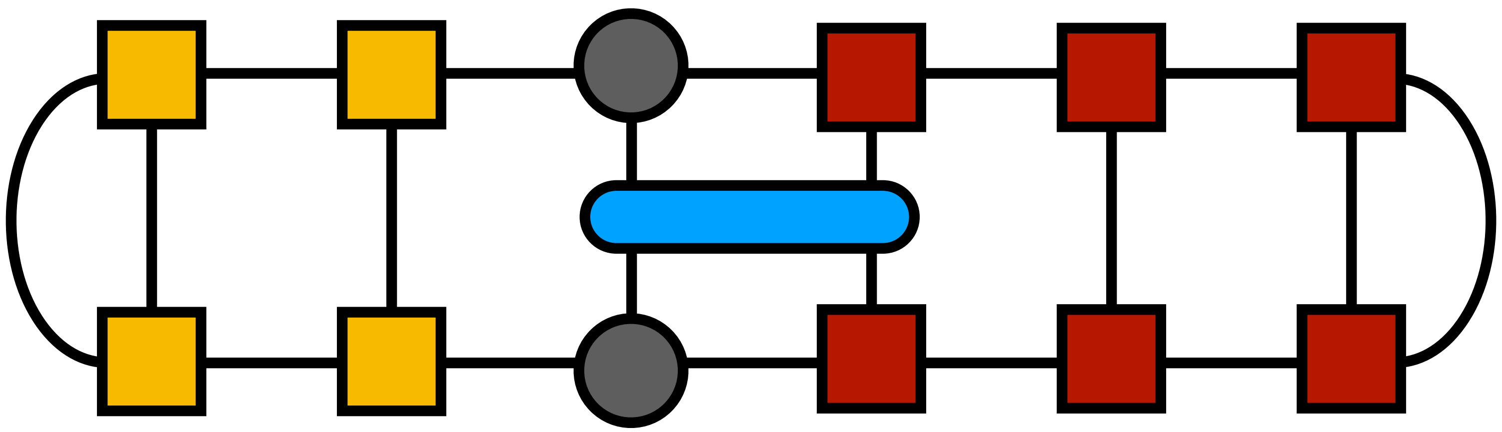
Tensor Networks and Applications



E.M. Stoudenmire

Apr 2018 - Sao Paulo

FLATIRON INSTITUTE
SIMONS FOUNDATION

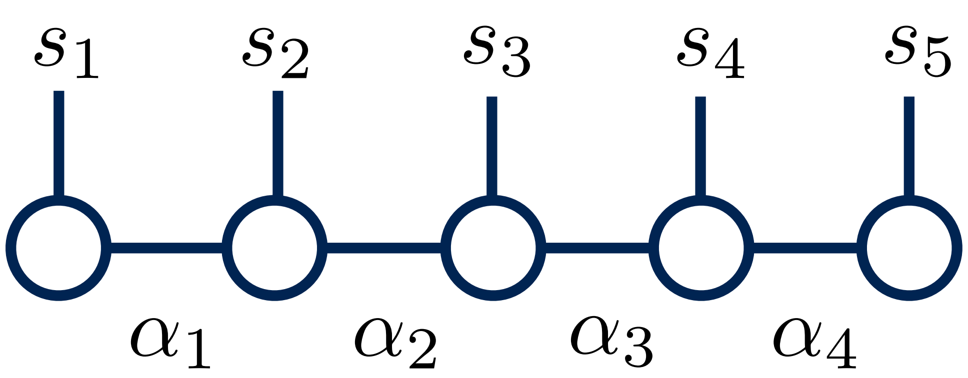
Review of Previous Lecture

Motivated *matrix product state* (MPS) ansatz for ground states

$$\Psi^{s_1 s_2 s_3 s_4 s_5} = M_1^{s_1} M_2^{s_2} M_3^{s_3} M_4^{s_4} M_5^{s_5}$$

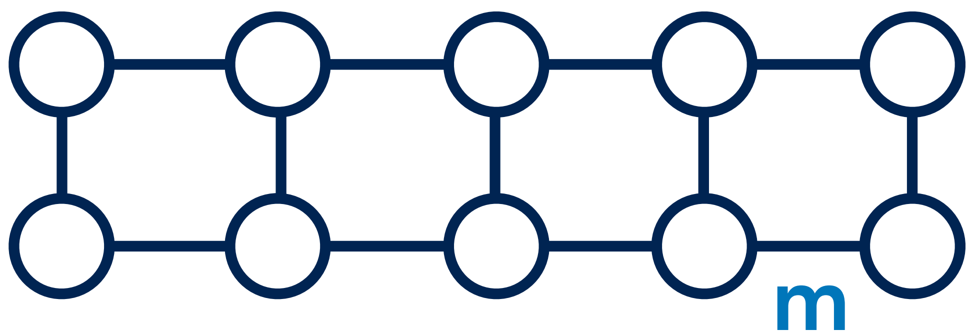
$$= \sum_{\{\alpha\}} M_{\alpha_1}^{s_1} M_{\alpha_1 \alpha_2}^{s_2} M_{\alpha_2 \alpha_3}^{s_3} M_{\alpha_3 \alpha_4}^{s_4} M_{\alpha_4}^{s_5}$$

$$=$$

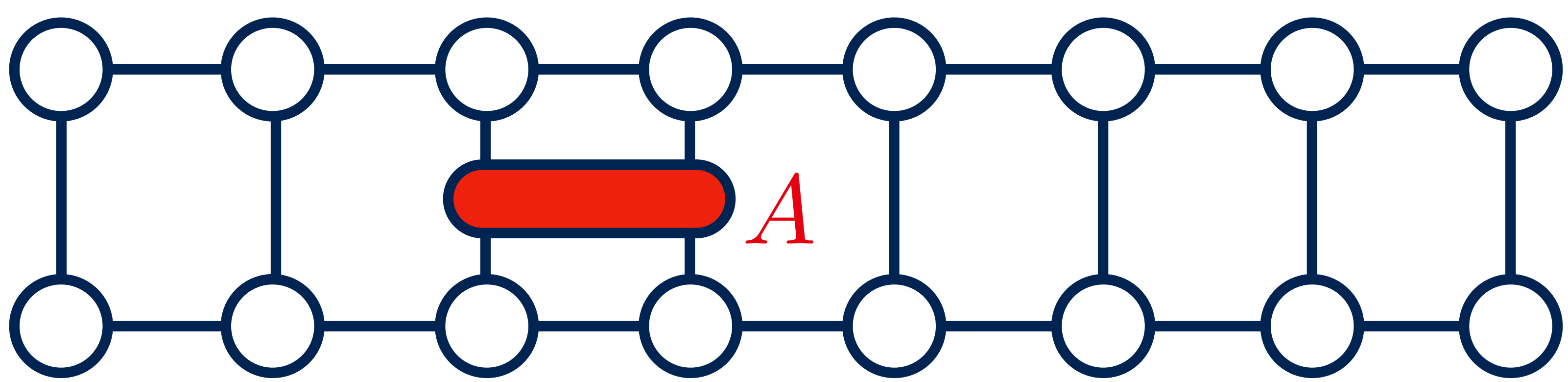


Calculations of MPS with bond dimension **m** scale as **m**^{3}

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

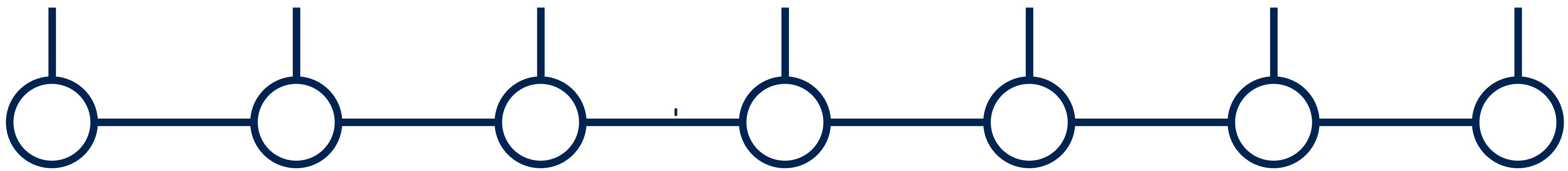


$$\langle \Psi | A | \Psi \rangle =$$

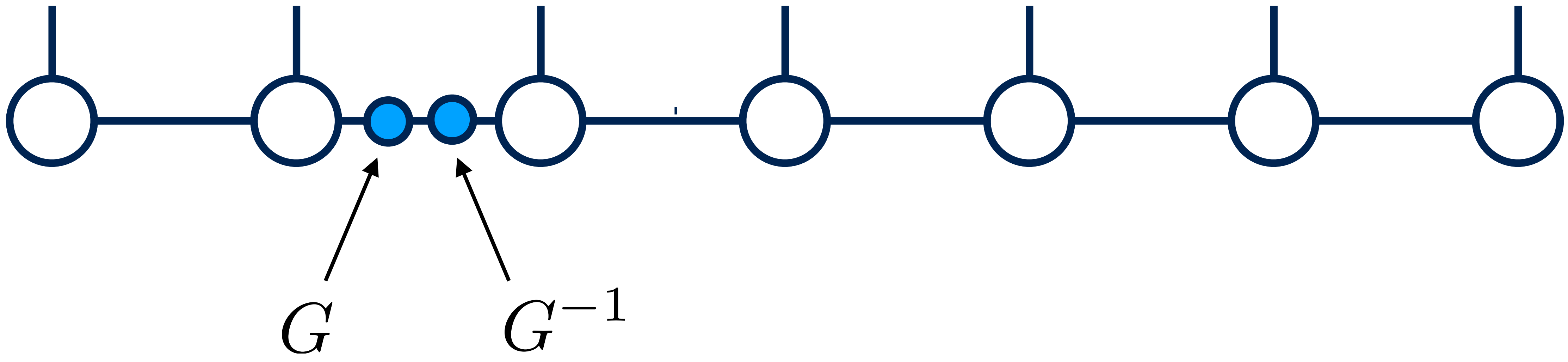


Gauging Matrix Product States

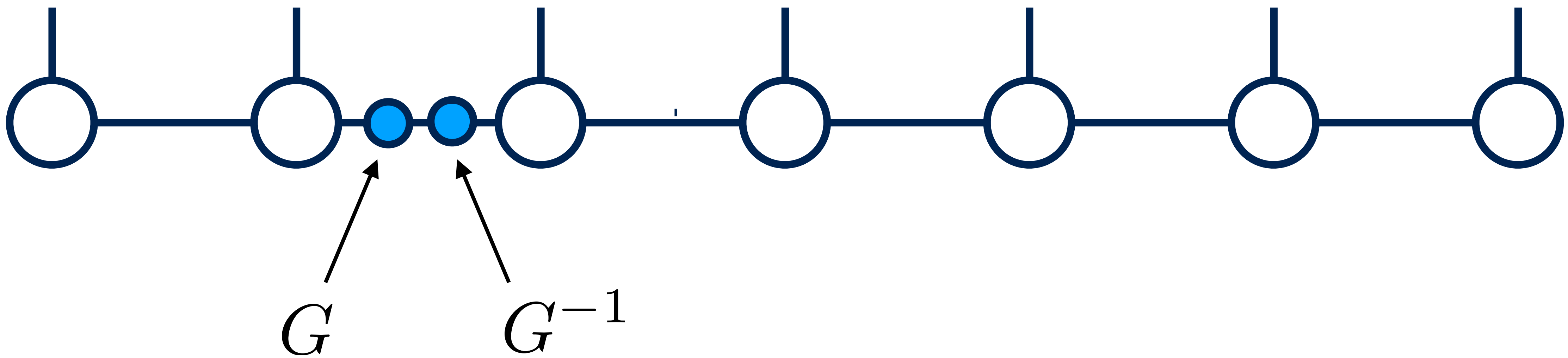
Matrix product state representation of a state is *highly redundant*

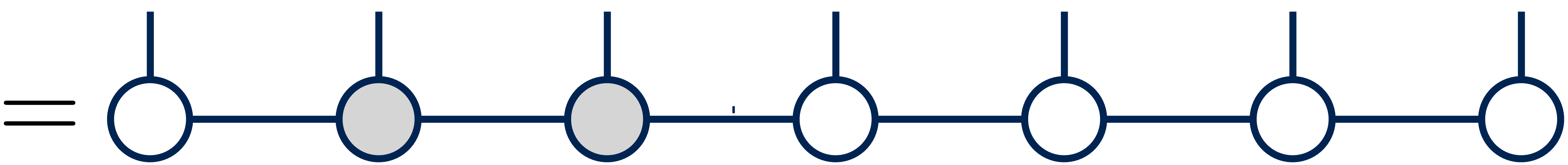


Matrix product state representation of a state is *highly redundant*



Matrix product state representation of a state
is *highly redundant*





Still represents the same state (same observables / amplitudes)

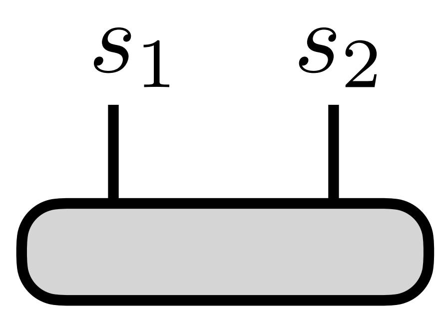
Internal parameters differ though

Huge freedom to manipulate parameters, but which such "gauge" transformations are interesting or useful?

Interesting gauges can be motivated from two-site MPS

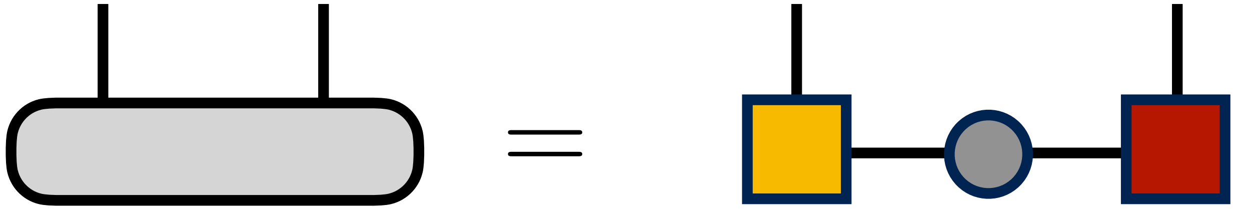
Consider arbitrary two-site wavefunction

$$\Psi^{s_1 s_2} =$$

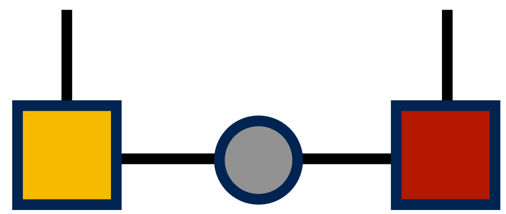


Use SVD to factorize the Ψ tensor

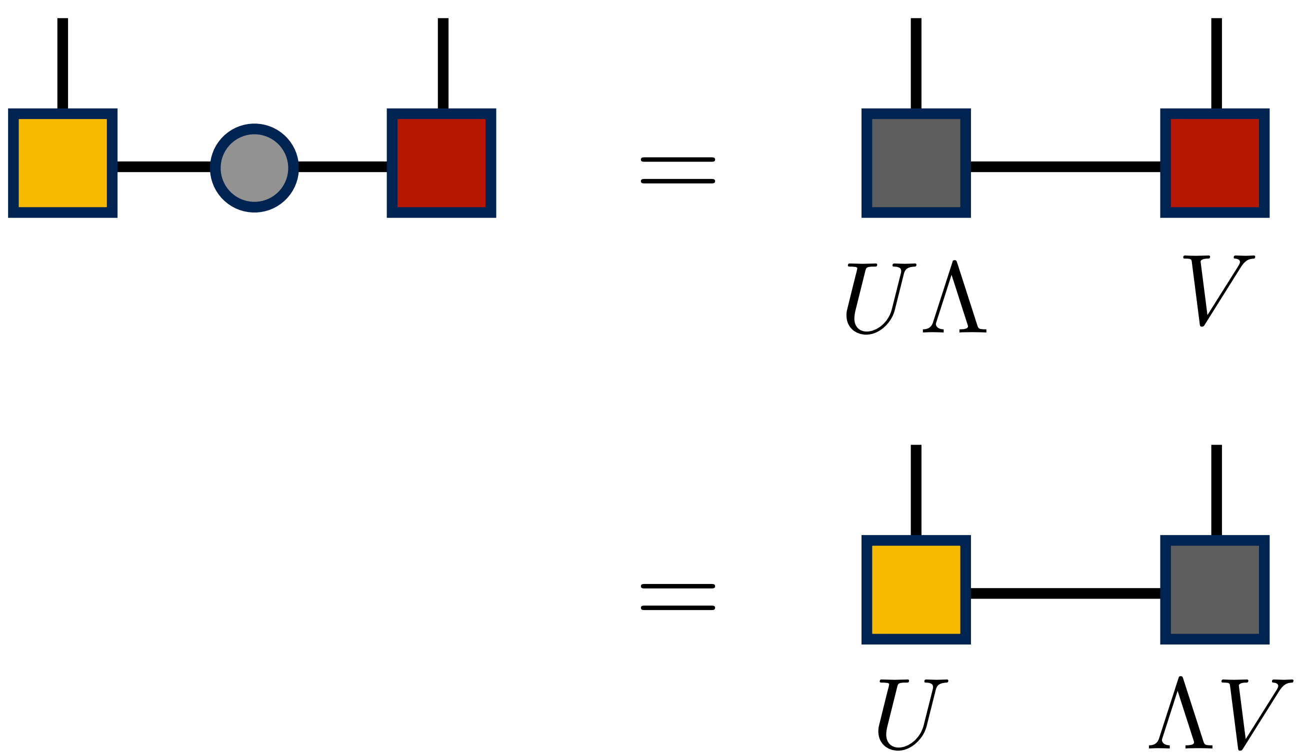
$$\Psi^{s_1 s_2} = \sum_n U_n^{s_1} \Lambda_n V_n^{s_2}$$



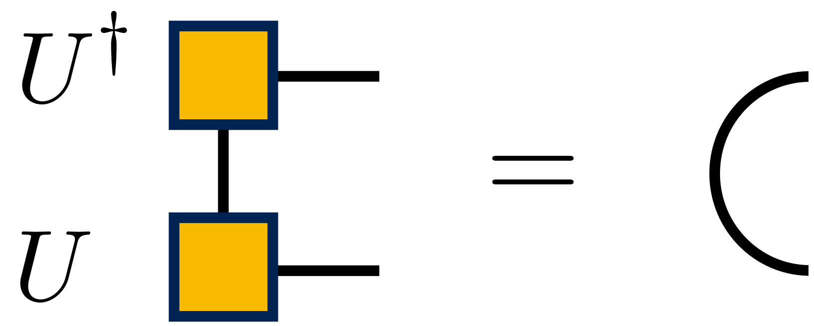
Could treat as an MPS, just with extra "bond tensor"

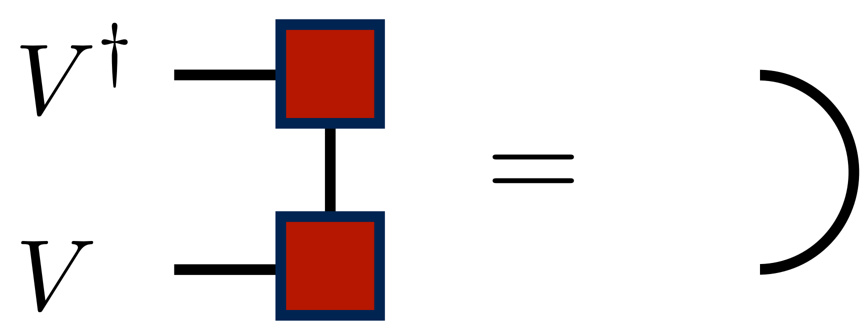


Or contract Λ with U or V to restore standard MPS form



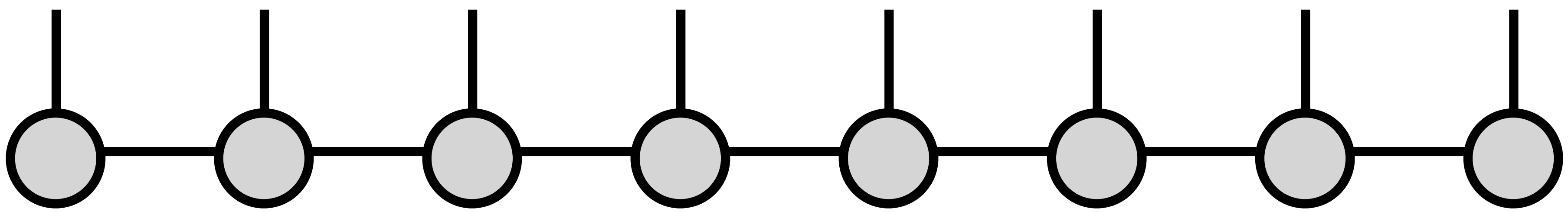
Note that U and V tensors have the following nice property



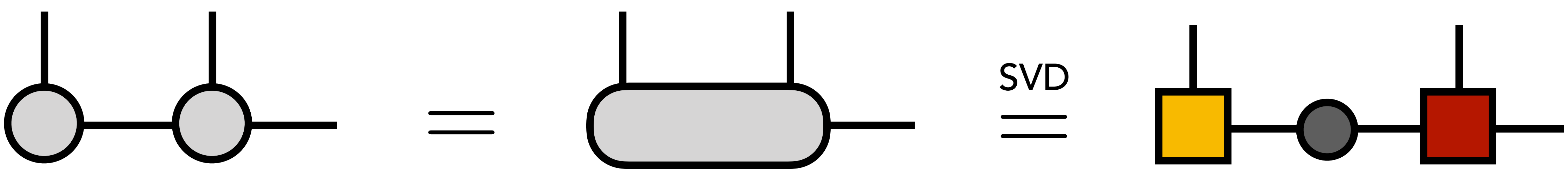


Realize a similar property beyond the two-site case?

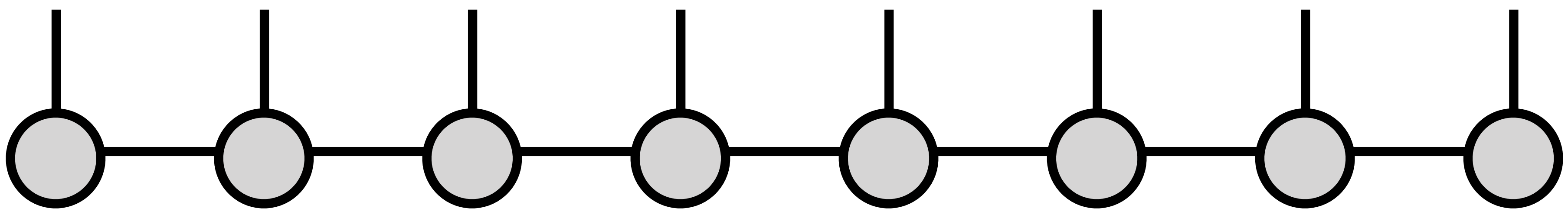
Start with generic MPS – no special properties



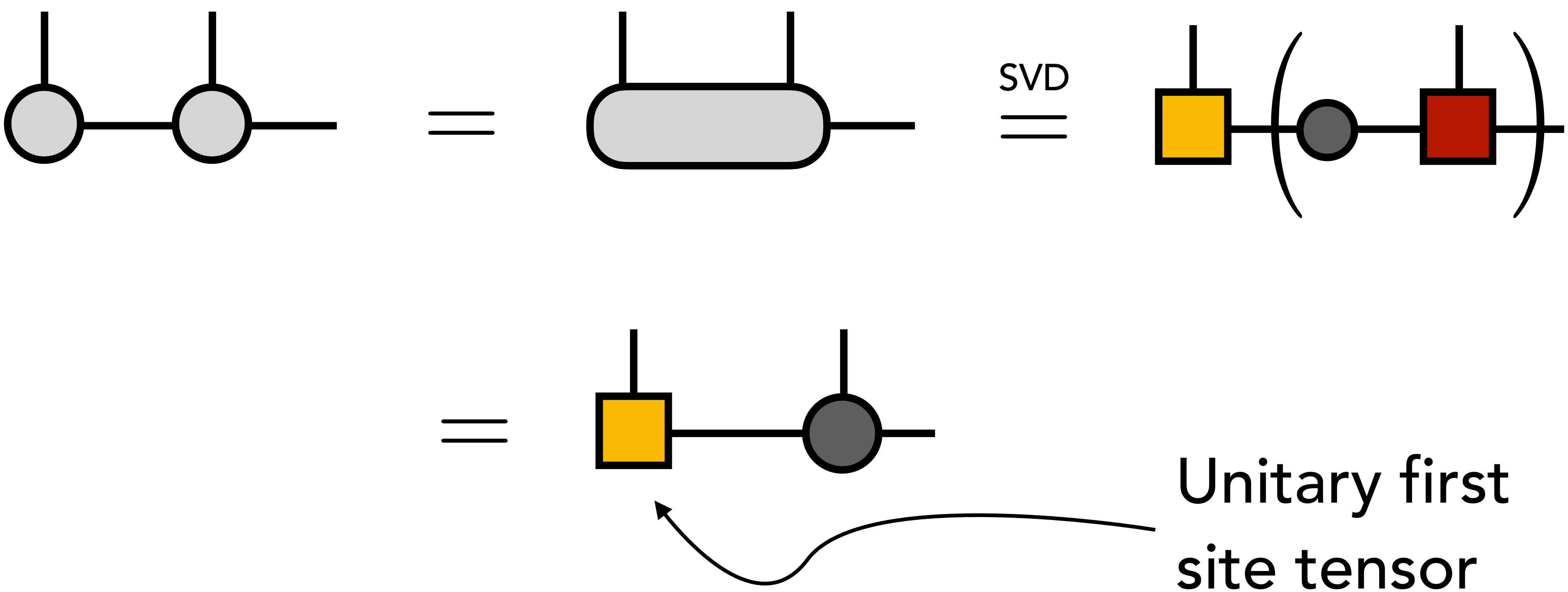
Multiply first two tensors together, then SVD (*no truncation!*)

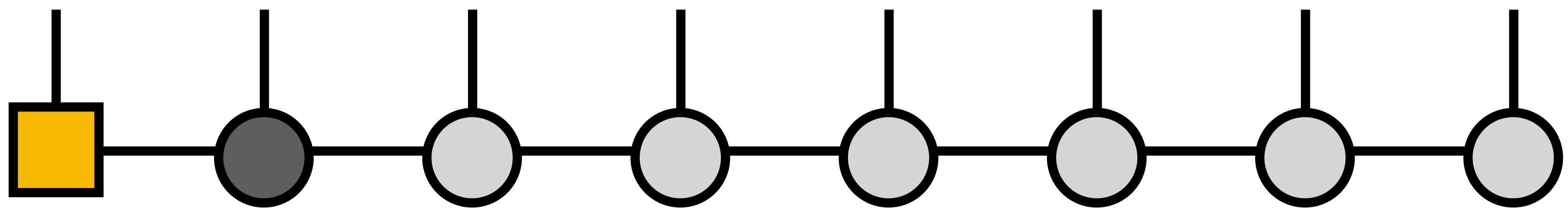


Start with generic MPS – no special properties

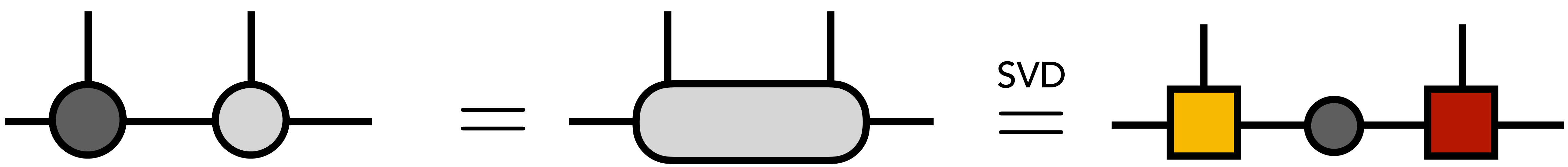


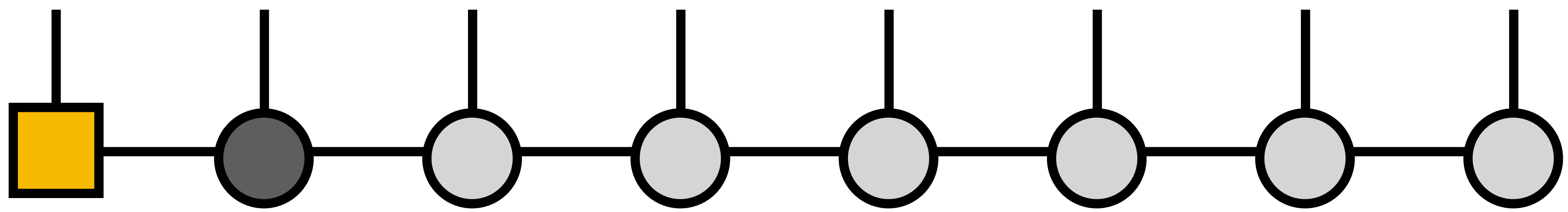
Multiply first two tensors together, then SVD (*no truncation!*)



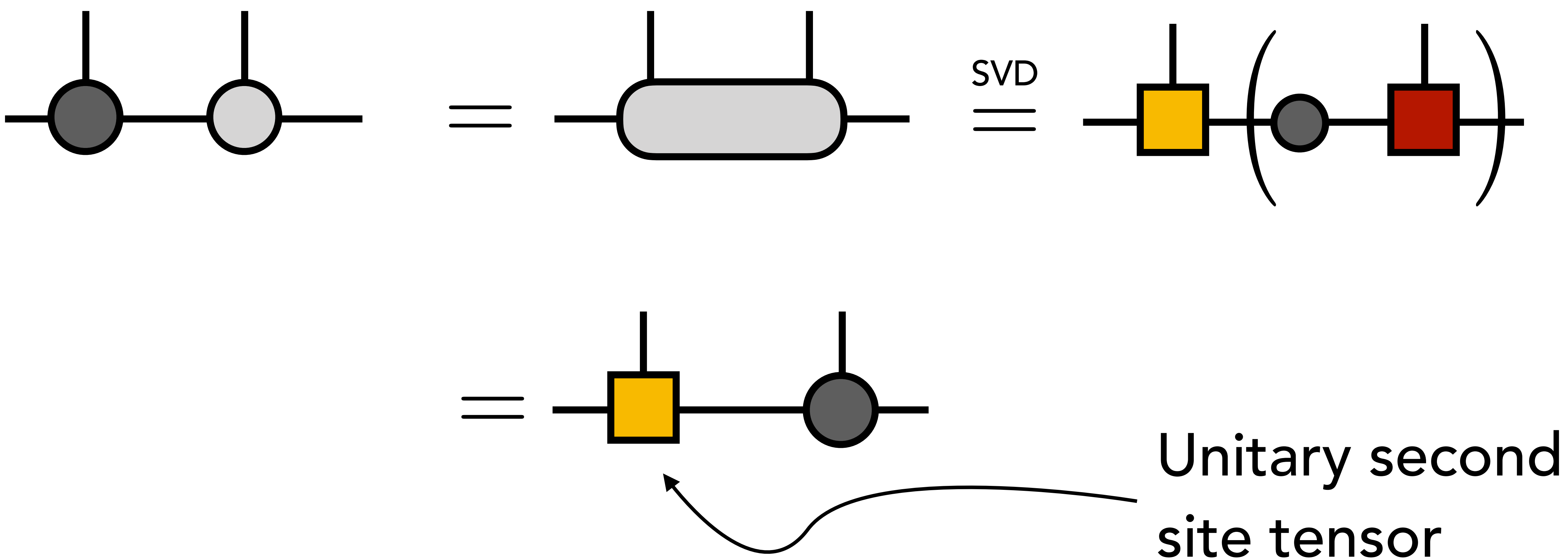


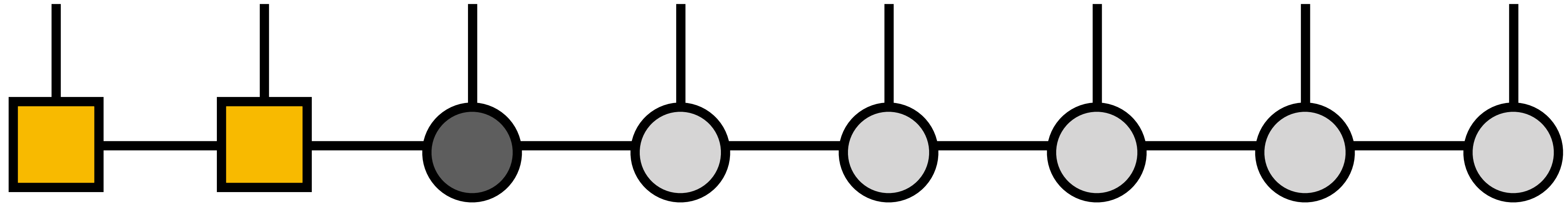
Multiply second two tensors together, then SVD (*no truncation!*)



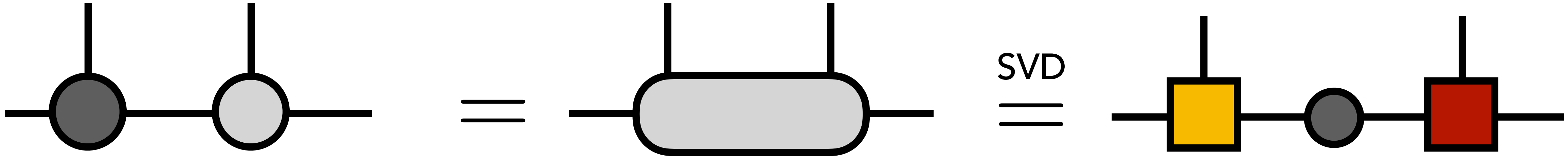


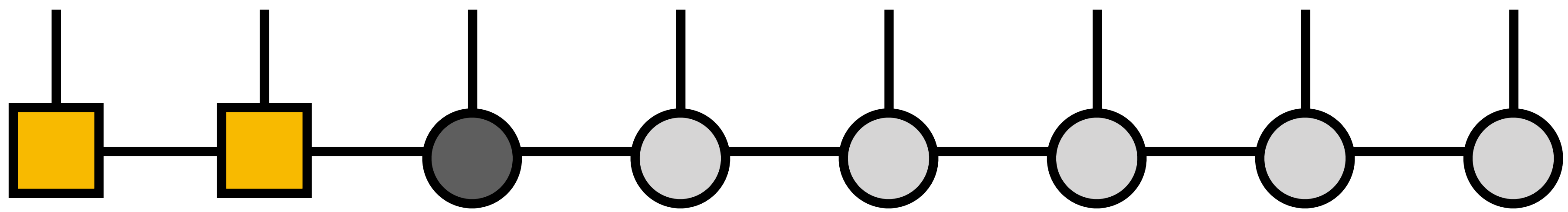
Multiply second two tensors together, then SVD (*no truncation!*)



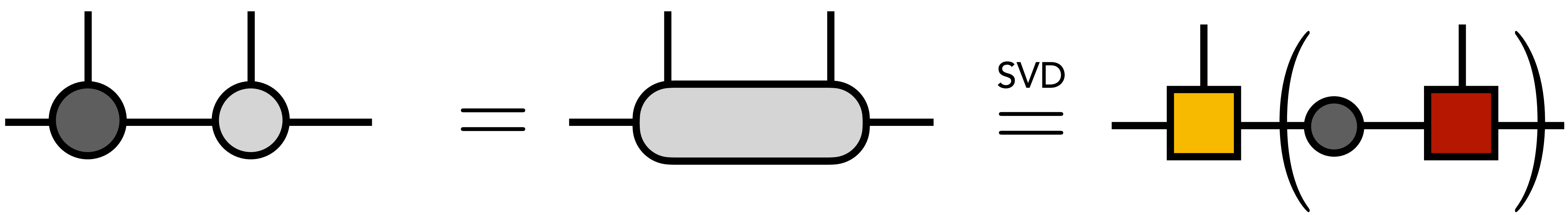


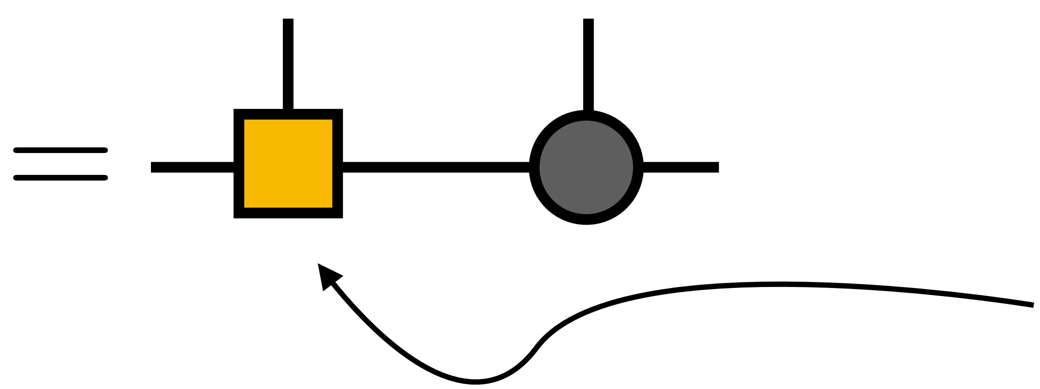
Multiply third pair together, then SVD (*no truncation!*)



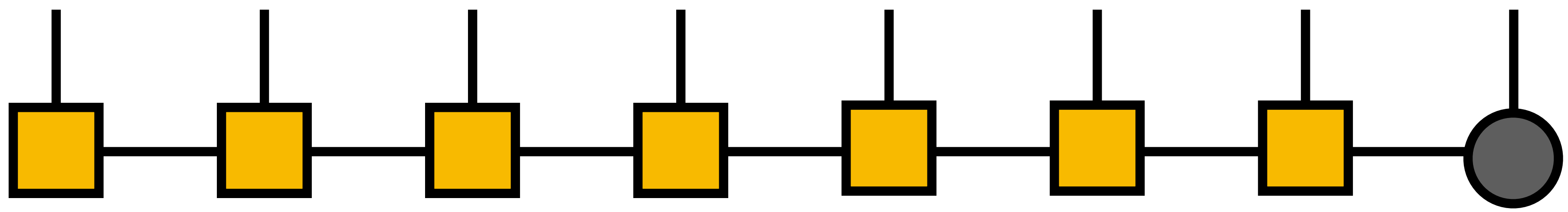


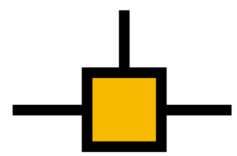
Multiply third pair together, then SVD (*no truncation!*)

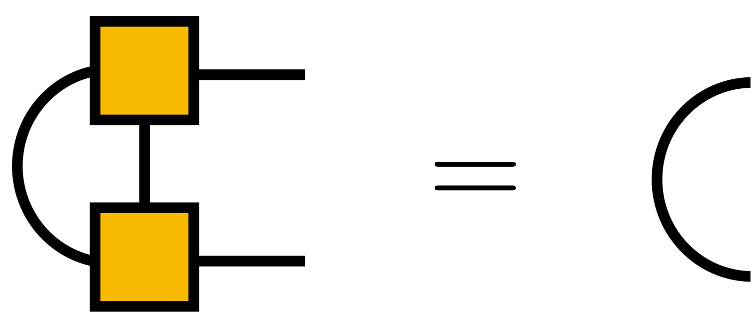




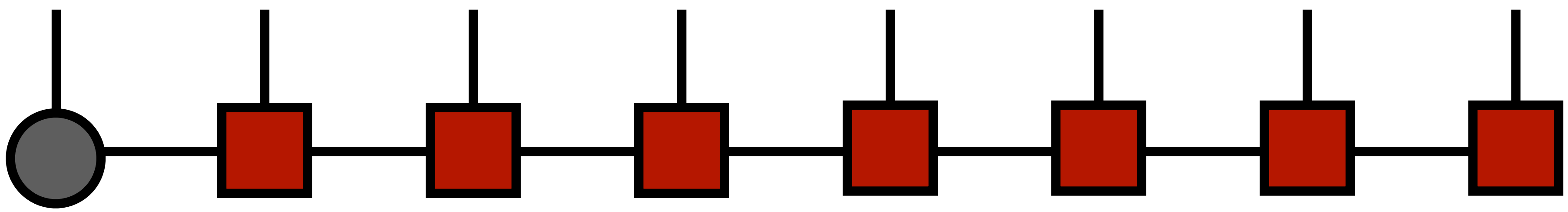
Repeating for all tensor pairs, left to right, gives

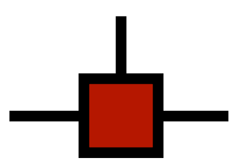


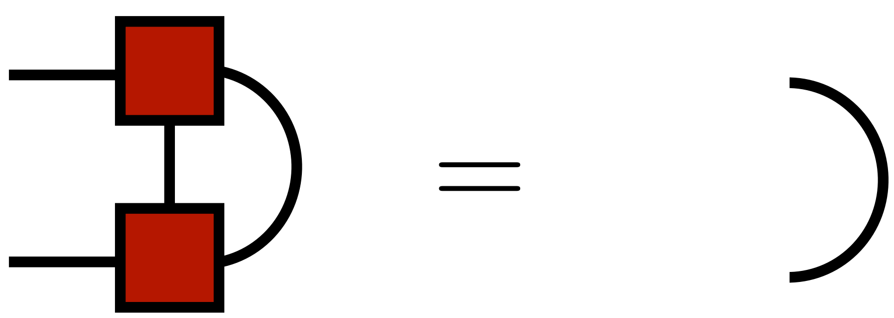
Because yellow  tensors were from left-hand side of SVD's, they are "left orthogonal"



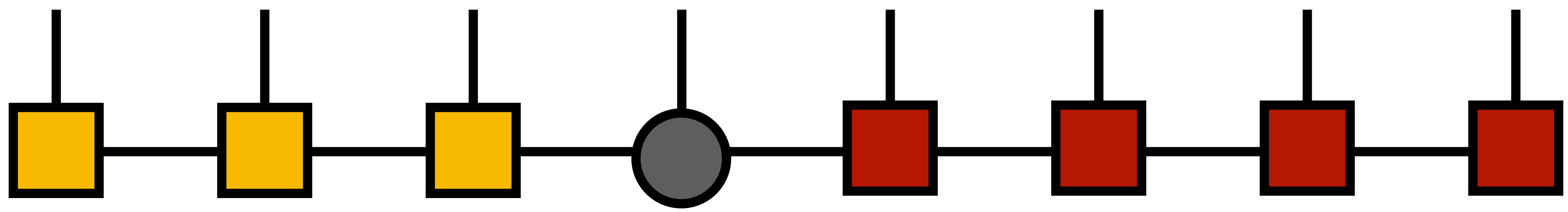
Can do the same procedure from right to left

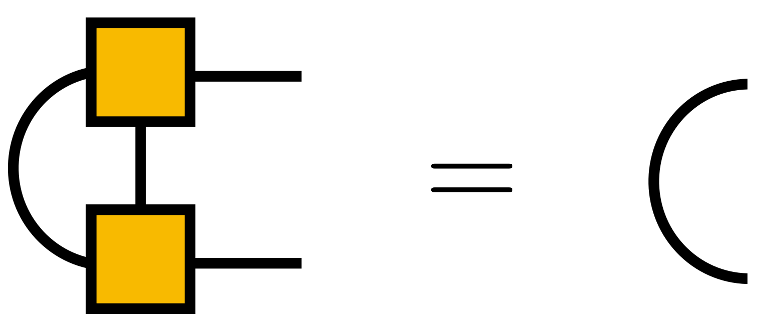


Because red  tensors were from left-hand side of SVD's, they are "right orthogonal"

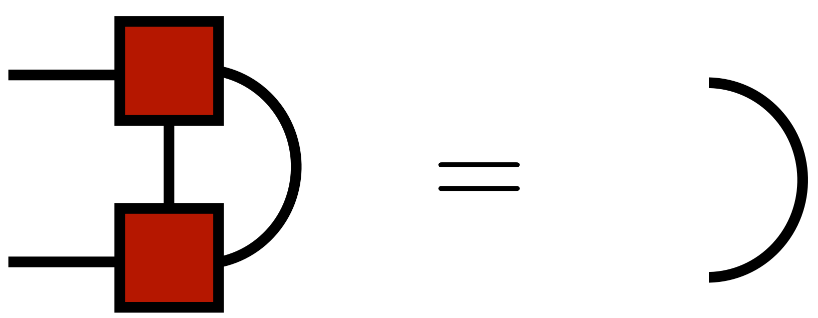


Or partway from left, partway from right



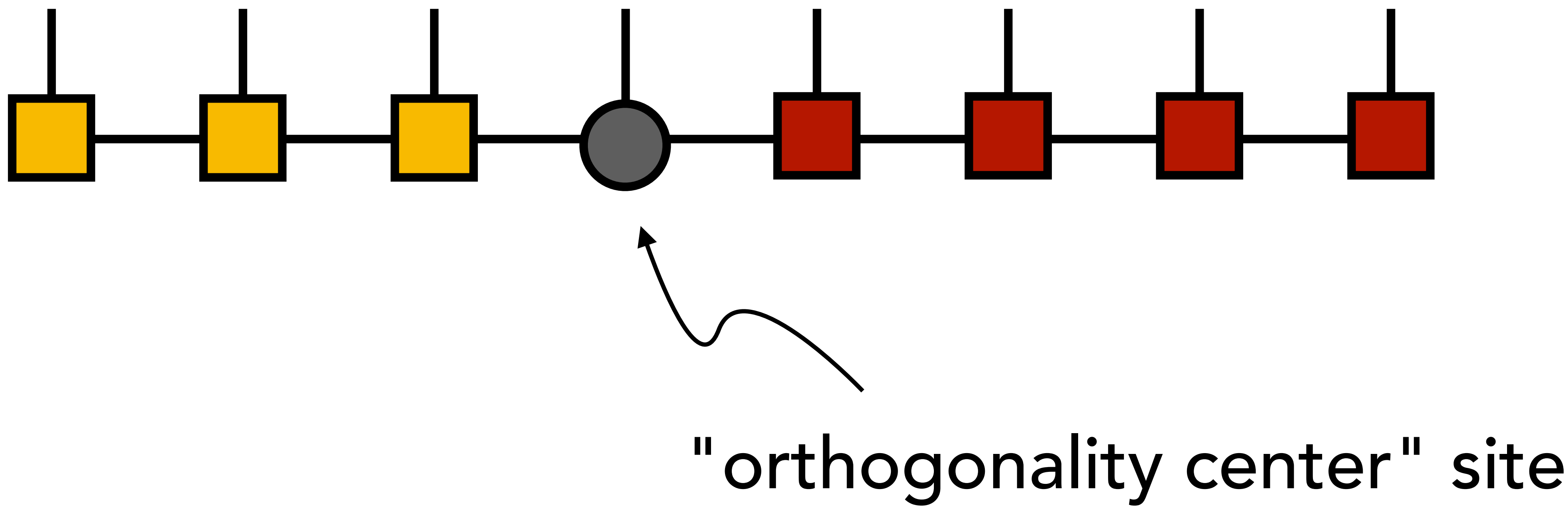


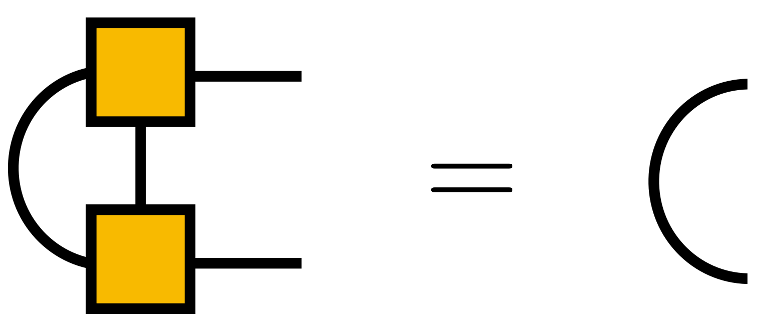
"left orthogonal"



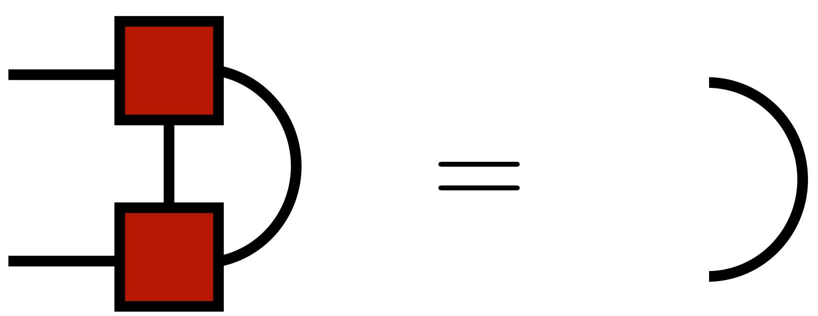
"right orthogonal"

Or partway from left, partway from right





"left orthogonal"

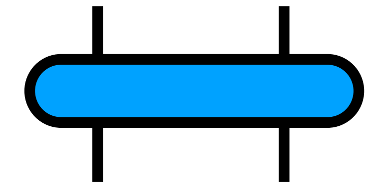


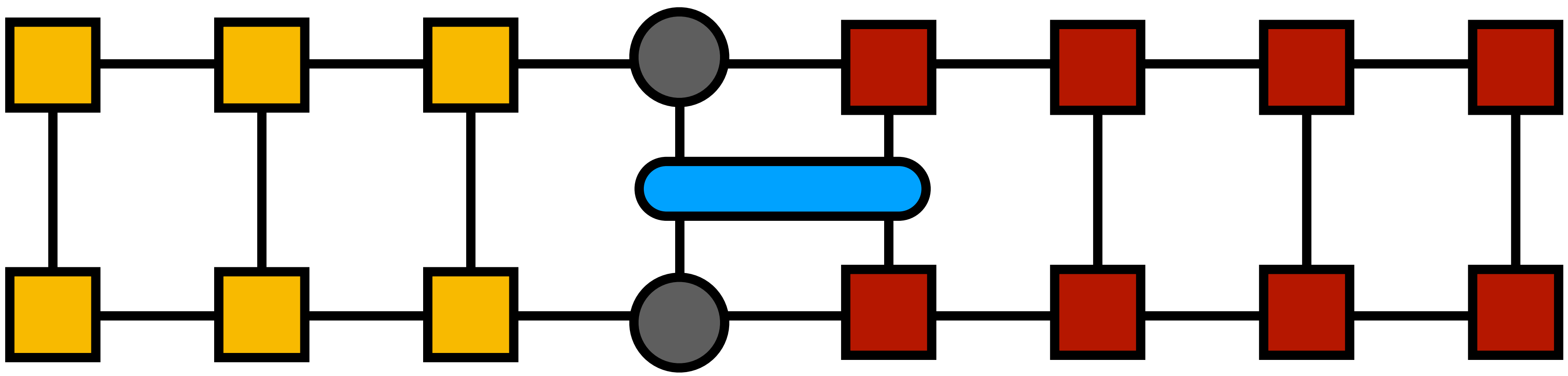
"right orthogonal"

MPS gauging important for many reasons:

- accurate truncations of MPS
- efficient computation of observables
- good conditioning properties for optimization algorithms
- connections to unitary quantum circuits (quantum computing)

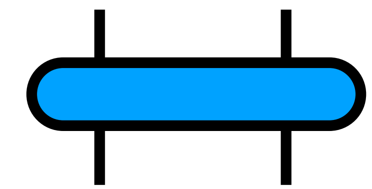
Efficient computation of observables

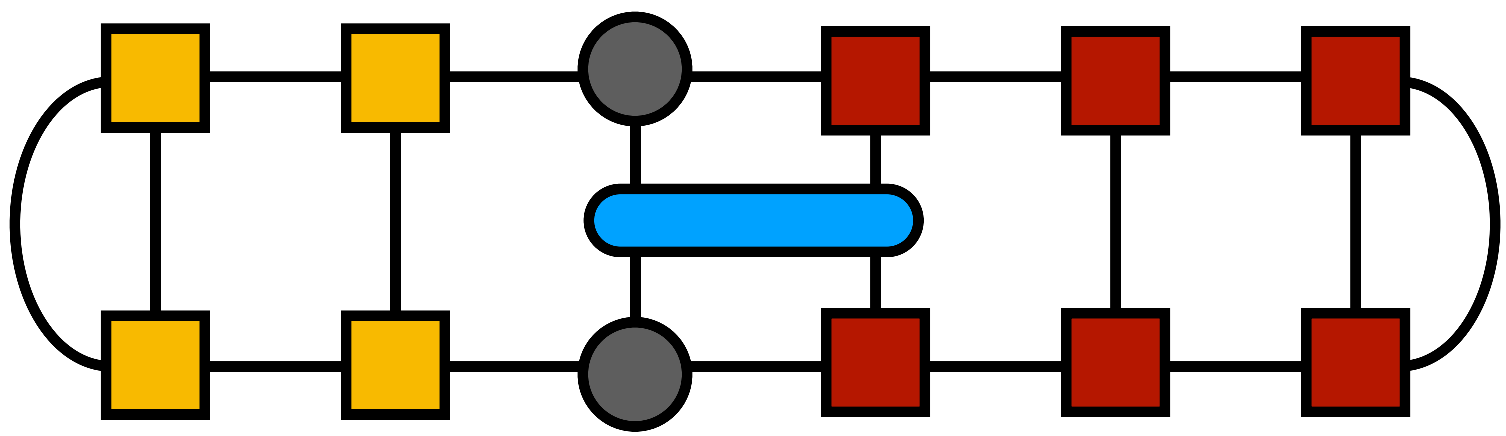
Say we want expectation value of operator $\mathcal{O} =$ 



If $\mathcal{O}$ acts on block of sites including 'center' site,
can cancel all other MPS tensors

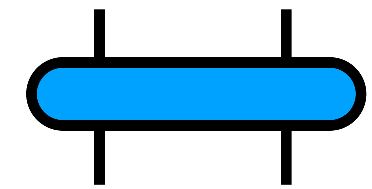
Efficient computation of observables

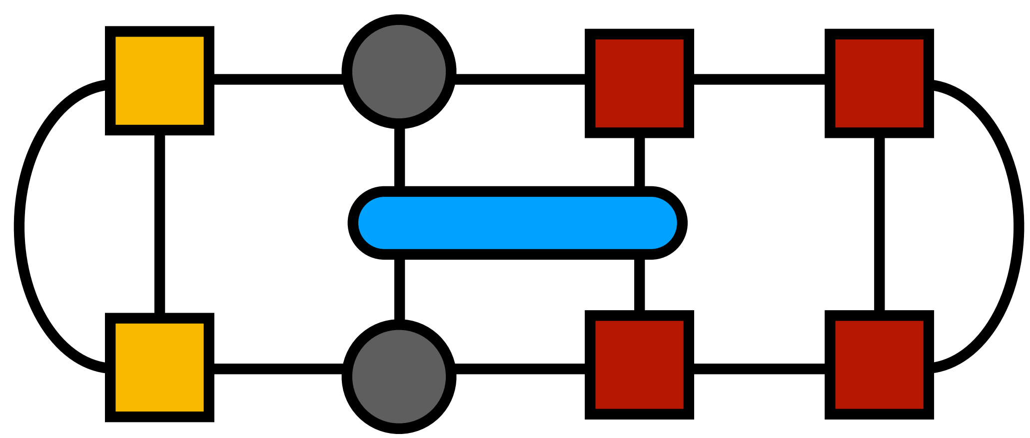
Say we want expecation value of operator $\mathcal{O} =$ 



If $\mathcal{O}$ acts on block of sites including 'center' site,
can cancel all other MPS tensors

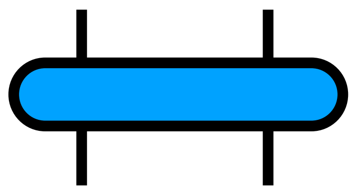
Efficient computation of observables

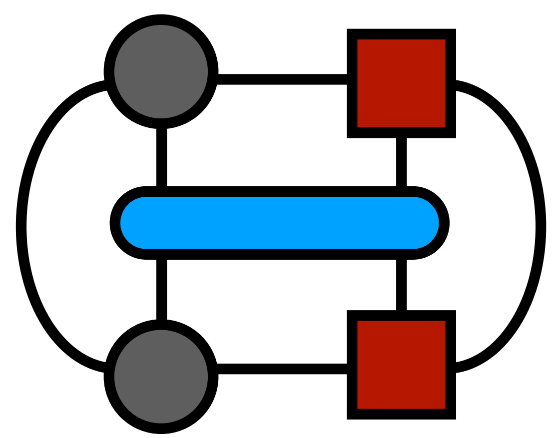
Say we want expecation value of operator $\mathcal{O} =$ 



If $\mathcal{O}$ acts on block of sites including 'center' site, can cancel all other MPS tensors

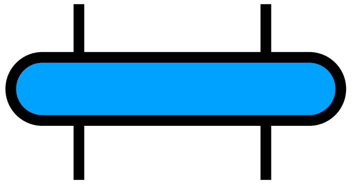
Efficient computation of observables

Say we want expecation value of operator $\mathcal{O} =$ 

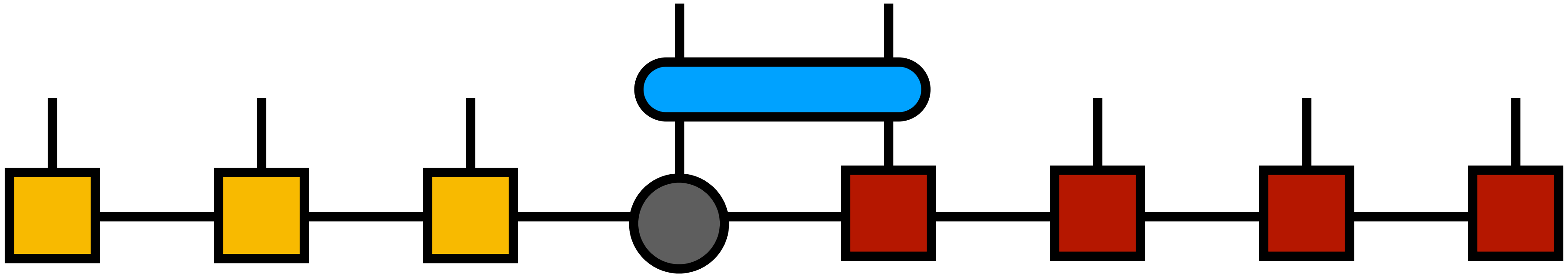


Much smaller diagram to compute!

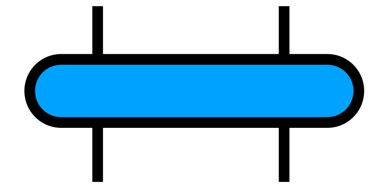
Accurate truncation of MPS

Say we act on the MPS with some operator $\mathcal{O} =$ 

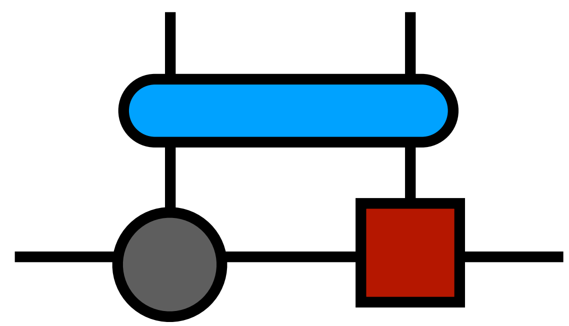
And affected sites include 'center' site



Accurate truncation of MPS

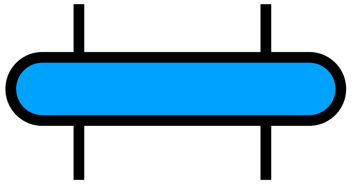
Say we act on the MPS with some operator $\mathcal{O} =$ 

And affected sites include 'center' site



Multiply into MPS tensors acted on by $\mathcal{O}$

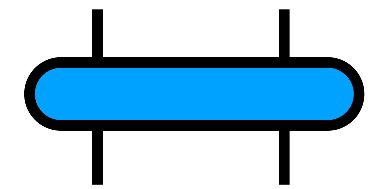
Accurate truncation of MPS

Say we act on the MPS with some operator $\mathcal{O} =$ 

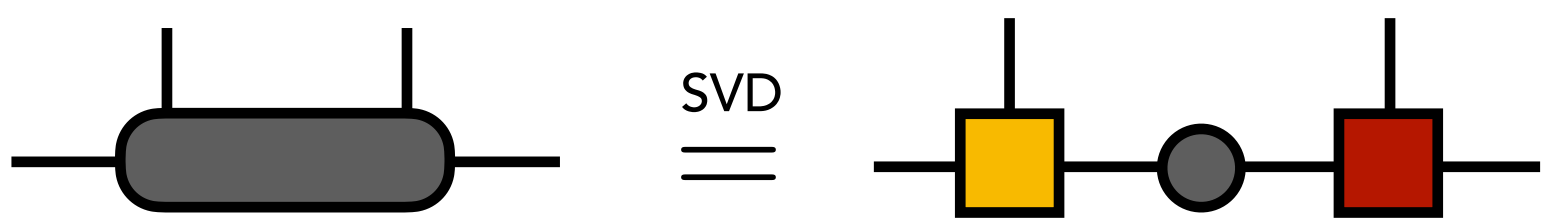
And affected sites include 'center' site

Contract to form new "bond tensor"

Accurate truncation of MPS

Say we act on the MPS with some operator $\mathcal{O} =$ 

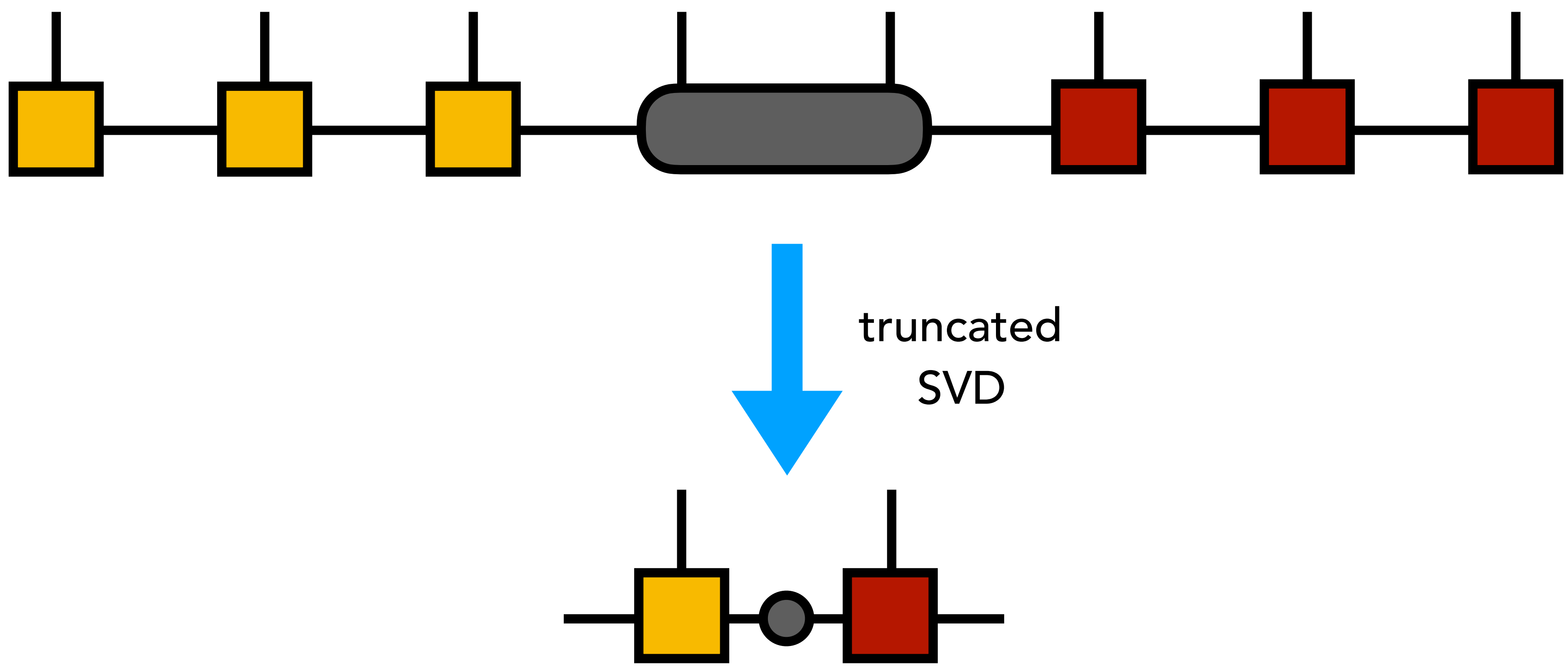
And affected sites include 'center' site



SVD to restore MPS form

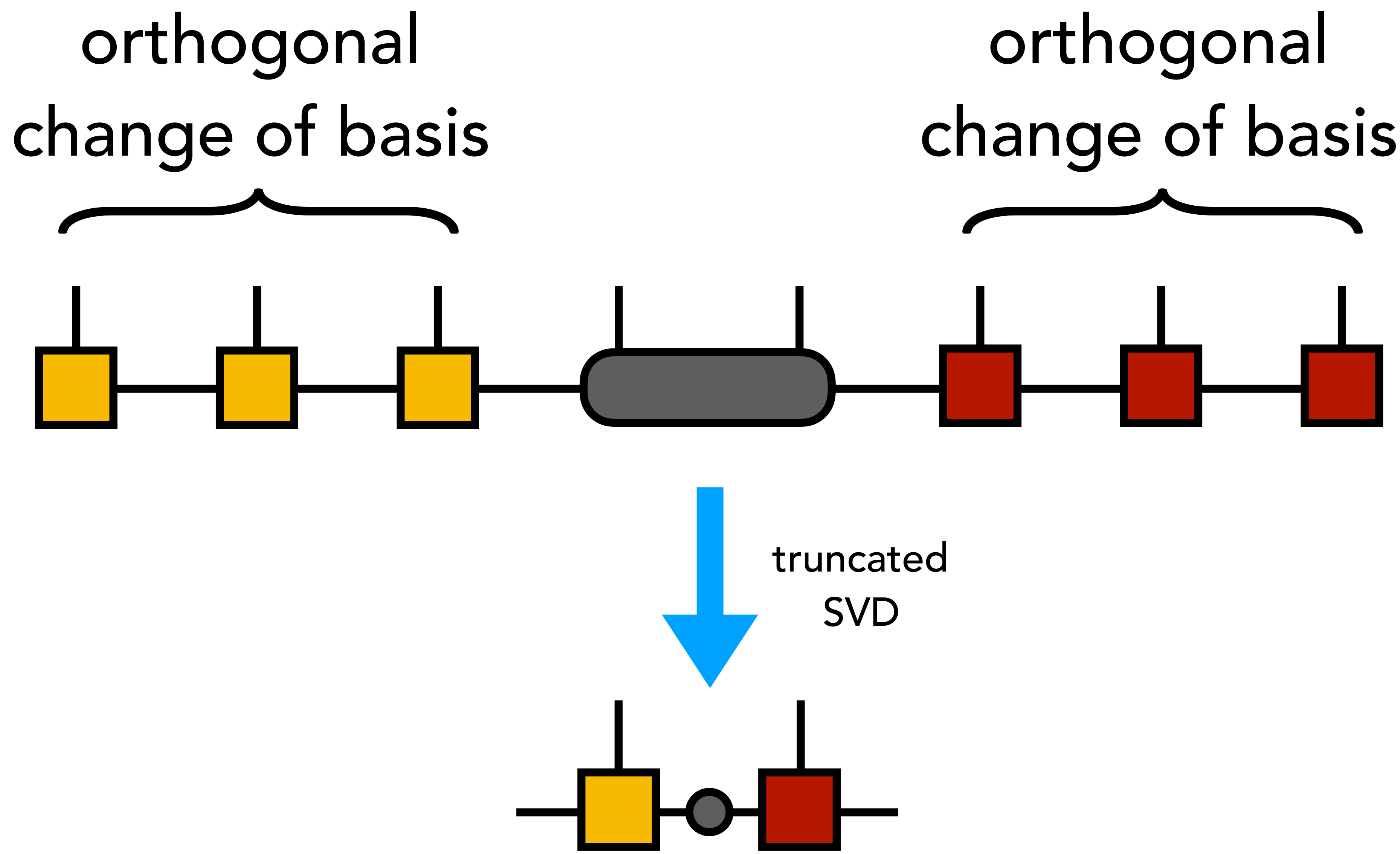
Ok to truncate SVD?

Accurate truncation of MPS



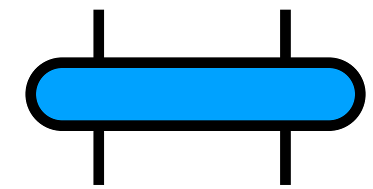
Can show small *local error* incurred in truncated SVD of bond translates to small *global error* for whole MPS

Accurate truncation of MPS

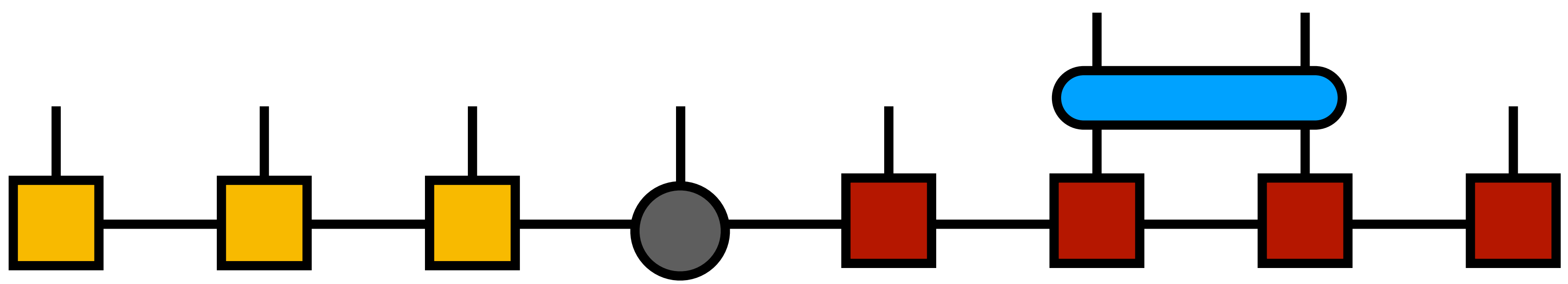


Can show small *local error* incurred in truncated SVD of bond translates to small *global error* for whole MPS

Accurate truncation of MPS

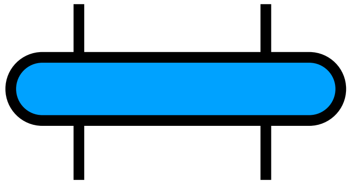
Important: acting with some operator $\mathcal{O} =$ 

away from orthogonality center

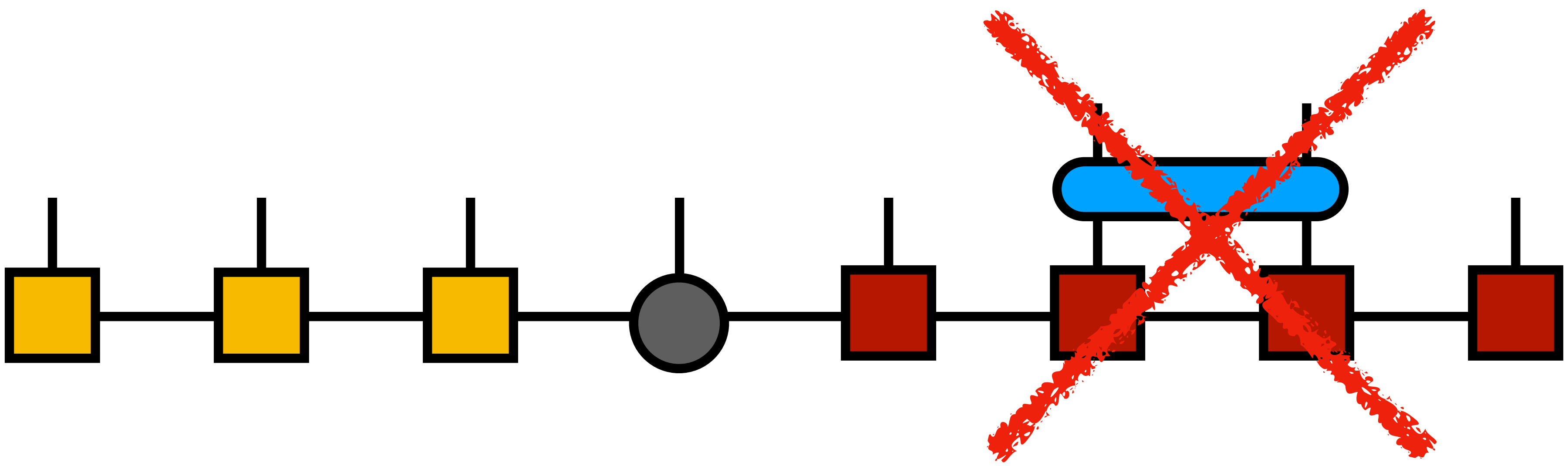


and performing local truncation could give a <u>large global error</u>

Accurate truncation of MPS

Important: acting with some operator $\mathcal{O} =$ 

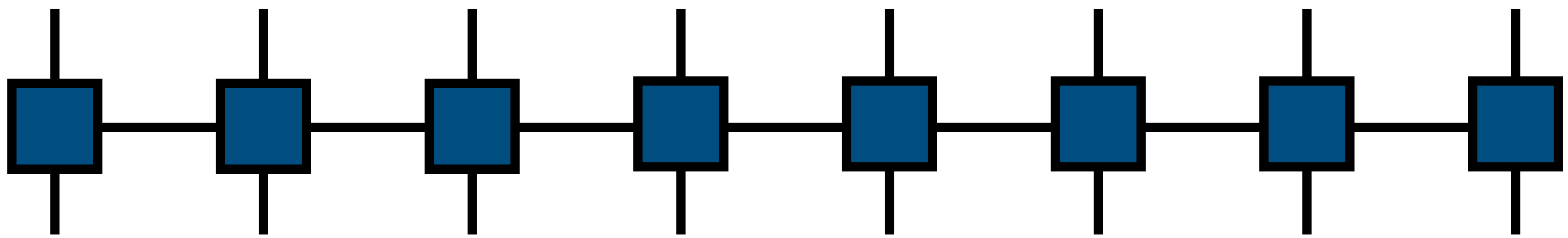
away from orthogonality center



and performing local truncation could give
a large global error

Matrix Product Operators

Idea of a matrix product operator (MPO):
chain of tensors like an MPS, but two sets of indices (up and down; bra and ket) just like an operator



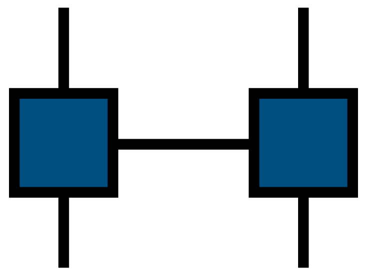
Very useful for algorithms involving MPS, such as DMRG

To motivate MPO construction, consider a two-site operator

$$\mathbf{S}_1 \cdot \mathbf{S}_2 = S_1^z S_2^z + \frac{1}{2} S_1^+ S_2^- + \frac{1}{2} S_1^- S_2^+$$

Write as dot product of operator-valued vectors

$$\mathbf{S}_1 \cdot \mathbf{S}_2 = \begin{bmatrix} S_1^z & \frac{1}{2}S_1^+ & \frac{1}{2}S_1^- \end{bmatrix} \begin{bmatrix} S_2^z \\ S_2^- \\ S_2^+ \end{bmatrix} =$$

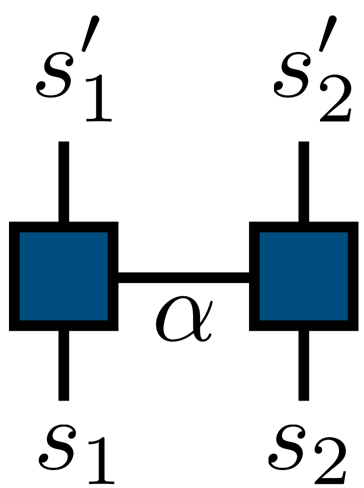


To motivate MPO construction, consider a two-site operator

$$\mathbf{S}_1 \cdot \mathbf{S}_2 = S_1^z S_2^z + \frac{1}{2} S_1^+ S_2^- + \frac{1}{2} S_1^- S_2^+$$

Write as dot product of operator-valued vectors

$$\mathbf{S}_1 \cdot \mathbf{S}_2 = \begin{bmatrix} S_1^z & \frac{1}{2} S_1^+ & \frac{1}{2} S_1^- \end{bmatrix}_\alpha \begin{bmatrix} S_2^z \\ S_2^- \\ S_2^+ \end{bmatrix} =$$



More generally, will involve operator-valued *matrices*

Consider the Hamiltonian:

$$H = S_1^z S_2^z + S_2^z S_3^z$$

More generally, will involve operator-valued *matrices*

Consider the Hamiltonian:

$$H = S_1^z S_2^z + S_2^z S_3^z \quad = S_1^z S_2^z I_3 + I_1 S_2^z S_3^z$$

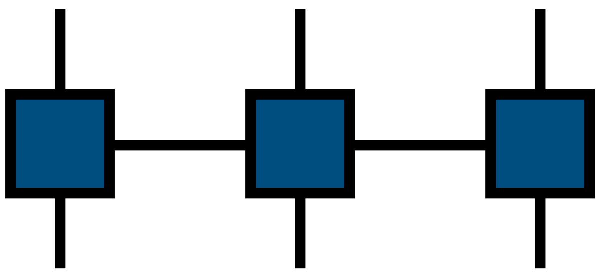
More generally, will involve operator-valued *matrices*

Consider the Hamiltonian:

$$H = S_1^z S_2^z + S_2^z S_3^z \quad = S_1^z S_2^z I_3 + I_1 S_2^z S_3^z$$

Can write as

$$\begin{bmatrix} 0 & S_1^z & I_1 \end{bmatrix} \begin{bmatrix} I_2 & 0 & 0 \\ S_2^z & 0 & 0 \\ 0 & S_2^z & I_2 \end{bmatrix} \begin{bmatrix} I_3 \\ S_3^z \\ 0 \end{bmatrix}$$



More generally, will involve operator-valued *matrices*

Consider the Hamiltonian:

$$H = S_1^z S_2^z + S_2^z S_3^z \quad (= S_1^z S_2^z I_3 + I_1 S_2^z S_3^z)$$

Can write as

$$\begin{bmatrix} 0 & S_1^z & I_1 \end{bmatrix} \left(\begin{bmatrix} I_2 & 0 & 0 \\ S_2^z & 0 & 0 \\ 0 & S_2^z & I_2 \end{bmatrix} \begin{bmatrix} I_3 \\ S_3^z \\ 0 \end{bmatrix} \right) = \begin{bmatrix} I_2 \, I_3 \\ S_2^z \, I_3 \\ S_2^z \, S_3^z \end{bmatrix}$$

More generally, will involve operator-valued *matrices*

Consider the Hamiltonian:

$$H = S_1^z S_2^z + S_2^z S_3^z \quad (= S_1^z S_2^z I_3 + I_1 S_2^z S_3^z)$$

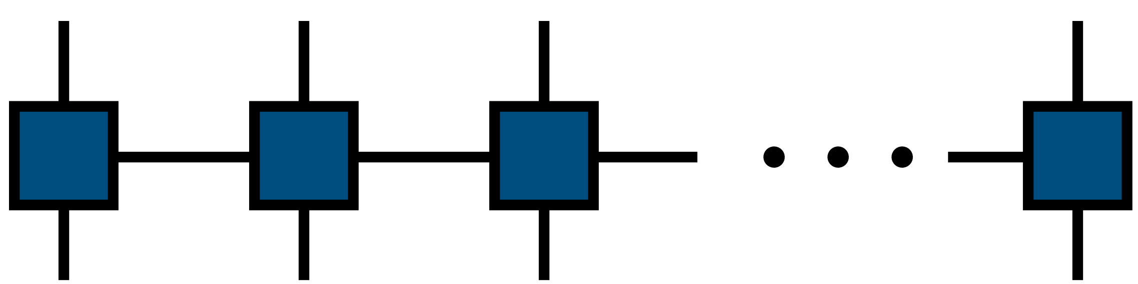
Can write as

$$\begin{bmatrix} 0 & S_1^z & I_1 \end{bmatrix} \begin{bmatrix} I_2 \; I_3 \\ S_2^z \; I_3 \\ S_2^z \; S_3^z \end{bmatrix} = S_1^z S_2^z I_3 + I_1 S_2^z S_3^z$$

Chaining the pattern will give Hamiltonian for arbitrarily big system

$$H = \sum_j S_j^z S_{j+1}^z$$

$$\begin{bmatrix} 0 & S_1^z & I_1 \end{bmatrix} \begin{bmatrix} I_2 & 0 & 0 \\ S_2^z & 0 & 0 \\ 0 & S_2^z & I_2 \end{bmatrix} \begin{bmatrix} I_3 & 0 & 0 \\ S_3^z & 0 & 0 \\ 0 & S_3^z & I_3 \end{bmatrix} \cdots \begin{bmatrix} I_N \\ S_N^z \\ 0 \end{bmatrix}$$



Why this pattern?

$$H = \sum_j S_j^z S_{j+1}^z$$

$$\begin{bmatrix} I_j & 0 & 0 \\ S_j^z & 0 & 0 \\ 0 & S_j^z & I_j \end{bmatrix}$$

View as a "machine" or "automaton"

$$\begin{bmatrix} I_1 & 0 & 0 \\ S_1^z & 0 & 0 \\ 0 & S_1^z & I_1 \end{bmatrix}$$

Result:

View as a "machine" or "automaton"

Start in state 3 →

$$\begin{bmatrix} I_1 & 0 & 0 \\ S^z_1 & 0 & 0 \\ 0 & S^z_1 & I_1 \end{bmatrix}$$

Result:

View as a "machine" or "automaton"

Start in state 3 →

$$\begin{bmatrix} I_1 & 0 & 0 \\ S^z_1 & 0 & 0 \\ 0 & S^z_1 & I_1 \end{bmatrix}$$

Choose an operator (bracing S^z_1 and I_1 in the bottom row)

Result:

View as a "machine" or "automaton"

Start in state 3

$$\begin{bmatrix} I_1 & 0 & 0 \\ S^z_1 & 0 & 0 \\ 0 & S^z_1 & I_1 \end{bmatrix}$$

Choose an operator

Result: I_1

View as a "machine" or "automaton"

State 3

Start in state 3

$$\begin{bmatrix} I_1 & 0 & 0 \\ S^z_1 & 0 & 0 \\ 0 & S^z_1 & I_1 \end{bmatrix}$$

Choose an operator

Result: I_1

View as a "machine" or "automaton"

State 3 →

$$\begin{bmatrix} I_2 & 0 & 0 \\ S^z_2 & 0 & 0 \\ 0 & S^z_2 & I_2 \end{bmatrix}$$

Result: I_1

View as a "machine" or "automaton"

State 3 →

$$\begin{bmatrix} I_2 & 0 & 0 \\ S^z_2 & 0 & 0 \\ 0 & S^z_2 & I_2 \end{bmatrix}$$

Choose an operator

Result: I_1

View as a "machine" or "automaton"

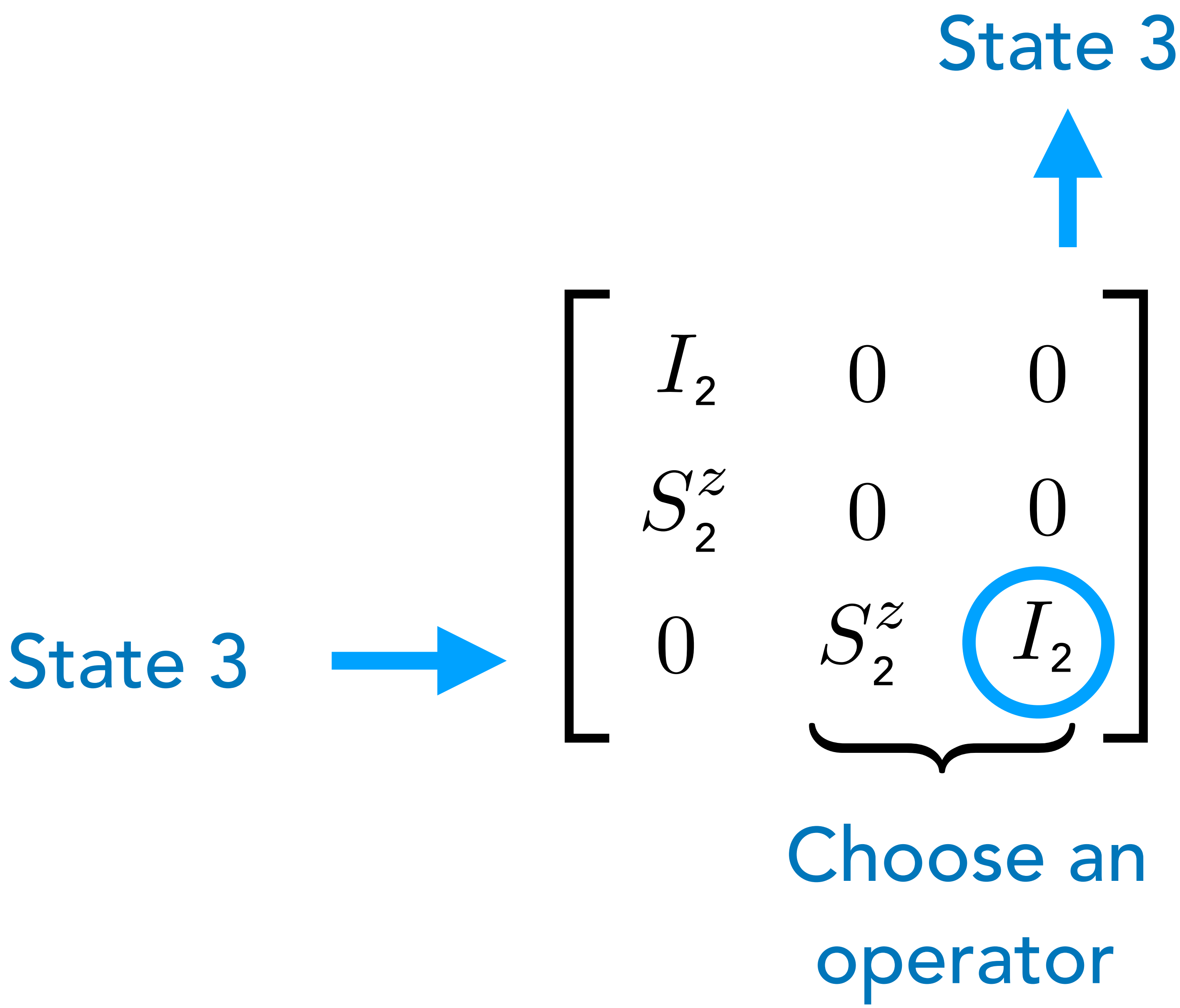
State 3 →

$$\begin{bmatrix} I_2 & 0 & 0 \\ S^z_2 & 0 & 0 \\ 0 & S^z_2 & I_2 \end{bmatrix}$$

Choose an operator

Result: $I_1 \quad I_2$

View as a "machine" or "automaton"



$$\begin{bmatrix} I_2 & 0 & 0 \\ S^z_2 & 0 & 0 \\ 0 & S^z_2 & I_2 \end{bmatrix}$$

Result: $I_1 \quad I_2$

View as a "machine" or "automaton"

State 3 →

$$\begin{bmatrix} I_3 & 0 & 0 \\ S_3^z & 0 & 0 \\ 0 & S_3^z & I_3 \end{bmatrix}$$

Result: $I_1 \quad I_2$

View as a "machine" or "automaton"

State 3 →

$$\begin{bmatrix} I_3 & 0 & 0 \\ S^z_3 & 0 & 0 \\ 0 & S^z_3 & I_3 \end{bmatrix}$$

Choose an operator

Result: $I_1 \quad I_2$

View as a "machine" or "automaton"

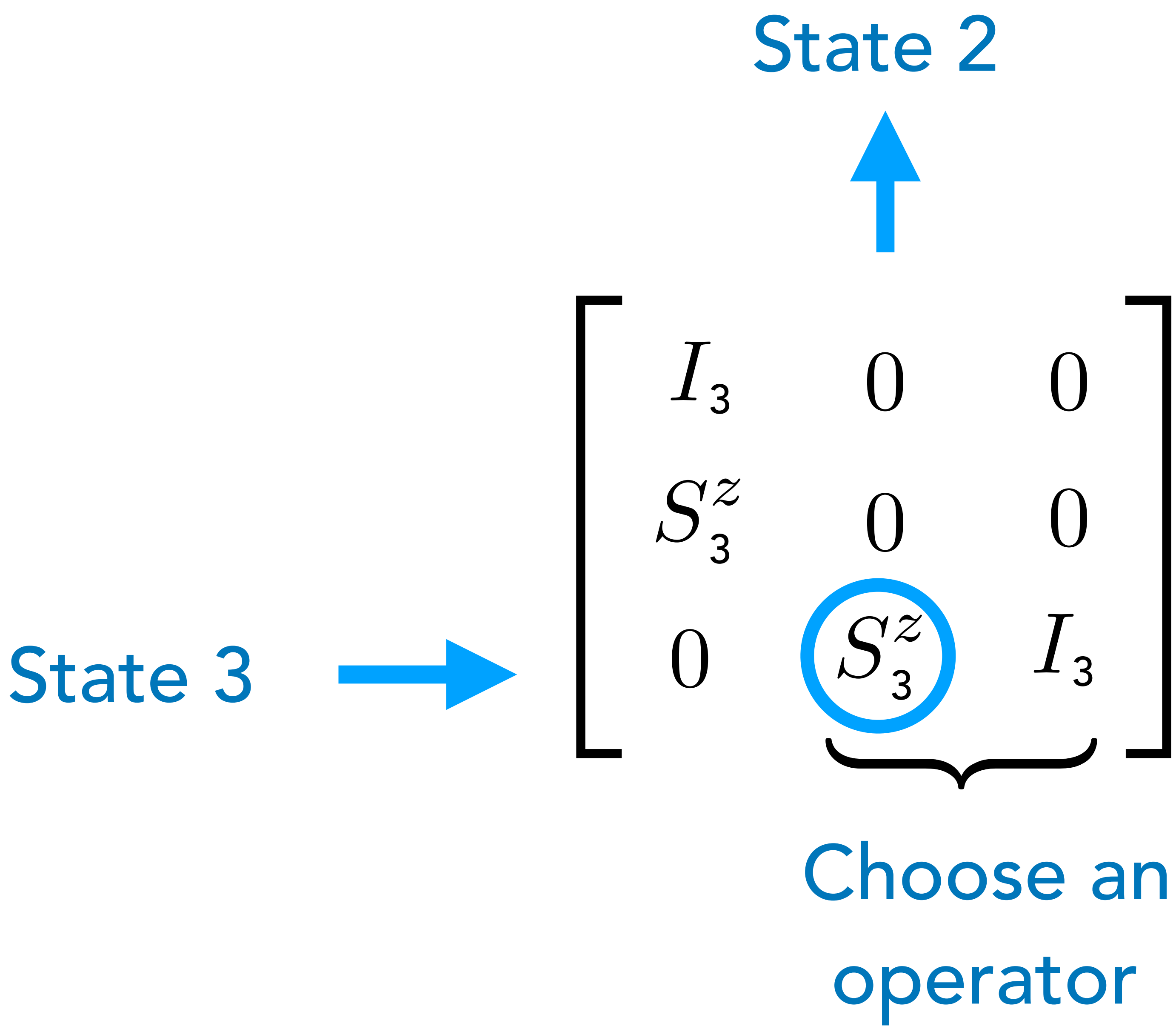
State 3 →

$$\begin{bmatrix} I_3 & 0 & 0 \\ S^z_3 & 0 & 0 \\ 0 & S^z_3 & I_3 \end{bmatrix}$$

Choose an operator

Result: $I_1 \quad I_2 \quad S^z_3$

View as a "machine" or "automaton"



$$\begin{bmatrix} I_3 & 0 & 0 \\ S^z_3 & 0 & 0 \\ 0 & S^z_3 & I_3 \end{bmatrix}$$

Result: $I_1 \quad I_2 \quad S^z_3$

View as a "machine" or "automaton"

State 2 →

$$\begin{bmatrix} I_4 & 0 & 0 \\ S^z_4 & 0 & 0 \\ 0 & S^z_4 & I_4 \end{bmatrix}$$

Result: $I_1 \quad I_2 \quad S^z_3$

View as a "machine" or "automaton"

State 2 →

$$\begin{bmatrix} I_4 & 0 & 0 \\ S^z_4 & 0 & 0 \\ 0 & S^z_4 & I_4 \end{bmatrix}$$

Result: $I_1 \quad I_2 \quad S^z_3 \quad S^z_4$

View as a "machine" or "automaton"

State 1

State 2

$$\begin{bmatrix} I_4 & 0 & 0 \\ S^z_4 & 0 & 0 \\ 0 & S^z_4 & I_4 \end{bmatrix}$$

Result: $I_1 \quad I_2 \quad S^z_3 \quad S^z_4$

View as a "machine" or "automaton"

State 1 →

$$\begin{bmatrix} I_5 & 0 & 0 \\ S^z_5 & 0 & 0 \\ 0 & S^z_5 & I_5 \end{bmatrix}$$

Result: $I_1 \quad I_2 \quad S^z_3 \quad S^z_4$

View as a "machine" or "automaton"

State 1

State 1

$$\begin{bmatrix} I_5 & 0 & 0 \\ S^z_5 & 0 & 0 \\ 0 & S^z_5 & I_5 \end{bmatrix}$$

Result: $I_1 \quad I_2 \quad S^z_3 \quad S^z_4$

View as a "machine" or "automaton"

State 1

State 1

$$\begin{bmatrix} I_5 & 0 & 0 \\ S^z_5 & 0 & 0 \\ 0 & S^z_5 & I_5 \end{bmatrix}$$

Result: $I_1 \quad I_2 \quad S^z_3 \quad S^z_4 \quad I_5$

Familiar 1D Hamiltonians as MPOs

Transverse-field Ising model

$$\begin{bmatrix} I_j & & \\ \sigma_j^z & & \\ -h\sigma_j^x & \sigma_j^z & I_j \end{bmatrix}$$

$$H = \sum_j \sigma_j^z \sigma_{j+1}^z - h\sigma_j^x$$

Heisenberg model

$$\begin{bmatrix} I_j & & & & \\ S_j^+ & & & & \\ S_j^- & & & & \\ S_j^z & & & & \\ 0 & \frac{1}{2}S_j^- & \frac{1}{2}S_j^+ & S_j^z & I_j \end{bmatrix}$$

$$H = \sum_j \mathbf{S}_j \cdot \mathbf{S}_{j+1}$$

To make MPO construction accessible,
AutoMPO in ITensor library

```
int N = 100;
auto sites = SpinOne(N);

auto ampo = AutoMPO(sites);
for(int j = 1; j < N; ++j)
    {
    ampo += 0.5,"S+",j,"S-",j+1;
    ampo += 0.5,"S-",j,"S+",j+1;
    ampo +=     "Sz",j,"Sz",j+1;
    }
auto H = MPO(ampo);
```

$$H = \sum_j \frac{1}{2} S^+_j S^-_{j+1} + \frac{1}{2} S^-_j S^+_{j+1} + S^z_j S^z_{j+1}$$

MPOs can even capture "long range" interactions

$$\begin{bmatrix} I_j & & \\ \sigma_j^z & \lambda I_j & \\ & \lambda\sigma_j^z & I_j \end{bmatrix}$$

$$H = \sum_{i<j} \lambda^{j-i} \, \sigma_i^z \sigma_j^z$$

MPOs can even capture "long range" interactions

$$\begin{bmatrix} I_j & & & \\ \sigma^z_j & \lambda_1 I_j & & \\ \sigma^z_j & & \lambda_2 I_j & \\ & \lambda_1 \sigma^z_j & \lambda_2 \sigma^z_j & I_j \end{bmatrix}$$

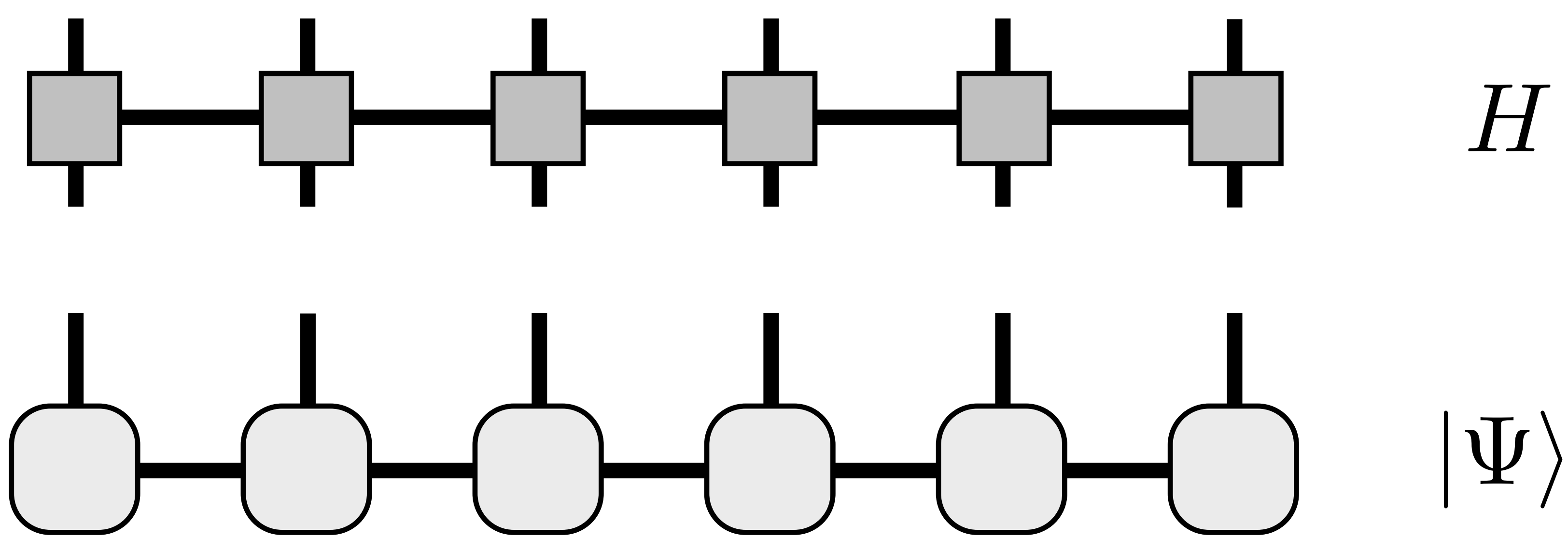
$$H = \sum_{i<j} \left(\lambda_1^{j-i} + \lambda_2^{j-i}\right) \sigma^z_i \sigma^z_j$$

DMRG

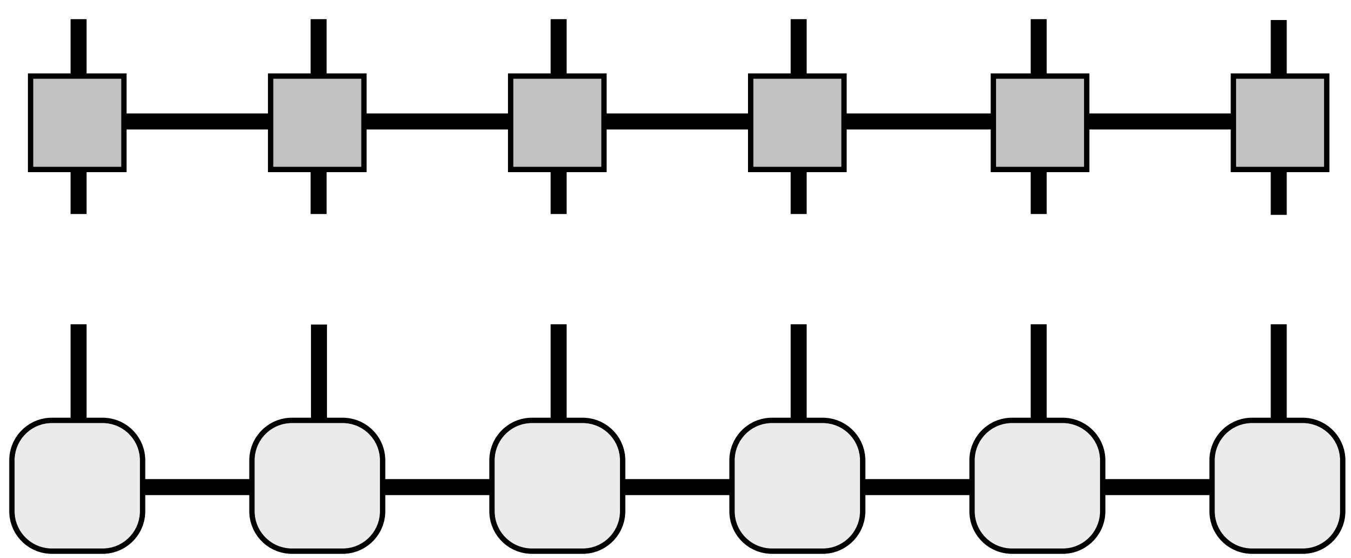
The density matrix renormalization group (DMRG) is the best method for finding ground states of 1D Hamiltonians

Want to solve $H|\Psi\rangle = E|\Psi\rangle$

Treat H as MPO

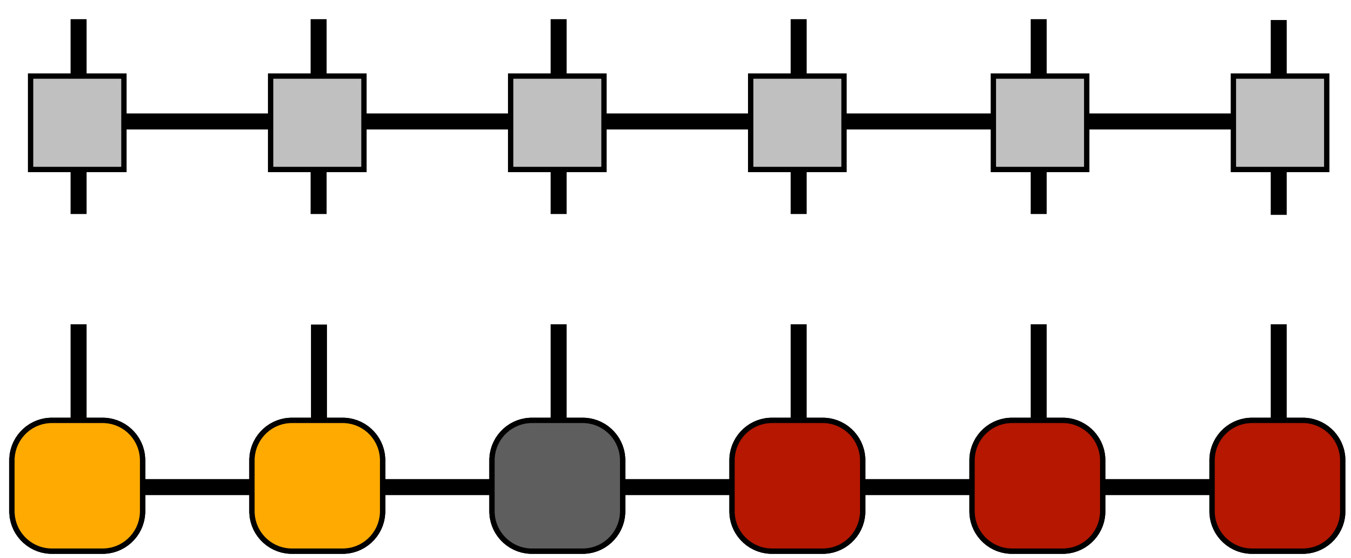


Important: MPS should be in definite gauge
I.e. most tensors unitary

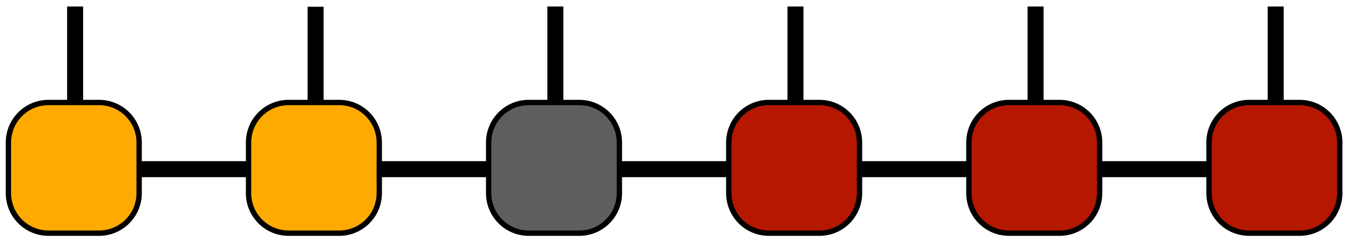


Important: MPS should be in definite gauge

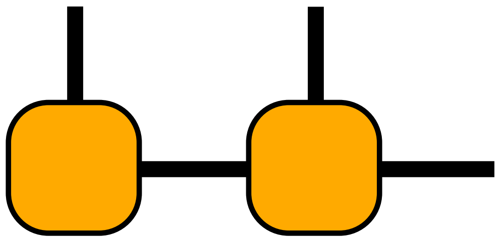
I.e. most tensors unitary



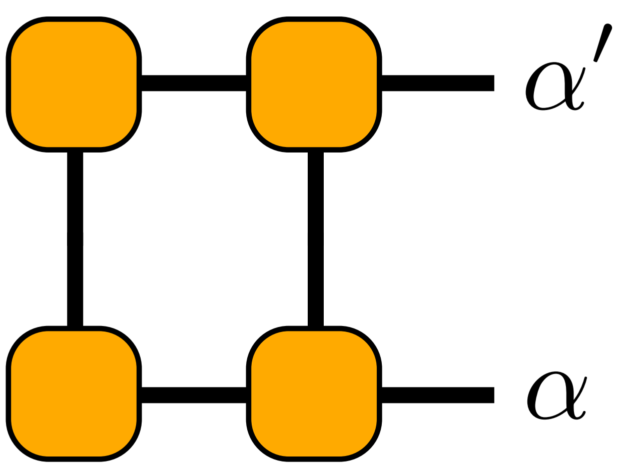
This way, left/right tensors define an *orthonormal basis*



This way, left/right tensors define an *orthonormal basis*

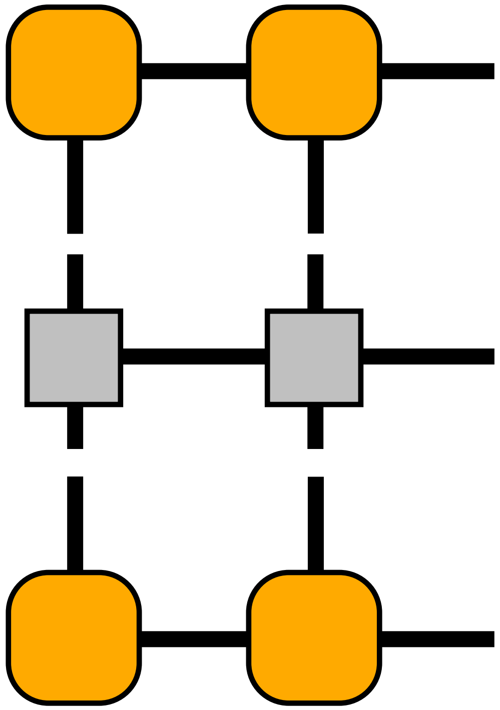


This way, left/right tensors define an *orthonormal basis*

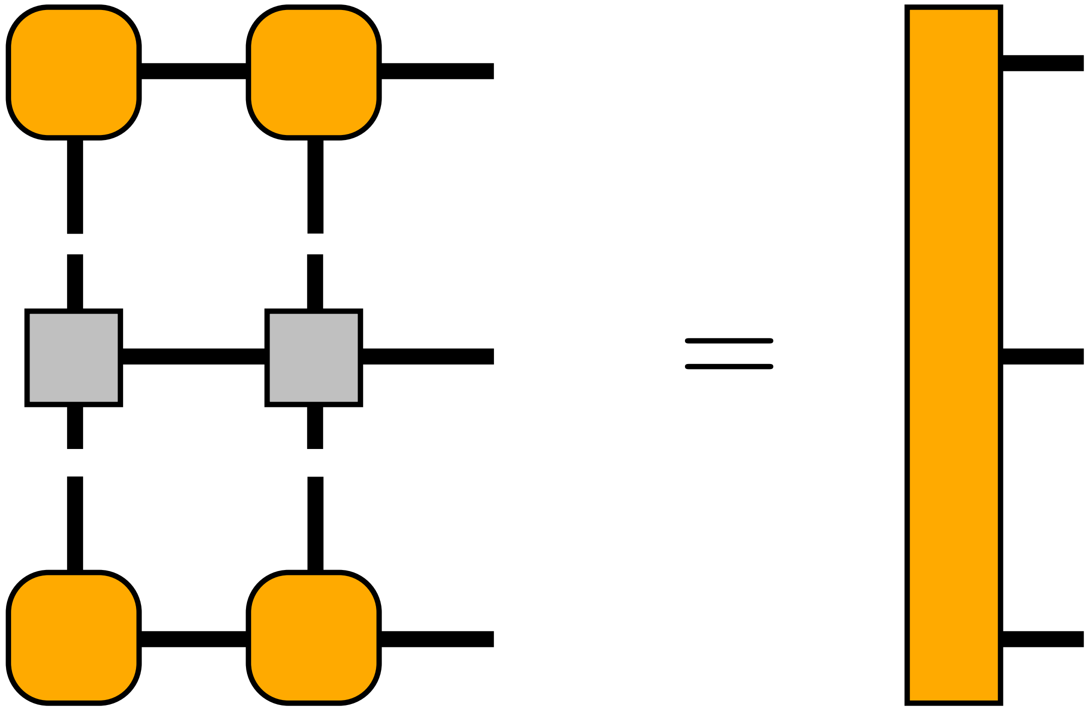


$$= \delta_{\alpha}^{\alpha'}$$

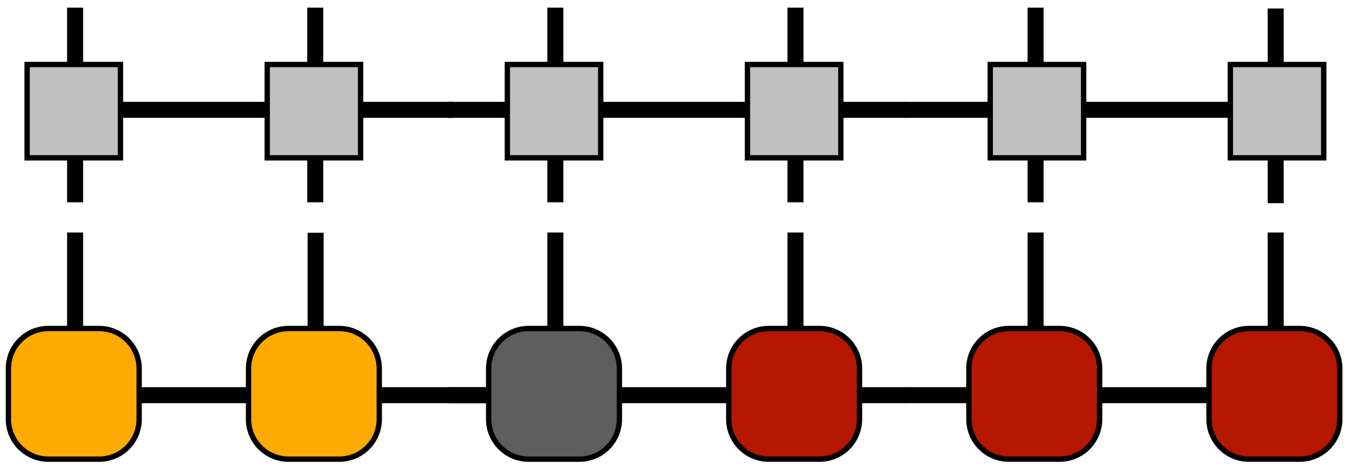
Project Hamiltonian into this basis



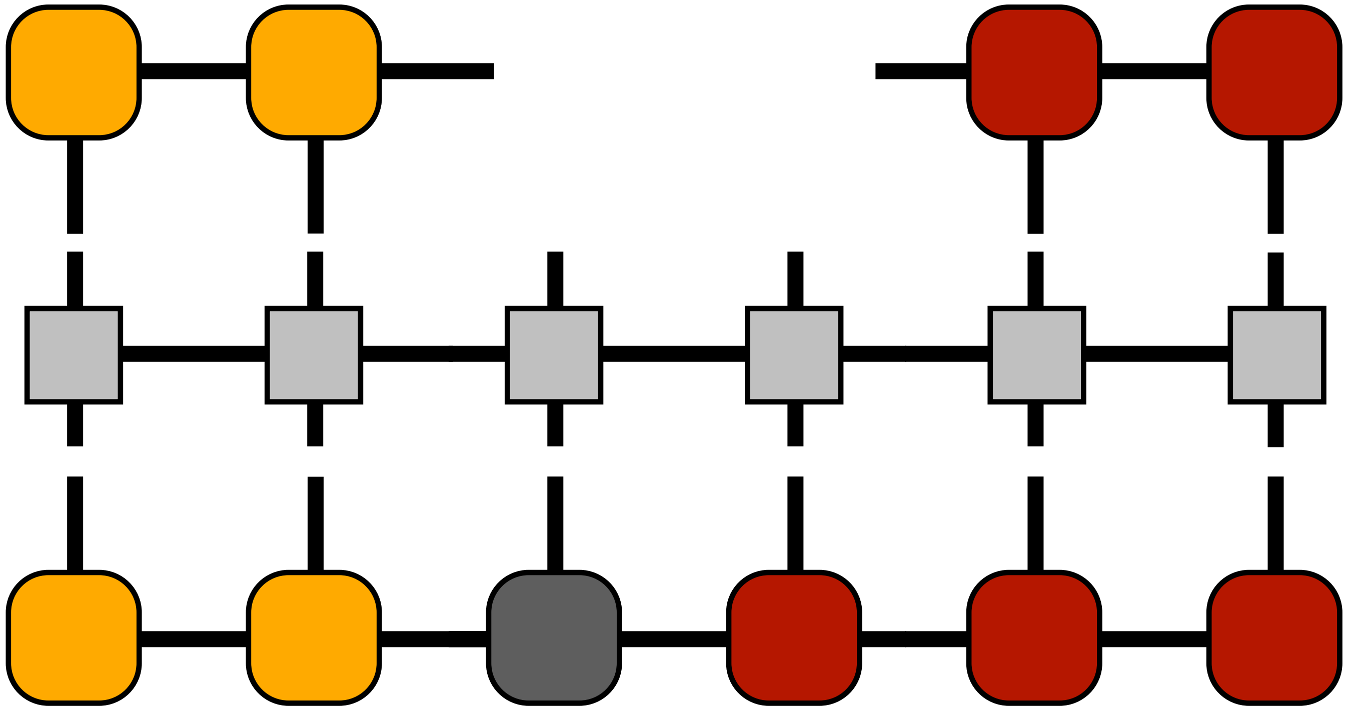
Project Hamiltonian into this basis



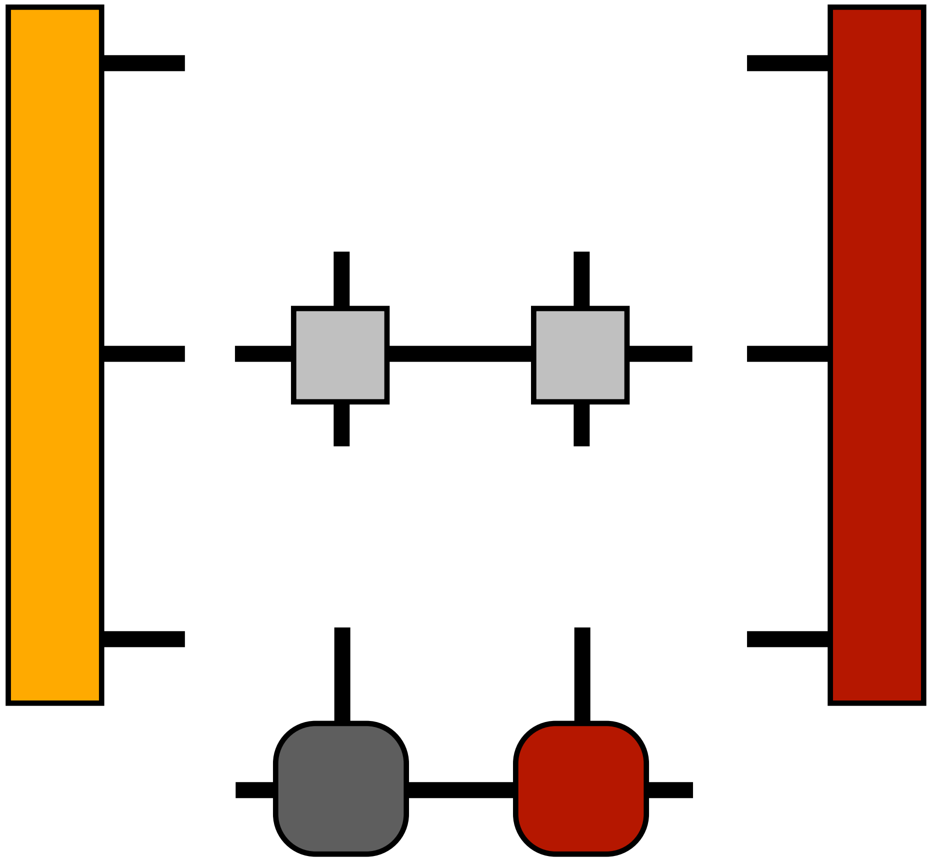
Doing the same on the right gives



Doing the same on the right gives



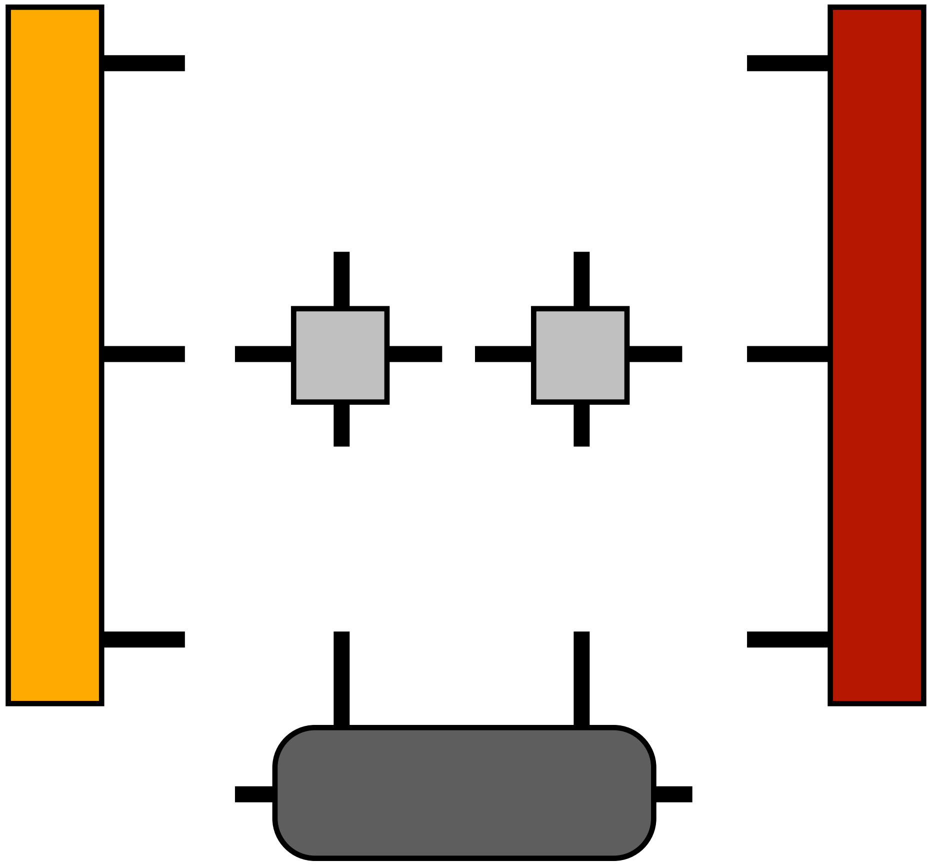
"Projected" eigenvalue problem



$$\tilde{H}|\tilde{\Psi}\rangle = \tilde{E}|\tilde{\Psi}\rangle$$

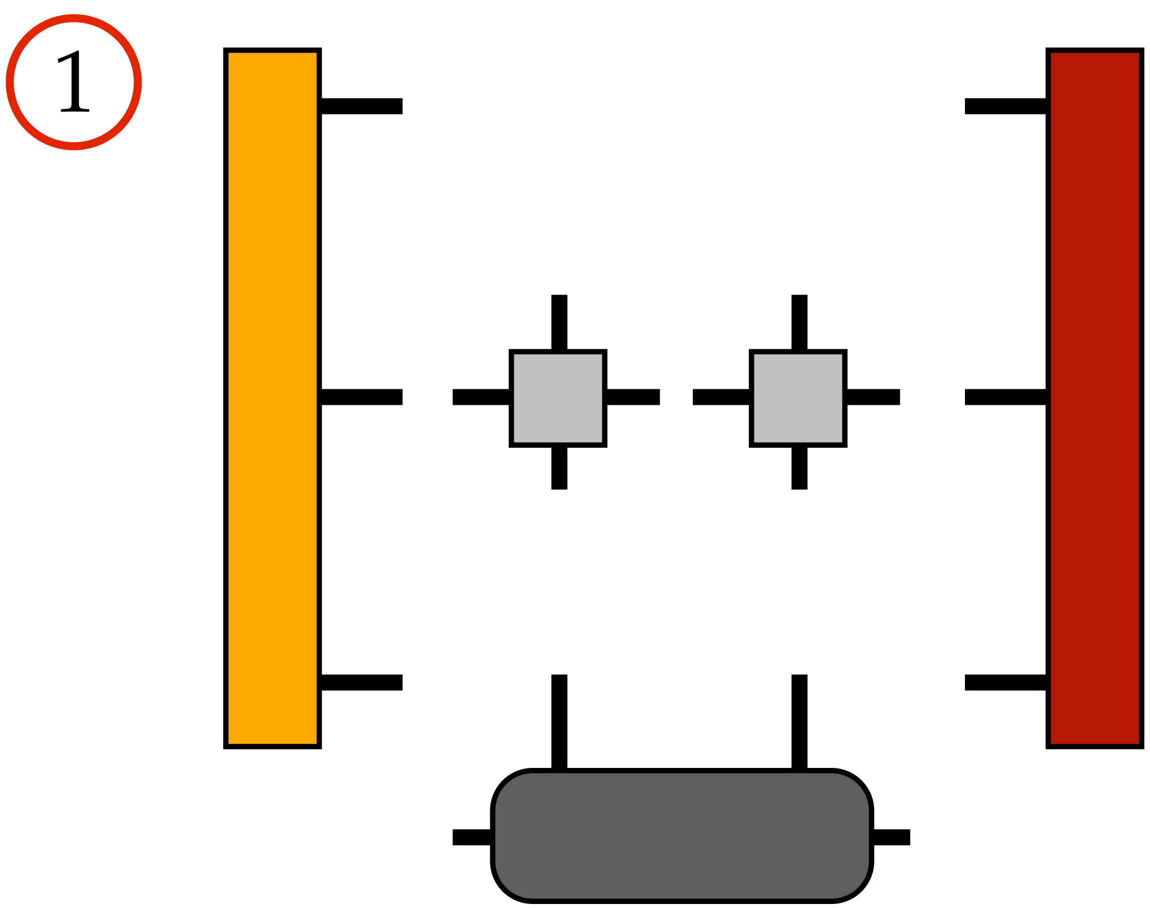
Can efficiently multiply projected $\tilde{H}$ times $|\tilde{\Psi}\rangle$

Order important!



Can efficiently multiply projected $\tilde{H}$ times $|\tilde{\Psi}\rangle$

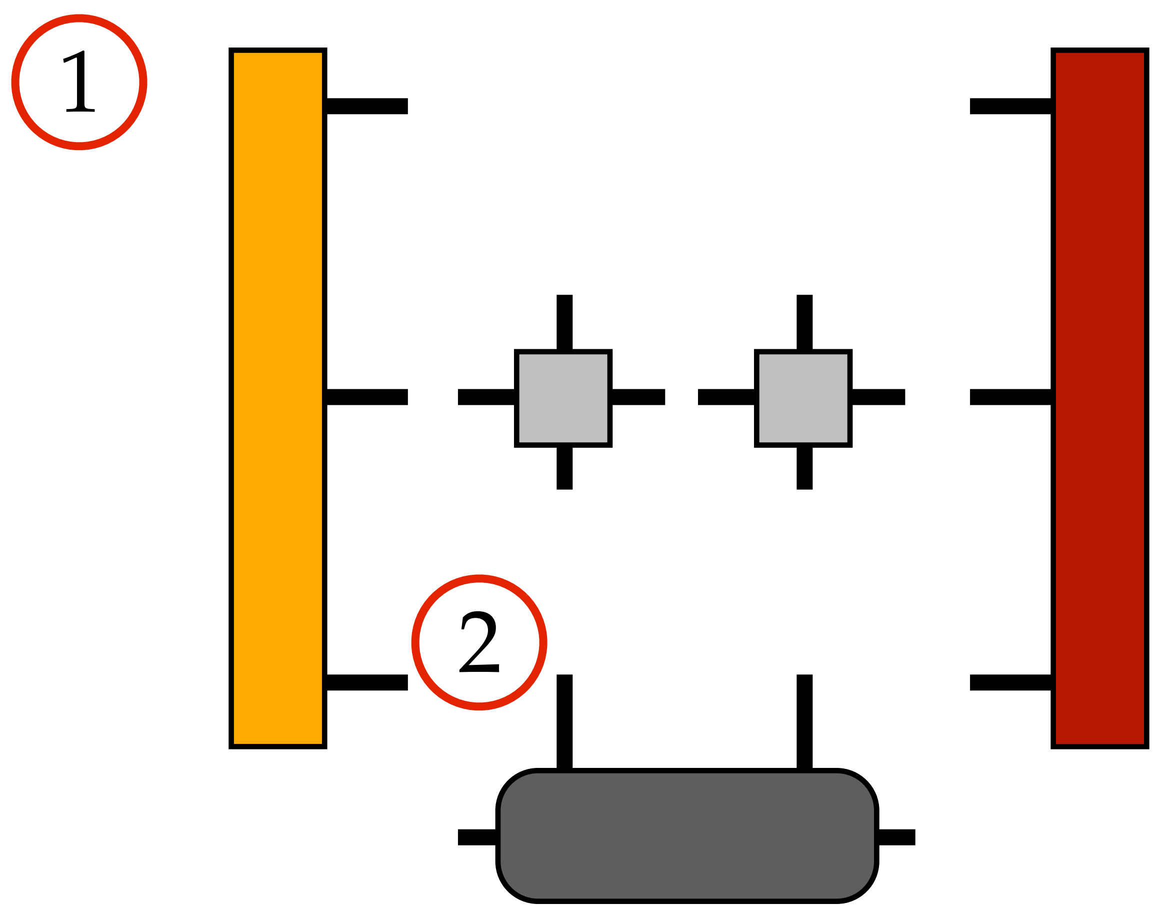
Order important!



Can efficiently multiply projected $\tilde{H}$ times $|\tilde{\Psi}\rangle$

Order important!

$2 \sim m^3$

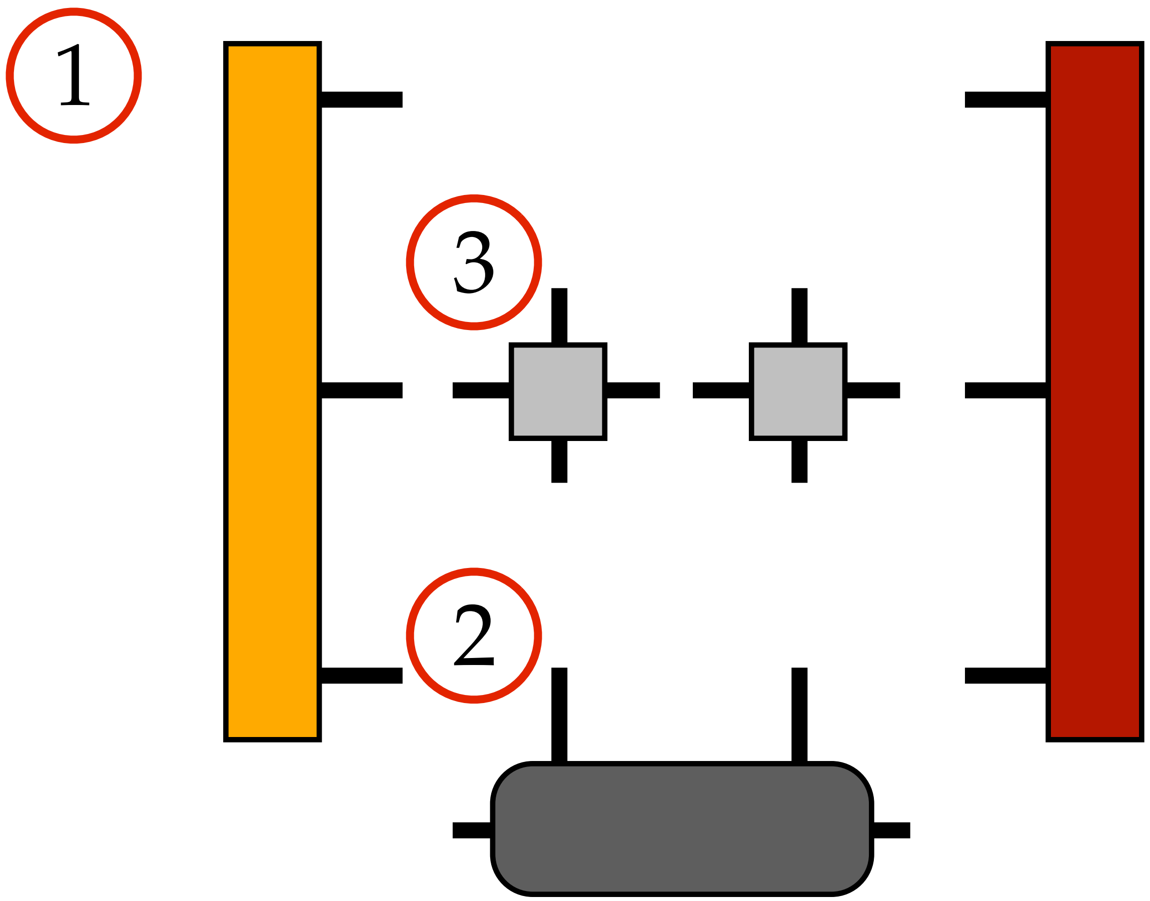


Can efficiently multiply projected $\tilde{H}$ times $|\tilde{\Psi}\rangle$

Order important!

$2 \sim m^3$

$3 \sim m^2$



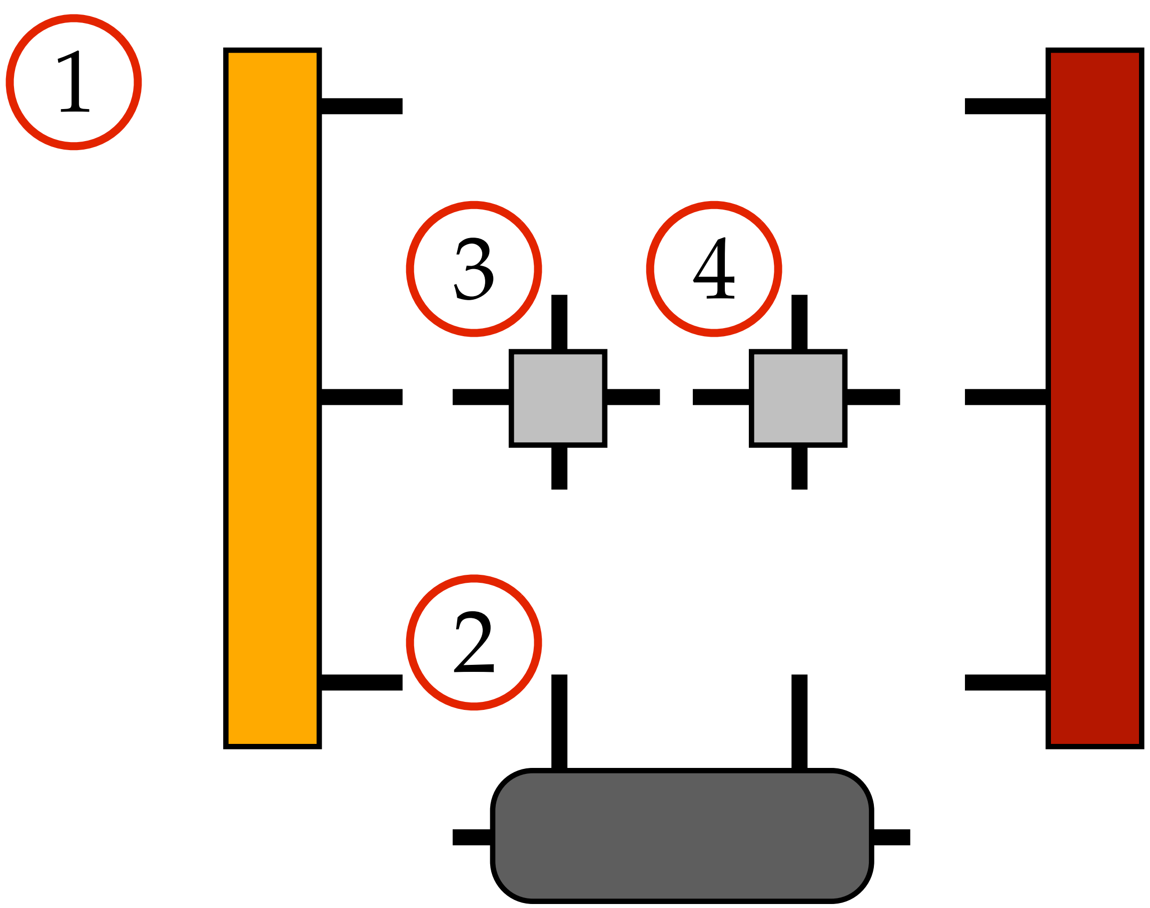
Can efficiently multiply projected $\tilde{H}$ times $|\tilde{\Psi}\rangle$

Order important!

$2 \sim m^3$

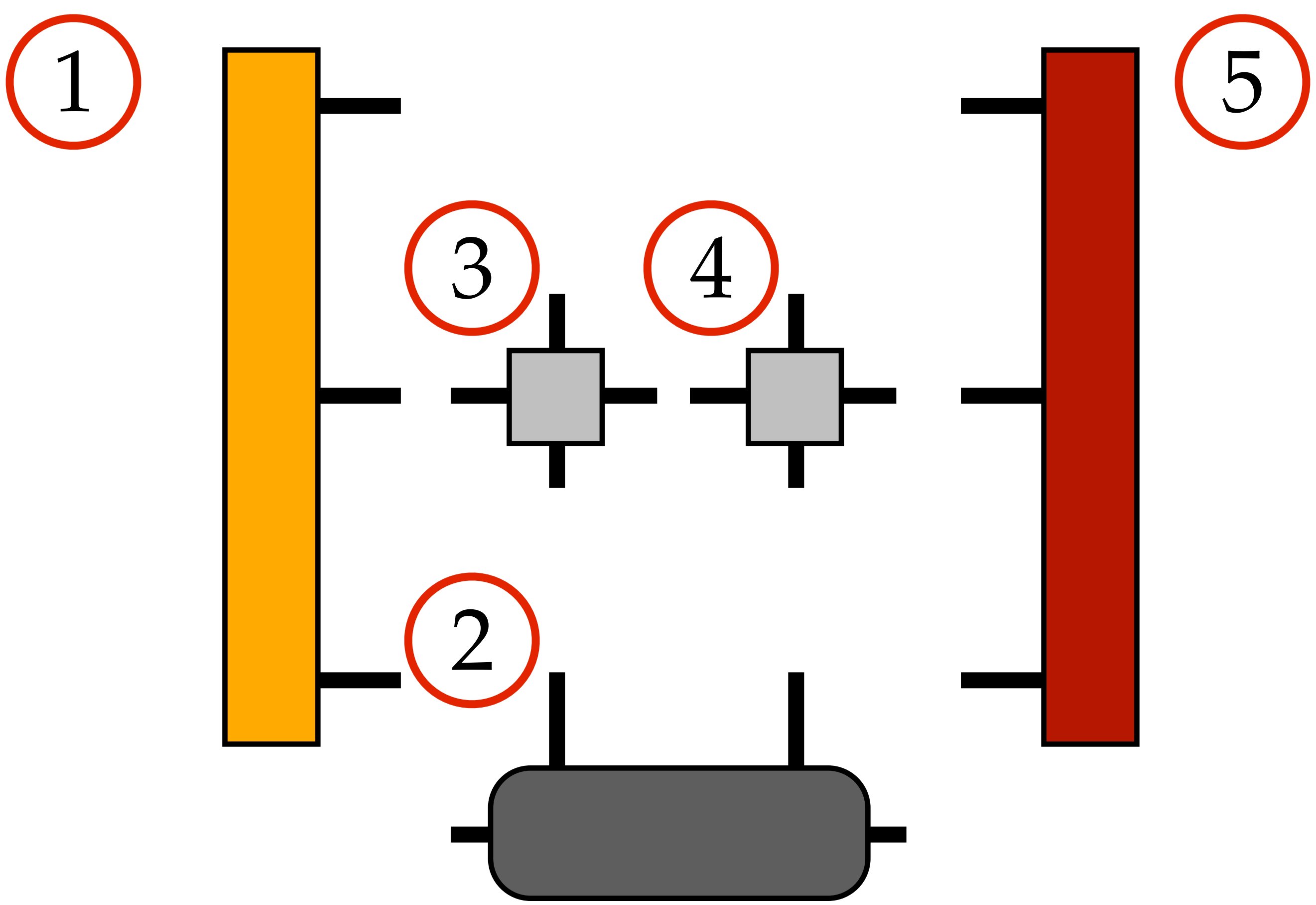
$3 \sim m^2$

$4 \sim m^2$



Can efficiently multiply projected $\tilde{H}$ times $|\tilde{\Psi}\rangle$

Order important!



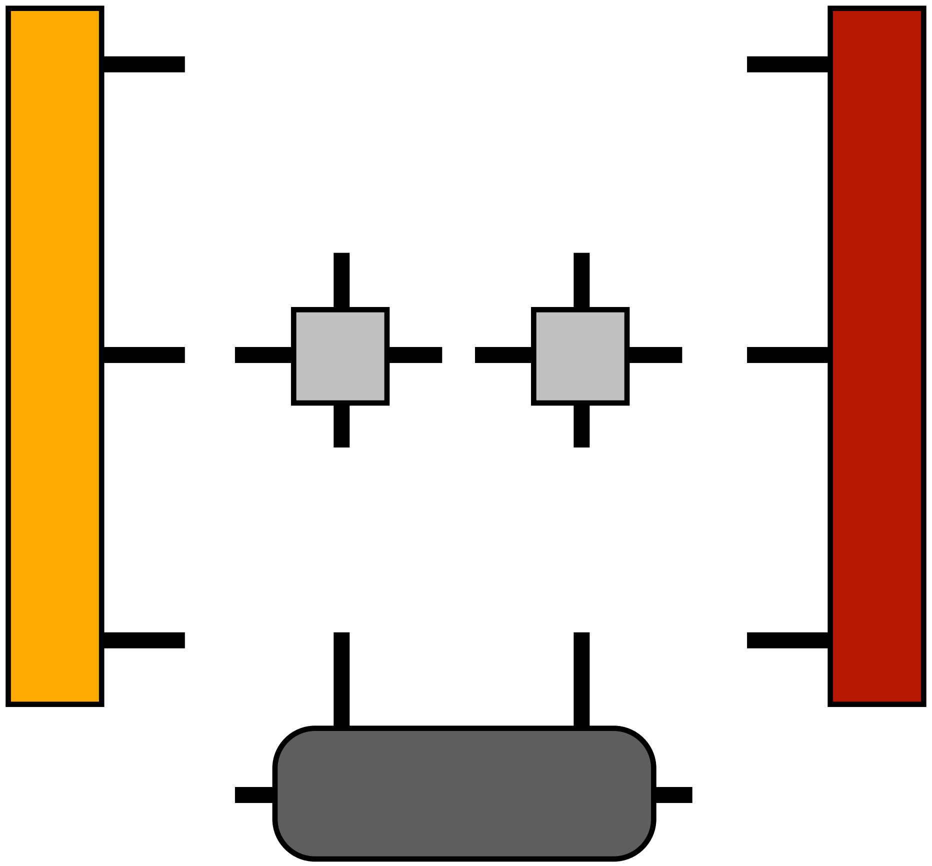
2 ~ m^3

3 ~ m^2

4 ~ m^2

5 ~ m^3

Use Lanczos or Davidson to solve
(iterative eigensolver)



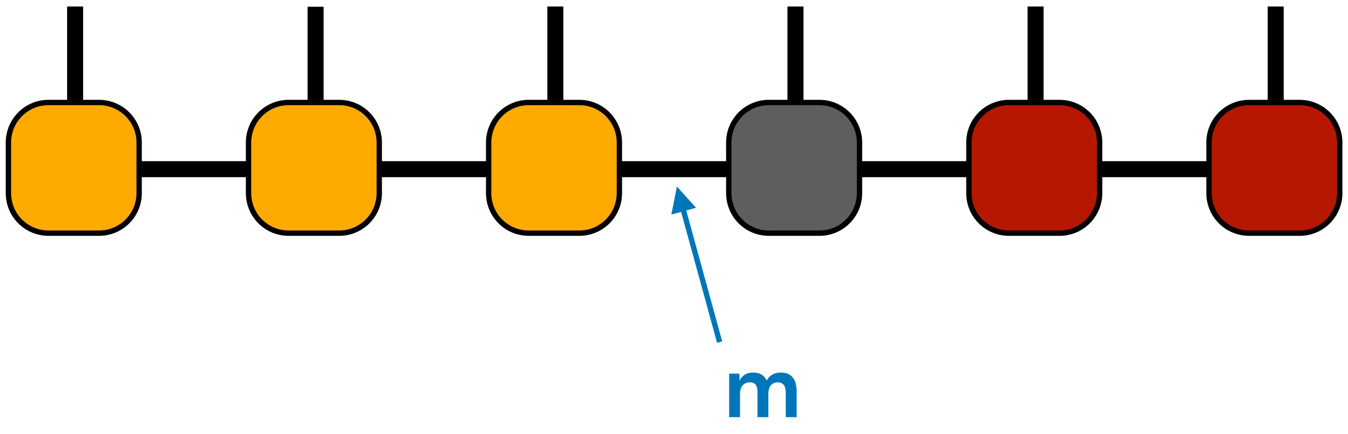
E.R. Davidson, J. Comput. Phys., **17**, 87 (1975)

SVD improved wavefunction (with truncation)
to restore MPS form and shift orthogonality center

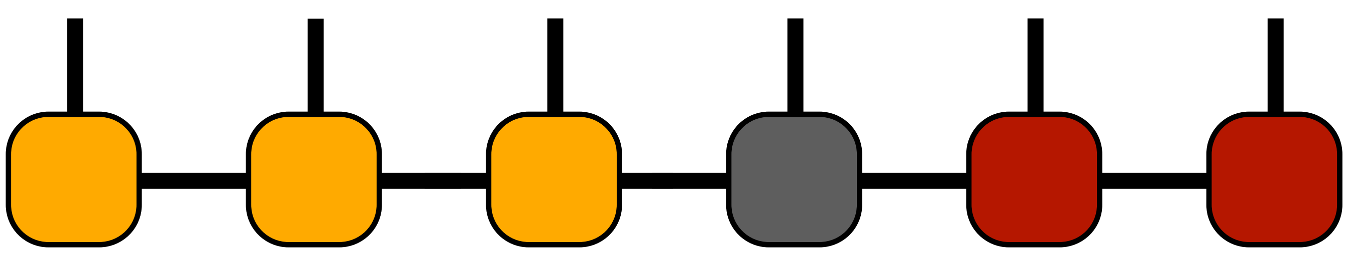
Number of singular values kept **m** is called
"number of states kept" in DMRG

SVD improved wavefunction (with truncation)
to restore MPS form and shift orthogonality center

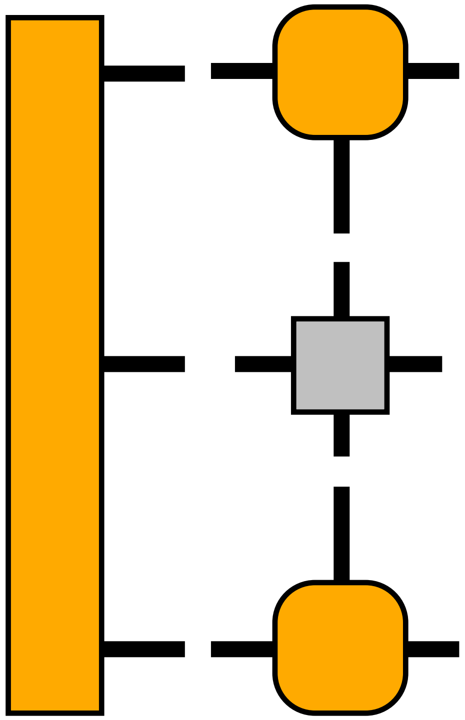
Number of singular values kept **m** is called
"number of states kept" in DMRG



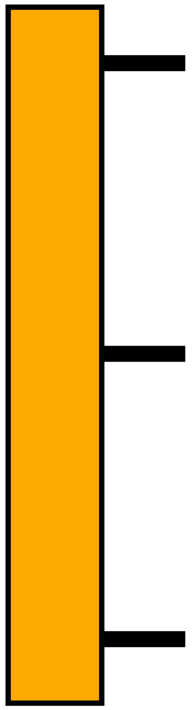
Grow projected Hamiltonian tensor from left



Grow projected Hamiltonian tensor from left

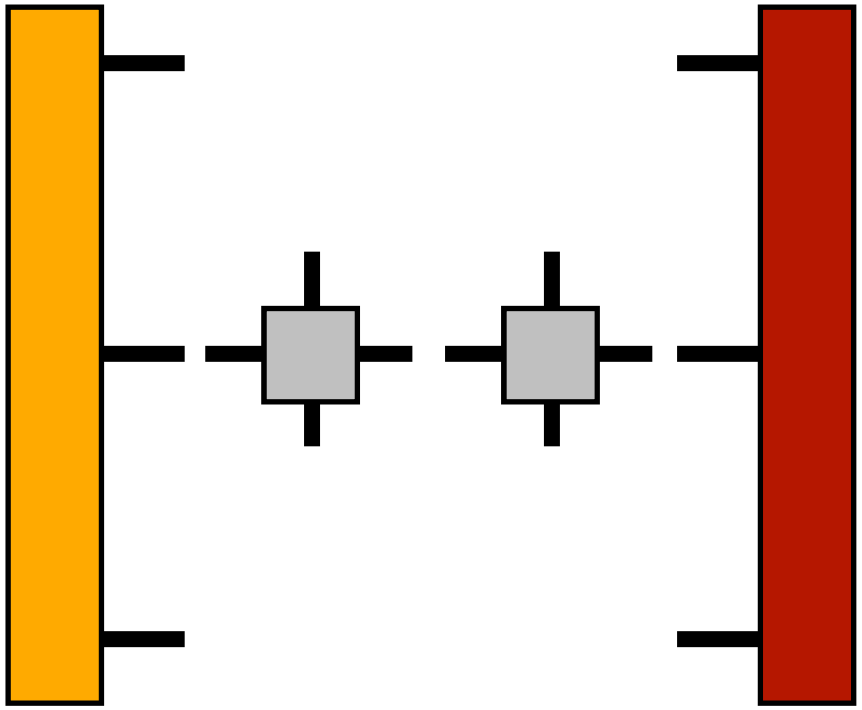


Grow projected Hamiltonian tensor from left

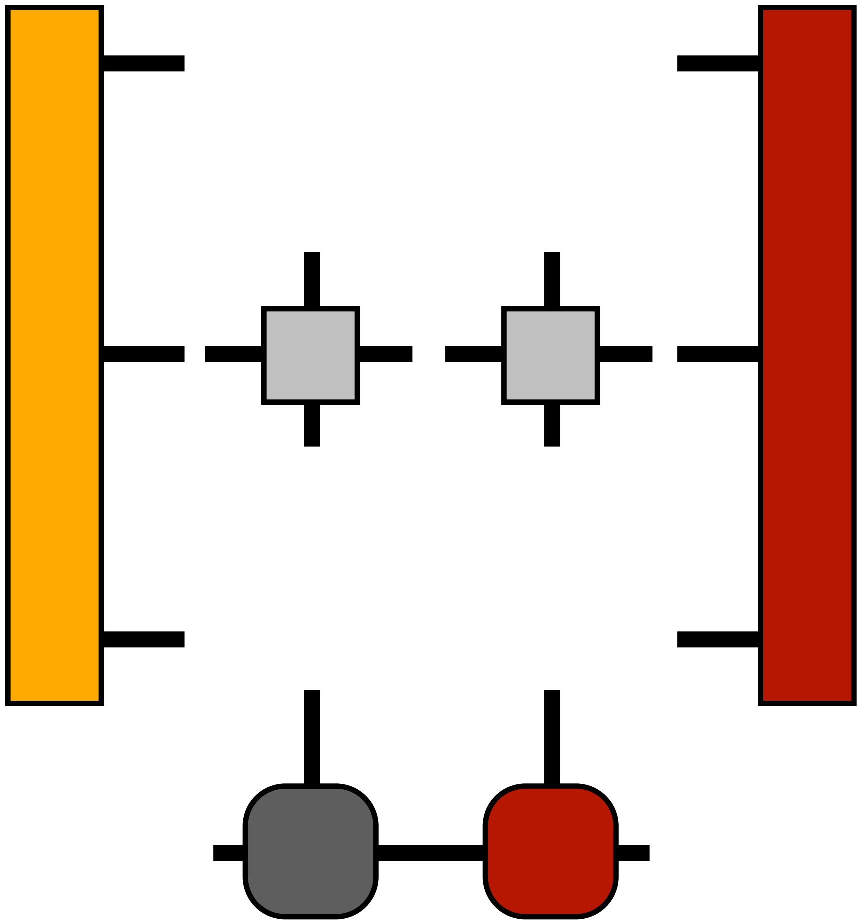


Grow projected Hamiltonian tensor from left

Recall right-hand projected H tensor from memory (saved in an array when made earlier)

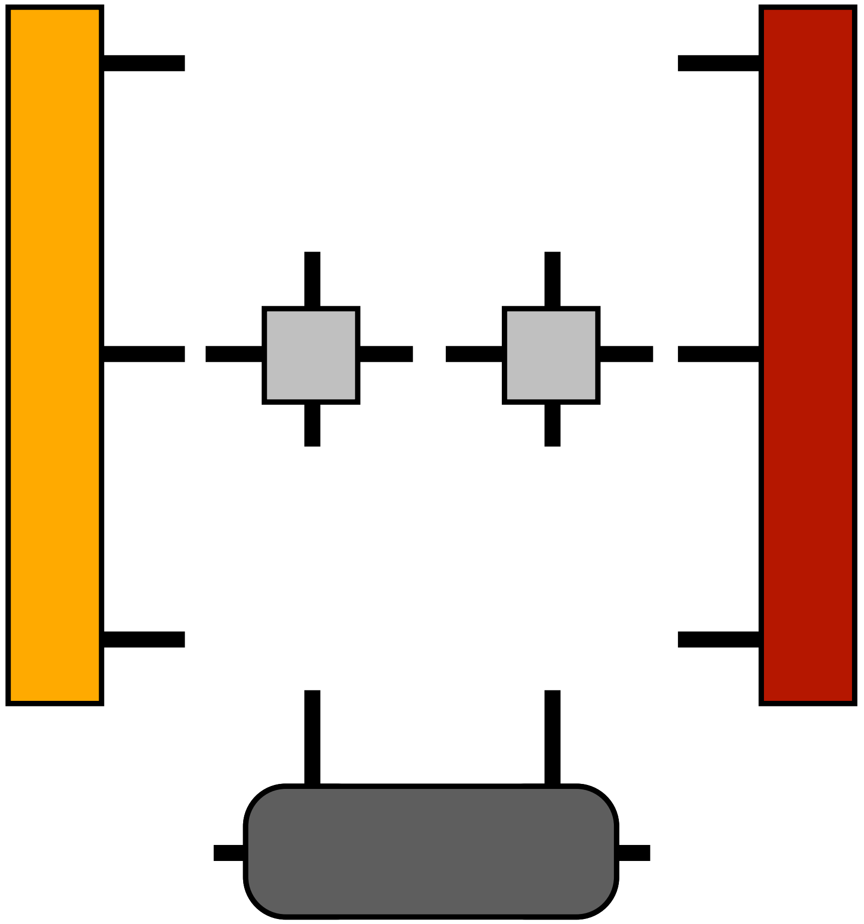


Iterating leads to sweeping procedure



Iterating leads to sweeping procedure

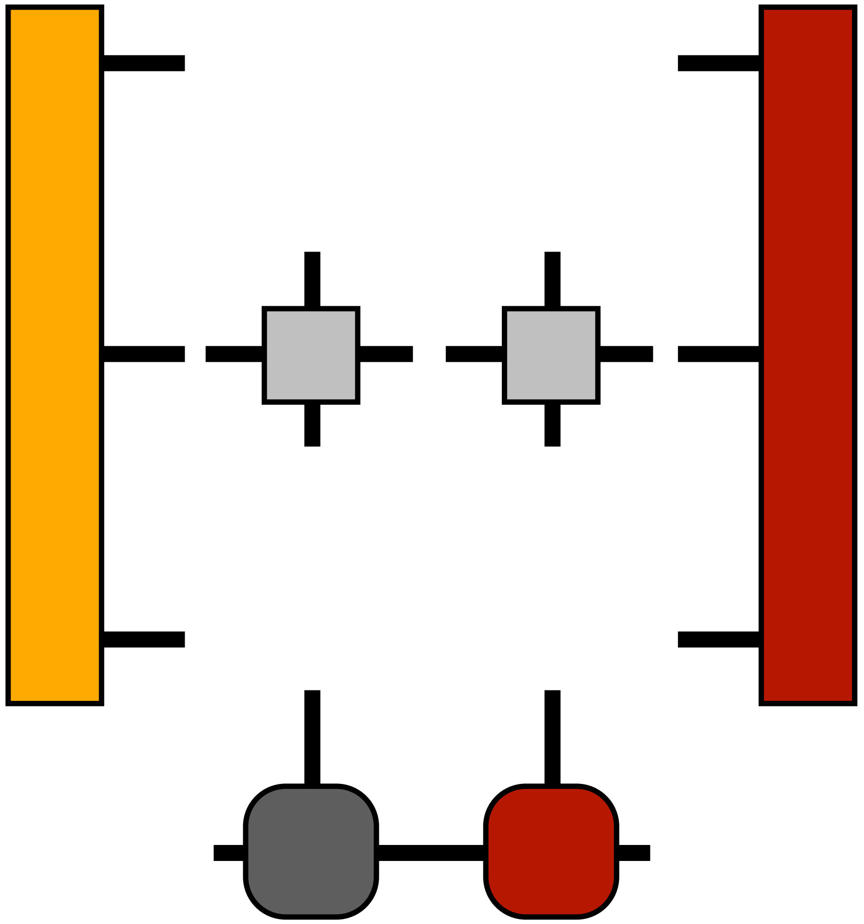
I. Solve eigenproblem



Iterating leads to sweeping procedure

I. Solve eigenproblem

II. SVD wavefunction

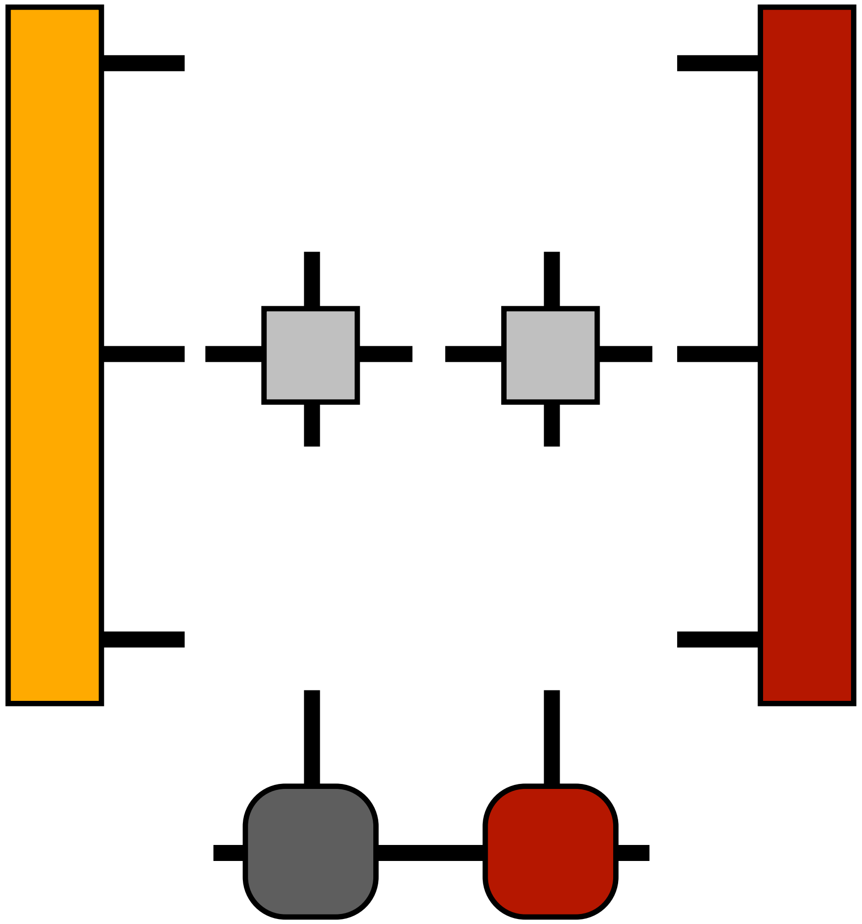


Iterating leads to sweeping procedure

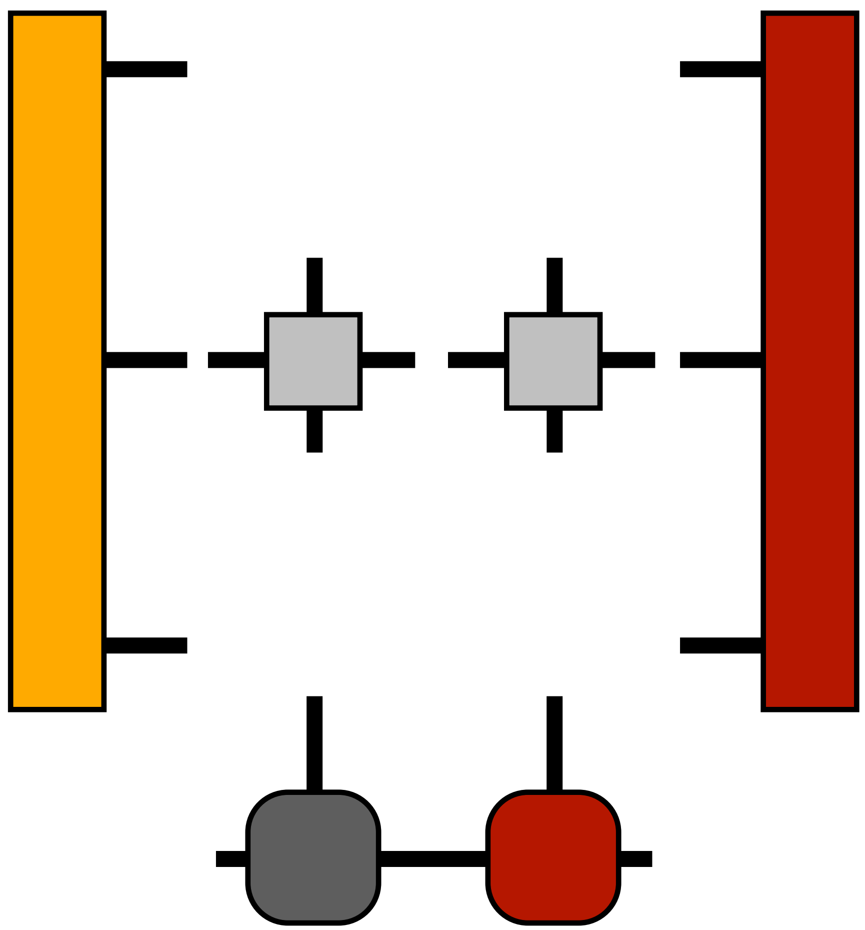
I. Solve eigenproblem

II. SVD wavefunction

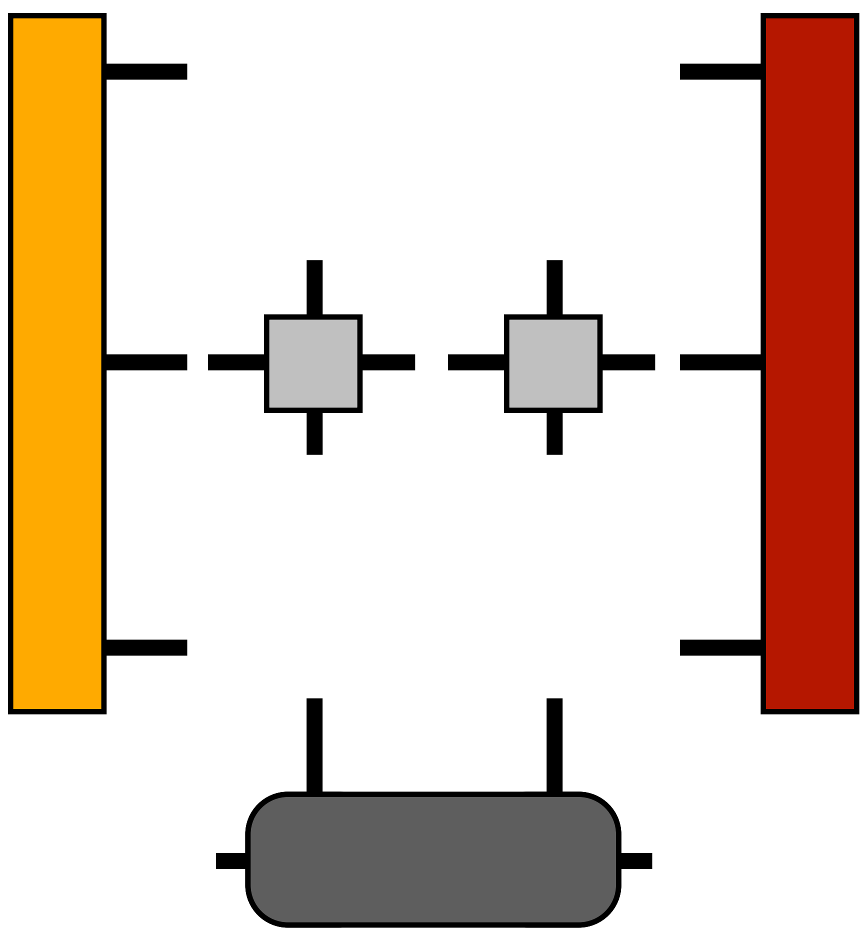
III. Shift projected H



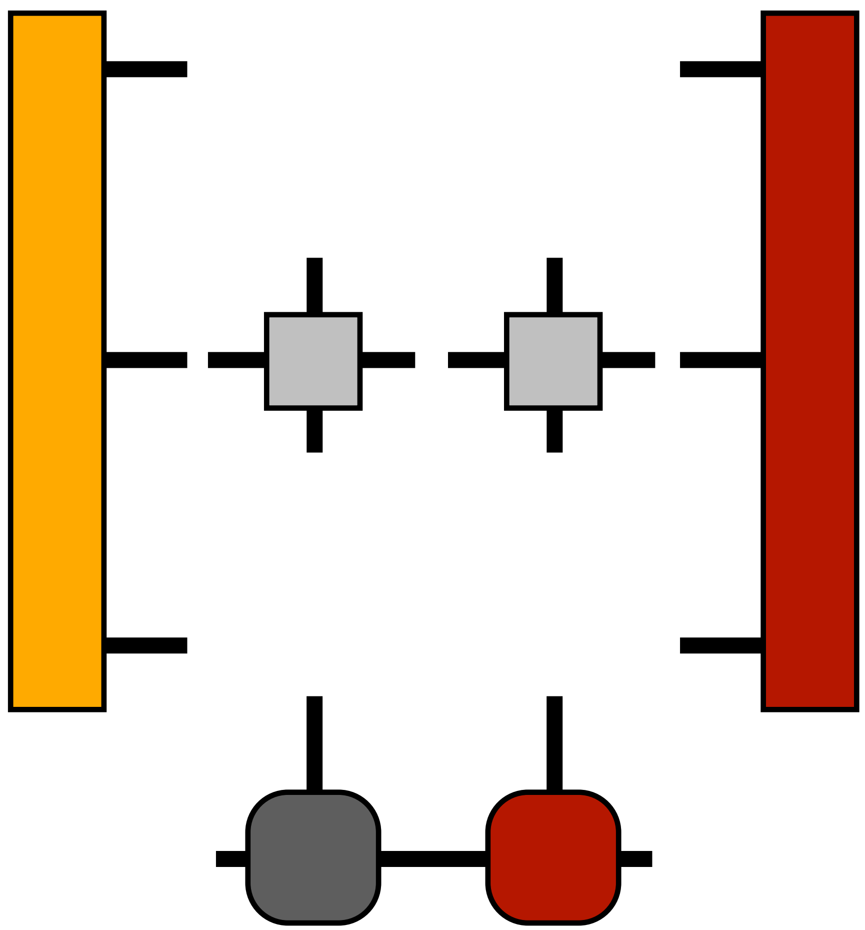
Iterating leads to sweeping procedure



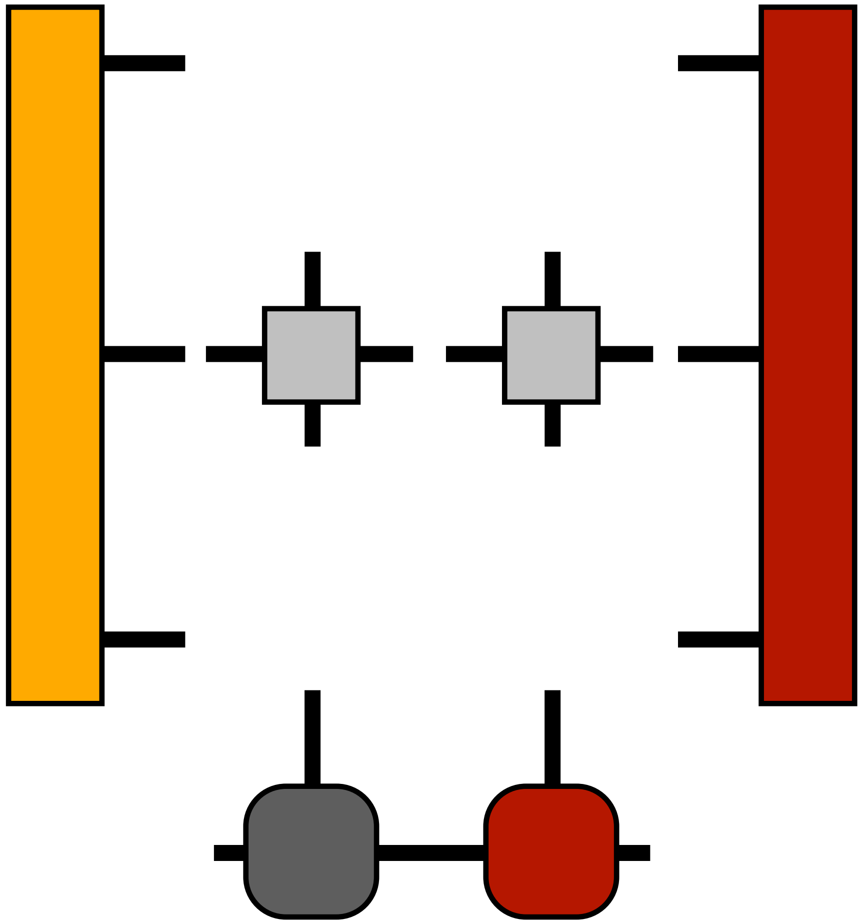
Iterating leads to sweeping procedure



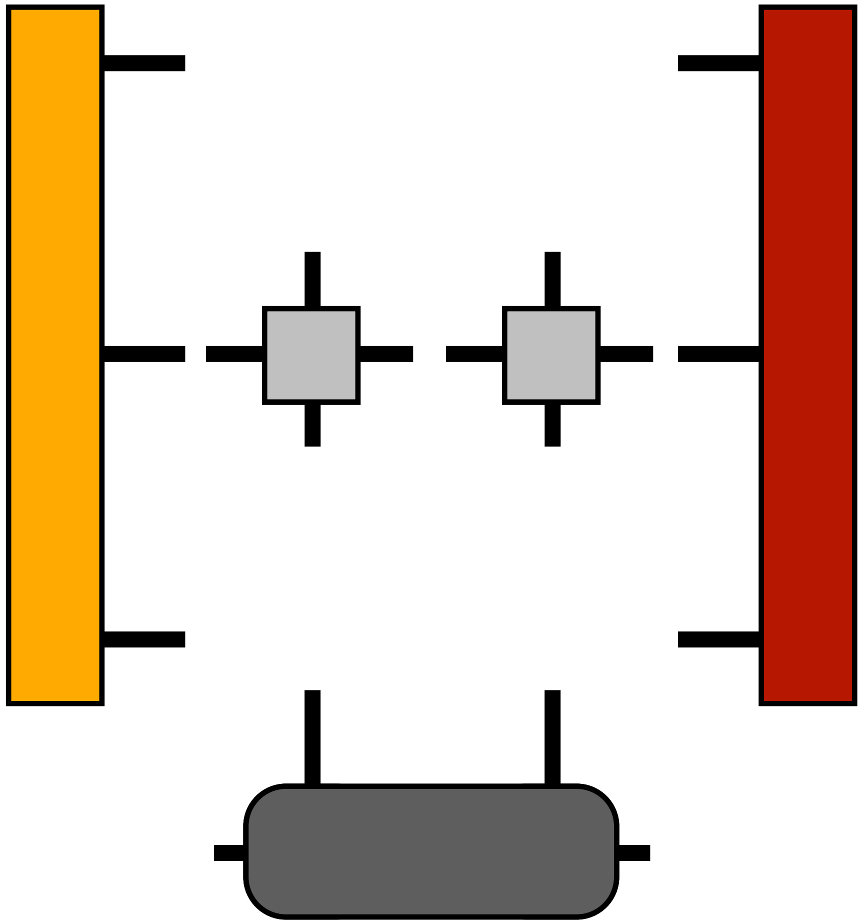
Iterating leads to sweeping procedure



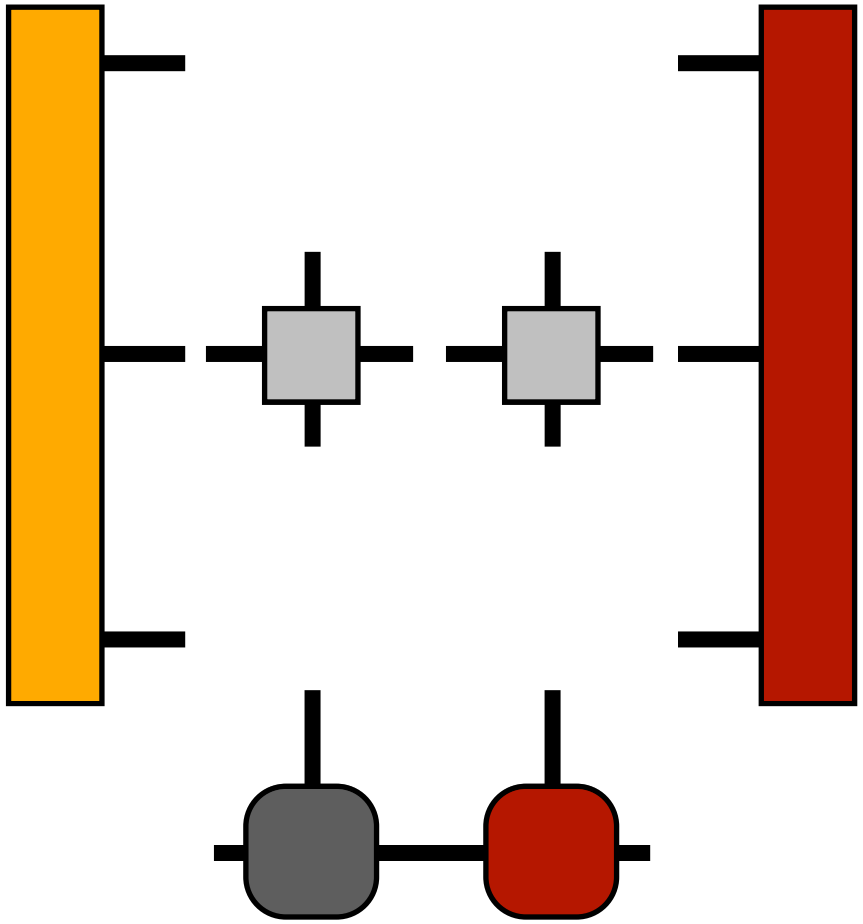
Iterating leads to sweeping procedure



Iterating leads to sweeping procedure



Iterating leads to sweeping procedure



DMRG can be used to get impressive results (in 1993!)

PHYSICAL REVIEW B VOLUME 48, NUMBER 6 1 AUGUST 1993-II

Numerical renormalization-group study of low-lying eigenstates of the antiferromagnetic $S = 1$ Heisenberg chain

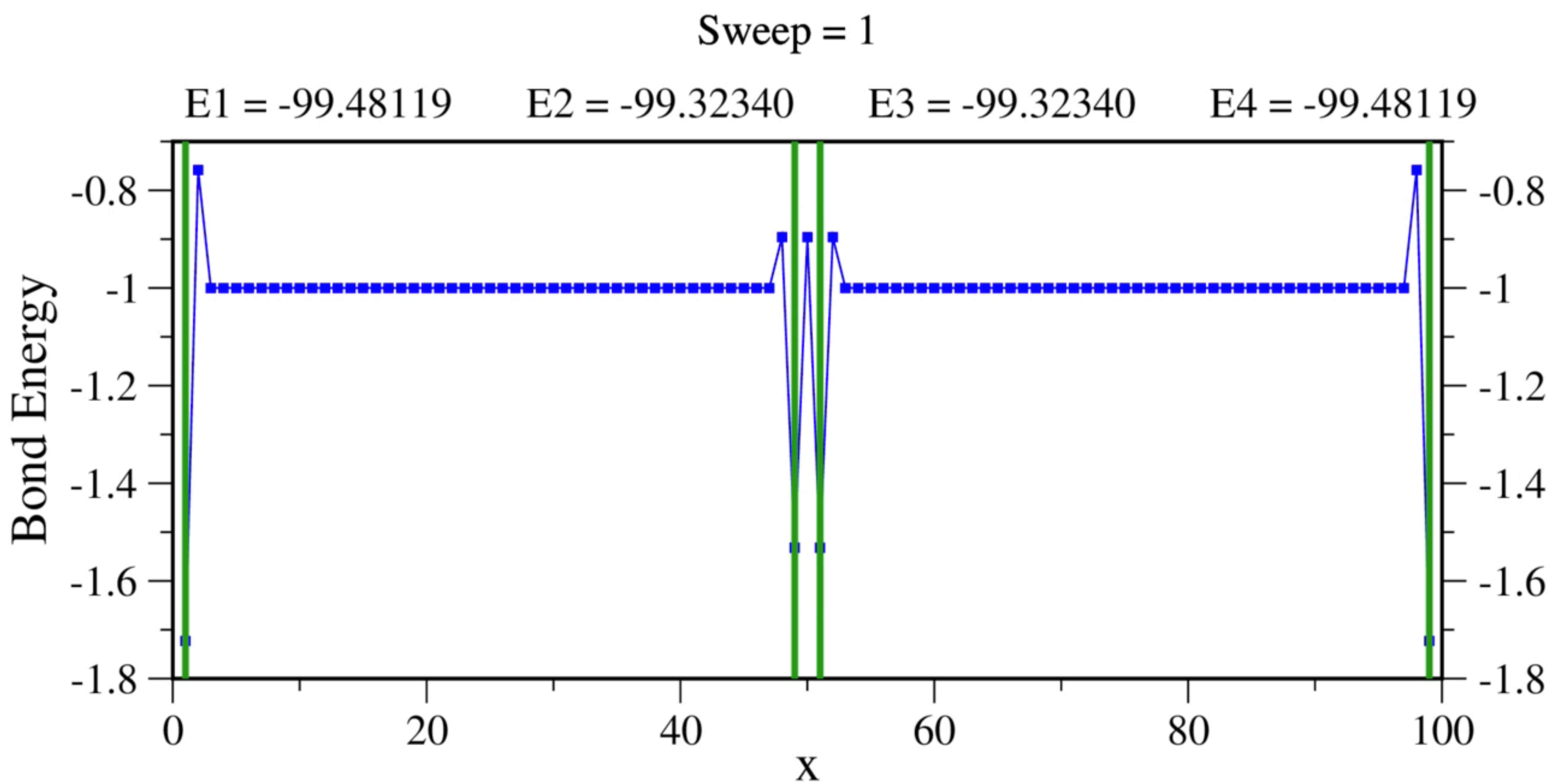
Steven R. White
Department of Physics, University of California, Irvine, California 92717

David A. Huse
AT&T Bell Labs, Murray Hill, New Jersey 07974

We present results of a numerical renormalization-group study of the isotropic $S = 1$ Heisenberg chain. The density-matrix renormalization-group techniques used allow us to calculate a variety of properties of the chain with unprecedented accuracy. The ground state energy per site of the infinite chain is found to be $e_0 \cong -1.401\,484\,038\,971(4)$. Open-ended $S = 1$ chains have effective $S = 1/2$ spins on each end, with exponential decay of the local spin moment away from the ends, with decay length $\xi \cong 6.03(1)$. The spin-spin correlation function also decays exponentially, and although the correlation length cannot be measured as accurately as the open-end decay length, it appears that the two lengths are identical. The string correlation function shows long-range order, with $g(\infty) \cong -0.374\,325\,096(2)$. The excitation energy of the first excited state, a state with one magnon with momentum $q = \pi$, is the Haldane gap, which we find to be $\Delta \cong 0.410\,50(2)$. We find many low-lying excited states, including one- and two-magnon states, for several different chain lengths.

DMRG can be run in parallel over separate computers

Parallel S=1 Heisenberg chain calculation:



Stoudenmire, White, PRB **87**, 155137 (2013)

DMRG can also be used to study quasi-2D systems

MPS path:

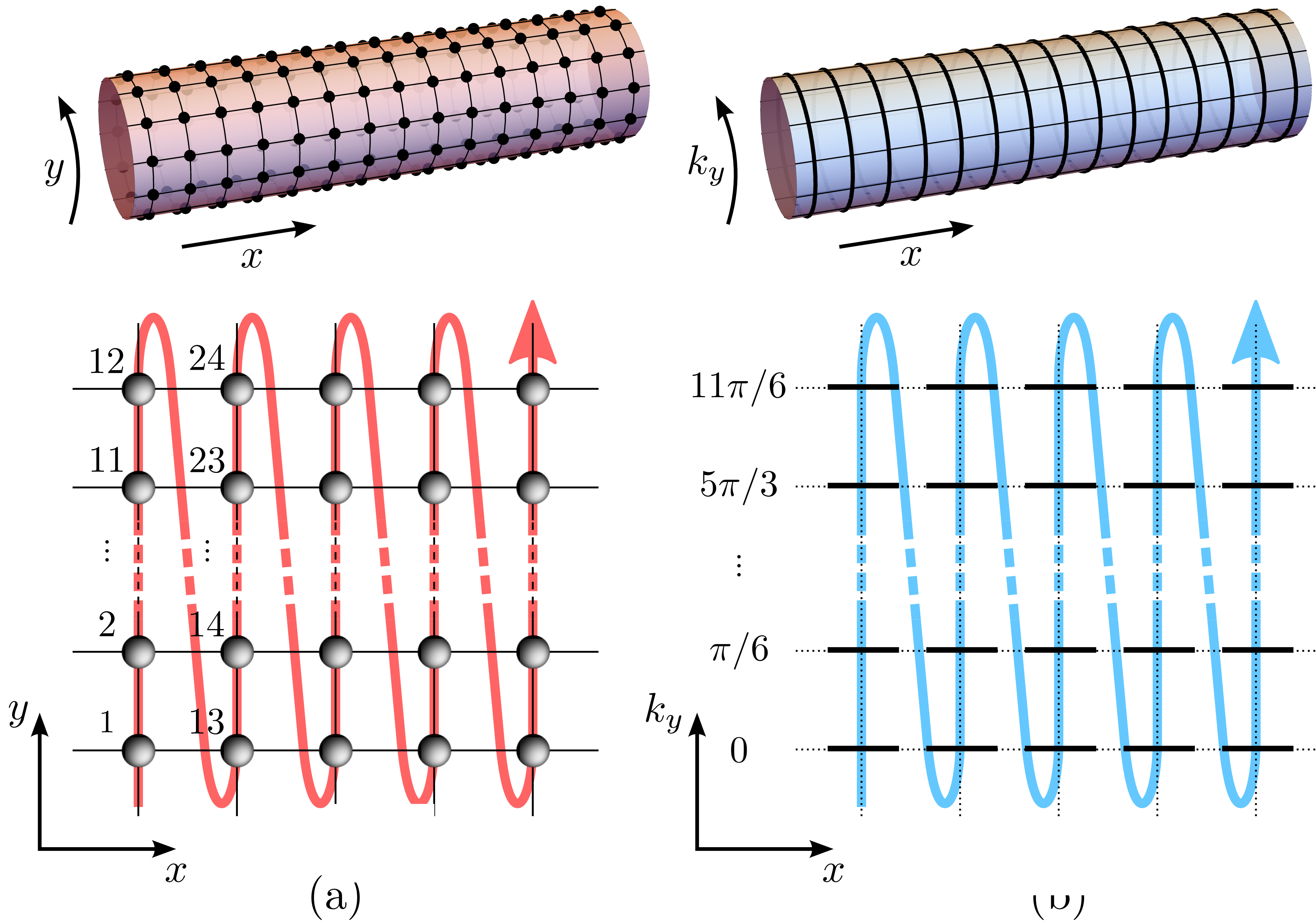
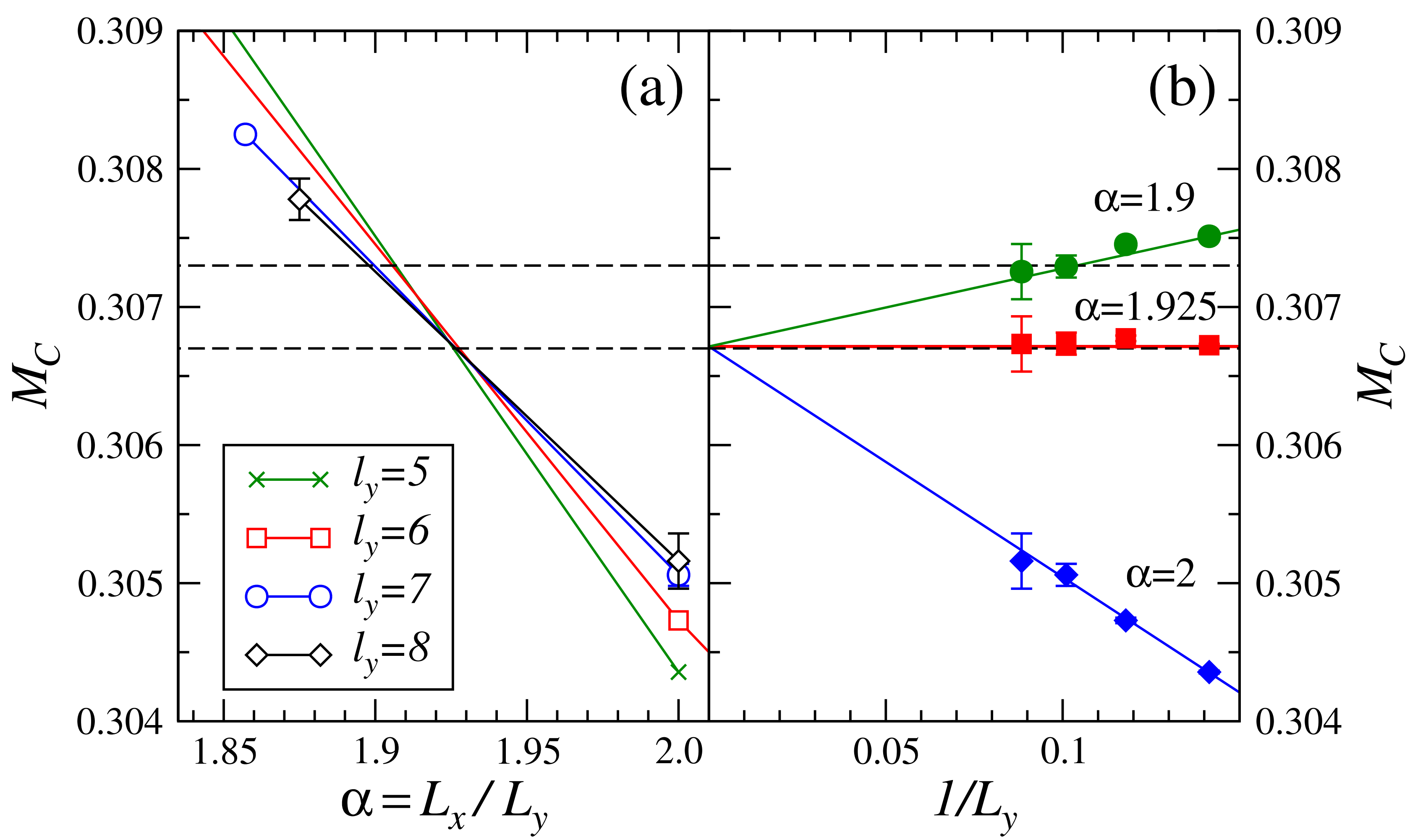


Figure from: Motruk, Zaletel, Mong, Pollmann, arxiv:1512.03318

With careful finite-size scaling,
2D DMRG competitive with quantum Monte Carlo

Magnetization of square-lattice Heisenberg model:

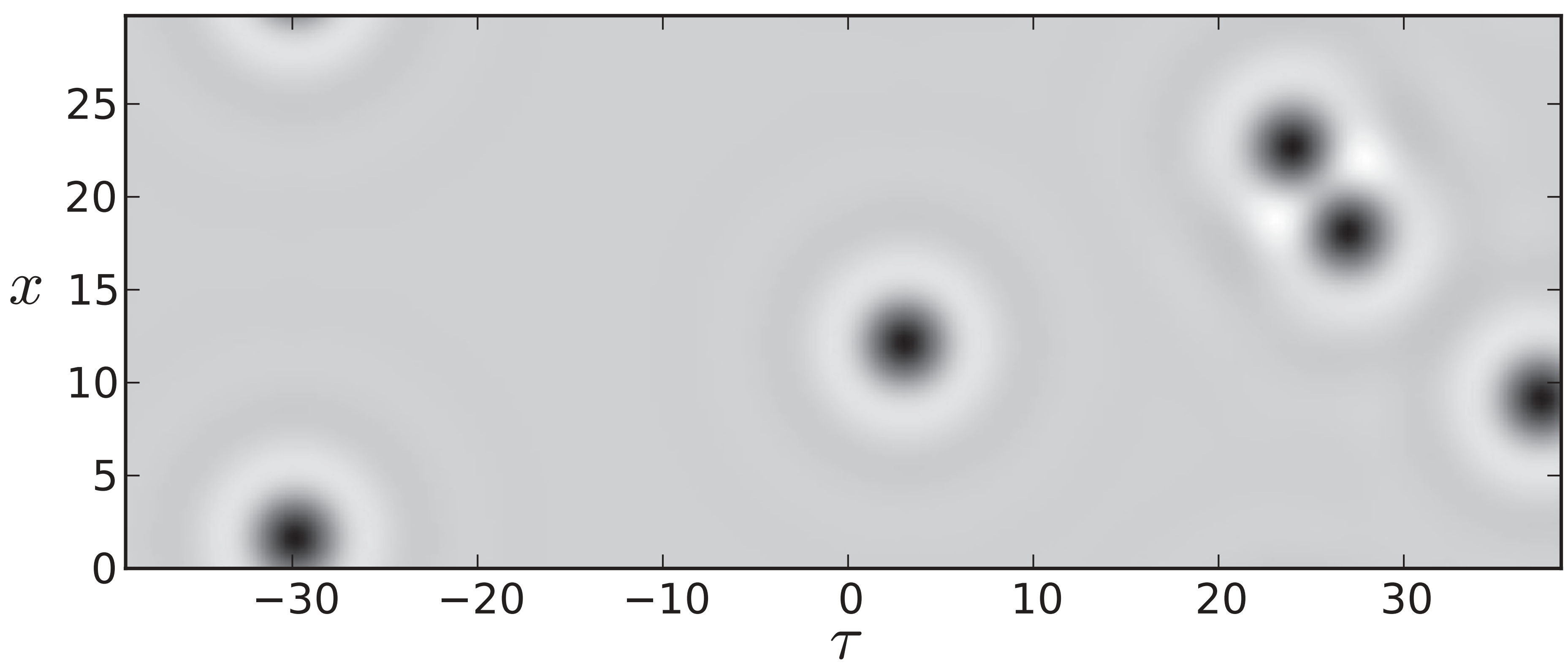
QMC bounds: {



White, Chernyshev, PRL **99**, 127004 (2007)

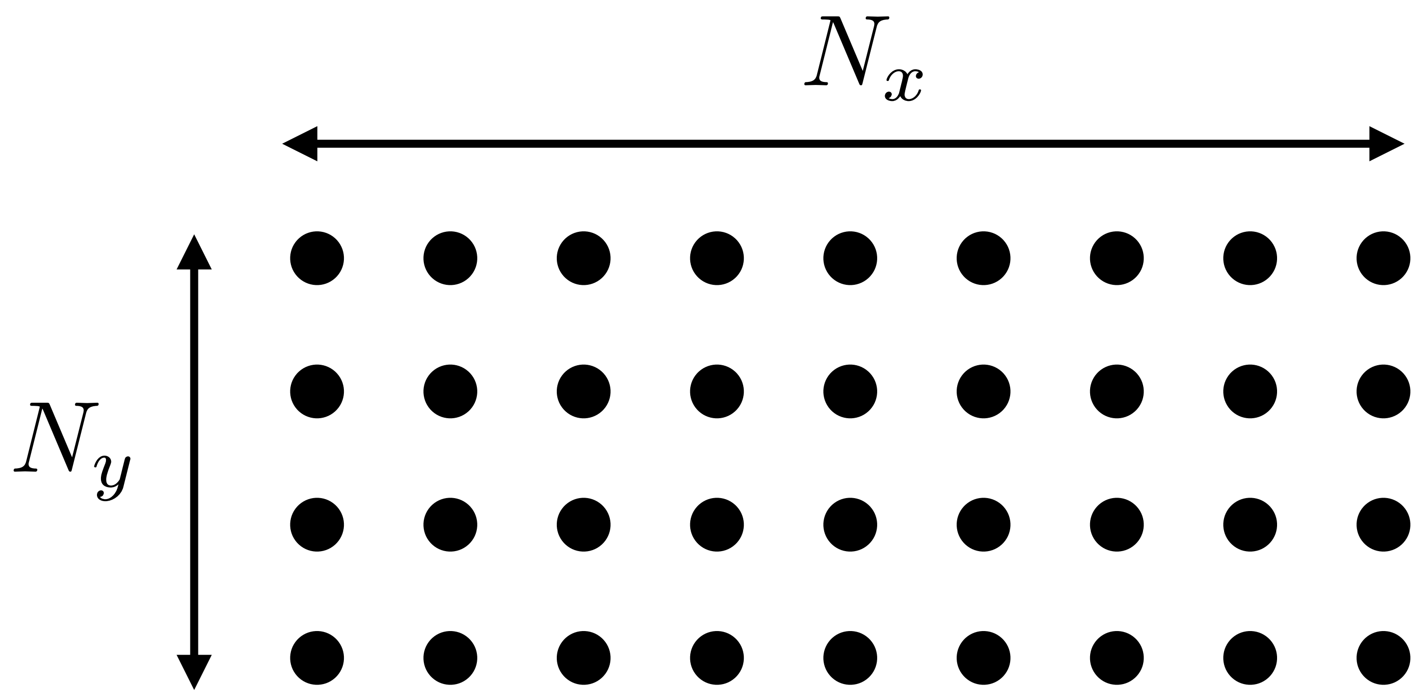
Using discrete set of 'orbitals', can study continuum quantum Hall systems on cylinders

Density plots of fractional "quasi-hole" excitations:



Zaletel, Mong, PRB **86**, 245305

DMRG for two-dimensional systems (cylinders) requires extreme care



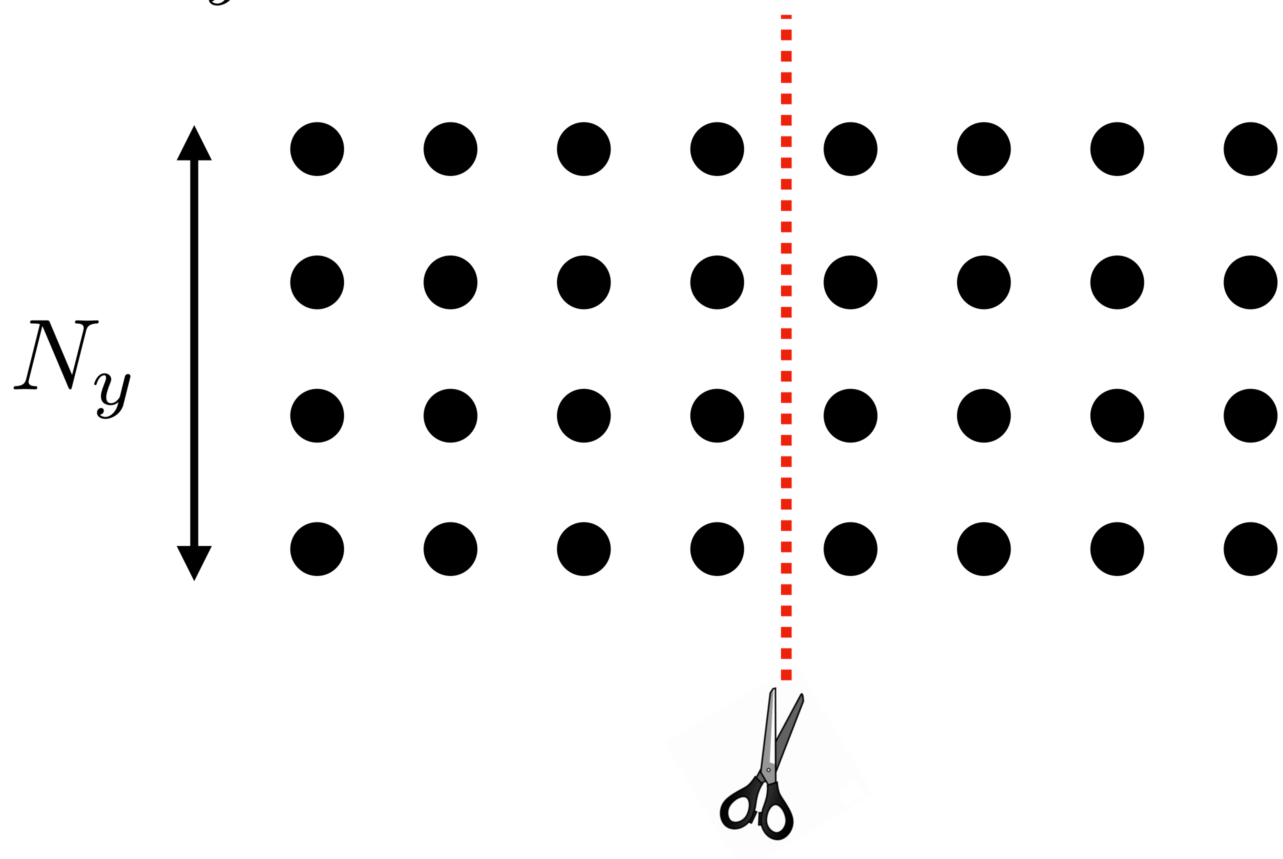
Scaling is: $N_x \, e^{a N_y}$

Like exact diagonalization, but only exponential in one direction (N_y), linear in other direction

Only N_y ~ 10-20 usually reachable

Why exponential in y direction?

If 2D ground state obeys boundary law, means $S \sim N_y$



Entanglement of MPS is bounded by log(m)

$$\implies S \sim N_y \sim \log(m)$$

$$\implies m \sim e^{N_y}$$

Takeaway

- 'Gauging' MPS important for accurate truncation, efficient measurement
- Matrix product operators (MPOs) can represent Hamiltonians in a generic way
- DMRG is a powerful algorithm for optimizing MPS