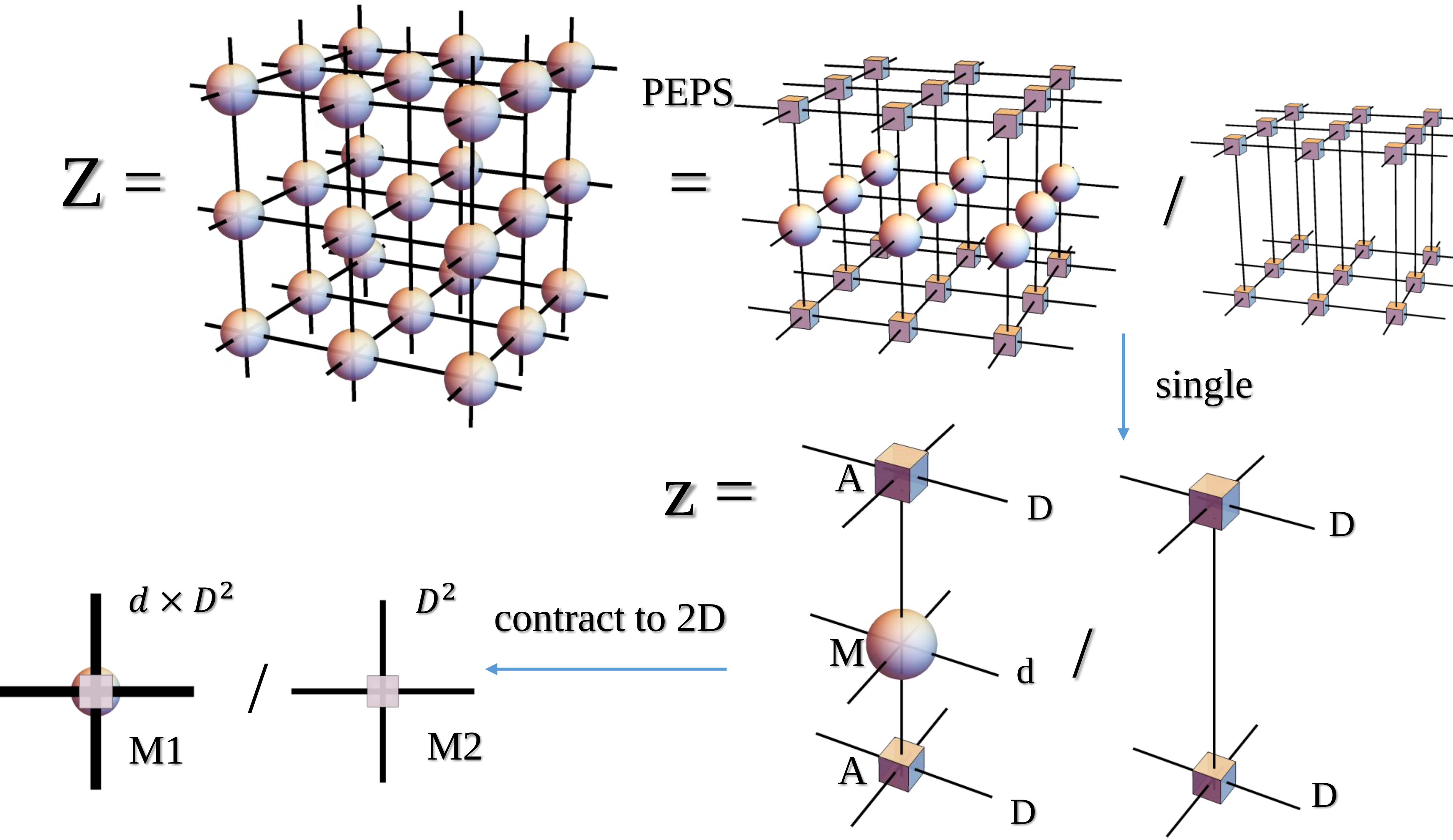
AD-BC-VUMPS

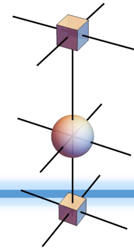
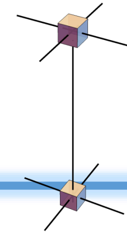
Contents

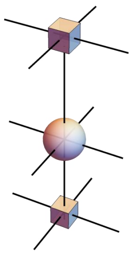
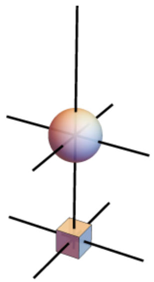
$$\begin{tikzcd}
\text{Why AD?} \arrow[r] & \text{AD-VUMPS} \arrow[d, "\text{big cell}"] \\
& \text{AD-BC-VUMPS}
\end{tikzcd}$$

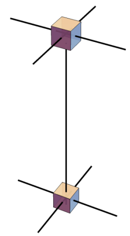
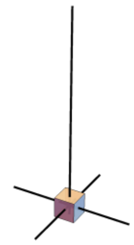
Why AD?

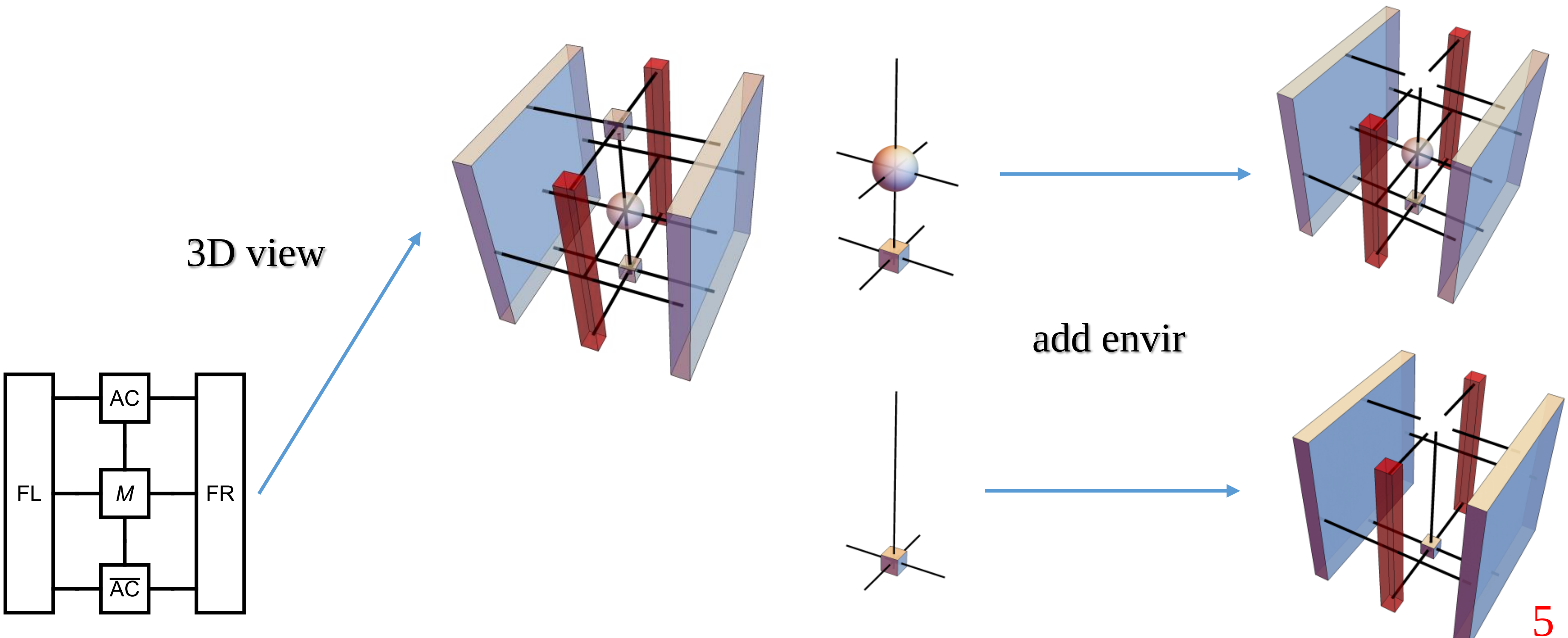
3D ising contract



Gradient to  and 

$\frac{\partial}{\partial A}($  $) = 2$ 

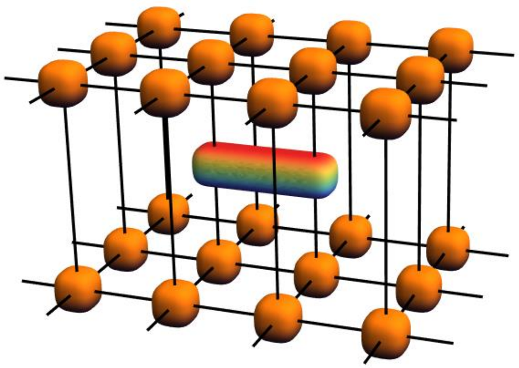
$\frac{\partial}{\partial A}($  $) = 2$ 

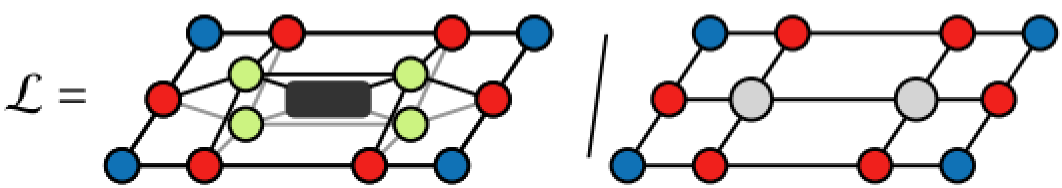


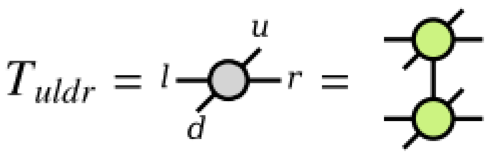
Quantum case

- 2D Energy

$$\min_{A} E(A) = \min_{A} \frac{\langle \Psi(A) | \hat{H} | \Psi(A) \rangle}{\langle \Psi(A) | \Psi(A) \rangle}$$

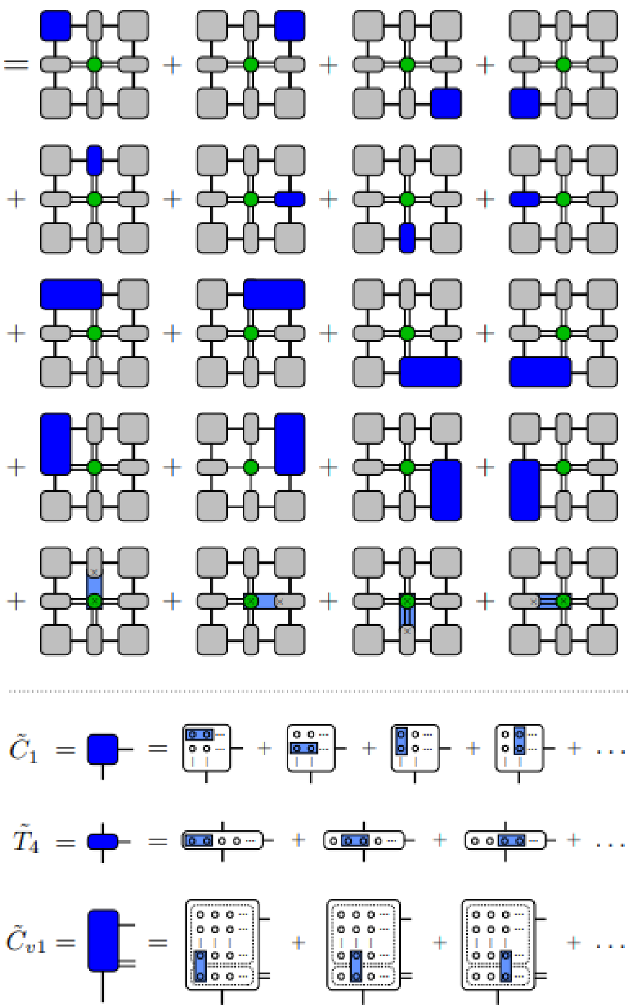


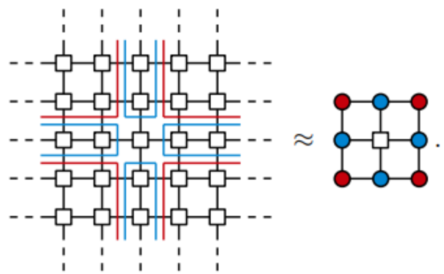


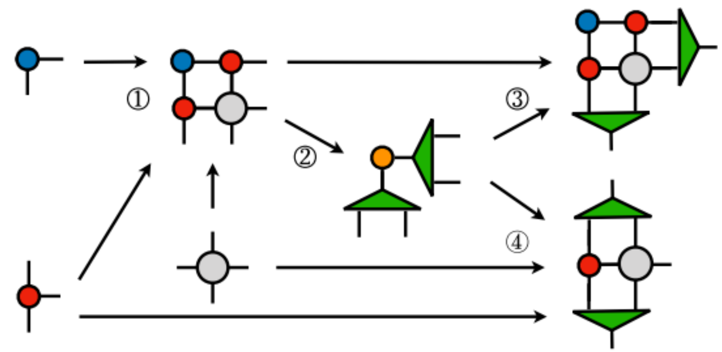


CTMRG

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







- Experimental optimization method

(1) Compute $E(1)$ (corresponding to the previous energy with the old tensor $A' = A$) and $E(0.5)$ (corresponding to the energy with $A' = \tilde{A}$).

(2) If $E(0.5) < E(1)$, take $A' = \tilde{A}$ as the solution and exit.

(3) Define an initial step size Δ_0 (e.g., $\Delta_0 = 0.1$) and a tiny step size h (e.g., $h = 10^{-4}$).

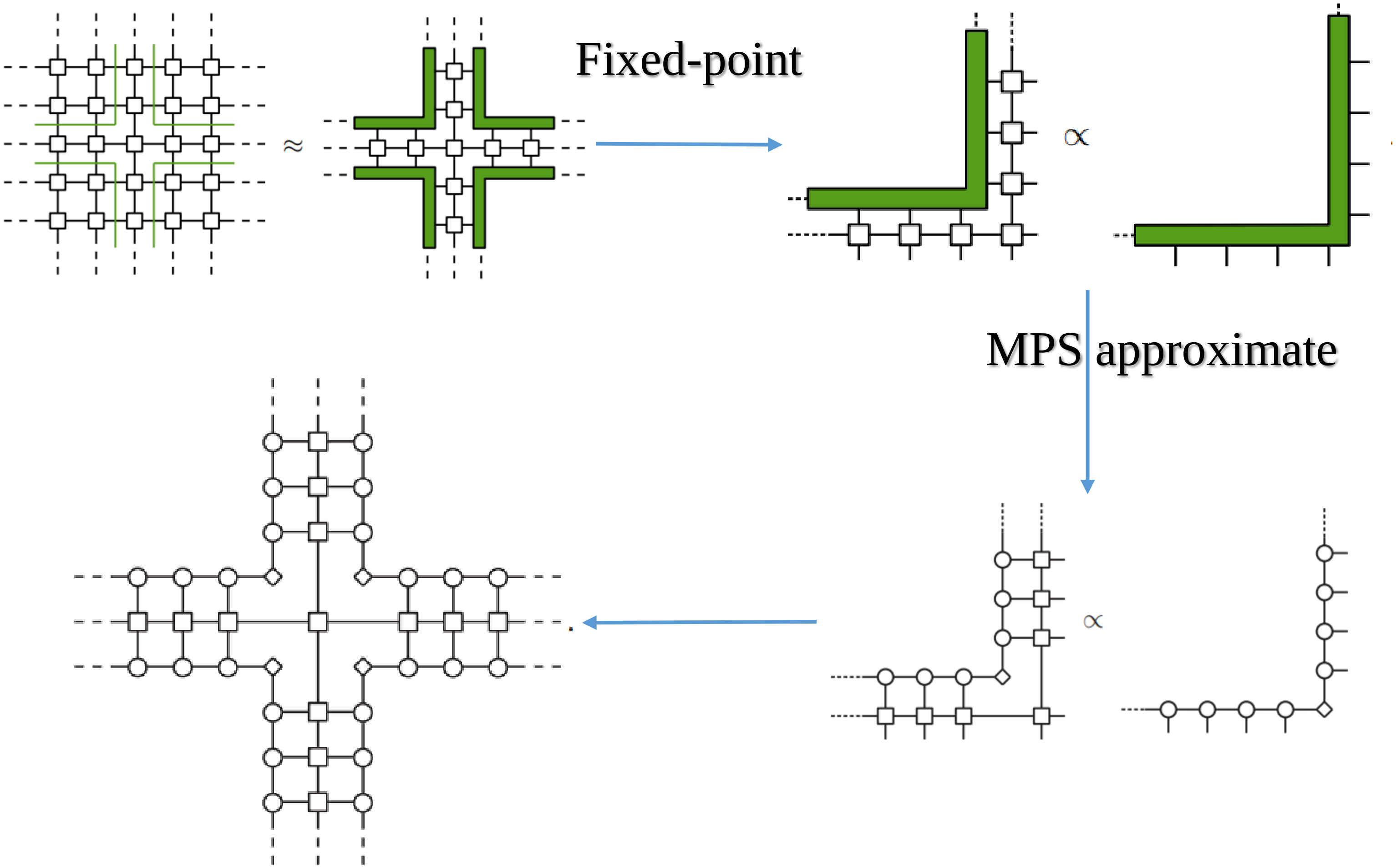
(4) If $E(1+h) < E(1)$, set $\Delta = \Delta_0$, else $\Delta = -\Delta_0$.

(5) For *iter* = 1 to *maxiter*

(a) If $E(1+\Delta) < E(1)$, accept solution [44] with $\lambda = 1 + \Delta$ and exit.

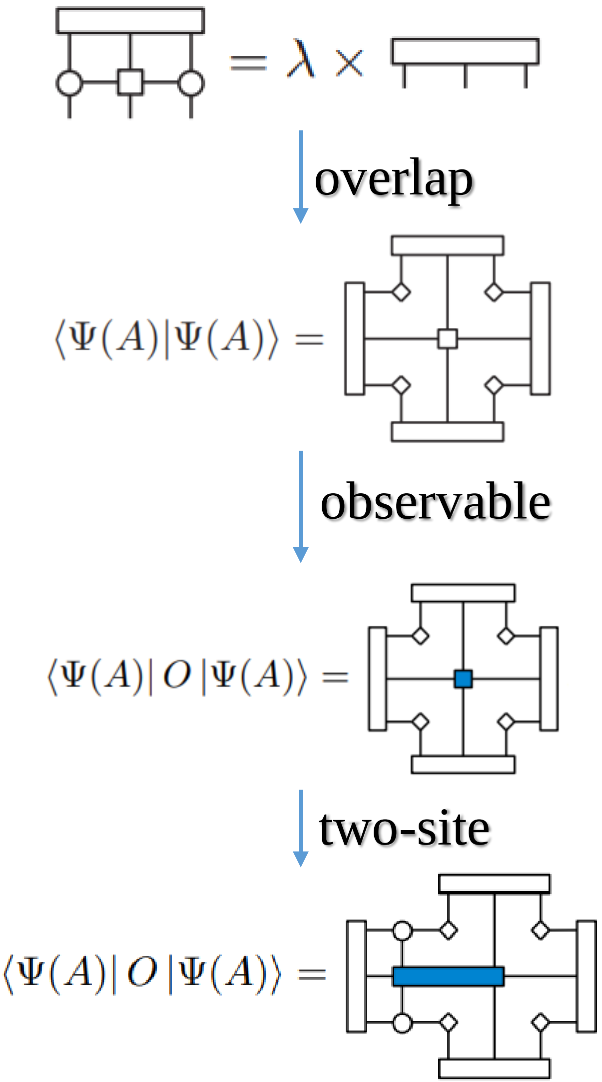
(b) Else $\Delta = \Delta/2$.

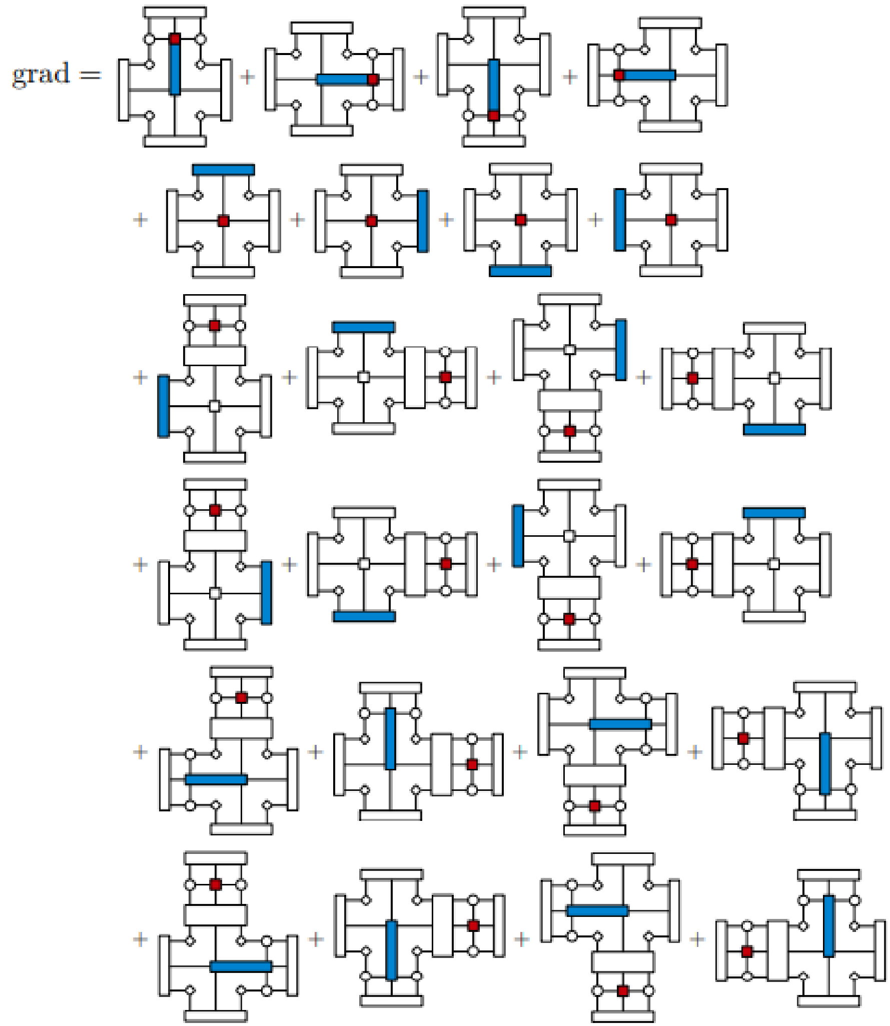
Channel environments



PRB 94, 155123 (2016)

Gradient

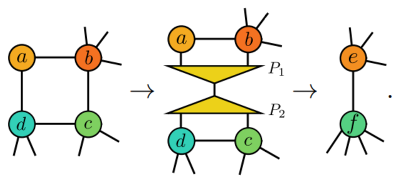




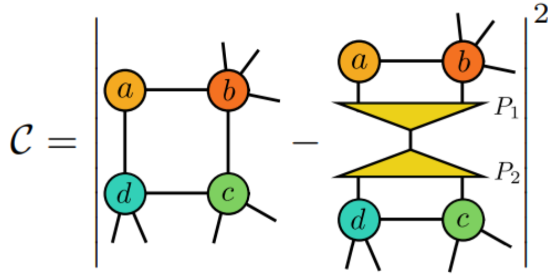
PRB 94, 155123 (2016)

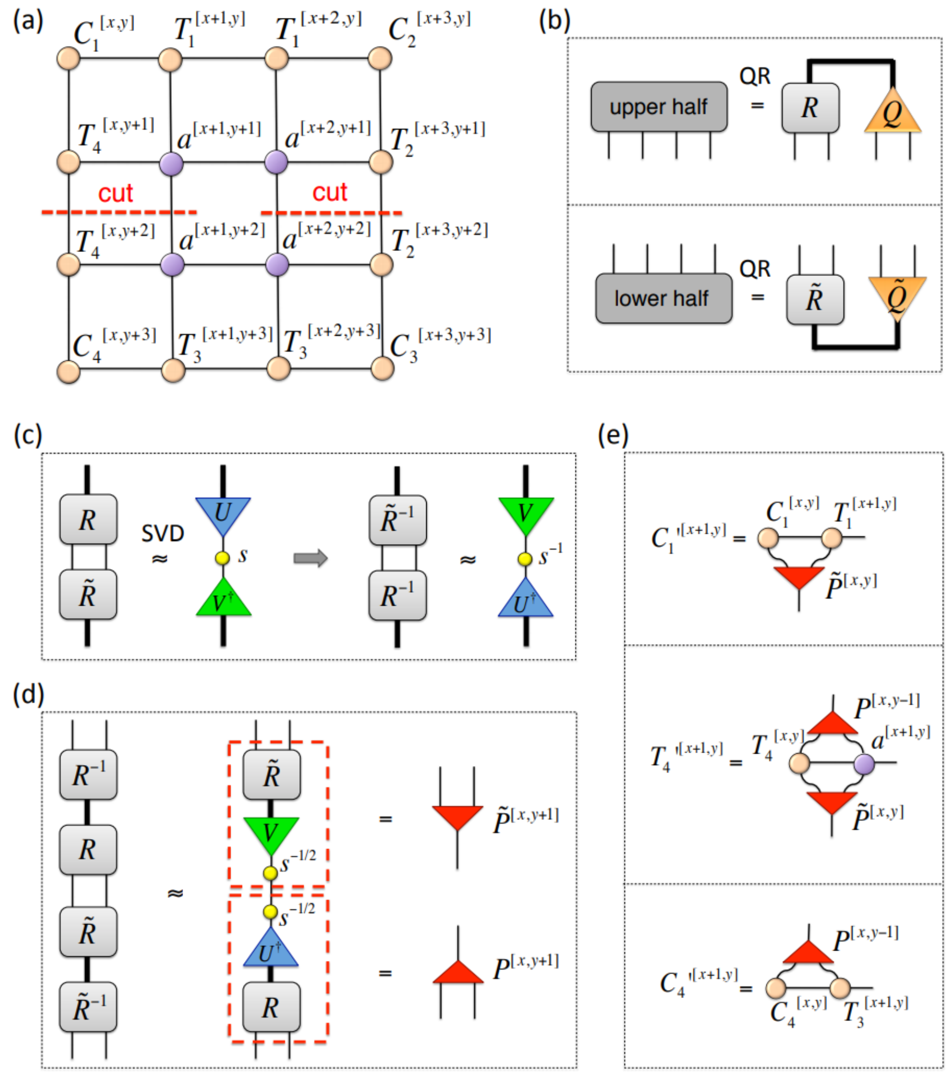
Big Cell CTMRG

- General case

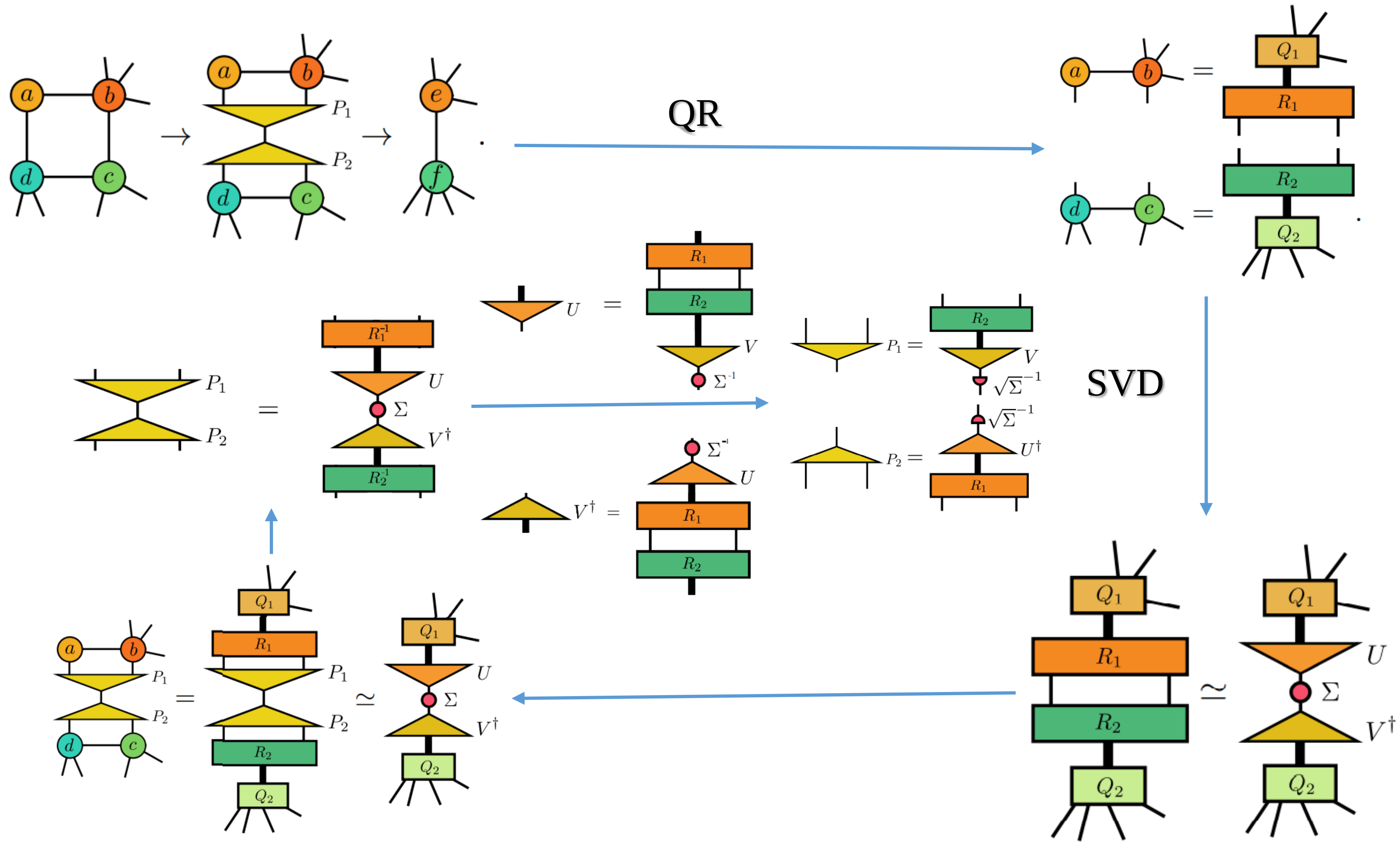


minimize the norm





Get P_1 and P_2

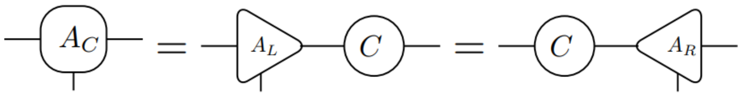


https://arxiv.org/abs/1905.02351

AD-VUMPS

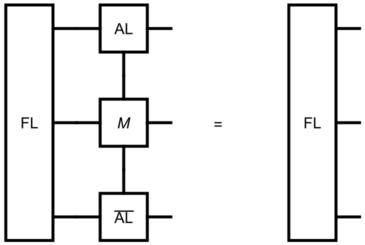
VUMPS-algorithm

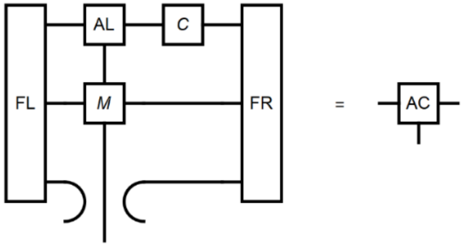
\# Initial orthonormal form MPS
M = classical2Disingmpo(β; J = 1.0, h = 0.)
AL,C = leftorth (A)
AR = rightorth (AL,C)



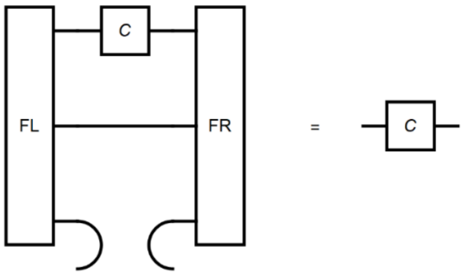
while error(AC,FL,FR,M) > tol_error

\# Get environment
FL = leftenv(AL, M)
FR = rightenv(AR, M)

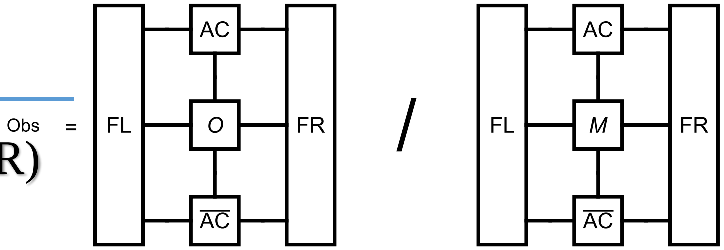




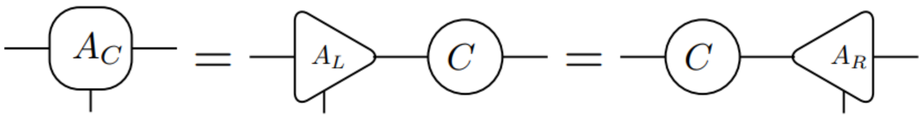
\# Get orthonormal form
AC, C = vumpsstep(AL, C, AR, M, FL, FR)



\# calculate observable
Obs = obser(O, AC, M, FL, FR)



QR to avoid inverse



$$A_C = Q_{A_C} \cdot R_{A_C} \qquad Q^T = Q^{-1}$$
$$C = Q_C \cdot R_C$$

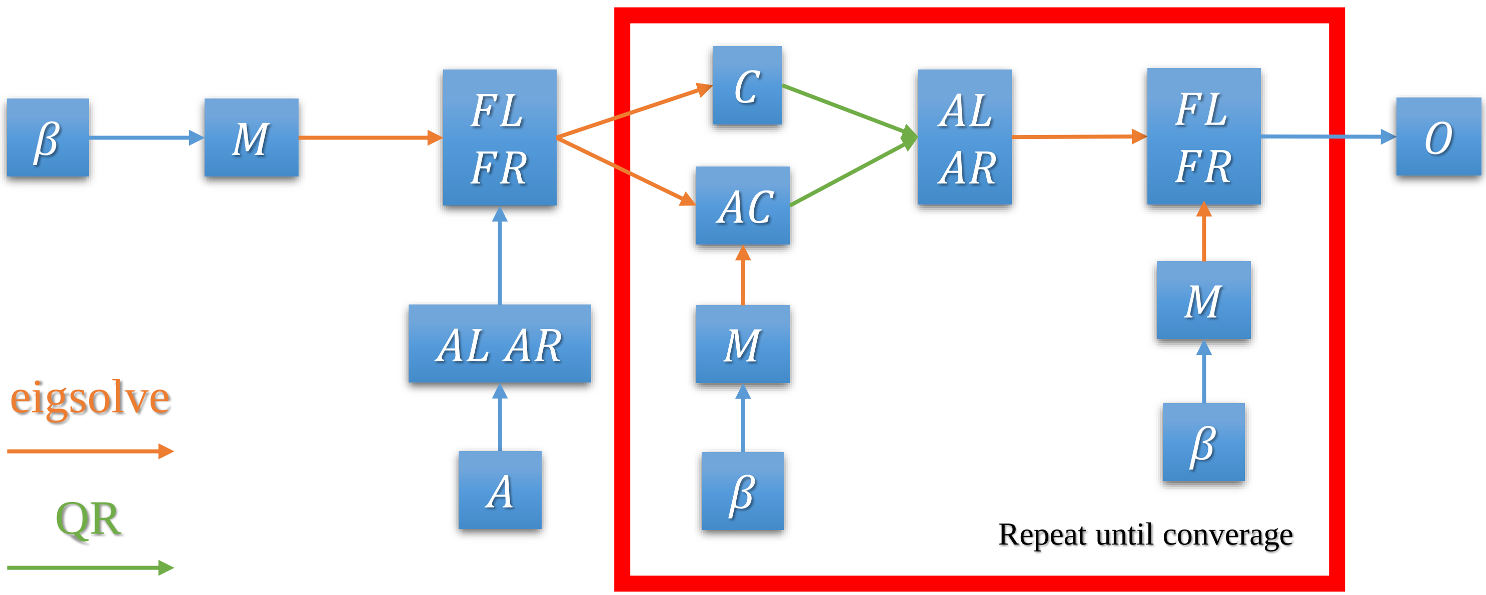
$$\begin{aligned} A_L &= A_C \cdot C^{-1} \\ &= Q_{A_C} \cdot R_{A_C} \cdot R_C^{-1} \cdot Q_C^{-1} \\ &= Q_{A_C} \cdot R_{A_C} \cdot R_C^{-1} \cdot Q_C^{T} \\ &\approx Q_{A_C} \cdot Q_C^T \end{aligned}$$

Error = R_{A_C}- R_C

$$A_L^T \cdot A_L = Q_C \cdot Q_{A_C}^T \cdot Q_{A_C} \cdot Q_C^T = 1$$

inverse and orthogonality at the same time

Computation Graphs



Adjoint of eigsolve

- $\boldsymbol{l}^T A = \lambda \boldsymbol{l}^T, \quad A\boldsymbol{r} = \lambda \boldsymbol{r}, \quad \boldsymbol{l}^T \boldsymbol{r} = 1,$

$$(A - \lambda I)\boldsymbol{\xi}_l = (1 - \boldsymbol{r}\boldsymbol{l}^T)\bar{\boldsymbol{l}}, \quad \boldsymbol{l}^T \boldsymbol{\xi}_l = 0$$
$$(A^T - \lambda I)\boldsymbol{\xi}_r = (1 - \boldsymbol{l}\boldsymbol{r}^T)\bar{\boldsymbol{r}}, \quad \boldsymbol{r}^T \boldsymbol{\xi}_r = 0$$

$$\longrightarrow \bar{A} = \bar{\lambda}\boldsymbol{l}\boldsymbol{r}^T - \boldsymbol{l}\boldsymbol{\xi}_l^T - \boldsymbol{\xi}_r \boldsymbol{r}^T$$

gauge invariant

$$\boldsymbol{l}^T \bar{\boldsymbol{l}} = \boldsymbol{r}^T \bar{\boldsymbol{r}} = 0$$

$$(A - \lambda I)\boldsymbol{\xi}_l = \bar{\boldsymbol{l}}, \quad \boldsymbol{l}^T \boldsymbol{\xi}_l = 0$$
$$(A^T - \lambda I)\boldsymbol{\xi}_r = \bar{\boldsymbol{r}}, \quad \boldsymbol{r}^T \boldsymbol{\xi}_r = 0$$

Only l

$$\boxed{(A - \lambda I)\boldsymbol{\xi}_l = \bar{\boldsymbol{l}}, \quad \boldsymbol{l}^T \boldsymbol{\xi}_l = 0}$$
$$(A^T - \lambda I)\boldsymbol{\xi}_r = 0, \quad \boldsymbol{r}^T \boldsymbol{\xi}_r = 0$$

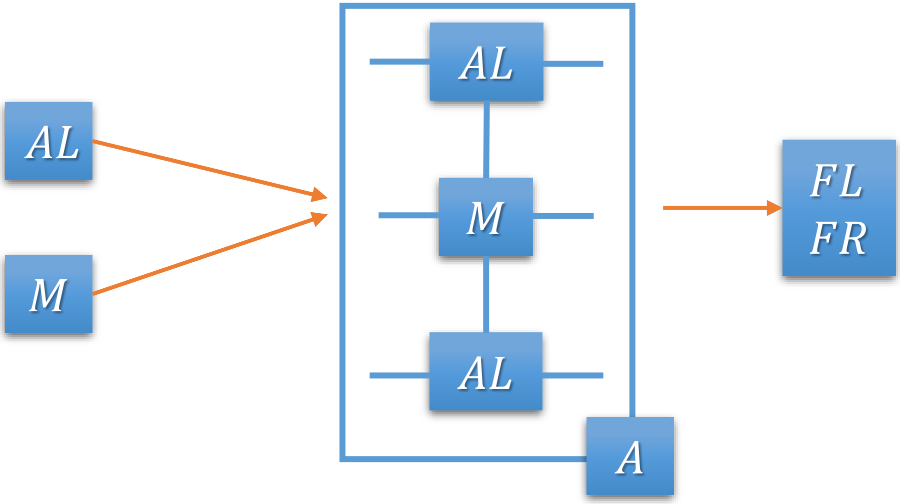
$$\boxed{\bar{A} = -\boldsymbol{l}\boldsymbol{\xi}_l^T}$$

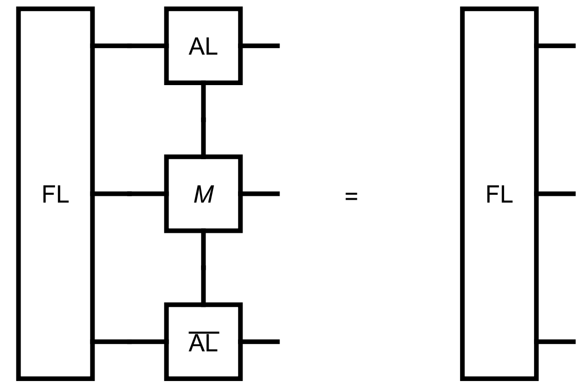
Only r

$$(A - \lambda I)\boldsymbol{\xi}_l = 0, \quad \boldsymbol{l}^T \boldsymbol{\xi}_l = 0$$
$$\boxed{(A^T - \lambda I)\boldsymbol{\xi}_r = \bar{\boldsymbol{r}}, \quad \boldsymbol{r}^T \boldsymbol{\xi}_r = 0}$$

$$\boxed{\bar{A} = -\boldsymbol{\xi}_r \boldsymbol{r}^T}$$

Adjoint of VUMPS envirment



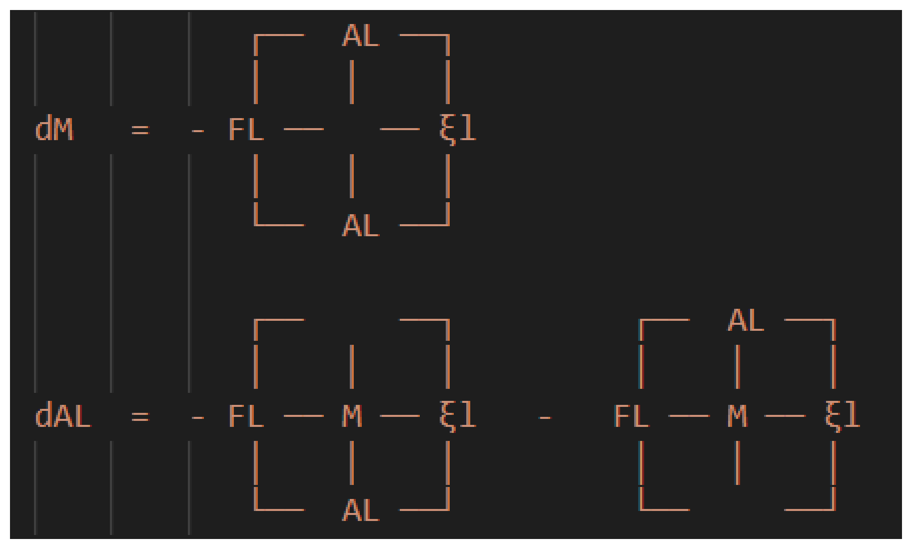


$$(A-\lambda I)\boldsymbol{\xi}_l = \bar{\boldsymbol{l}}, \quad \boldsymbol{l}^T \boldsymbol{\xi}_l = 0$$

$$\bar{A} = -\boldsymbol{l}\boldsymbol{\xi}_l^T$$

$$\overline{AL} = \bar{A} \cdot \frac{\partial A}{\partial AL}$$

$$\bar{M} = \bar{A} \cdot \frac{\partial A}{\partial M}$$



Adjoint of QR

- $$A = QR$$

$$\bar{A} = [\bar{Q} + Q\texttt{copyltu}(M)]R^{-T}$$

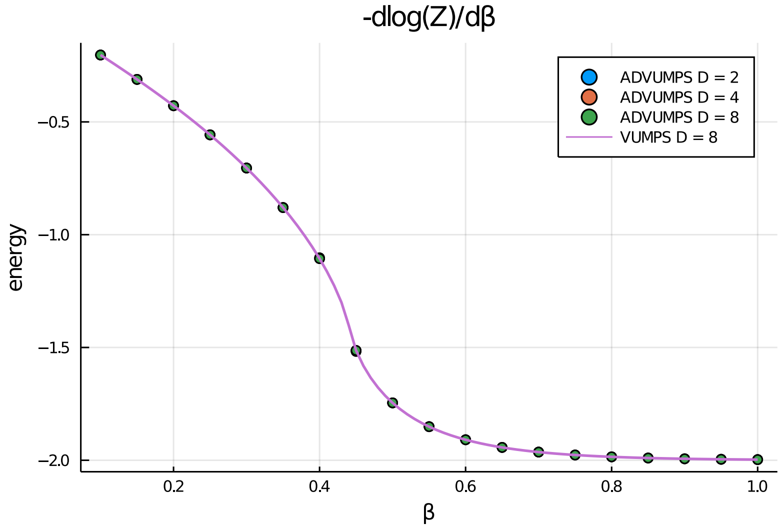
$$M = R\bar{R}^T - \bar{Q}^T Q$$

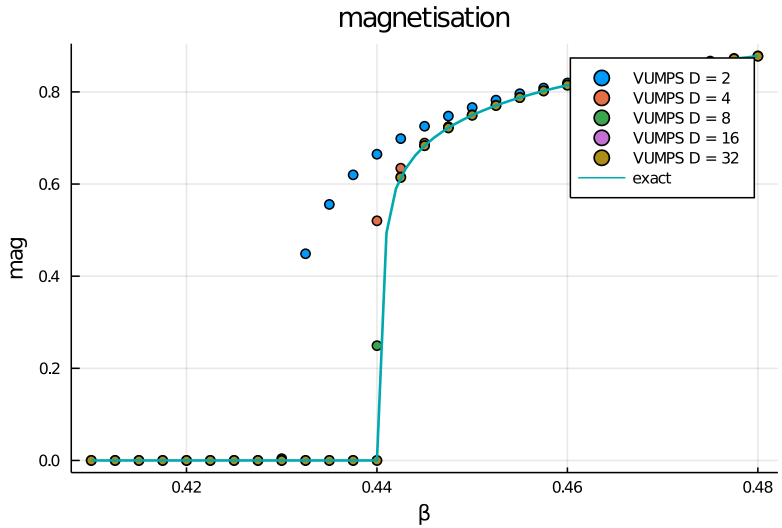
$$[\texttt{copyltu}(M)]_{ij} = M_{\max(i,j),\min(i,j)}$$

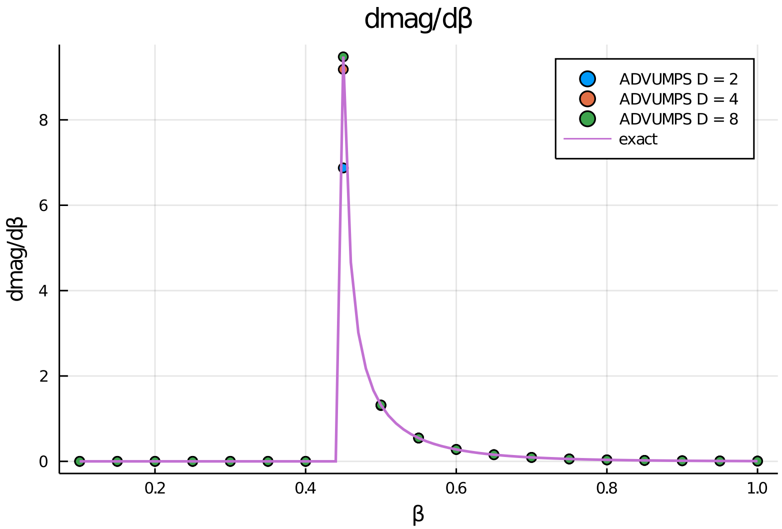
- Trick: add $\delta = 10^{-6}$ to R's diagonal element for the stability of inverse

PRX 9, 031041 (2019)

2D Classical Ising Model





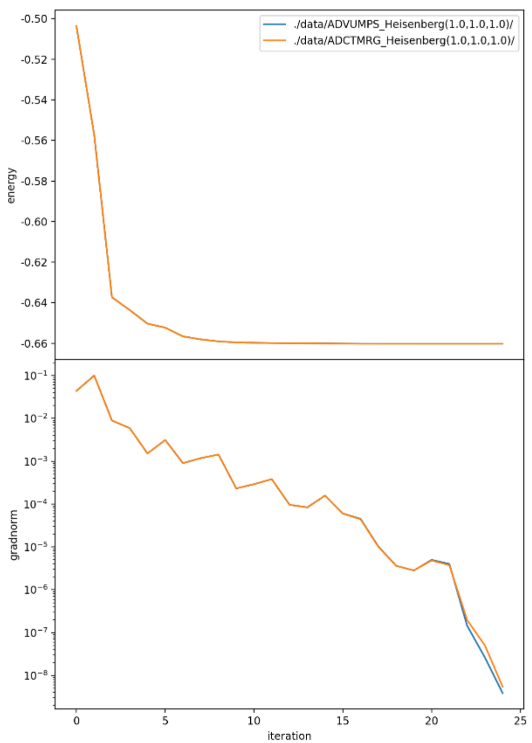


https://github.com/XingyuZhang2018/ADVUMPS.jl

Finding the Ground State of infinite 2D Heisenberg model

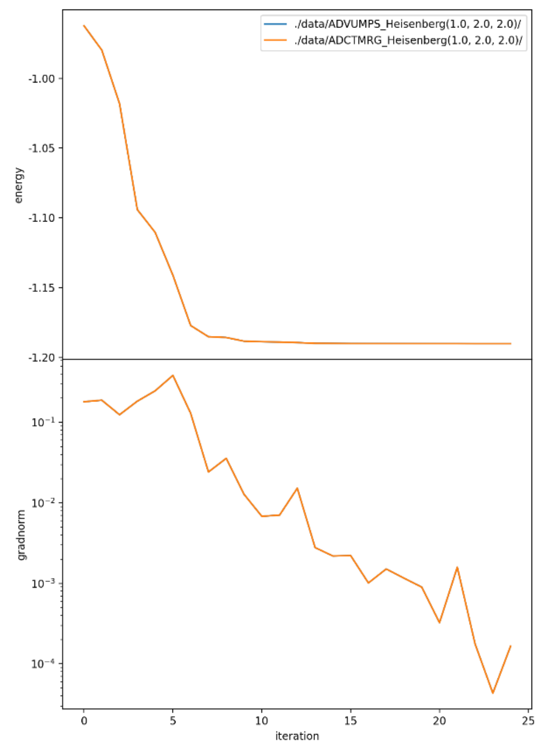
ADVUMPS vs ADCTMRG $D = 2\ \chi = 20\ \delta = 10^{-12}$

33.76s vs 15.62s



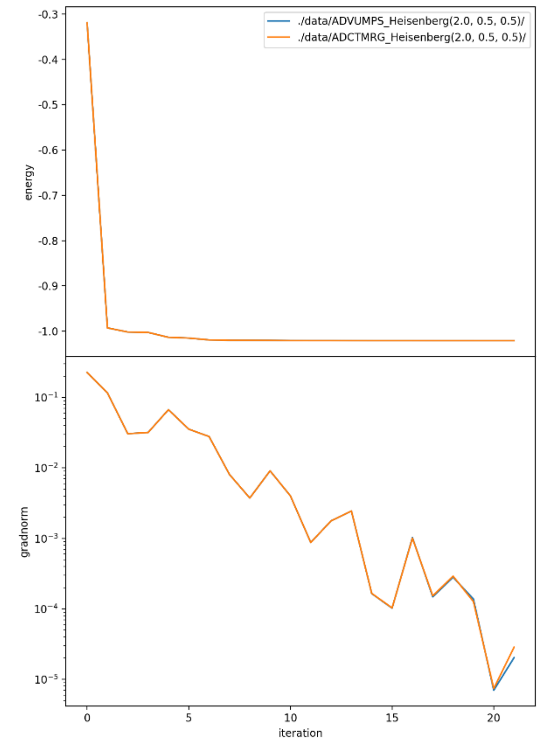
Difference ~1e-14

29.11s vs 15.58s



Difference ~1e-9

16.61s vs 4.94s

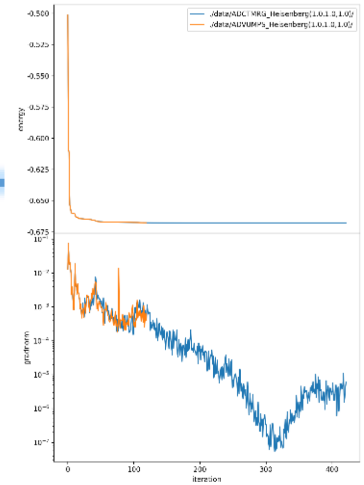


Difference ~1e-9

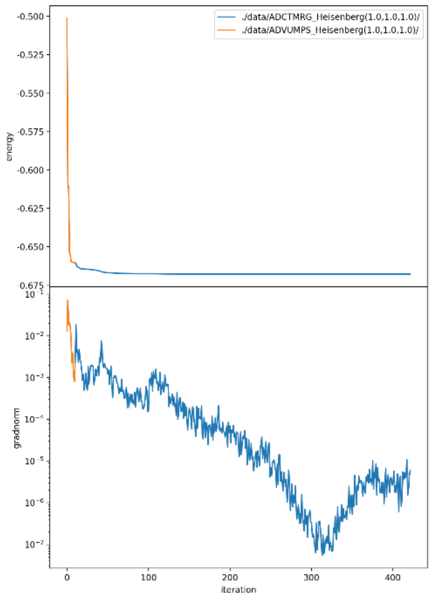
		Time			Allocations		
Tot / % measured:		8.66s / 100%			7.54GiB / 100%		
Section	ncalls	time	%tot	avg	alloc	%tot	avg
backward	61	6.22s	72.1%	102ms	6.11GiB	81.0%	103MiB
forward	61	2.41s	27.9%	39.5ms	1.43GiB	19.0%	24.0MiB

		Time			Allocations		
Tot / % measured:		30.2s / 100%			26.2GiB / 100%		
Section	ncalls	time	%tot	avg	alloc	%tot	avg
backward	62	22.8s	75.8%	368ms	20.1GiB	76.6%	332MiB
forward	62	7.30s	24.2%	118ms	6.13GiB	23.4%	101MiB

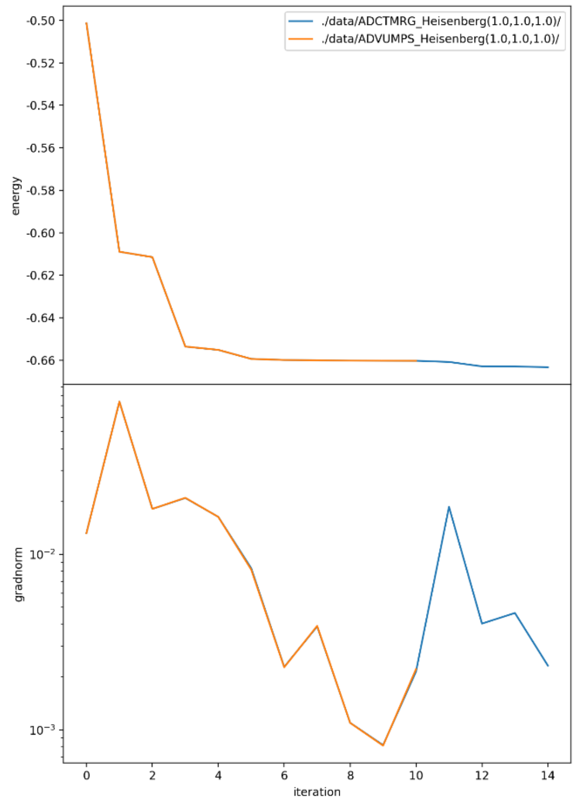
Problem of D=3



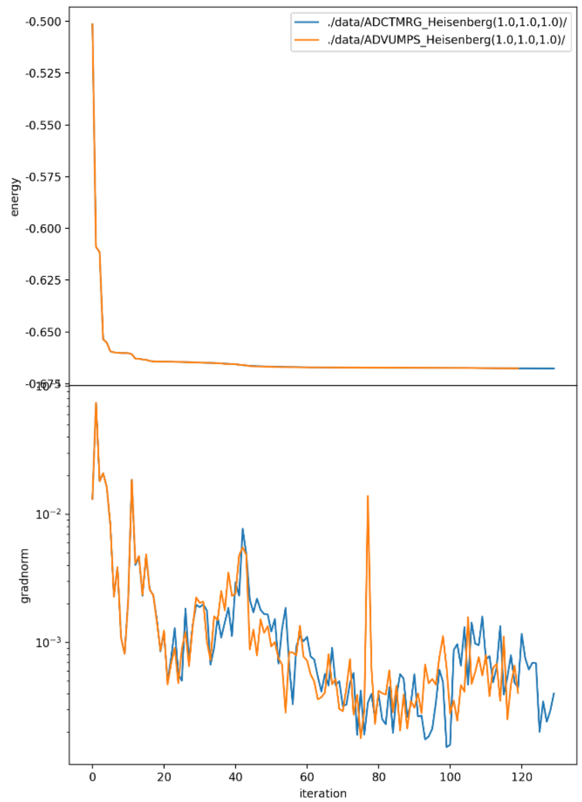
- QR isn't stable for R^{-1}



tol=1e-10



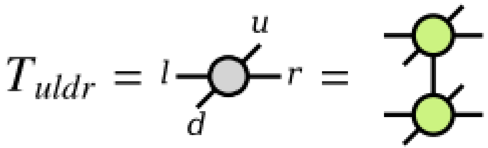
tol=1e-20



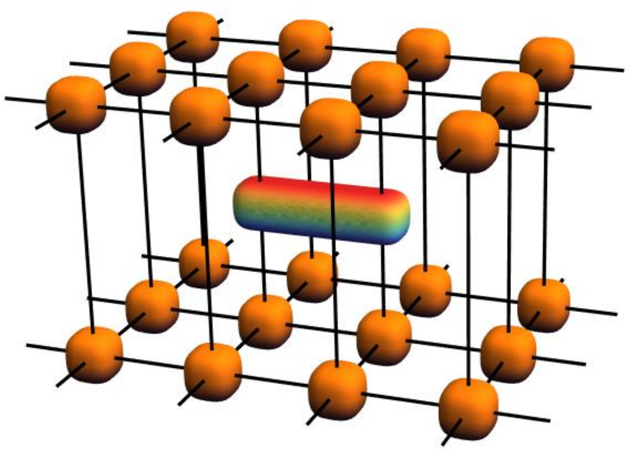
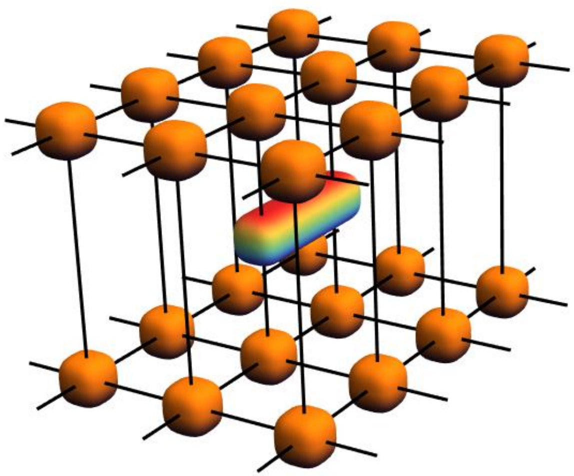
Problem of iPEPS symmetry

- When only require up and down symmetry, Heisenberg(1,1,1)

  $E_0 = -0.85 < -0.66$



- Vertical and horizontal bond energy should be optimized at the same time

AD-BC-VUMPS

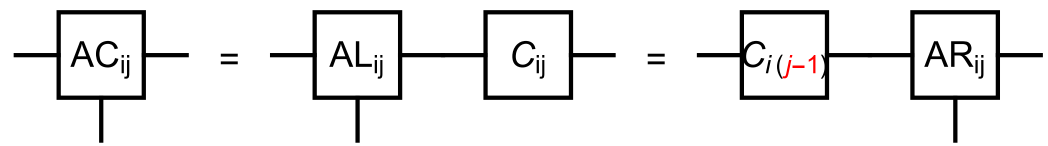
BCVUMPS-algorithm

Initial orthonormal form MPS

M_{ij} = mpo(params)

AL_{ij}, C_{ij} = leftorth (A_{ij})

AR_{ij} = rightorth (AL_{ij}, C_{ij})

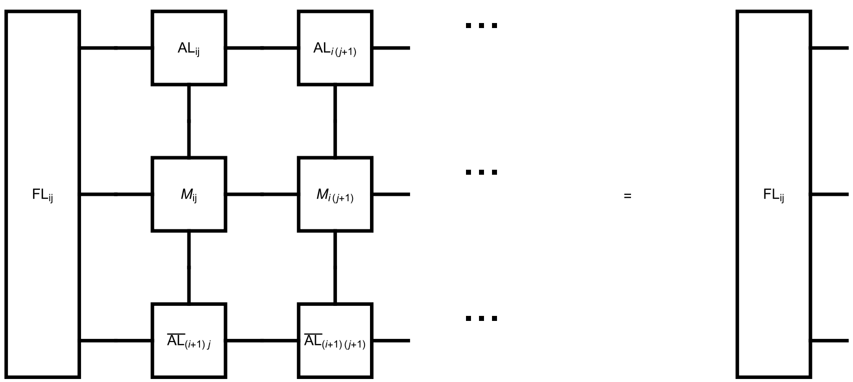


while error(AC,FL,FR,M) > tol_error

Get environment

AL[i, j] = leftenv(AL[i , :], M [i , :])

AR[i, j]= rightenv(AR[i , :], M [i , :])

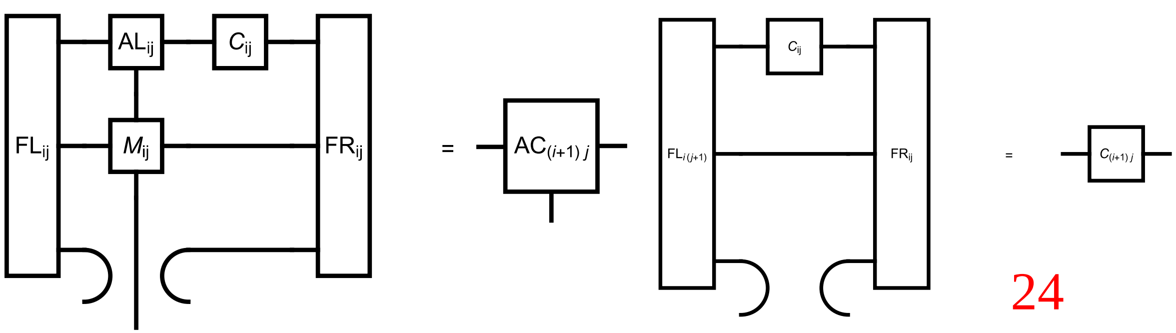


Get orthonormal form

AC[i, j], C[i, j] = vumpsstep(AL[:, j], C[:, j], AR[:, j], M[:, j], FL[:, j], FR[:, j])

calculate observable

Obs = obser(O, AC, M, FL, FR)



AD for array of array

```
# example to solve differential of array of array
# use `[]` list then reshape
A = Array{Array,2}(undef, 2, 2)
for j = 1:2,i = 1:2
    A[i,j] = rand(2,2)
end
function foo2(x)
    # B[i,j] = A[i,j].*x   # mistake
    B = reshape([A[i].*x for i=1:4],2,2)
    return sum(sum(B))
end
@test Zygote.gradient(foo2, 1)[1] ≈ num_grad(foo2, 1)
```

Be careful of index and replaced function!