Nov 10, 2025

# Selected Topics in Computational Quantum Physics

# 量子物理计算方法选讲

Shuo Yang（杨硕）

*Department of Physics, Tsinghua University*

Email: shuoyang@tsinghua.edu.cn

WeChat: condmat-ys

# Density Matrix Renormalization Group and Matrix Product States

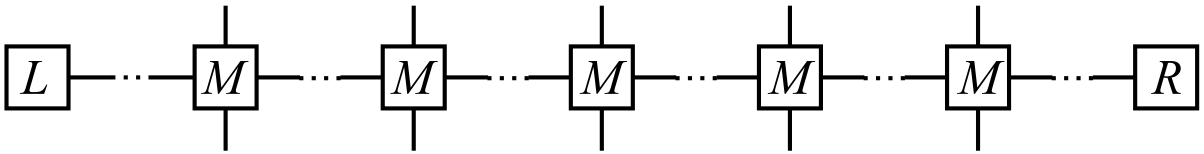
- origin of Density Matrix Renormalization Group method (DMRG)
- many-body entanglement
- traditional DMRG method
- Matrix Product State (MPS) and Matrix Product Operator (MPO)
- MPS algorithms
- various applications

selected review articles:
U. Schollwock, arXiv: 1008.3477
N. Schuch, arXiv: 1306.5551.
F. Verstraete, J.I. Cirac, V. Murg, arXiv: 0907.2796.

# Matrix Product Operator (MPO)

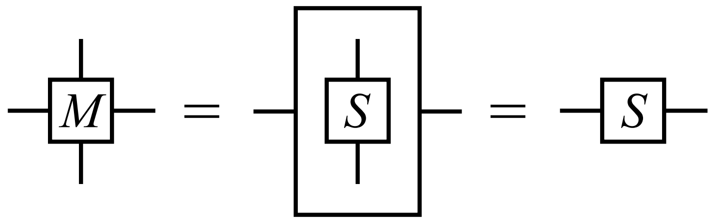
- 1D Hamiltonian can be written as an Matrix Product Operator (MPO)

  e.g., under Open Boundary Condition (OBC)

  $H =$ 
  

  upper and lower bonds of $M$ are physical bonds

  left and right bonds of $M$ are virtual bonds



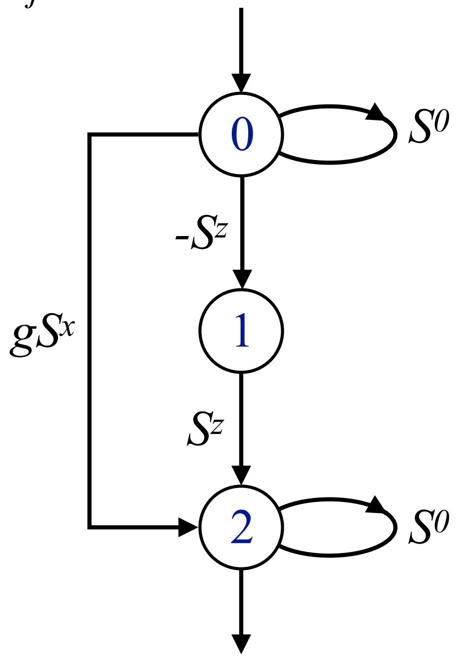

- we regard $M$ as a matrix

  matrix product operator → matrix elements are physical operators

# Finite automata for MPO

- transverse field Ising model

$$H = \sum_j \left( -S^z_j S^z_{j+1} + g S^x_j \right)$$

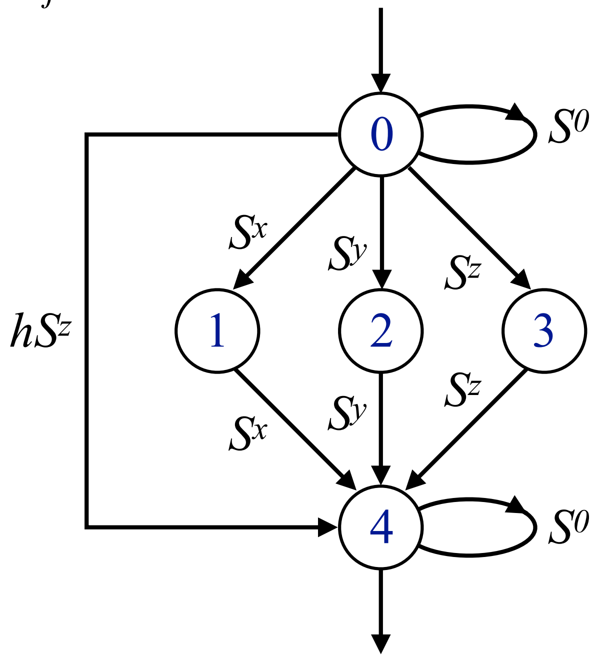



$$D_{\mathrm{mpo}} = 3 \quad \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & & & \\ 1 & & \left(\begin{array}{ccc} S^0 & -S^z & gS^x \\ 0 & 0 & S^z \\ 0 & 0 & S^0 \end{array}\right) & \\ 2 & & & \end{array}$$

$$\begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline \begin{array}{c} 0 \\ 1 \\ 2 \end{array} & \multicolumn{3}{c}{\left(\begin{array}{ccc} S^0 & -S^z & gS^x \\ 0 & 0 & S^z \\ 0 & 0 & S^0 \end{array}\right)} \end{array}$$

- Heisenberg model

$$H = \sum_j \left( S^x_j S^x_{j+1} + S^y_j S^y_{j+1} + S^z_j S^z_{j+1} + h S^z_j \right)$$




$$D_{\mathrm{mpo}} = 5 \quad \begin{array}{c|ccccc} & 0 & 1 & 2 & 3 & 4 \\ \hline \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{array} & \multicolumn{5}{c}{\left(\begin{array}{ccccc} S^0 & S^x & S^y & S^z & hS^z \\ 0 & 0 & 0 & 0 & S^x \\ 0 & 0 & 0 & 0 & S^y \\ 0 & 0 & 0 & 0 & S^z \\ 0 & 0 & 0 & 0 & S^0 \end{array}\right)} \end{array}$$

G. M. Crosswhite, D. Bacon, Phys. Rev. A 78, 012356 (2008)

# Finite automata for MPO

- for an arbitrary Hamiltonian

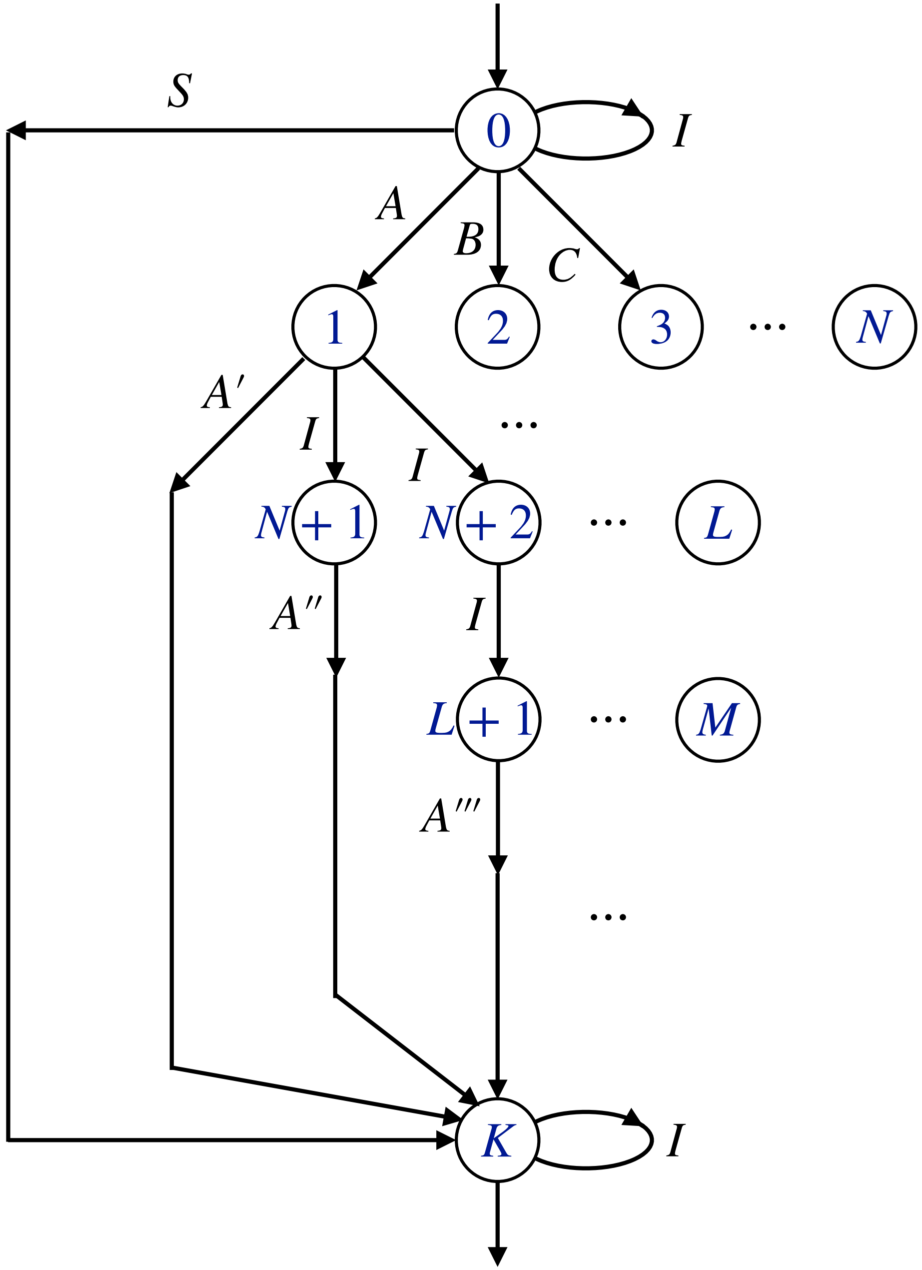
$$H = \sum_i S_i + \sum_i A_i A'_{i+1} + \sum_i B_i B'_{i+1} + \sum_i C_i C'_{i+1} + \cdots + \sum_i A_i A''_{i+2} + \cdots + \sum_i A_i A'''_{i+3} + \cdots$$

(1) initial site 0 and on-site term $S$

(2) add arrow and circle for each $A_i$, $B_i$, $C_i$, …

(3) add arrow and circle for each 2-body term, 3-body term, 4-body term …

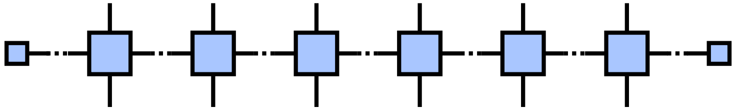
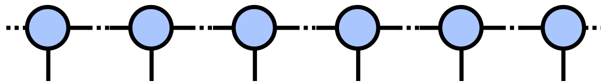
(4) connect to the final site $K$

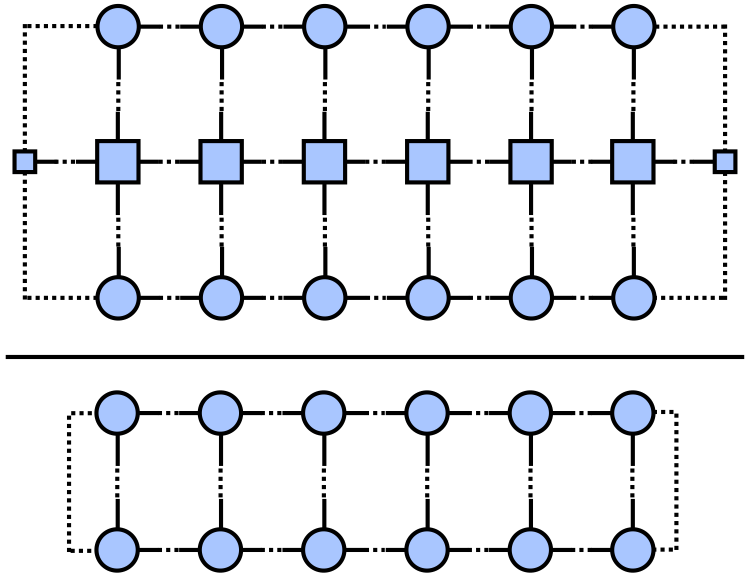



https://www.zhihu.com/question/360201849/answer/1127202896

# Variational MPS algorithm (1-site, OBC)

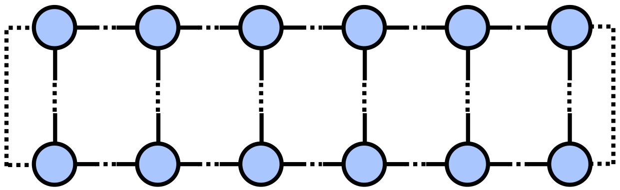
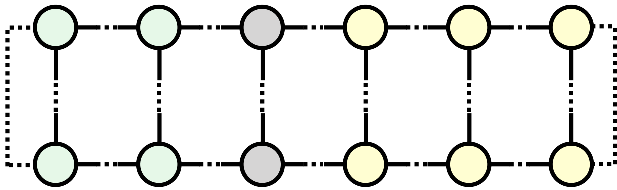
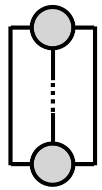
- given a Hamiltonian, we may find its ground state by minimizing its energy
- 1D system with open boundary condition (OBC)

$H =$  $|\psi\rangle =$ 

$E = \frac{\langle\psi| H |\psi\rangle}{\langle\psi|\psi\rangle} =$ 

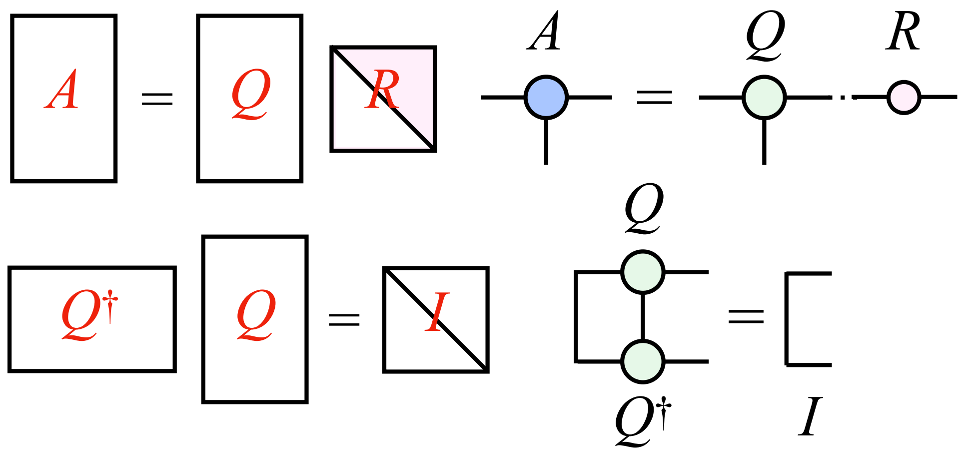
- if we use the canonical form of MPS

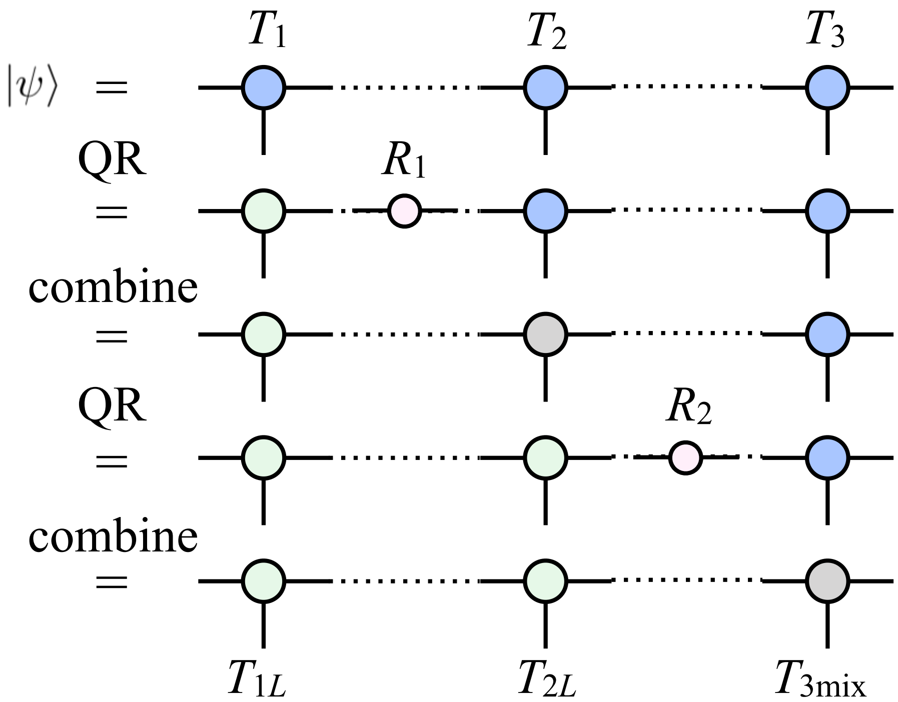
$T_1$ $T_2$ $T_3$ $T_4$ $T_5$ $T_6$ $T_{1L}$ $T_{2L}$ $T_{3\mathrm{mix}}$ $T_{4R}$ $T_{5R}$ $T_{6R}$ $T_{3\mathrm{mix}}$

$\langle\psi|\psi\rangle =$  $=$  $=$  $= 1$

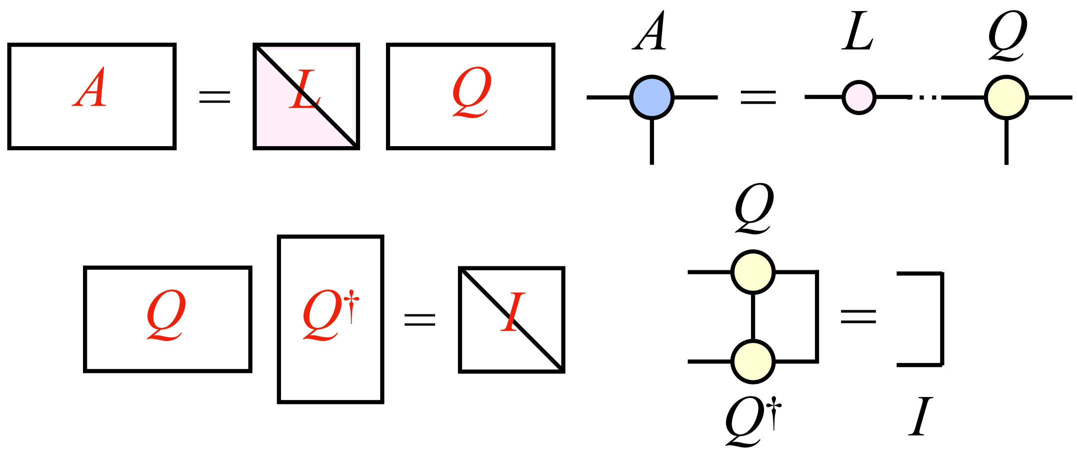
# Find canonical form using QR and LQ

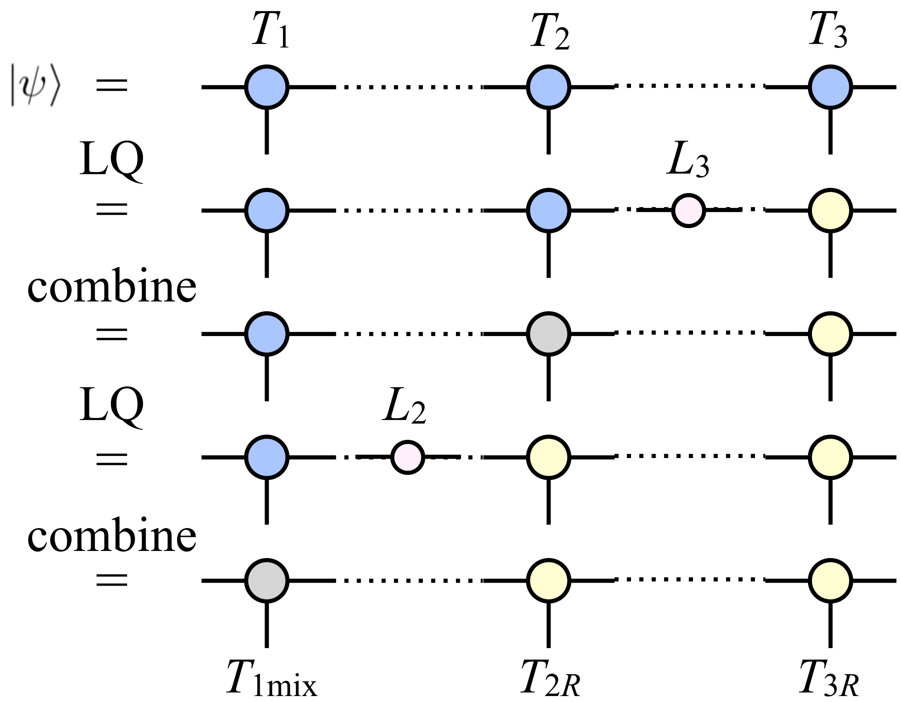
- QR decomposition




- LQ decomposition

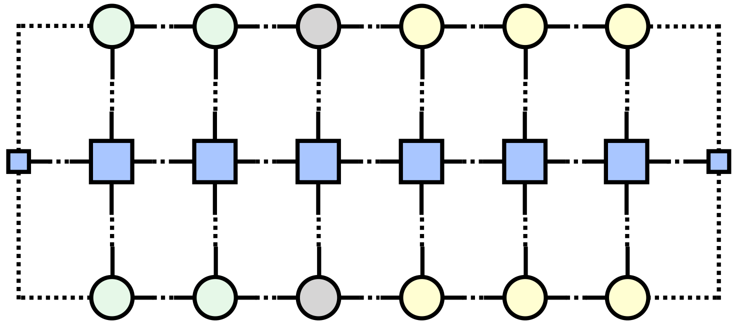


# Variational MPS algorithm (1-site, OBC)

- with open boundary condition and canonical form, we only need to minimize

$$E = \langle\psi| H |\psi\rangle =$$

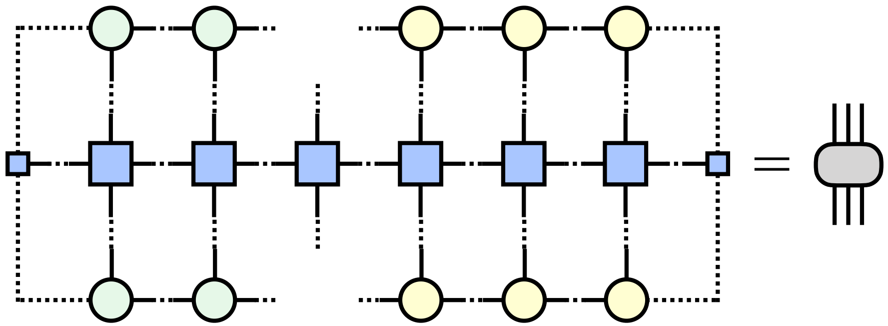


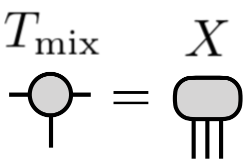
- we may fix all $T_L$ and $T_R$ and only optimize $T_{\text{mix}}$

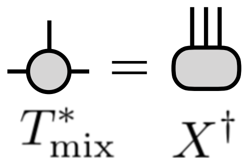
  the optimal $T_{\text{mix}}$ can be found by solving the eigenvalue problem

$$H_{\text{eff}} X = E X$$

$$H_{\text{eff}} =$$





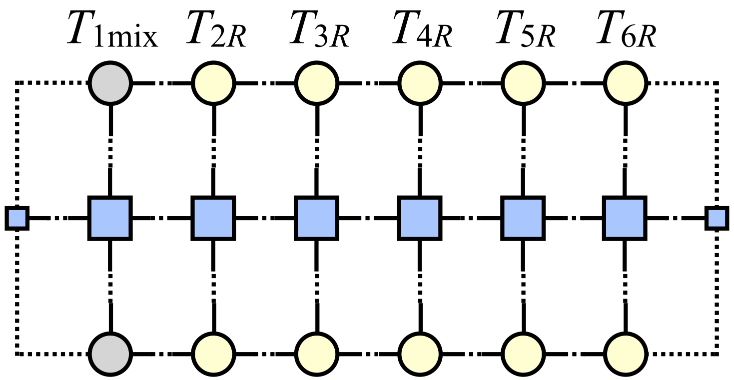




$T_{\text{mix}}$ corresponds to the ground state of $H_{\text{eff}}$

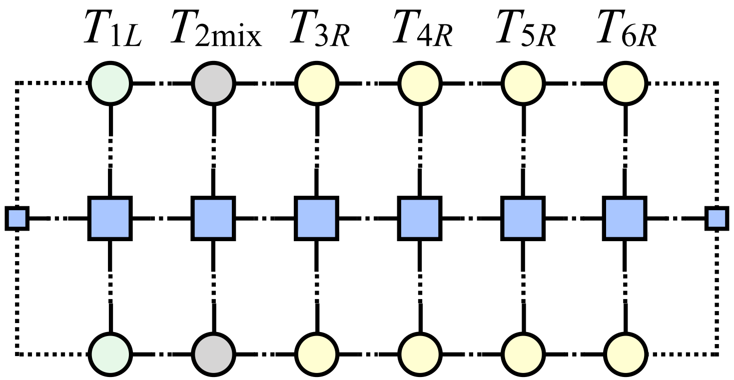
# Variational MPS algorithm (1-site, OBC)

- we optimize each site one by one, sweeping back and forth until convergence
- sweep from left to right:

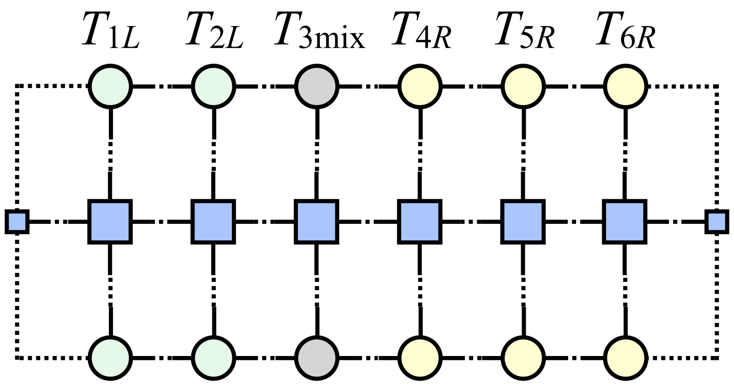
(1)



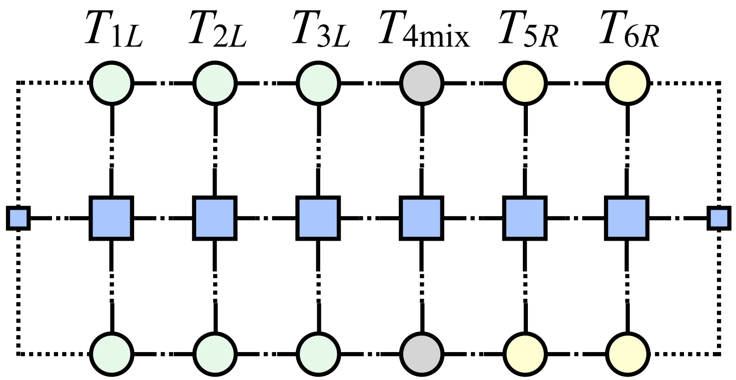

(2)



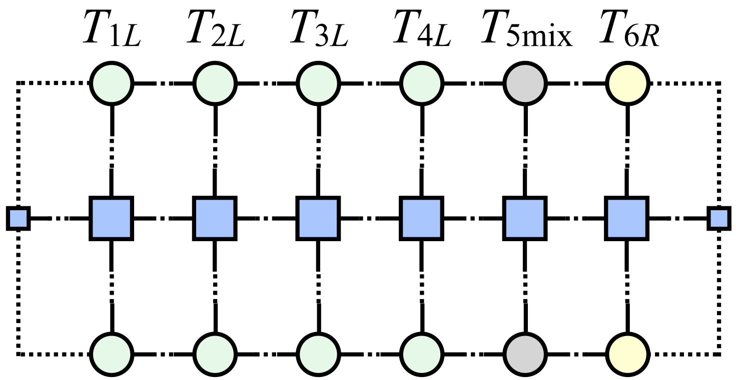

(3)



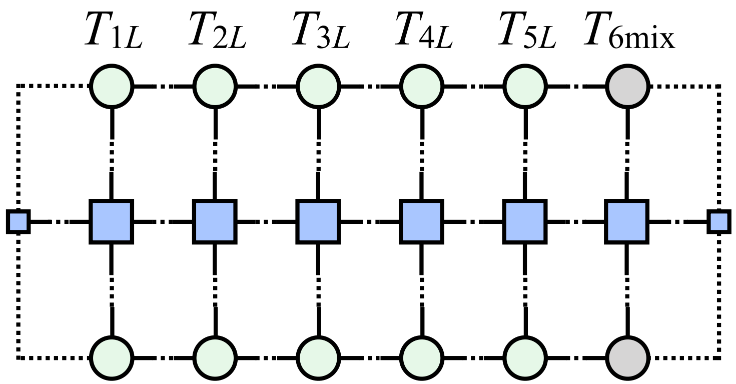

(4)




(5)



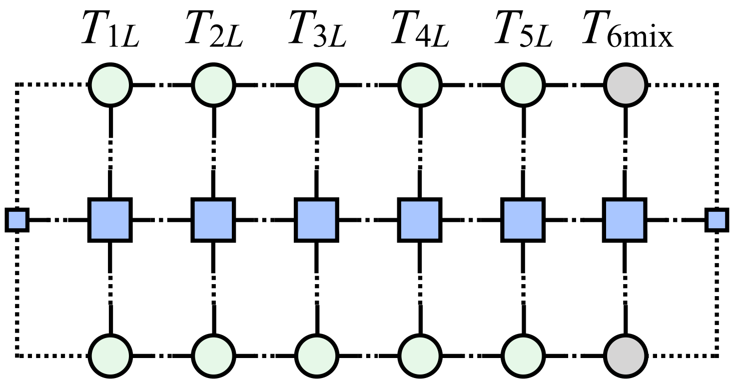

(6)



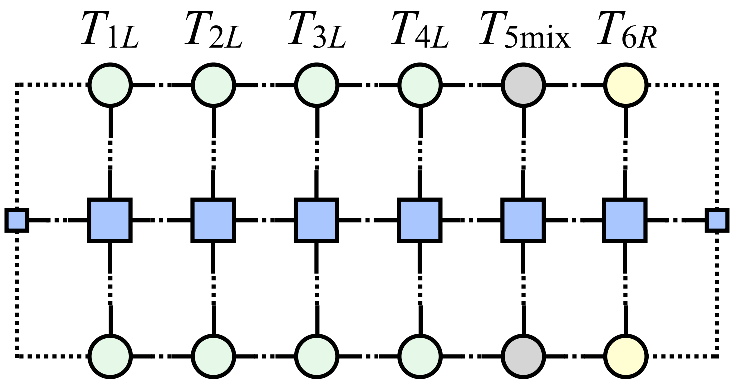
# Variational MPS algorithm (1-site, OBC)

- we optimize each site one by one, sweeping back and forth until convergence
- sweep from right to left:

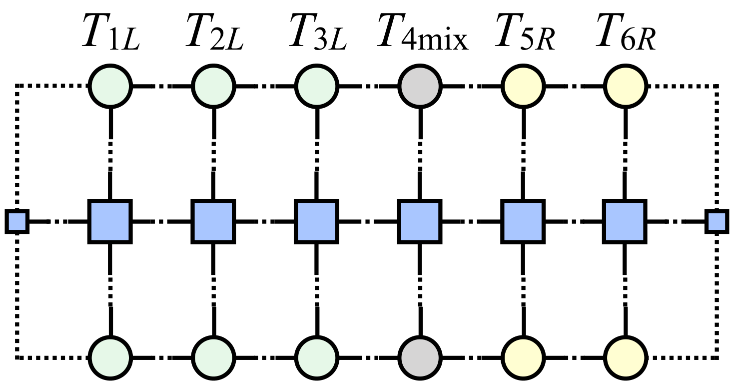
(6)



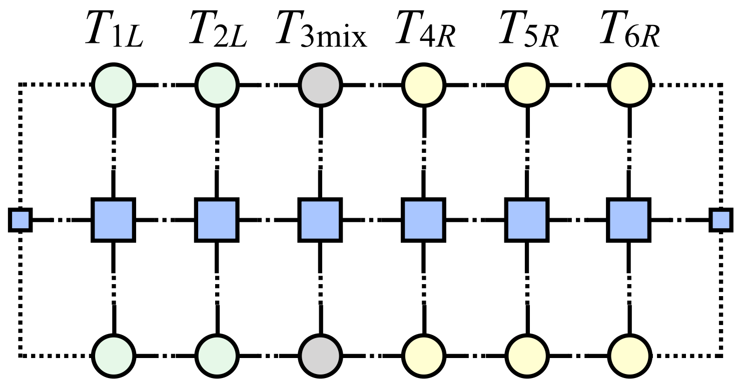

(7)



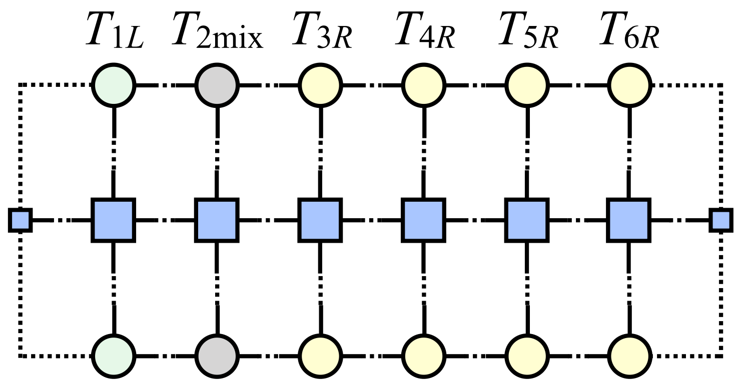

(8)



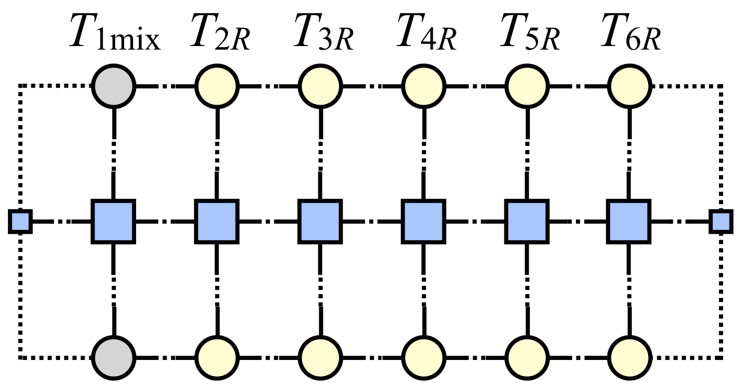

(9)




(10)

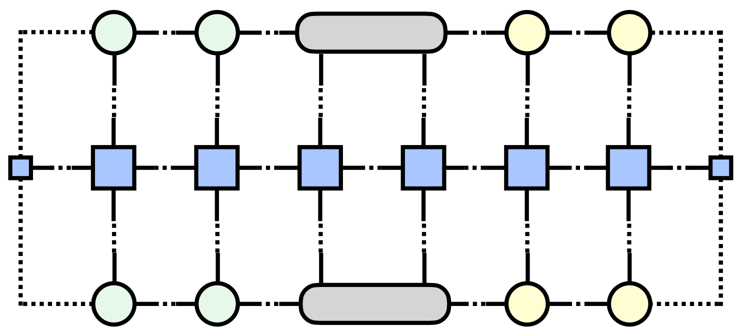



(1)

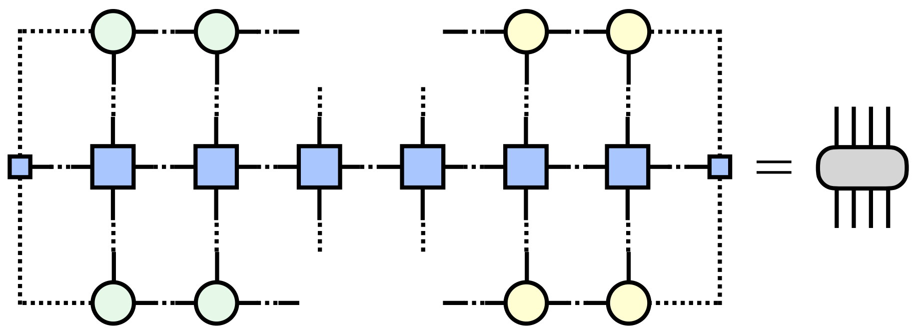


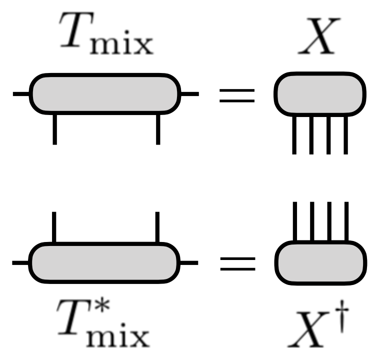
# Variational MPS algorithm (2-site, OBC)

- another option is to optimize two sites at once

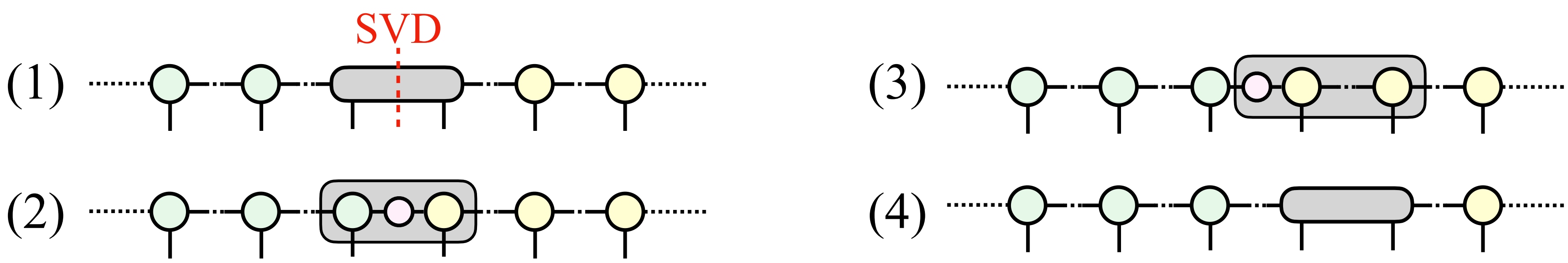
$E = \langle\psi|\, H \,|\psi\rangle =$ 

- the optimal $T_{\mathrm{mix}}$ can be found by solving the eigenvalue problem $H_{\mathrm{eff}} X = E X$

$H_{\mathrm{eff}} =$ 



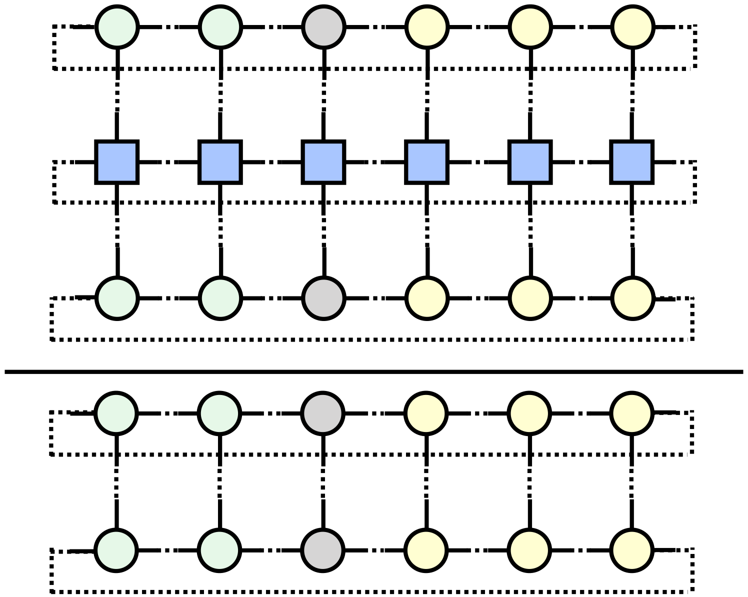

- split $T_{\mathrm{mix}}$ using SVD and move on to the next two sites



# Variational MPS algorithm (1-site, PBC)

- 1D system with periodic boundary condition (PBC)

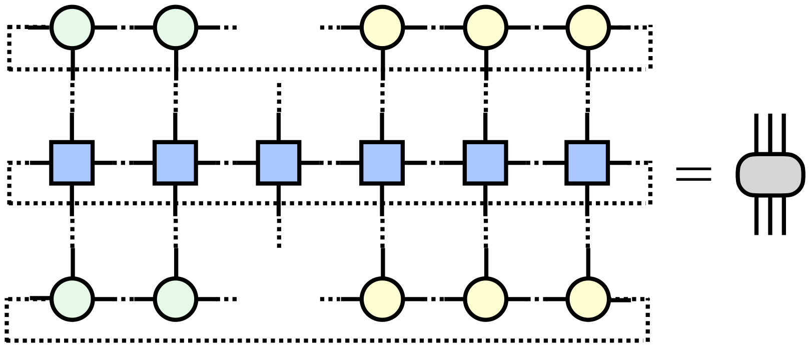
$$E = \frac{\langle\psi|\, H \,|\psi\rangle}{\langle\psi|\, \psi\rangle} =$$



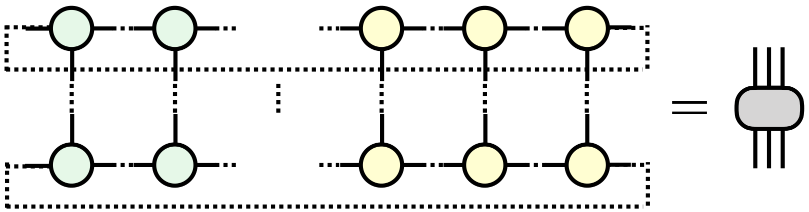
- because of PBC, $\langle\psi|\,\psi\rangle \neq 1$ even if we use the canonical form of MPS
- but we still use the canonical form for stabilization

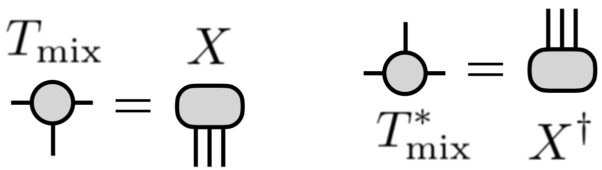
- we need to solve a generalized eigenvalue problem $H_{\mathrm{eff}} X = E N_{\mathrm{eff}} X$

$H_{\mathrm{eff}} =$



$N_{\mathrm{eff}} =$






$T_{\mathrm{mix}}$ corresponds to the ground state of $H_{\mathrm{eff}}$ in the presence of $N_{\mathrm{eff}}$

# Subroutine for tensor operations

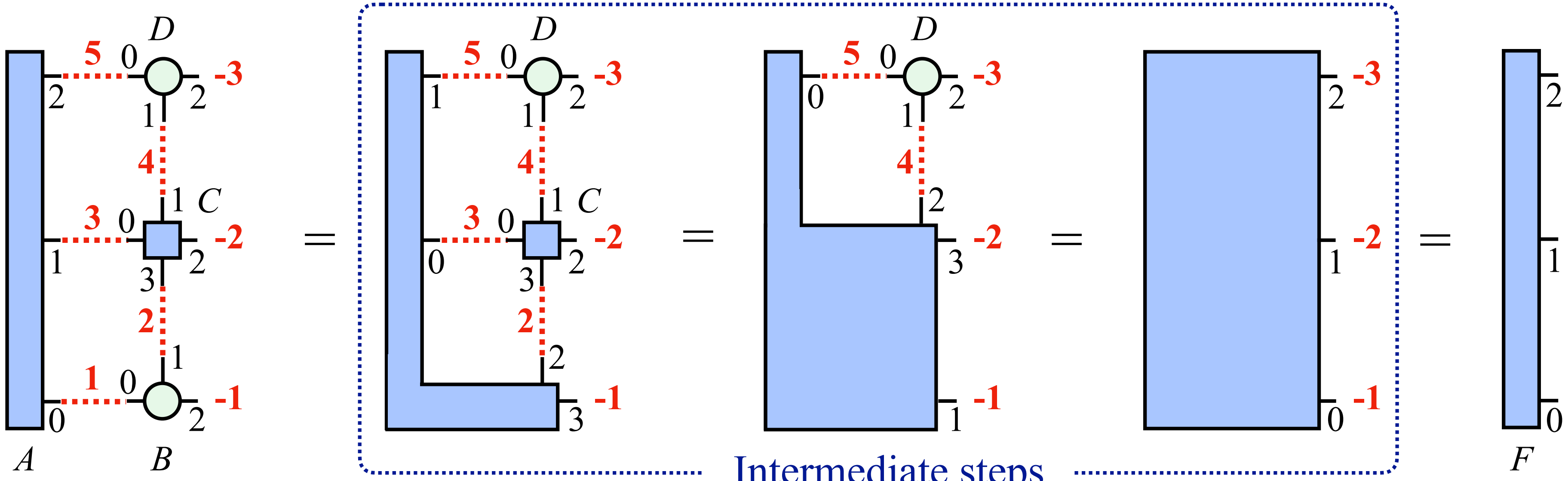
- contract several tensors one after another: NCon(Tensor, Index)

rules: (1) all tensors has its own index order, labeled by 0,1,2, … in black

(2) all indices to be contracted are labeled by positive numbers **1**,**2**,**3**, **…** in red

(3) all indices to be left open are labeled by negative numbers **-1**,**-2**,**-3**, **…** in red, they become the indices of the final tensor with index order 0,1,2, … respectively

(4) tensors are contracted one by one according to the contraction order **1**,**2**,**3**, **…**



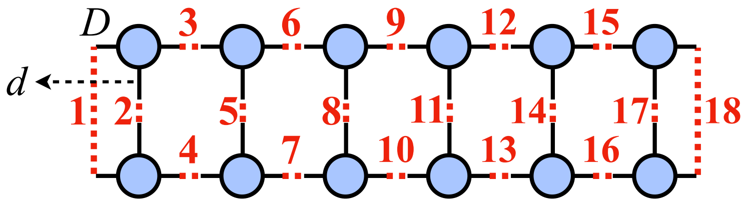

code: $F$ = NCon( [$A$, $B$, $C$, $D$], [ [**1**, **3**, **5**], [**1**, **2**, **-1**], [**3**, **4**, **-2**, **2**], [**5**, **4**, **-3**] ] )

| | | | |
|---|---|---|---|
| 0, 1, 2 | 0, 1, 2 | 0, 1, 2, 3 | 0, 1, 2 |
| from A | from B | from C | from D |

R. N. C. Pfeifer, G. Evenbly, S. Singh, G. Vidal, arXiv:1402.0939

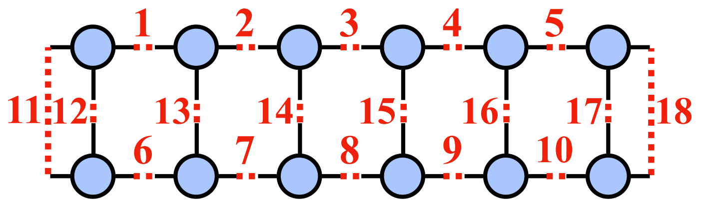
# Subroutine for tensor operations

- contraction order is important



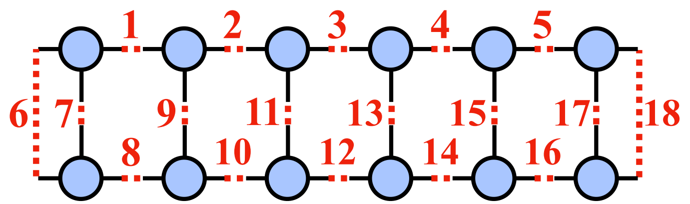

computational cost:
$6dD^3 + dD^2$

best order



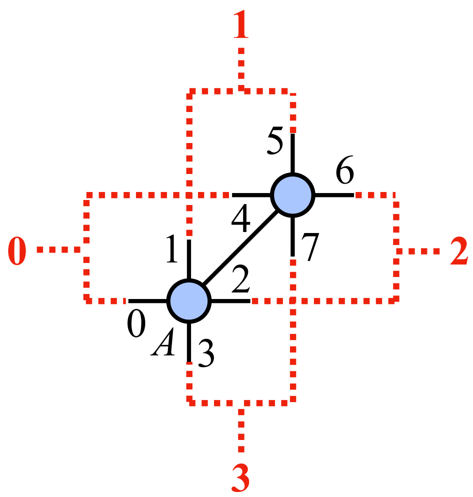

computational cost:
$2\left(d^2 + d^3 + d^4 + d^5 + d^6\right) D^3 + d^6 D^2$

worst order




computational cost:
$2\left(d^2 + d^3 + d^4 + d^5\right) D^3 + d^6 D^3 + dD^2$

- group tensor indices to form a new tensor: Group(A, shapeA)



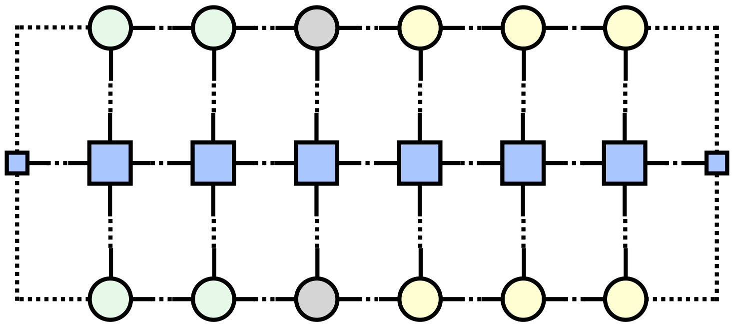

code: $F = \mathrm{Group}(\, A, [\, \underbrace{[0, 4]}_{\text{sub-index from 0}}, \underbrace{[1, 5]}_{\text{sub-index from 1}}, \underbrace{[2, 6]}_{\text{sub-index from 2}}, \underbrace{[3, 7]}_{\text{sub-index from 3}} \,]\,)$

# Code for variational MPS algorithm

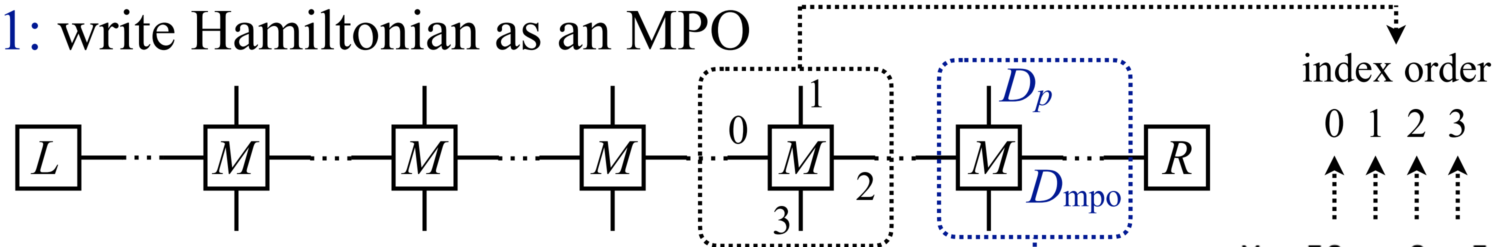
- we consider the spin-1/2 anti-ferromagnetic Heisenberg chain

$$H = \sum_i \left( S_i^x S_{i+1}^x + S_i^y S_{i+1}^y + S_i^z S_{i+1}^z \right) = \sum_i \left[ \frac{1}{2} \left( S_i^+ S_{i+1}^- + S_i^- S_{i+1}^+ \right) + S_i^z S_{i+1}^z \right]$$

- we will find the ground state by minimizing the energy

$E = \langle \psi | H | \psi \rangle =$ 

- Step 1: write Hamiltonian as an MPO

$H =$ 


$$L = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$M = \begin{pmatrix} S^0 & S^+ & S^- & S^z & 0 \\ 0 & 0 & 0 & 0 & S^-/2 \\ 0 & 0 & 0 & 0 & S^+/2 \\ 0 & 0 & 0 & 0 & S^z \\ 0 & 0 & 0 & 0 & S^0 \end{pmatrix}$$

$$R = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \end{pmatrix}^{\mathrm{T}}$$

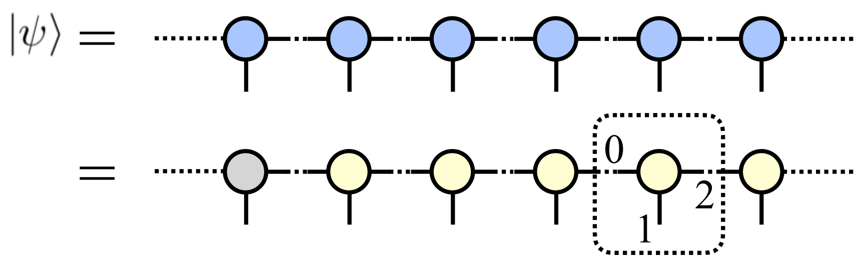
```
def GetMpo_Heisenberg_Obc(Dp):
    S0,Sp,Sm,Sz,Sx,Sy = Sub.SpinOper(Dp)

    Dmpo = 5
    Mpo = np.zeros((Dmpo,Dp,Dmpo,Dp))
```

tensor shape

```
    Mpo[0,:,0,:] = S0
    Mpo[0,:,1,:] = Sp
    Mpo[0,:,2,:] = Sm
    Mpo[0,:,3,:] = Sz
    Mpo[1,:,4,:] = Sm/2.0
    Mpo[2,:,4,:] = Sp/2.0
    Mpo[3,:,4,:] = Sz
    Mpo[4,:,4,:] = S0
```

# Code for variational MPS algorithm

- Step 2: tensor initialization




LQ decompositions start from the right end

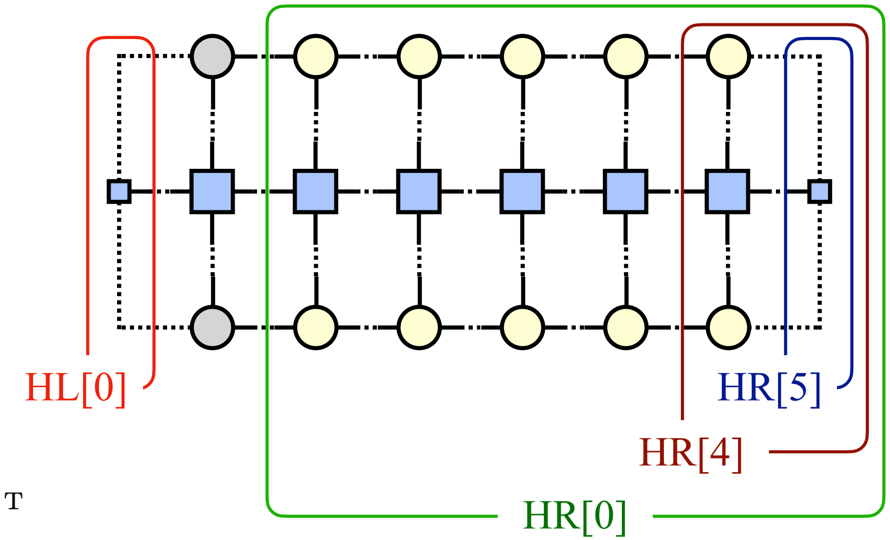
```python
def InitMps(Ns,Dp,Ds):
    T = [None]*Ns
    for i in range(Ns):
        Dl = min(Dp**i,Dp**(Ns-i),Ds)
        Dr = min(Dp**(i+1),Dp**(Ns-1-i),Ds)
        T[i] = np.random.rand(Dl,Dp,Dr)

    U = np.eye(np.shape(T[-1])[-1])
    for i in range(Ns-1,0,-1):
        U,T[i] = Sub.Mps_LQP(T[i],U)

    return T
```

- Step 3: environment initialization

```python
def InitH(Mpo,T):
    Ns = len(T)
    Dmpo = np.shape(Mpo)[0]

    HL = [None]*Ns
    HR = [None]*Ns

    HL[0] = np.zeros((1,Dmpo,1))
    HL[0][0,0,0] = 1.0
    HR[-1] = np.zeros((1,Dmpo,1))
    HR[-1][0,-1,0] = 1.0

    for i in range(Ns-1,0,-1):
        HR[i-1] =
Sub.NCon([HR[i],T[i],Mpo,np.conj(T[i])],[[1,3,5],[-1,2,1],[-2,2,3,4],[-3
,4,5]])

    return HL,HR
```

$L = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \end{pmatrix}$

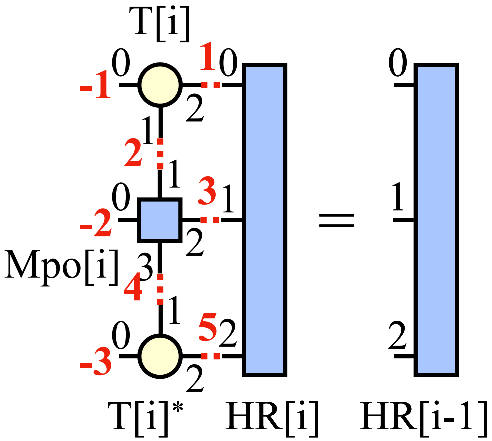
$R = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 \end{pmatrix}^{\mathrm{T}}$




HR[5] → HR[4] → … → HR[0]

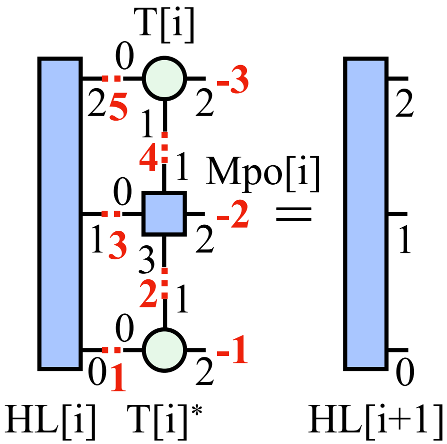
# Code for variational MPS algorithm

- update HR: HR[i] → HR[i-1]




```
HR[i-1] =
Sub.NCon([HR[i],T[i],Mpo,np.conj(T[i])],[[1,3,5],[-1,2,1],[-2,2,3,4],[-3
,4,5]])
```

- update HL: HL[i] → HL[i+1]




```
HL[i+1] =
Sub.NCon([HL[i],np.conj(T[i]),Mpo,T[i]],[[1,3,5],[1,2,-1],[3,4,-2,2],[5,
4,-3]])
```

# Code for variational MPS algorithm

- Step 4: sweep back and forth to optimize each tensor

```
def OptT(Mpo,HL,HR,T):
    Ns = len(T)
    Eng0 = np.zeros(Ns)
    Eng1 = np.zeros(Ns)

    for r in range(100):
        print r

        for i in range(Ns-1):
            T[i],Eng1[i] = OptTSite(Mpo,HL[i],HR[i],T[i],Method=1)
            # print i,Eng1[i]
            T[i],U = Sub.Mps_QR0P(T[i])
            HL[i+1] = 
Sub.NCon([HL[i],np.conj(T[i]),Mpo,T[i]],[[1,3,5],[1,2,-1],[3,4,-2,2],[5,
4,-3]])
            T[i+1] = np.tensordot(U,T[i+1],(1,0))

        for i in range(Ns-1,0,-1):
            T[i],Eng1[i] = OptTSite(Mpo,HL[i],HR[i],T[i],Method=1)
            # print i,Eng1[i]
            U,T[i] = Sub.Mps_LQ0P(T[i])
            HR[i-1] = 
Sub.NCon([HR[i],T[i],Mpo,np.conj(T[i])],[[1,3,5],[-1,2,1],[-2,2,3,4],[-3
,4,5]])
            T[i-1] = np.tensordot(T[i-1],U,(2,0))

        print Eng1
        if abs(Eng1[1]-Eng0[1]) < 1.0e-7:
            break
        Eng0 = copy.copy(Eng1)

    print Eng1/float(Ns)

    return T
```

sweep back and forth for r times

sweep from left to right

- optimize tensor T[i]
- change T[i] to canonical form
- update HL for the next site
- update initial guess for next T

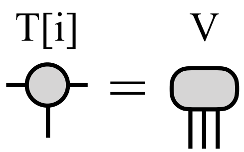
sweep from right to left

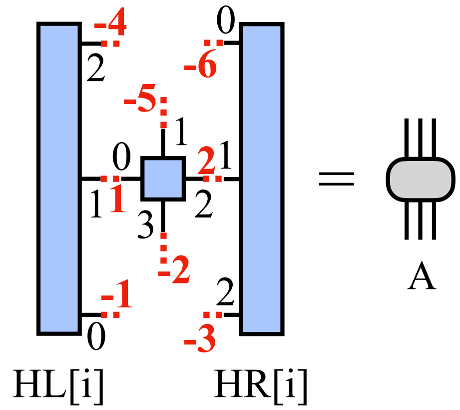
- optimize tensor T[i]
- change T[i] to canonical form
- update HR for the next site
- update initial guess for next T

stop if energy is converged

# Code for variational MPS algorithm

- optimize each T[i]



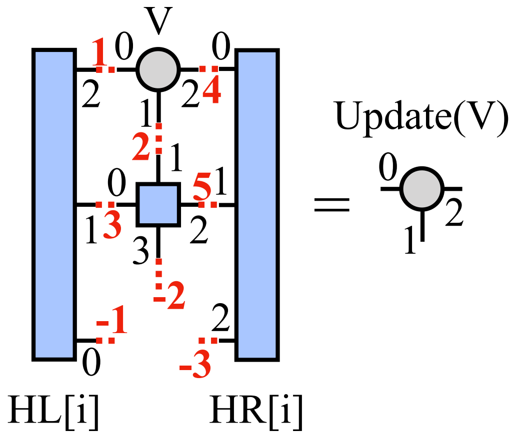

```python
def OptTSite(Mpo,HL,HR,T,Method=0):
    DT = np.shape(T)
    Dl = np.prod(DT)
```

direct method

```python
    if Method == 0:
        A = Sub.NCon([HL,Mpo,HR],[[-1,1,-4],[1,-5,2,-2],[-6,2,-3]])
        A = Sub.Group(A,[[0,1,2],[3,4,5]])
        Eig,V = LAs.eigsh(A,k=1,which='SA')
        T = np.reshape(V,DT)
```

iterative method

```python
    if Method == 1:
        def UpdateV(V):
            V = np.reshape(V,DT)
            V =
Sub.NCon([HL,V,Mpo,HR],[[-1,3,1],[1,2,4],[3,2,5,-2],[4,5,-3]])
            V = np.reshape(V,[Dl])
            return V

        V0 = np.reshape(T,[Dl])

        MV = LAs.LinearOperator((Dl,Dl),matvec=UpdateV)
        Eig,V = LAs.eigsh(MV,k=1,which='SA',v0=V0)
        # print Eig
        T = np.reshape(V,DT)
        Eig = np.real(Eig)

    return T,Eig
```



# Code for variational MPS algorithm

- main part

```python
if __name__ == "__main__":
    Ns = 6
    Dp = 2
    Ds = 4

    Mpo = GetMpo_Heisenberg_Obc(Dp)
    T = InitMps(Ns,Dp,Ds)
    HL,HR = InitH(Mpo,T)
    T = OptT(Mpo,HL,HR,T)
```

Ns = 6: system has 6 sites

Dp = 2: spin-1/2 system, 2 physical basis

Ds = 4: maximal bond dimension is 4

Step 1 to Step 4

- results

```
0
[-0.18161427 -2.49049059 -2.49049059 -2.47873556 -2.47873556 -2.47873556]
1
[-2.49049059 -2.49084768 -2.49084768 -2.49083341 -2.49083341 -2.49083341]
2
[-2.49084768 -2.49092554 -2.49092554 -2.49085865 -2.49085865 -2.49085865]
3
[-2.49092554 -2.49204873 -2.49204873 -2.49127928 -2.49127928 -2.49127928]
4
[-2.49204873 -2.49252309 -2.49252309 -2.49244379 -2.49244379 -2.49244379]
5
[-2.49252309 -2.492541   -2.492541   -2.49253797 -2.49253797 -2.49253797]
6
[-2.492541   -2.49254176 -2.49254176 -2.49254162 -2.49254162 -2.49254162]
7
[-2.49254176 -2.49254179 -2.49254179 -2.49254178 -2.49254178 -2.49254178]
[-0.41542363 -0.41542363 -0.41542363 -0.41542363 -0.41542363 -0.41542363]
```

total energy after each sweep

after 7 sweeps, energy is converged

converged energy per site

# Homework rules

- this homework has only one task

  please submit through http://learn.tsinghua.edu.cn

- please submit the source code and a detailed note including:

  (1) a brief summary of theory and algorithm

  (2) the structure and technical details of the code

  (3) problems encountered and solutions

  (4) summary of results and your understanding

- deadline is Nov. 17, 23:00

  if you submit after the deadline, you will have less points

- if you are an expert in variational MPS method, this homework may be waived

  please contact me in person and submit something you have done before

# Homework

- consider a 1D spin model with open boundary condition

$$H = \sum_j \left[ -(\sigma_j^x + \sigma_j^z \sigma_{j+1}^z) + g(\sigma_j^x \sigma_{j+1}^z \sigma_{j+2}^z + \sigma_j^z \sigma_{j+1}^z \sigma_{j+2}^x) \right]$$

where $\sigma^x$, $\sigma^y$ and $\sigma^z$ are Pauli matrices

for single-body terms, $j = 1, \cdots, N$

for two-body terms, $(j,\ j+1) = (1,\ 2),\ (2,\ 3),\ \cdots\ ,\ (N-1,\ N)$

for three-body terms, $(j,\ j+1,\ j+2) = (1,\ 2,\ 3),\ (2,\ 3,\ 4),\ \cdots\ ,\ (N-2,\ N-1,\ N)$

(1) write down the MPO for this Hamiltonian, draw a finite automata figure (Page 4)

(2) write a code for the 2-site variational MPS method (Page 11)

(3) choose $N = 10$, $g = 0.428$, $D_s = 4$ and 6, calculate the ground state energy $E$ per site, and the magnetization per site $\langle \sigma_i^z \rangle$ and $\langle \sigma_i^x \rangle$

(4) compare with the results obtained from exact diagonalization and 1-site variational MPS method

# Partial sample output

- please note the input parameters are different from homework

```
(Ns,Ds,g)= (6, 4, 1)
==================== 2-site ====================

(2) Energy per Site: [-1.16342161 -1.16342161 -1.16342161 -1.16415328 -1.16342161 -1.16342161]
(2) Energy average: -1.163543555161248

(3) Sigma_X per Site: [0.9148724 +0.00000000e+00j 0.76888575-5.55111512e-17j
 0.53312368+2.62376926e-17j 0.53171004-2.46519033e-32j
 0.76888575+3.46944695e-18j 0.9148724 +9.86076132e-32j]
(3) Sigma_X average: (0.7387250020058149-4.300668617525797e-18j)

(4) Sigma_Z per Site: [-3.89201475e-16+4.93038066e-32j -3.65057732e-15+2.46283300e-17j
  1.00188412e-15-6.72205347e-18j -3.08585622e-15+2.46519033e-31j
  1.38777878e-15-6.93889390e-18j -8.32667268e-16+1.23259516e-32j]
(4) Sigma_Z average: (-9.28106564628587e-16+1.8278971001748395e-18j)

==================== 1-site ====================

(2) Energy per Site: [-1.16345424 -1.16345424 -1.16345424 -1.16345424 -1.16345424 -1.16345424]
(2) Energy average: -1.1634542441033482

(3) Sigma_X per Site: [0.91662215-2.77555756e-17j 0.7706066 +0.00000000e+00j
 0.53659188+0.00000000e+00j 0.53659195+1.38777878e-17j
 0.77060656+0.00000000e+00j 0.91662215+0.00000000e+00j]
(3) Sigma_X average: (0.7412735480680963-2.3129646346357427e-18j)

(4) Sigma_Z per Site: [-1.50475832e-08+8.67361738e-19j -1.12981599e-08+3.12250226e-17j
 -1.65477624e-08+4.51028104e-17j  4.18026120e-09+1.89735380e-17j
 -4.54925561e-09+1.73472348e-18j -3.67335007e-09-1.38777878e-17j]
(4) Sigma_Z average: (-7.82264166021209e-09+1.4004278061271098e-17j)

==================== ED ====================

(2) Energy per Site: -1.1641532762479179

(3) Sigma_X per Site: [(0.9101004634715449+0j), (0.7663268746827528+0j), (0.5314901295323278+0j), (0.53149012953232
84+0j), (0.7663268746827523+0j), (0.9101004634715452+0j)]
(3) Sigma_X average: (0.7359724892288753+0j)

(4) Sigma_Z per Site: [(-4.8433479449272454e-15+0j), (-2.1371793224034263e-15+0j), (-3.2959746043559335e-15+0j),
(1.4155343563970746e-15+0j), (-4.3021142204224816e-15+0j), (-4.08006961549745e-15+0j)]
(4) Sigma_Z average: (-2.87385855853491e-15+0j)
```

# Grading of this homework

- academic integrity [+50]

  completing assignments independently, creating and expressing your own ideas, DON'T copy answers from others or allow others to copy your answers

- the source code can be executed [+10] and can provide correct results [+10]

  the source code has high readability [+10]

- there is a detailed note file to explain the basic principle, source code and results [+10]

  the note file is well-written and easy to understand [+10]

- did not show MPO and the finite automata figure [-20]

  did not use the 2-site variational MPS method [-20]

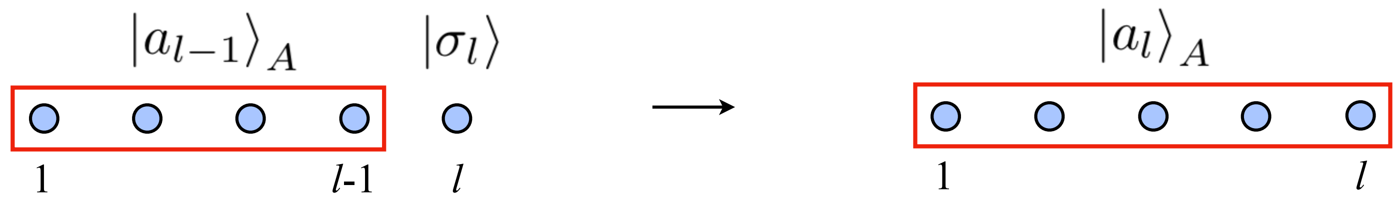
  did not compare with ED and 1-site variational MPS method [-20]

- time-dependent score: after DDL: [-1 per day]

- self-motivation score: new ideas or extra results [+10]

# Equivalence between traditional finite-size DMRG and variational MPS

- in the traditional DMRG scheme, we grow blocks while decimating basis



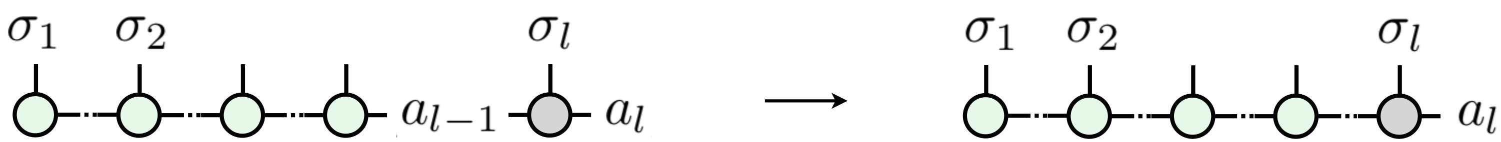

$$|a_l\rangle = \sum_{a_{l-1},\sigma_l} \langle a_{l-1}, \sigma_l | \, a_l\rangle \, |a_{l-1}\rangle \, |\sigma_l\rangle = \sum_{a_{l-1},\sigma_l} M^{\sigma_l}_{a_{l-1},a_l} \, |a_{l-1}\rangle \, |\sigma_l\rangle$$

- simple rearrangement of expansion coefficients into matrices

$$M^{\sigma_l}_{a_{l-1},a_l} = \langle a_{l-1}, \sigma_l | \, a_l\rangle$$

- recursion easily expressed as matrix multiplication

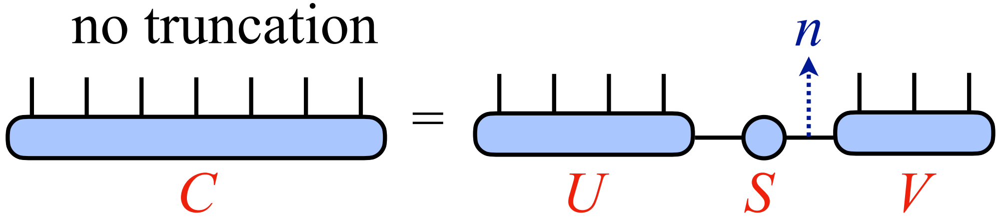
$$|a_l\rangle = \sum_{\sigma_1,\sigma_2,\cdots,\sigma_l} \left(M^{\sigma_1} M^{\sigma_2} \cdots M^{\sigma_l}\right)_{1,a_l} |\sigma_1, \sigma_2, \cdots, \sigma_l\rangle$$

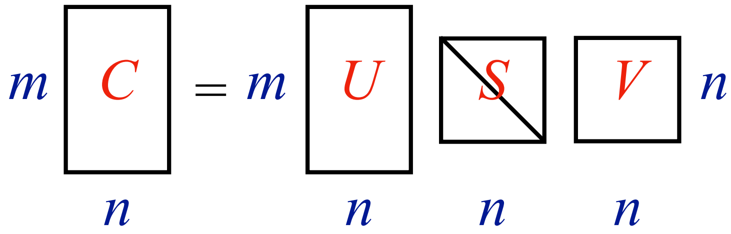


# Truncation method: MPS vs DMRG

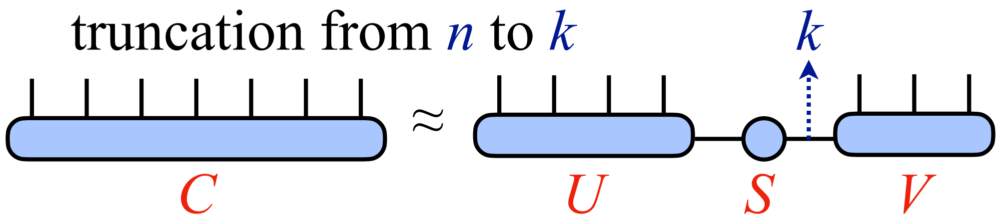
- variational MPS: truncation using singular value decomposition (SVD)

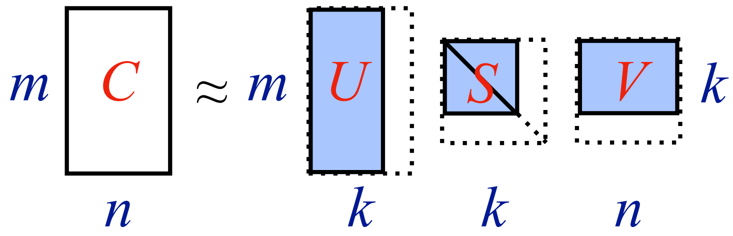
no truncation



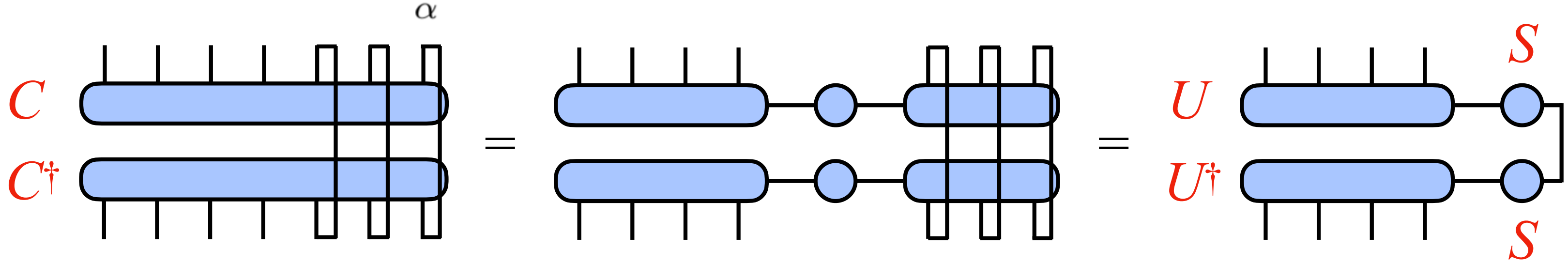

truncation from $n$ to $k$



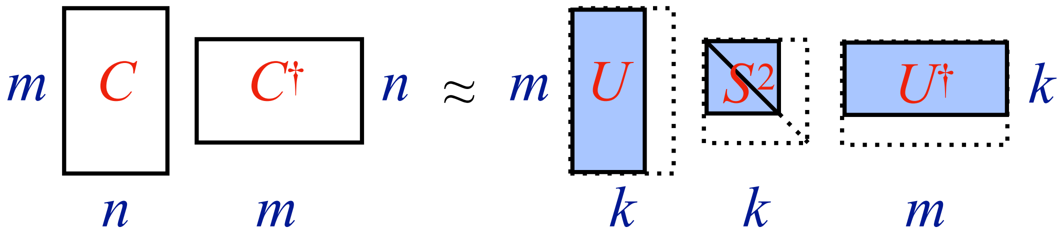




- traditional DMRG: truncation using eigenvalue decomposition (EVD)

$$\rho_A = \mathrm{Tr}_B \rho = \sum_\alpha s_\alpha^2 \, |u^\alpha\rangle_A \, {}_A\langle u^\alpha| = CC^\dagger = USVV^\dagger SU^\dagger = US^2U^\dagger$$

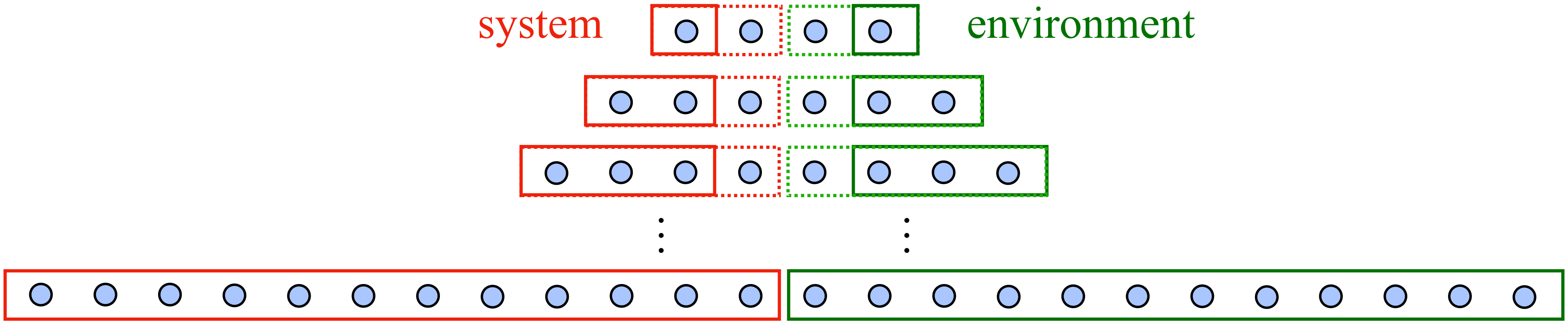



truncation from $n$ to $k$

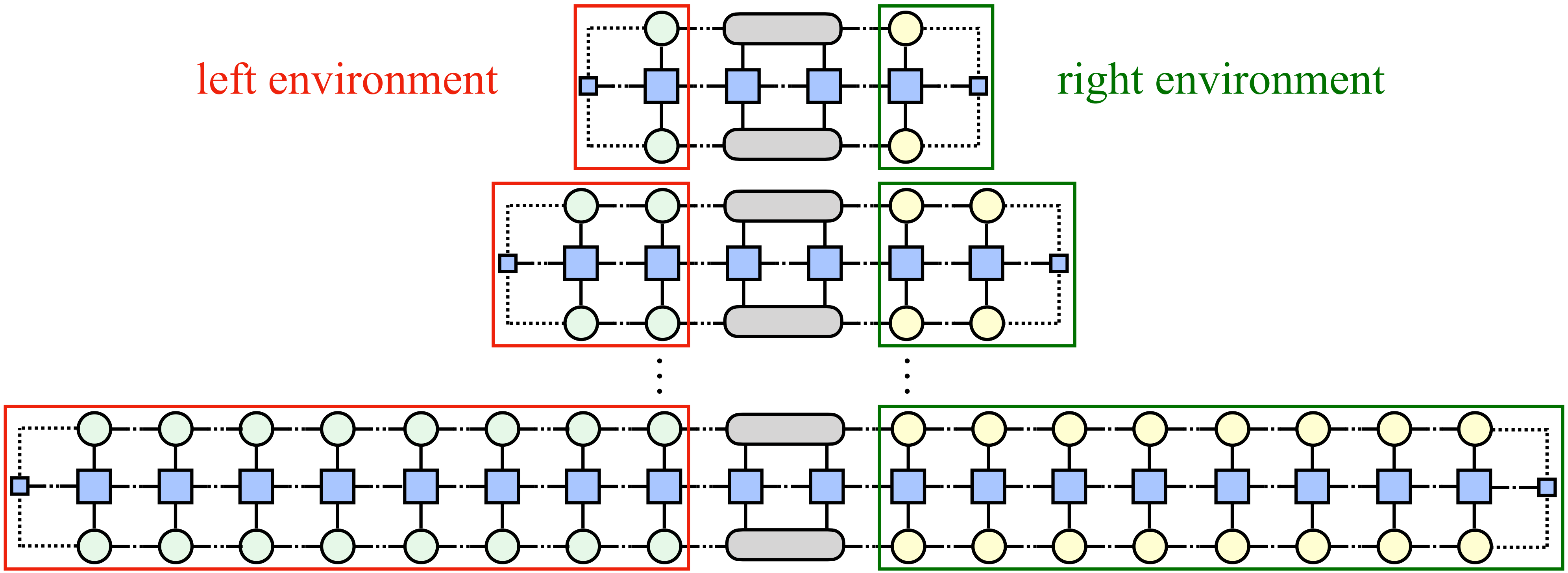


# iDMRG in the age of MPS

- traditional infinite-size DMRG (iDMRG)

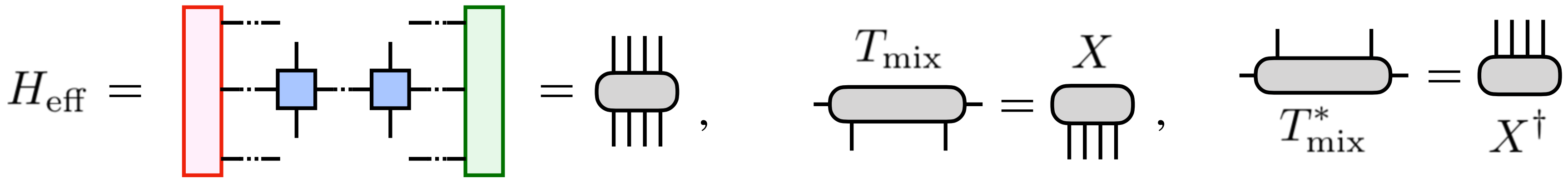



- iDMRG based on 2-site update MPS

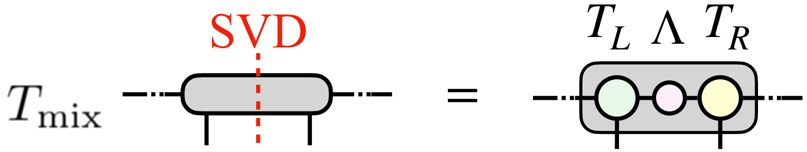


# iDMRG in the age of MPS

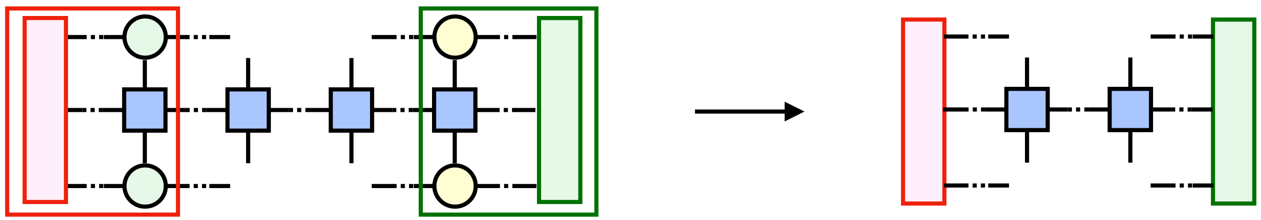
- the optimal $T_{\text{mix}}$ can be found by solving the eigenvalue problem $H_{\text{eff}} X = E X$




- we use SVD to split $T_{\text{mix}}$ into three parts




- $T_L$ is absorbed into left environment, and $T_R$ is absorbed into right environment
  the environments are updated, the effective system size is enlarged
  in the next iteration, we use new environments to find out new $T_{\text{mix}}$



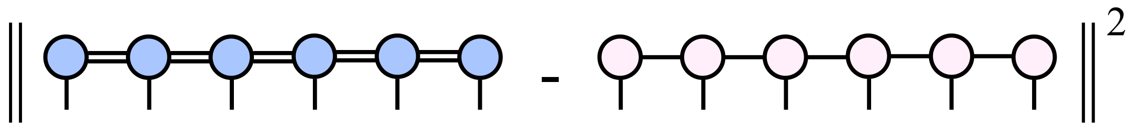
- the unit cell $T_{\text{mix}}$ may contain an arbitrary number of sites

I.P. McCulloch, arXiv: 0804.2509.

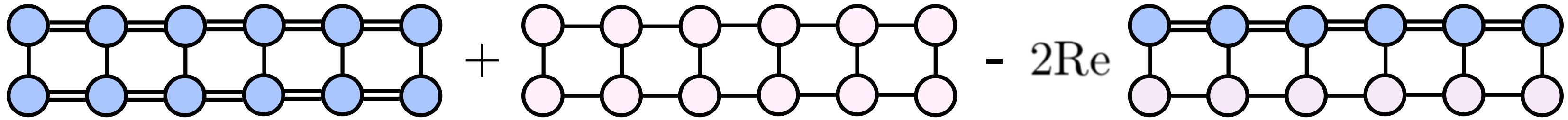
# Variational-based MPS compression

- we try to minimize the cost function

$$\varepsilon = \| |\psi_A\rangle - |\psi_B\rangle \|^2 =$$



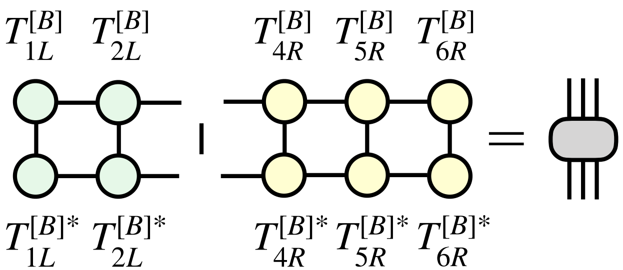

$$= \langle \psi_A | \psi_A \rangle + \langle \psi_B | \psi_B \rangle - 2\mathrm{Re} \langle \psi_B | \psi_A \rangle$$

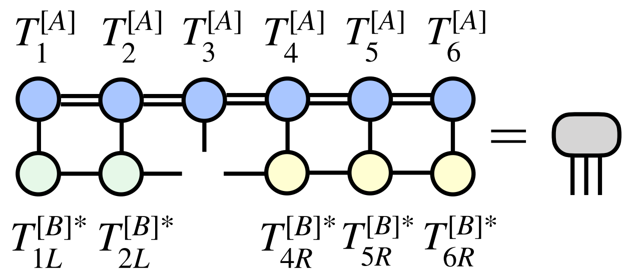
$=$ 

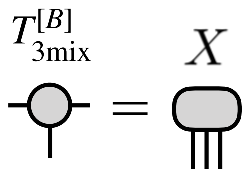

- we use the canonical form of MPS

  the optimal $T$ is given by the solution of the linear equation

$$N_{\text{eff}} X = W_{\text{eff}}$$

$N_{\text{eff}} =$ 


$W_{\text{eff}} =$ 


- we optimize each tensor, sweeping back and forth until convergence