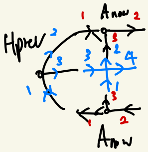
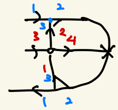
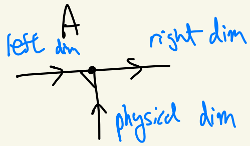
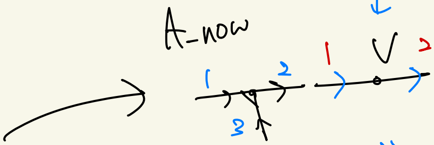
* iterative diagonalization.

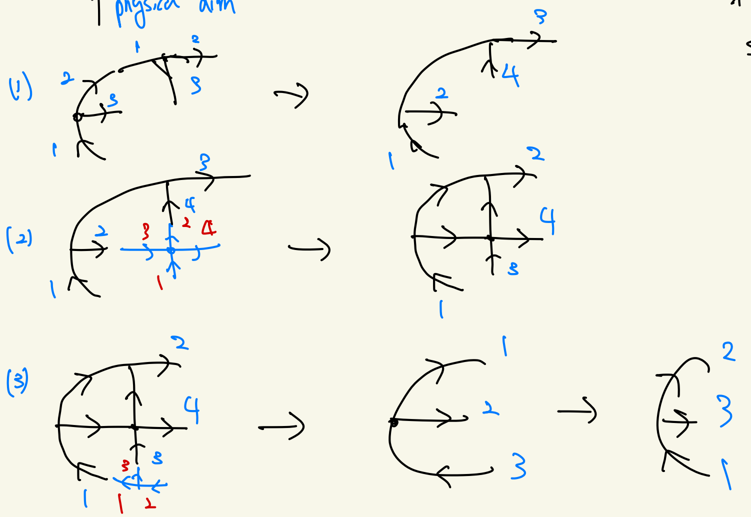




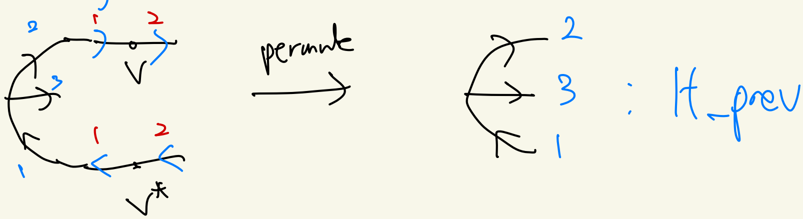




= i번째 MPS state.



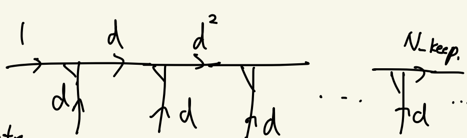
basis change



* random MPS

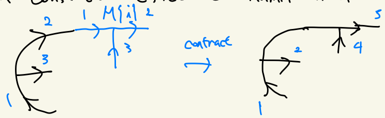
차원만 맞춰서

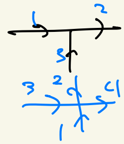
random left isometry



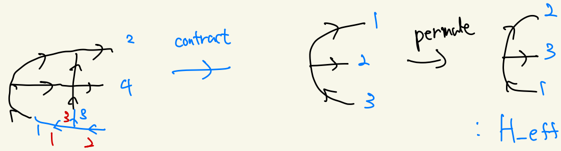
↳ M{i}에 넣기. ↳ 임의의 m×n (m≥n) 행렬 QR decom. → Q 가져옴. → reshape

* Construct effective hamiltonian



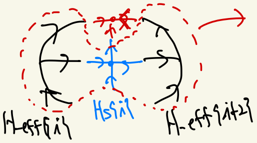


H_eff{i}



: H_eff{i+1}

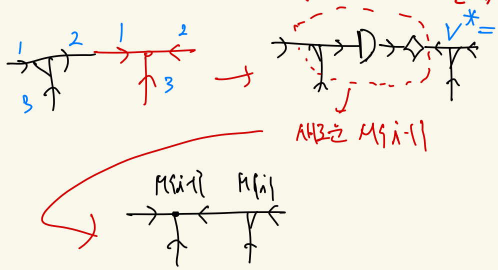
* right to left sweep (By Lanczos)



eigen vector

: M{i}

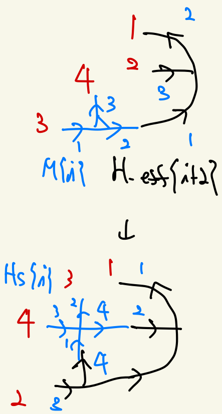
eigenvalue: E_list에 저장
(2·iteration−1 열에)



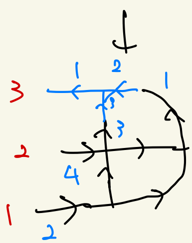
새로운 $V^* = M\{i\}$

새로운 M{i-1}

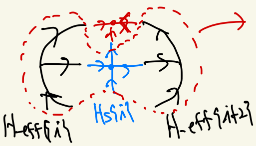
* update right effective hamiltonian

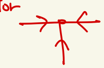


permute



* left to right sweep

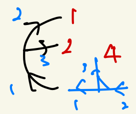


→ eigen vector  : M{i}

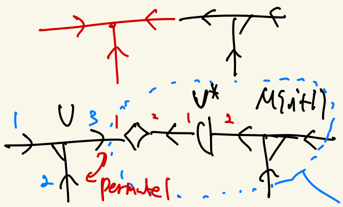
eigenvalue: E_list에 저장.

↳ 2x iteration 열 에!

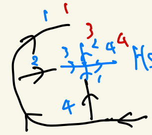
- update Heff

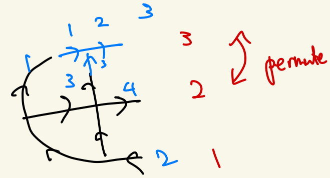


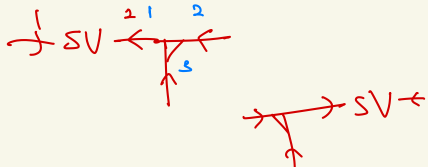
M{i}



new M{i+1}





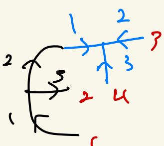


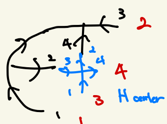
* Lanczos method

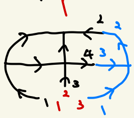
V0 저장

(Mold normalize)

$v(:,:,:,i)$







vectors flatten.

$w_i = Hv_i$

$X = \sum v_j v_j^\dagger w_i$

$w_i = w_i - X$

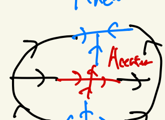
$v_{i+1} = w_i / \|w_i\|$ 로 vector 추가

$\alpha_i = v_i^\dagger w_i$

$\beta_i = \|w_i\|$

→ Tri-diagonal matrix T 생성.

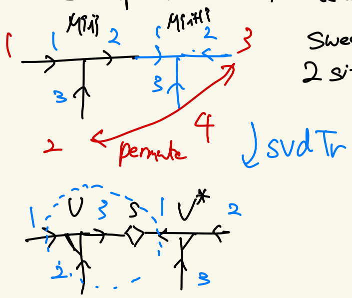
→ diagonalize → eigenvalue 최소인 eigenvector : M_{new}

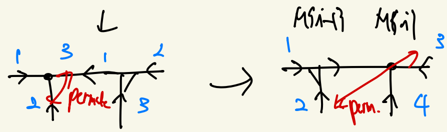


Hright ⇒ E_G 계산.

*2 site update : 1 site 와 차이점만 기록.

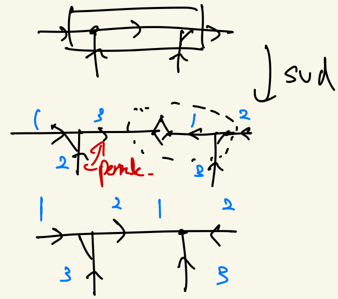
Sweep 돌기 전
2 site로 묶어주기.





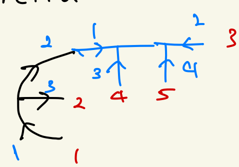
다시 묶어주기

오른쪽으로 갈 땐 반대로..



- Lanczos method

difference :



다른 것은 1 site와 동일.

(flatten 해서 T 만들고
대각화해서 최소 에너지 구하기).