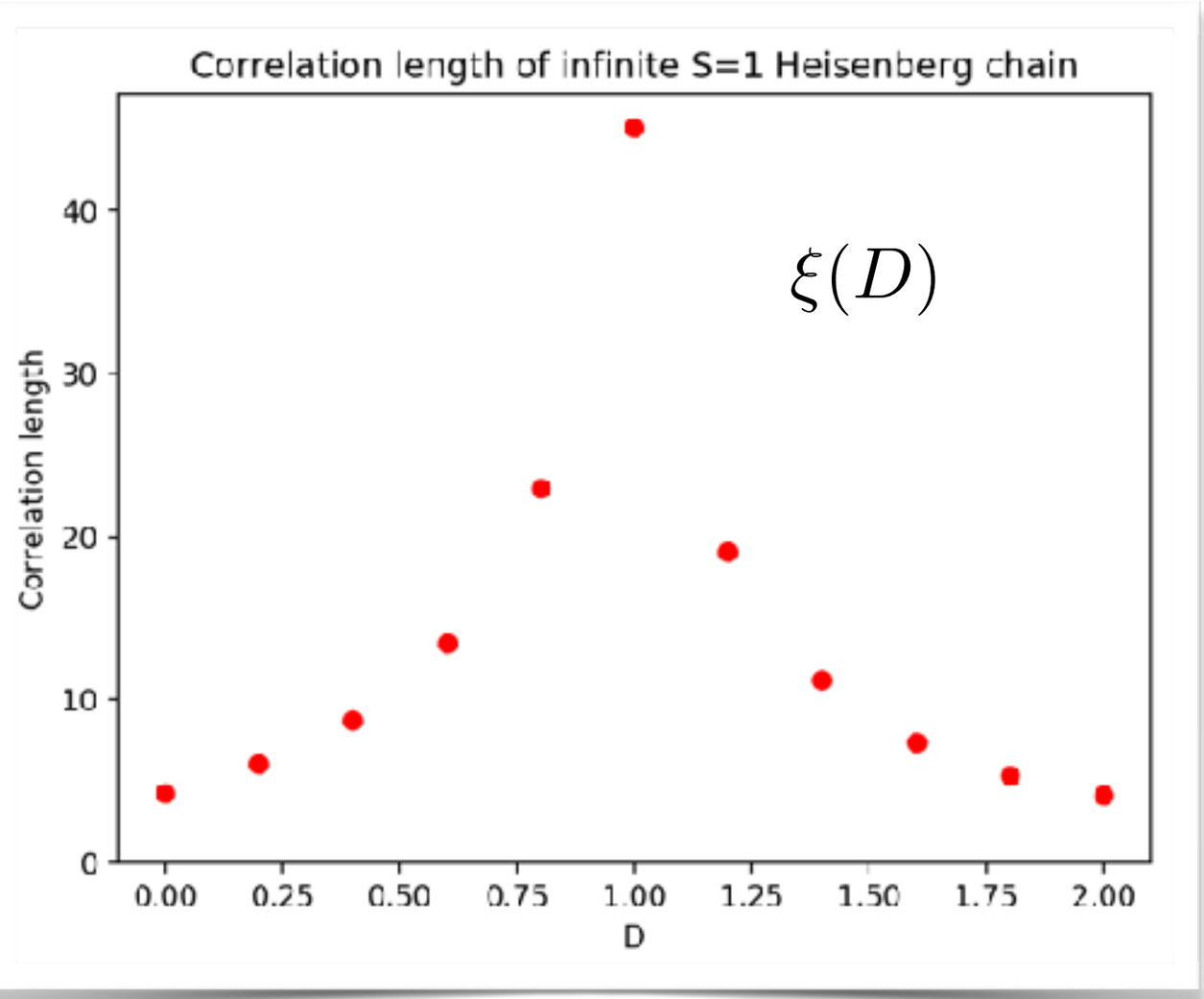
Sample code: Ex4.py

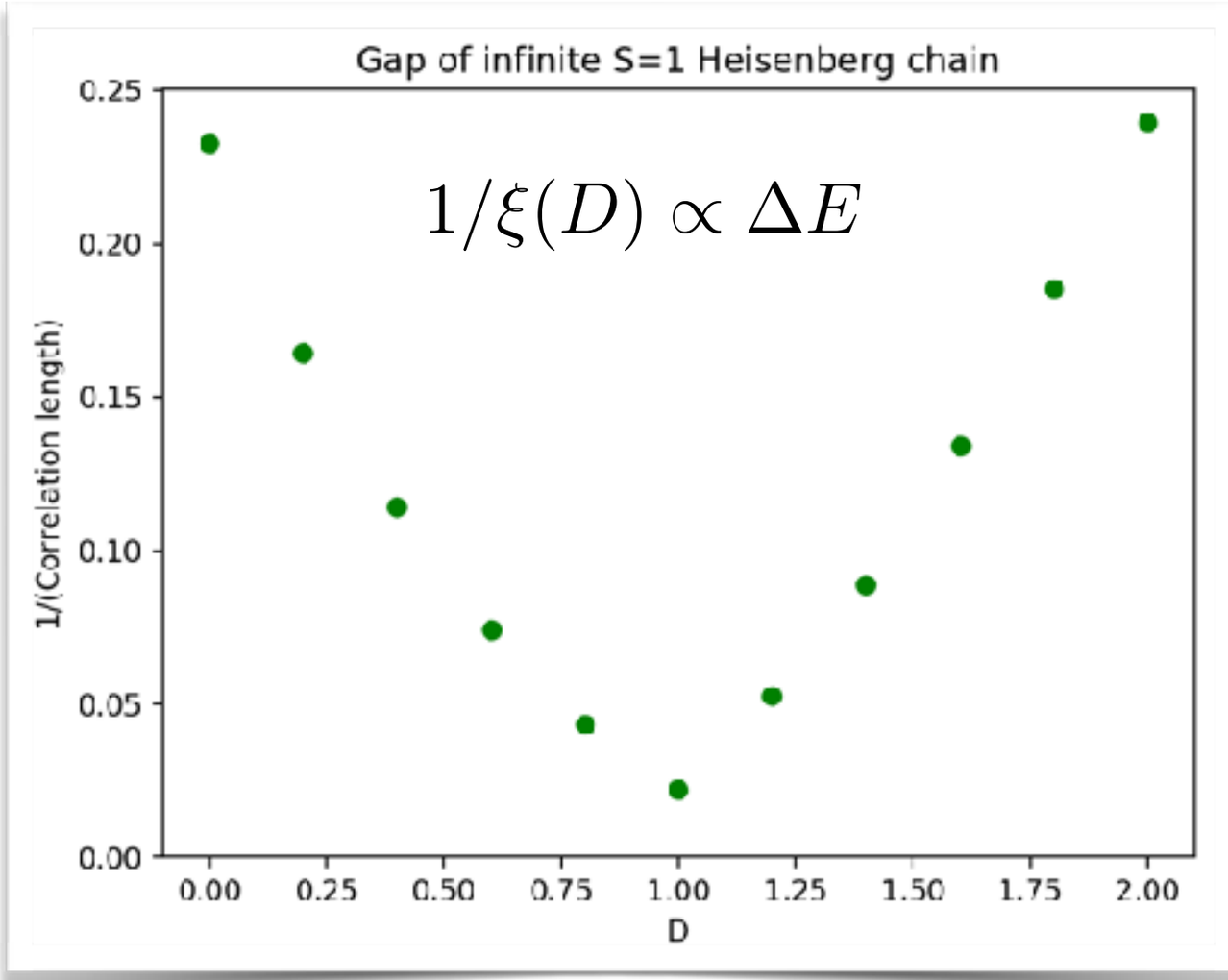
(Calculate the correlation length from the transfer matrix)

python Ex4.py

* It takes about 5 minutes to complete the calculation
* In order to obtain more accurate results, we need to increase χ.
  In addition, more longer ITE steps are needed.

4: Correlation length and gap





(Y.-C. Tzeng, H. Onishi, T. Okubo, Y.-J. Kao, Phys. Rev. B **96**, 060404(R) (2017))

$$D_c = 0.9684713(1)$$

$D < D_c$: Haldane phase

$D > D_c$: Large-D phase

Introduction to Matrix Product State
Detecting Symmetry