Note for IsoPEPS

[1]Yuqing Rong

May 08, 2025

Abstract

1 Introduction

This is a note document for the project: IsoPEPS of Rydberg atom arays. It includes basic knowledge (tensor network, matrix computation, Lie Algebra and Lie group, quantum mechanics, quantum computation and quantum information), coding technique of Julia and related paper review.

2 Basic knowledge

2.1 Tensor network

2.2 Automatic Differentiation (AD)

Use the chain rule, seperate the gradients as a flow.

2.2.1 basis

control flow:

examble: $y = f(x_1, x_2) = \ln(x_1) + x_1 x_2 - \sin(x_2)$

- input variables: $v_{i-n}, i = 1, 2, ..., n$

$$v_{-1} = x_1$$
$$v_0 = x_2$$

- intermediate variables: $v_i, i = 1, 2, ..., l$

$$v_1 = \ln(x_1) = \ln(v_{-1})$$
$$v_2 = x_1 x_2 = v_{-1} v_0$$
$$v_3 = \sin(x_2) = \sin(v_0)$$
$$v_4 = \ln(x_1) + x_1 x_2 = v_1 + v_2$$
$$v_5 = v_4 - v_3$$

- output variables: $y_{m-i} = v_{l-i}, i = m-1, ..., 0$

$$y = v_5$$

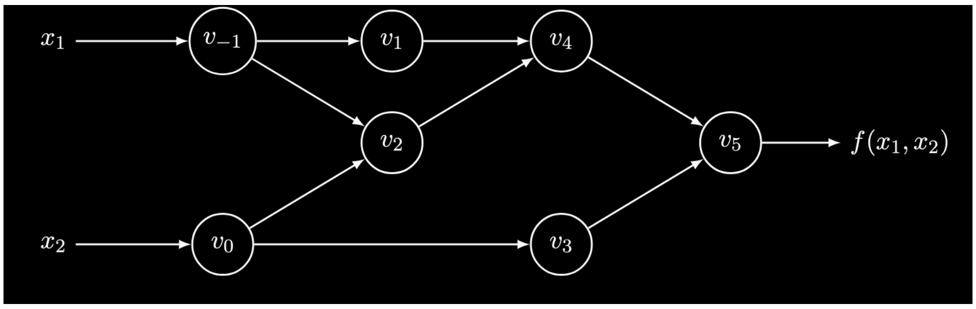
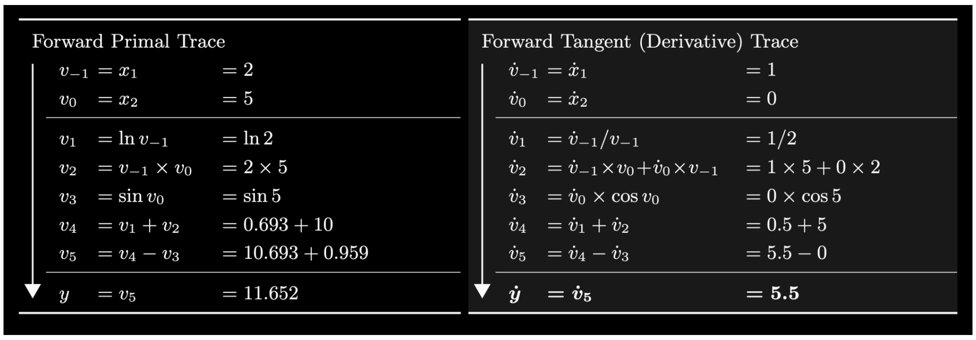


Figure 1: Computational graph

2.2.2 forward mode

$\dot{y_j} = \frac{\partial y_j}{\partial x_i}|_{x=a}, j = 1, ..., m; i = 1, ..., n$

First, we should set at which point we want to calculate the gradients. Then we calcaulate the gradient to $x_1, x_2, ..., x_n$ in order. When we calculate the gradient to x_i, we set $\dot{x_i} = 1$, the others $\dot{x_j} = 0, j \neq i$(they are independent variables).

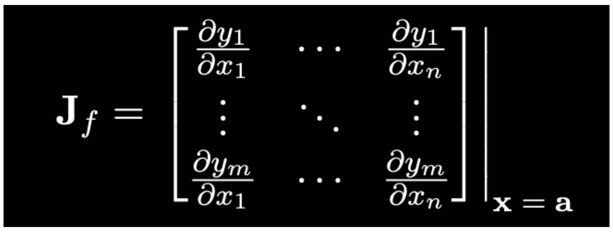


Forward Primal Trace		
v_{-1}	$= x_1$	$= 2$
v_0	$= x_2$	$= 5$
v_1	$= \ln v_{-1}$	$= \ln 2$
v_2	$= v_{-1} \times v_0$	$= 2 \times 5$
v_3	$= \sin v_0$	$= \sin 5$
v_4	$= v_1 + v_2$	$= 0.693 + 10$
v_5	$= v_4 - v_3$	$= 10.693 + 0.959$
y	$= v_5$	$= 11.652$

Forward Tangent (Derivative) Trace		
$\dot{v}_{-1}$	$= \dot{x}_1$	$= 1$
$\dot{v}_0$	$= \dot{x}_2$	$= 0$
$\dot{v}_1$	$= \dot{v}_{-1}/v_{-1}$	$= 1/2$
$\dot{v}_2$	$= \dot{v}_{-1} \times v_0 + \dot{v}_0 \times v_{-1}$	$= 1 \times 5 + 0 \times 2$
$\dot{v}_3$	$= \dot{v}_0 \times \cos v_0$	$= 0 \times \cos 5$
$\dot{v}_4$	$= \dot{v}_1 + \dot{v}_2$	$= 0.5 + 5$
$\dot{v}_5$	$= \dot{v}_4 - \dot{v}_3$	$= 5.5 - 0$
$\dot{y}$	$= \dot{v}_5$	$= \mathbf{5.5}$

Figure 2: forward mode

After n evaluations, all gradients to x_i has been calculate. We put them into Jabian matrix.



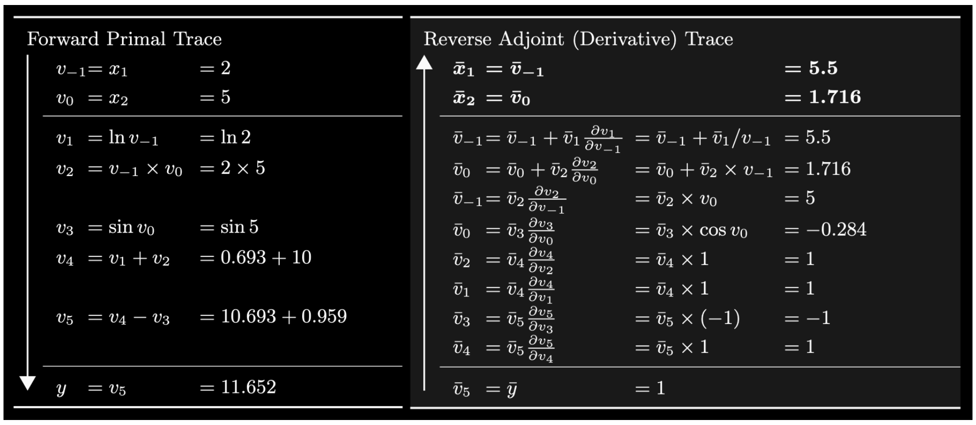
$$\mathbf{J}_f = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix} \Bigg|_{\mathbf{x} = \mathbf{a}}$$

Figure 3: Jacobian matrix

2.2.3 reverse mode

Suppose the ajiont: $\overline{v}_i = \frac{\partial y}{\partial v_i}$

As forward mode, we should set the point we want to calculate, and calculate $y_1, y_2, .., y_m$ in order. We set $\overline{V_5} = \overline{y} = 1$.



Forward Primal Trace

v_{-1}	$= x_1$	$= 2$
v_0	$= x_2$	$= 5$
v_1	$= \ln v_{-1}$	$= \ln 2$
v_2	$= v_{-1} \times v_0$	$= 2 \times 5$
v_3	$= \sin v_0$	$= \sin 5$
v_4	$= v_1 + v_2$	$= 0.693 + 10$
v_5	$= v_4 - v_3$	$= 10.693 + 0.959$
y	$= v_5$	$= 11.652$

Reverse Adjoint (Derivative) Trace

$\bar{x}_1$	$= \bar{v}_{-1}$		$= 5.5$
$\bar{x}_2$	$= \bar{v}_0$		$= 1.716$
$\bar{v}_{-1}$	$= \bar{v}_{-1} + \bar{v}_1 \frac{\partial v_1}{\partial v_{-1}}$	$= \bar{v}_{-1} + \bar{v}_1 / v_{-1}$	$= 5.5$
$\bar{v}_0$	$= \bar{v}_0 + \bar{v}_2 \frac{\partial v_2}{\partial v_0}$	$= \bar{v}_0 + \bar{v}_2 \times v_{-1}$	$= 1.716$
$\bar{v}_{-1}$	$= \bar{v}_2 \frac{\partial v_2}{\partial v_{-1}}$	$= \bar{v}_2 \times v_0$	$= 5$
$\bar{v}_0$	$= \bar{v}_3 \frac{\partial v_3}{\partial v_0}$	$= \bar{v}_3 \times \cos v_0$	$= -0.284$
$\bar{v}_2$	$= \bar{v}_4 \frac{\partial v_4}{\partial v_2}$	$= \bar{v}_4 \times 1$	$= 1$
$\bar{v}_1$	$= \bar{v}_4 \frac{\partial v_4}{\partial v_1}$	$= \bar{v}_4 \times 1$	$= 1$
$\bar{v}_3$	$= \bar{v}_5 \frac{\partial v_5}{\partial v_3}$	$= \bar{v}_5 \times (-1)$	$= -1$
$\bar{v}_4$	$= \bar{v}_5 \frac{\partial v_5}{\partial v_4}$	$= \bar{v}_5 \times 1$	$= 1$
$\bar{v}_5$	$= \bar{y}$	$= 1$	

Figure 4: Jacobian matrix

As forward mode, reveerse mode also has Jacobian after m evaluations.

$$\mathbf{J}_f^\top \mathbf{r} = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_1}{\partial x_n} & \cdots & \frac{\partial y_m}{\partial x_n} \end{bmatrix} \begin{bmatrix} r_1 \\ \vdots \\ r_m \end{bmatrix}$$

Figure 5: Jacobian matrix

2.2.4 conclusion

$f : \mathbb{R}^n \to \mathbb{R}^m$. If $n > m$, the reverse mode is more efficient, elseif $m > n$, the forward mode is more efficient.

2.2.5 code

2.3 Matrix computation

2.3.1 Matrix/Vector operations

(i) pointwise multiplication: $C = A.*B \Rightarrow c_{ij} = a_{ij} * b_{ij}$

pointwise division: $C = A./B \Rightarrow c_{ij} = a_{ij}/b_{ij}$

(ii) saxpy: (scalar*vector) $y = \alpha * x + y \Rightarrow y_i = \alpha * x_i + y_i$, update vector y as $\alpha * x + y$. "saxpy" is used in LAPACK.

(iii) gaxpy: (matrix*vector, matrix is sum of scalars) $y = y + Ax \Rightarrow y_i = y_i + \sum_j a_{ij} x_j$, $y \in R^m$, $x \in R^n$, $A \in R^{m*n}$.

```julia
#Row-oriented gaxpy
for i in 1:m
  for j in 1:n
    y[i] = y[i] + A[i, j] * x[j]
  end
end

#or
for i in 1:m
  y[i] = y[i] + A[i, :] * x
end
```

```julia
#Column-oriented gaxpy
for j in 1:n
  for i in 1:m
    y[i] = y[i] + A[i, j] * x[j]
  end
end

#or
for j in 1:n
  y = y + A[:, j] * x[j]
end
```

(iv) outer product: $A = A + xy^T \Rightarrow a_{ij} = a_{ij} + x_i y_j$, $x \in R^m$, $y \in R^n$, $A \in R^{m*n}$.

```julia
for i in 1:m
  A[i, :] = A[i, :] + x[i] * y^T
end
#or
for j in 1:n
  A[:, j] = A[:, j] + x * y[j]
end
```

(v) matrix-matrix multiplication: $C = C + AB$. $A \in R^{m*r}$, $B \in R^{r*n}$, $C \in R^{m*n}$.

- saxpy formulation

  each row of C: $c_i = c_i + a_i^T B$

  $$A = \begin{pmatrix} a_1^T \\ a_2^T \\ \cdots \\ a_m^T \end{pmatrix}, \quad C = \begin{pmatrix} c_1^T \\ c_2^T \\ \cdots \\ c_m^T \end{pmatrix}$$

```julia
for i in 1:m
  C[i, :] = C[i, :] + A[i, :] * B
end
```

each column of C: $c_j = c_j + Ab_j$

$$B = \begin{pmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{pmatrix}, \; C = [c_1|\; c_2| \;\dots\; |c_n]$$

```julia
for j in 1:n
  C[:, j] = C[:, j] + A * B[:, j]
end
```

- outer product formulation

$$A = [a_1|\; a_2|\; \dots|\; a_r]\;, \quad B = \begin{pmatrix} b_1^T \\ b_2^T \\ \dots \\ b_r^T \end{pmatrix}$$

```julia
for i in 1:r
  C = C + A[:,k] * B[k,:]
end
```

2.3.2 band matrices: can improve efficiency of matrix-matrix multiplication.

$A \in R^{m*n}$ has lower bandwidth p if $a_{ij} = 0$ for $i > j + p$ and upper bandwidth q if $a_{ij} = 0$ for $j > i + q$.

band matrices includes: diagonal, tridiagonal, bidiagonal, Hessenberg, and tridiagonal.

(i) band storage: $a_{ij} = A.band(i - j + q + 1, j)$

eg. : $y = y + Ax$, we can replace A with $A.band$ to improve efficiency.

```julia
for j in 1:n
  a1 = max(1, j-q); a2 = min(n, j+p);
  b1 = max(1, q+2-j); b2 = b1 + a2-a1;
  y(a1:a2) = y(a1:a2) + A.band(b1:b2,j) * x(j)
end
```

(ii) vec storage of symmetric matrices: $a_{ij} = A.vec\Big(\big(n - \frac{j}{2}\big)(j-1) + i\Big), 1 \le j \le i \le n$,

store the lower triangular part of the matrix in vector.

eg.: $A = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{pmatrix} \Leftrightarrow A.\mathrm{vec} = [1, 2, 3, 4, 5, 6]$

2.3.3 permutation matrices: reordered rows I_n

eg.: $P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$, it can be rewrited as $v = [2, 4, 3, 1]$, which tell us the position of "1" in each row. $P = I_n[v, :]$

(i) get submatrix using v

eg.: $v = 1 : 2 : n \Rightarrow v = [1357]$

eg,: $A[i,j] = A[1:2:n, 2:2:n]$, the elements would be reordered.

(ii) 3 examples

- exchange permutation:

  $$v = [n:-1:1] \Rightarrow P = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

- downshift permutation:

  $$v = [4,1,2,3] \Rightarrow P = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

- upshift permutation:

  $$v = [2,3,4,1] \Rightarrow P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

- mod-p perfect shuffle:

  $P_{p,r} = I_n([(1:r:n)(2:r:n)...(r:r:n)],:)$

2.3.4 Kronecker product

(i) properties:

- $(B \otimes C)^T = B^T \otimes C^T$
- $(B \otimes C)(D \otimes F) = BD \otimes CF$
- $(B \otimes C)^{-1} = B^{-1} \otimes C^{-1}$
- $B \otimes (C \otimes D) = (B \otimes C) \otimes D$
- perfect shufffle permutation: $P(B \otimes C)Q^T = C \otimes B$

(ii) kronecker peoduct $\Rightarrow$ matrix product (improve efficiency)

$Y = (B \otimes C)X^T = \left(CXB^T\right)^T = BX^TC^T$

2.3.5 Hamiltonian and Symplectic matrix

(i) Hamiltonian matrix:

$M = \begin{pmatrix} A & G \\ F & -A^T \end{pmatrix}$, $F^T = G$ or

$JMJ^T = -M^T$, $J = \begin{pmatrix} 0 & I_n \\ -I_n & 0 \end{pmatrix}$

(ii) symplectic matrix:

$S^TJS = J$, $J = \begin{pmatrix} 0 & I_n \\ -I_n & 0 \end{pmatrix}$

2.3.6 Strassen matrix multiplication

2.3.7 Fast Fourier Transform (FFT)

For a polynomial $P(x) = p_0 + p_1x + p_2x^2 + ... + p_nx^{n-1}$, we have 2 representations:

(i) coefficient representation: $[p_0, p_1, p_2, ..., p_{n-1}]$

(ii) point-value representation: $[(x_0, P(x_0)), (x_1, P(x_1)), (x_2, P(x_2)), ..., (x_{n-1}, P(x_{n-1})]$

If we use value representation,

$$\begin{pmatrix} P(x_0) \\ P(x_1) \\ P(x_2) \\ \vdots \\ P(x_{n-1}) \end{pmatrix} = \begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^{n-1} \\ 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \\ 1 & x_{n-1} & x_{n-1}^2 & \dots & x_{n-1}^{n-1} \end{pmatrix} \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ \vdots \\ p_{n-1} \end{pmatrix}$$

needs n points?

$\downarrow$

if $P(x) = x^2 \Rightarrow P(-x) = P(x), n$ points $\Rightarrow 2n$ points

if $P(x) = x^3 \Rightarrow P(-x) = -P(x), n$ points $\Rightarrow 2n$ points

$P(x)$ can be cut into even and odd parts:

$P(x) = p_0 + p_1 x + p_2 x^2 + \dots + p_n x^{n-1} = p_e(x^2) + x p_o(x^2)$

n points $[x_1, x_2, \dots, x_n]$

let $x_2 = -x_1, x_4 = -x_3, \dots, x_n = -x_{n-1}$

$\downarrow$

$\frac{n}{2}$ points $\left[x_1^2, x_2^2, \dots, x_{\frac{n}{2}}^2\right]$

find the next pairs, let $x_2^2 = -x_1^2, x_4^2 = -x_3^2, \dots, x_{\frac{n}{2}}^2 = -x_{\frac{n}{2}-1}^2$

$\downarrow$

$\frac{n}{4}$ points $\left[P(x_1^2), P(x_2^2), \dots, P\left(x_{\frac{n}{4}}^2\right)\right]$

$\vdots$

finally, we need only 1 point.

So, how to find these points?

N roots of unity (on a unit circle):

$z^n = 1 \Rightarrow z = e^{2\pi \frac{i}{n}}$

then we set the points as: $\omega^0, \omega^1, \omega^2, \dots, \omega^n$, we can easily find their pairs and cut half of them.

2.3.8 vector norms

$p - norms : \|x\|_p = (|x_1|^p + |x_2|^p + \dots + |x_n|^p)^{\frac{1}{p}}, p \geq 1$

(i) Holder inequality: $|x^T y| \leq \|x\|_p \, \|y\|_q, \frac{1}{p} + \frac{1}{q} = 1$

Cauchy-Schwarz inequality: $|x^T y| \leq \|x\|_2 \, \|y\|_2$

(ii) convergence: if $\lim_{k\to\infty} \|x^k - x\|_p \to 0, p \geq 1$, then x^k converges to x.

2.3.9 matrix norms

(i) $p - norms$: $\|A\|_p = \max_{x\neq 0} \frac{\|Ax\|_p}{\|x\|_p}$
- $\|A\|_2 = \sqrt{\lambda_{\max}(A^T A)}$, equal to the largest singular value of A.
- $\|A\|_1 = \max_j \sum_i |a_{ij}|$, largest sum over column.

- $\|A\|_\infty = \max_i \sum_j |a_{ij}|$, largest sum over row.

(ii) Frobenius norms: $\|A\|_F = \sqrt{\sum_i \sum_j |a_{ij|^2}} = \sqrt{\text{tr}(A^H A)}$

2.3.10 SVD

Intuitive understanding: A matrix is an operation. An arbitrary object transformation can be seen as rotation, dimension adding(erasing), stretching, rotation. $A = U\Sigma V^T$ V^T for rotation, Σ for dimension adding(erasing)and stretching, U for rotation.

2.3.11 LU decomposition

$A = LU$, L is a lower triangular matrix, U is a upper triangular matrix.

eg.: $A = \begin{pmatrix} 3 & 5 \\ 6 & 7 \end{pmatrix}$

$A = IA = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 3 & 5 \\ 6 & 7 \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}\begin{pmatrix} 3 & 5 \\ 0 & -3 \end{pmatrix}$

$r_2 - 2 * r_1$, so that it should add $2 * r_1$ to become A.

2.4 Lie Algebra and Lie group

2.4.1 group introduction:

(i) **definition**: a set of elements with a binary operation that satisfies closure, associativity, identity, and invertibility.
- **closure**: for any $a, b \in G$, $a * b \in G$
- **associativity**: for any $a, b, c \in G$, $(a * b) * c = a * (b * c)$
- **identity**: there exists an unique element $e \in G$, such that for any $a \in G$, $e * a = a * e = a$
- **invertibility**: for any $a \in G$, there exists an unique element $b \in G$, such that $a * b = b * a = e$

(ii) **example**:
- **abelian group**: a group that satisfies commutativity. $a * b = b * a$
- **non-abelian group**: a group that does not satisfy commutativity. $a * b \neq b * a$
- **general linear group**: $GL(n, \mathbb{C}) = \{n \times n \text{ invertible matrices} \mid \mathbb{C}\}$, non-abelian group except n=1.
- **symmetric group**: $S_n = \{n! \text{ permutations} \mid n\}$.

  eg: all permutations of set {1,2,3,4}, so there are 4!=24 elements.
- **integers mod n**: $Z_n = \{0, 1, 2, ..., n-1\}$ 余数

(iii) **subgroup**: a subset of a group that is also a group. $H \subset G$

properties:
- same identity as the G.
- same product operation as the G.
- inverse: $h^{-1} \in H$
- closure: $h_1 * h_2 \in H$

center of a group: $Z(G) = \{g \in G \mid g * h = h * g, \forall h \in G\}$ (is subgroup of G)

direct product of groups: $G \times H = \{(g, h) \mid g \in G, h \in H\}$, operation is $(g_1, h_1) * (g_2, h_2) = (g_1 * g_2, h_1 * h_2)$

(iv) **Homomorphism**: G and H are groups, $\varphi : G \to H$ is a homomorphism if $\varphi(g * h) = \varphi(g) * \varphi(h)$

kernel: $\ker(\varphi) = \{g \in G \mid \varphi(g) = e_H\}$

Isomorphism: G and H are groups, $\varphi : G \to H$ is an isomorphism if φ is a homomorphism and one-to-one and on-to.

(v) **Quotient group**: $\frac{G}{H} = \{gH \mid g \in G\}$

properties: $(g_1 H)(g_2 H) = (g_1 g_2)H$

eg.: $H = 3Z$ in Z, $\frac{G}{H} = \{0 + 3Z, 1 + 3Z, 2 + 3Z ...\}$

$(0 + 3Z) + (1 + 3Z) = (0 + 1) + 3Z$

(vi) **normal matrix**: $AA^* = A^*A$

Normal matrix such as Hermitian($A^* = A$), skew-Hermitian($A^* = -A$), unitary($A^* = A^{-1}$) can be diagonalized. And it ia possible an orhtogonal basis for $\mathbb{C}^n$ consisting of eigenvectors for A.

2.4.2 general concepts

(i) Lie group:

a Lie group is a smooth manifold that is also a group.

- S^1: a circle, geometric object
- $GL(n, \mathbb{C})$: general linear group, $\{n \times n$ invertible matrices $\mid \mathbb{C}\}$

(ii) Lie Algebra:

algebra with "Lie bracket" $[,]$, $[X, Y] = XY - YX$

- $gl(n; \mathbb{C})$: general linear algebra. $\{n \times n$ matrices $\mid \mathbb{C}\}$

2.5 Quantum computation and quantum information

(i) measurement basis:

It comes from the Bloch sphere. We know that the Z operator has 2 eigenvectors: $|0\rangle$ and $|1\rangle$, X operator has 2 eigenvectors: $|+\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, Y operator has 2 eigenvectors: $|i\rangle = \frac{1}{\sqrt{2}}(|0\rangle + i|1\rangle)$ and $|-i\rangle = \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$. And they all can be expressed as a point on the Bloch sphere. Z,X,Y operators has coresponding x-axis, y-axis, z-axis on the Bloch sphere.

Therefore, for a given state $|\psi\rangle$, we can break it into one of the basis and do measurement (get the possibility of the result). For example, $|\psi\rangle = \frac{1}{\sqrt{2}}(|+\rangle + |-\rangle)$, the result of getting $|+\rangle$ is $\frac{1}{2}$, and the result of getting $|-\rangle$ is also $\frac{1}{2}$.

2.5.1 transfer matrix

$\psi_{n+1} = T\psi_n$, T is transfer matrix.

2.6 Topology

3 Coding technique

3.1 Module

(i) differences betweeb **using** and **import**:

using: can use the fields directly which were exported by module.

import: should be used with module name, Module.cat

(ii)

```julia
using .NiceStuff: nice, DOG
# will import the name nice and DOG
```

3.2 Function

(i) objectid(x): return a unique identifier for x, something like what it was store in the storage system in computer.

(ii) haskey(collection, key) –> Bool: judge wether collection has the key. (has key)

(iii) Closures: The return of the function is a function.

```julia
function make_multiplier(factor)
    return x -> x * factor
end

multiplier = make_multiplier(2)
multiplier(3)
6
```

(iv) promote(x,y): convert x and y to the same type.

```julia
promote(1, 2.5)
(1.0, 2.5)
```

3.3 Type

(i) Ref: If a function argument type is Ref, it can be modified in-place. $a = \text{Ref}(10) \longrightarrow a[] = 10$, we can modify $a[] = 20$.

(ii) We can define an abstract type

```julia
abstract type «name» end
abstract type «name» <: «supertype» end
```

(iii) parametric type:

```julia
struct Point{T} # T is a type parameter
  field1::T
  field2::T
end

# We can use the parametric type to create a concrete type
Point{Float64}
```

```julia
Point{Float64}<:Point{Real}
false

Point{Float64}<:Point{<:Real}
true
```

(iv) tuple type:

```julia
 struct Tuple2{A,B}
    a::A
    b::B
 end
```

(v) operations on type

  (i) isa: check if a object is a given type.

```julia
isa(1, Int)
true
```

  (ii) supertype: get the supertype of a type.

```julia
supertype(Point{Float64})
Point{Real}
```

3.4 Constructors

(i) outer constructor: define function out of the struct, more flexible.

```julia
struct Foo
  x::T
  y::T
end
#function can be defined like this
Foo(x) = Foo(x, x)
Foo() = Foo(x)
```

(ii) inner constructor: define function inside the struct, more restrictive.

```julia
 struct OrderedPair
  x::Real
  y::Real
  OrderedPair(x,y) = x > y ? error("out of order") : new(x,y)
  # constraints and function definition.
end
```

3.5 Operation

(i) $x \div y$: integer divide

(ii) updating operators: $+=$, -= , $/=$, $=$, $\div=$, $\%=$, $\hat{}=$,&= , $|=$, $\veebar=$,>>>=, >>= , <<=.

(iii) vectorized “dot” operator: $[1,2,3].\hat{}3 = [1,8,27]$

(iv) “pipe” operator |>: x |> f pass x to f→f(x)
(v) cbrt(x): $\sqrt[3]{x}$

3.6 Key words

(i) global: a=1; global a+=1 change the value of a.

(ii) break: break the iteration. eg:

```julia
if j >= 3
  break
end
```

a similar use is continue

(iii) baremodule: a lightweight module, module contains base functions when construct, but baremodule doesn’t. It has to imports manually.

(iv) outer: change value of the variables out of the block.

```julia
function f()
  i = 0
  for outer i = 1:3
    # empty
  end
  return i
end;

f()
3
```

(v) const: give specific, unchanging value

```julia
const a=1
```

3.7 blocks

(i) for, zip var1 = value1

```julia
for (j, k) in zip([1 2 3], [4 5 6 7])
  println((j,k))
end

(1, 4)
(2, 5)
(3, 6)
```

(ii) let: create local variables without pollute the global ones.

```julia
let
  var1 = value1
  var2 = value2
  operations on var1, var2...
end
```

4 Paper review

4.1 Simulating 2D topological quantum phase transitions on a digital quantum computer [1]

(i) sign function:

$sign = +1(g > 0); 0(g = 0); -1(g < 0)$

(ii) $\mathbb{Z}_2$ symmetry:

$\mathbb{Z}_2$ group: {0,1}, a system has only two distinct states. $\mathbb{Z}_2$ operates twice on a state will return the oroginal state.

eg.: bit flip: $X|0\rangle = |1\rangle, XX|0\rangle = |0\rangle$

phrase flip: $Z|1\rangle = -|1\rangle, ZZ|1\rangle = |1\rangle$

(iii) time reversal symmetry:

$T|\psi(t)\rangle = |\psi(-t)\rangle$

(iv) $\mathbb{Z}_2^T$ symmetry (Time-Reversal Symmetry in $\mathbb{Z}_2$ Systems):

$TXT^{-1} = X$, $TYT^{-1} = Y$, $TZT^{-1} = Z$

4.2 Efficient Simulation of Dynamics in Two-Dimensional Quantum Spin Systems with Isometric Tensor Networks [2]

(i) **isometry is a linear map**:

$W : V_s \to V_l$

$W^\dagger W = \mathbb{I}, WW^\dagger = P_{V_s}$, P_{V_s} is projection operator of V_s to V_l.

(ii) $\mathbb{L}_2$ **norm**: Euclidean length of the vector

$v = (v_1, v_2, ..., v_n)$ in $\mathbb{R}^n$ or $\mathbb{C}^n$,

$\|v\|_2 = \sqrt{|v_1|^2 + |v_2|^2 + ... + |v_n|^2}$

(iii) **norm tensor N_l**:

The contraction $\langle\psi|\psi\rangle$ but leave out tensor at site l. Because of the isometric condition, $N_l = \mathbb{I}_{\partial V}$.

4.3 Variational Quantum Eigensolver with Fewer Qubits [3]

(i) **Jordan-Wigner transformation**:

Map Pauli operators to Fermion operators (Fermion generation operator and annihilation operator). $\sigma_j^x, \sigma_j^y, \sigma_j^z \to c_j, c_j^\dagger$

$\sigma_j^+ = \frac{1}{2}\left(\sigma_j^x + i\sigma_j^y\right)$, $\sigma_j^- = \frac{1}{2}\left(\sigma_j^x - i\sigma_j^y\right)$

$c_j = \left(\prod_{k=1}^{j-1} \sigma_k^z\right)\sigma_j^-$, $c_j^\dagger = \left(\prod_{k=1}^{j-1} \sigma_k^z\right)\sigma_j^+$

It's a non-local transformation, because c_j depends on σ_k^z for $k < j$. Besides, $\left\{c_j, c_j^\dagger\right\} = 0$, $\left\{c_j, c_j\right\} = 0$, $\left\{c_j^\dagger, c_j^\dagger\right\} = 0$

Second quantization: a general quantum many-body system, whose Hamiltonian is represented by generation and annihilation operators.

$H = \sum_{j=1}^{N} c_j^\dagger c_j + \frac{1}{2}\sum_{j,k=1}^{N}\left(c_j^\dagger c_k + c_k^\dagger c_j\right)$

$H = \sum_{j=1}^{N} c_j^\dagger c_j + \frac{1}{2}\sum_{j,k=1}^{N}\left(c_j^\dagger c_k + c_k^\dagger c_j\right)$

$H = \sum_{j=1}^{N} c_j^\dagger c_j + \frac{1}{2}\sum_{j,k=1}^{N}\left(c_j^\dagger c_k + c_k^\dagger c_j\right)$

(ii) **U(1)**:
- U(1) group: all complex number
- a system has U(1) symmetry means under the transformation $\psi \to e^{i\theta}\psi$, it remain the same.

The Heisenberg model has a U(1) symmetry with good quantum number S_z. The Hamiltonian and S_z (spin projection along z-axis) is invariant under the transformation $U(\theta) = e^{-i\theta\frac{\sigma_i^z}{2}}$.

According to **Baker-Campbell-Hausdorff formula**:

$e^A B e^{-A} = B + [A, B] + \frac{1}{2}[A, [A, B]] + \frac{1}{6}[A, [A, [A, B]]] + ...$

$U(\theta)\sigma_i^x U^\dagger(\theta) = \sigma_i^x \cos\left(\frac{\theta}{2}\right) - \sigma_i^y \sin\left(\frac{\theta}{2}\right)$

$U(\theta)\sigma_i^y U^\dagger(\theta) = \sigma_i^x \sin\left(\frac{\theta}{2}\right) + \sigma_i^y \cos\left(\frac{\theta}{2}\right)$

$U(\theta)\sigma_i^z U^\dagger(\theta) = \sigma_i^z$

$H = \sigma_i^x\sigma_j^x + \sigma_i^y\sigma_j^y + \sigma_i^z\sigma_j^z = \left(\sigma_i^x \cos\left(\frac{\theta}{2}\right) - \sigma_i^y \sin\left(\frac{\theta}{2}\right)\right)\left(\sigma_j^x \cos\left(\frac{\theta}{2}\right) - \sigma_j^y \sin\left(\frac{\theta}{2}\right)\right) + \left(\sigma_i^x \sin\left(\frac{\theta}{2}\right) + \sigma_i^y \cos\left(\frac{\theta}{2}\right)\right)\left(\sigma_j^x \sin\left(\frac{\theta}{2}\right) + \sigma_j^y \cos\left(\frac{\theta}{2}\right)\right) + \sigma_i^z\sigma_j^z = \sigma_i^x\sigma_j^x + \sigma_i^y\sigma_j^y + \sigma_i^z\sigma_j^z$

So H is invariant under the transformation $U(\theta)$. And $[H, S_z] = 0$, $\frac{dO}{dt} = -i[H, O] \Rightarrow \frac{dS_z}{dt} = 0$, S_z is invariant.

(iii) **SWAP$^\alpha$ gate**:

$\alpha = \frac{\theta}{\pi}$. $\alpha = 0$: No swap; $\alpha = 1$: Full swap; $\alpha \in (0, 1)$: Partial swap.

$$\text{SWAP}^\alpha = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\left(\frac{\alpha\pi}{2}\right) & -i\sin\left(\frac{\alpha\pi}{2}\right) & 0 \\ 0 & -i\sin\left(\frac{\alpha\pi}{2}\right) & \cos\left(\frac{\alpha\pi}{2}\right) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(iv) **SU(2)**:
- $SU(2)$ group:

all 2×2 unitary matrices with determinant 1. $UU^\dagger = U^\dagger U = \mathbb{I}$, $\det(U) = 1$

- $SU(2)$ symmetry: $UHU^\dagger = H$

(v) **Abelian symmetry** and **non-Abelian symmetry**:
- Abelian symmetry: $g_q g_2 = g_2 g_1$ eg. U(1) $e^{i\theta_1}e^{i\theta_2} = e^{i\theta_2}e^{i\theta_1}$

- Non-Abelian symmetry: $g_q g_2 \neq g_2 g_1$ eg. Pauli matrix

(vi) $\overline{CNOT(1,a)}H(1)X(1)|0\rangle_1 \otimes |0\rangle_a = \overline{CNOT(1,a)}H(1)|1\rangle_1 \otimes |0\rangle_a = \overline{CNOT(1,a)}\frac{1}{\sqrt{2}}(|0\rangle_1 - |1\rangle_1) \otimes |0\rangle_a = \frac{1}{\sqrt{2}}(|0\rangle \otimes |1\rangle_a - |1\rangle \otimes |0\rangle_a)$

4.4 Quantum circuit learning [4]

(i) **gradient of expectation value** with respect to a circuit parameter θ_i :

unitary gate: $U(\theta) = \prod_{j=1}^{l} U_j(\theta_j) = U_l(\theta_l)...U_1(\theta_1)$, let $U_{j:k} = U_j(\theta_j)...U_k(\theta_k)$

$U_j(\theta) = e^{-\frac{i}{2}\theta P_j} = \cos(\frac{\theta}{2})\mathbb{I} - i\sin(\frac{\theta}{2})P_j$, P_j is a Pauli peoduct, satisfies $P_j^2 = \mathbb{I}$, $P_j\rho P_j = \rho$

$U_j(\frac{\pi}{2})\rho U_j^\dagger(\frac{\pi}{2}) = \left(\cos(\frac{\pi}{4})\mathbb{I} - i\sin(\frac{\pi}{4})P_j\right)\rho\left(\cos(\frac{\pi}{4})\mathbb{I} + i\sin(\frac{\pi}{4})P_j\right) = \frac{1}{2}(\mathbb{I} + iP_j)\rho(\mathbb{I} - iP_j) = \frac{1}{2}\left(2\rho - i[P_j, \rho]\right)$

$U_j(-\frac{\pi}{2})\rho U_j^\dagger(-\frac{\pi}{2}) = \left(\cos(-\frac{\pi}{4})\mathbb{I} - i\sin(-\frac{\pi}{4})P_j\right)\rho\left(\cos(-\frac{\pi}{4})\mathbb{I} + i\sin(-\frac{\pi}{4})P_j\right) = \frac{1}{2}(\mathbb{I} - iP_j)\rho(\mathbb{I} + iP_j) = \frac{1}{2}\left(2\rho + i[P_j, \rho]\right)$

$\Rightarrow [P_j, \rho] = i\left[U_j(\frac{\pi}{2})\rho U_j^\dagger(\frac{\pi}{2}) - U_j(-\frac{\pi}{2})\rho U_j^\dagger(-\frac{\pi}{2})\right]$..(1)

$\langle B(\theta)\rangle = \mathrm{Tr}\left(BU_{l:1}\rho_{in}U_{l:1}^\dagger\right) = \mathrm{Tr}\left(BU_{l:j}\left(U_{j-1:1}\rho_{in}U_{j-1:1}^\dagger\right)U_{l:j}^\dagger\right)$

$\frac{\partial U_j(\theta_j)}{\partial\theta_j} = -\frac{i}{2}P_jU_j(\theta_j)$

$\frac{\partial\langle B\rangle}{\partial\theta_j} = \mathrm{Tr}\left(B\frac{\partial}{\partial\theta_j}\left(U_{l:j}\left[\left(U_{j-1:1}\rho_{in}U_{j-1:1}^\dagger\right)\right]U_{l:j}^\dagger\right)\right) = \mathrm{Tr}\ \left(BU_{l:j+1}(-\frac{i}{2}P_jU_j)\left[\left(U_{j-1:1}\rho_{in}U_{j-1:1}^\dagger\right)\right]U_{l:j}^\dagger\right) + \mathrm{Tr}\ \left(BU_{l:j}\left[U_{j-1:1}\rho_{in}U_{j-1:1}^\dagger\right](-\frac{i}{2}P_jU_j)^\dagger U_{l:j+1}^\dagger\right) = -\frac{i}{2}\ \mathrm{Tr}\ \left(BU_{l:j}\left[P_j, U_{j-1:1}\rho_{in}U_{j-1:1}^\dagger\right]U_{l:j}^\dagger\right) = -\frac{i}{2}\ \mathrm{Tr}\ \left(BU_{l:j}[P_j, \rho_j]U_{l:j}^\dagger\right)$..(2)

Substitute (1) into (2):

$\frac{\partial\langle B\rangle}{\partial\theta_j} = -\frac{i}{2}\ \mathrm{Tr}\ \left(BU_{l:j}[P_j, \rho_j]U_{l:j}^\dagger\right) = \frac{1}{2}\ \mathrm{Tr}\ \left[BU_{l:j}\left(U_j(\frac{\pi}{2})\rho_jU_j(\frac{\pi}{2})^\dagger\right)U_{l:j}^\dagger\right] - \frac{1}{2}\ \mathrm{Tr}\ \left[BU_{l:j}\left(U_j(-\frac{\pi}{2})\rho_jU_j(-\frac{\pi}{2})^\dagger\right)U_{l:j}^\dagger\right] = \frac{1}{2}\left(\langle B(\theta_j + \frac{\pi}{2})\rangle - \langle B(\theta_j - \frac{\pi}{2})\rangle\right) = \frac{1}{2}\left(\langle B\rangle_j^+ - \langle B\rangle_j^-\right)$

4.5 Time Dependent Variational Principle with Ancillary Krylov Subspace [5]

(i) **TDVP**:

- We suppose a MPS (L sites) state $|\psi\rangle$.

  The fulll Hilbert space can be decomposed into $H_j^L \otimes H_j^R$, where $H_j^L = H_1 \otimes ... \otimes H_j^L$ and $H_j^R = H_{j+1} \otimes ... \otimes H_L^R$.

  If we do SVD on the j-th site, the left and right part of j-th site will be orthogonal basis. We define this operator to the state as:

  $\hat{P}_j^{L,|\psi\rangle} = \sum A_1...A_j\overline{A_j}...\overline{A_1}|\sigma_1...\sigma_j\rangle\langle\overline{\sigma_1}...\overline{\sigma_j}| \otimes \hat{I}_{j+1}^R$

  $\hat{P}_j^{R,|\psi\rangle} = \hat{I}_{j-1}^L \otimes \sum \overline{B_L}...\overline{B_j}B_j...B_L\ |\sigma_j...\sigma_L\rangle\langle\overline{\sigma_j}...\overline{\sigma_L}|$

  Their tensor network representations are:

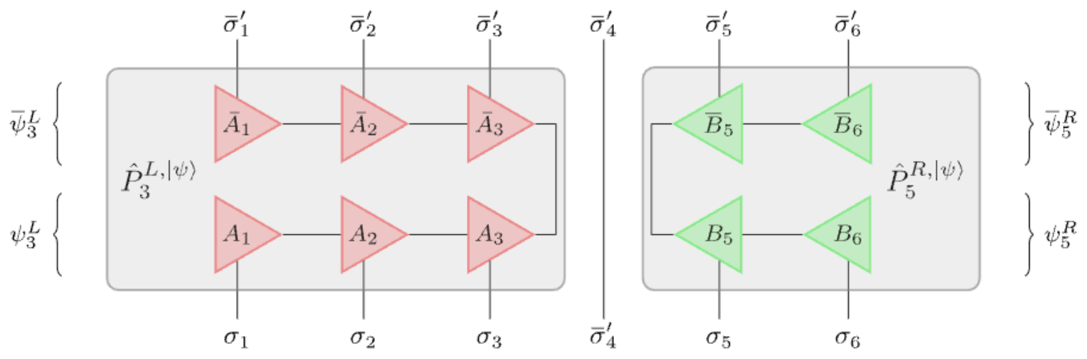


Figure 6: projector

- Then we define:

  $\hat{P}_{T\ |\psi\rangle} = \sum_{j=1}^{L} \hat{P}^{L,|\psi\rangle}_{j-1} \otimes \hat{I}_j \otimes \hat{P}^{R,|\psi\rangle}_{j+1} - \sum_{j=1}^{L-1} \hat{P}^{L,|\psi\rangle}_{j} \otimes \hat{P}^{R,|\psi\rangle}_{j+1}$ (1)

  (1) The first term: Only the j-th site is free, the others are fixed in orthogonal basis.

  (2) The second term: The bonds are fixed twice. So we have to move ones.

  Thus, $\hat{P}_{T\ |\psi\rangle}$ is exactly the tangent space projector.

- From local Krylov method, we know the approximation:

  $\frac{\partial\ |\psi\rangle}{\partial t} = -i\hat{P}_{T,\ |\psi\rangle}\hat{H}\ |\psi\rangle$ (2)

  Substitute (1) into (2), we know that we have to solve:

  (1)L forward-evolving equations:

  $\frac{\partial\ |\psi\rangle}{\partial t} = -i\hat{P}^{L,|\psi\rangle}_{j-1} \otimes \hat{I}_j \otimes \hat{P}^{R,|\psi\rangle}_{j+1}\hat{H}\ |\psi\rangle = -i\hat{H}^{\text{eff}}_j\ |\psi\rangle$

  We know that except the j-th site, the others are identity when contract with their ajoint part. So

  $\frac{\partial M_j}{\partial t} = -i\hat{H}^{\text{eff}}_j M_j$

  The solution of M_j has the form: $M_j = e^{-i\hat{H}^{\text{eff}}_j t} M^0_j$

  We update it at $\Delta t = \frac{\delta}{2}$.

  $M_j \leftarrow e^{-i\hat{H}^{\text{eff}}_j (t+\frac{\delta}{2})} = e^{-i\hat{H}^{\text{eff}}_j \frac{\delta}{2}} M_j$ (3)

  (2)L-1 backward-evolving equations: $\frac{\partial\ |\psi\rangle}{\partial t} = +i\hat{P}^{L,|\psi\rangle}_{j} \otimes \hat{P}^{R,|\psi\rangle}_{j+1}\hat{H}\ |\psi\rangle$

  $\frac{\partial C_j}{\partial t} = -i\hat{H}^{\text{eff}}_j C_j$

  We update it at $\Delta t = \frac{\delta}{2}$.

  $C_j \leftarrow e^{-i\hat{H}^{\text{eff}}_j (t+\frac{\delta}{2})} C_j = e^{-i\hat{H}^{\text{eff}}_j \frac{\delta}{2}} C_j$ (4)

- TDVP is similar to DMRG, The only difference is that:

  DMRG update the state by local eigen state (through Lanczos), while TDVP update the state by applying local effective Hamiltonian ((3) and (4)).

All the sites are updated in turn.(right and left sweep, QR decomposition)

- Why imaginary time evolution will get the ground state?

  Expand an initial state $|\psi\rangle_0$ in the energy eigenbasis $\{|E_n\rangle\}$

  $|\psi(0)\rangle = \sum_n c_n \, |E_n\rangle$

  After imaginary time τ:

  $|\psi(\tau)\rangle = e^{-H\tau} \, |\psi(0)\rangle = \sum_n c_n e^{-E_n\tau} \, |E_n\rangle$

  High-energy states decay exponentially faster than the ground state. As $\tau \to \infty$, only the ground state $|E_0\rangle$ survives.

  $|\psi(\tau)\rangle \approx c_0 e^{-E_0\tau} \, |E_0\rangle$

(i) **Heisenberg ladder**:

  Two rows chains, $J_\parallel$ is coefficient of the parallel spin-spin interaction, $J_\perp$ is coefficient of the perpendicular spin-spin interaction.

  $H = J_\parallel \sum_{i=1}^{N} \left(\hat{S}_{1,i} \cdot \hat{S}_{1,i+1} + \hat{S}_{2,i} \cdot \hat{S}_{2,i+1} \right) + J_\perp \sum_{i=1}^{N} \hat{S}_{1,i} \cdot \hat{S}_{2,i}$

  The first term is Intra-leg couplings, the second term is Rung couplings couplings.

- rung-decoupled Heisenberg ladder: $J_\perp = 0$

  Map the 2D system to 1D system:

  origin: (1,1) (1,2) (1,3) (1,4)

  (2,1) (2,2) (2,3) (2,4)

  - zig-zag path:

    (1,1) (2,1) (1,2) (2,2) (1,3) (2,3) (1,4) (2,4)

    We can know that the interaction is between:

  $(1,1) - (1,3)$, $(1,2) - (1,3)$, $(1,3) - (1,4)$

  $(2,1) - (2,2)$, $(2,2) - (2,3)$, $(2,3) - (2,4)$

  - snake path:

  (1,1) (2,1) (2,2) (1,2) (1,3) (2,3) (2,4) (1,4)

  Interaction is the same as above.

- rung-coupled Heisenberg ladder: $J_\perp \neq 0$

  We can know that the interaction is between:

  $(1,1) - (1,3)$, $(1,2) - (1,3)$, $(1,3) - (1,4)$

  $(2,1) - (2,2)$, $(2,2) - (2,3)$, $(2,3) - (2,4)$

  Also, the interaction between two rows:

  $(1,1) - (2,1)$, $(1,2) - (2,2)$, $(1,3) - (2,3)$, $(1,4) - (2,4)$

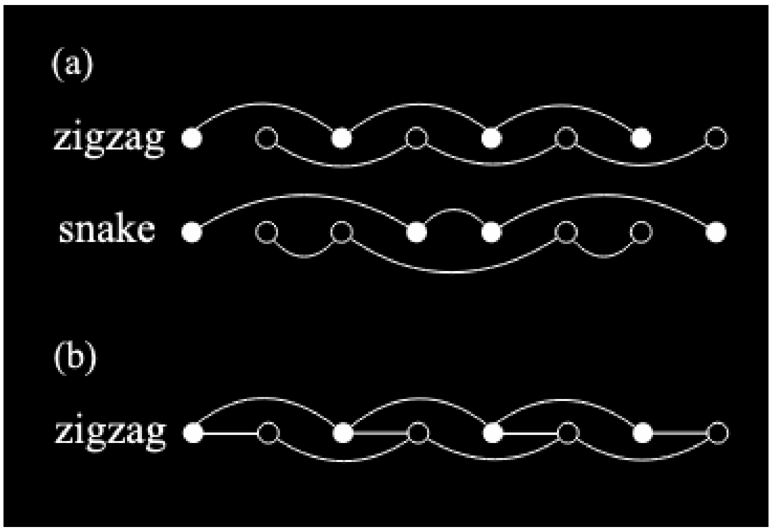


Figure 7: map to zig-zag or snake

From above, we can know that we should build entanglement between two sites which are not nearest neighbor. So the 2TDVP fails. We need to enlarge the bond dimension (tangent space of the MPS manifold).

(i) **GSE-TDVP1**:

- basis extension:

  Like doing DMRG to multiple sites.

  Suppose the original state is an MPS state in left canonical form:

  $|\psi\rangle = \sum_{s_1 \dots s_N} A_1^{s_1} \dots A_{N-1}^{s_{N-1}} C_N^{s_N} \; |s_1 \dots s_N\rangle$

  We extend the bond basis by

  $|\tilde{\psi}\rangle = \sum_{s_1 \dots s_N} \tilde{A}_1^{s_1} \dots \tilde{A}_{N-1}^{s_{N-1}} \tilde{C}_N^{s_N} \; |s_1 \dots s_N\rangle$

  Then we can sum them directly

  $\binom{|\psi\rangle}{|\tilde{\psi}\rangle} = \sum_{s_1 \dots s_N} A_1'^{s_1} \dots A_{N-1}'^{s_{N-1}} C_N'^{s_N} \; |s_1 \dots s_N\rangle = \sum_{s_1 \dots s_N} \begin{pmatrix} A_1^{s_1} & 0 \\ 0 & \tilde{A}_1^{s_1} \end{pmatrix} \dots \begin{pmatrix} A_{N-1}^{s_{N-1}} & 0 \\ 0 & \tilde{A}_{N-1}^{s_{N-1}} \end{pmatrix} \begin{pmatrix} C_N^{s_N} & 0 \\ 0 & \tilde{C}_N^{s_N} \end{pmatrix} \; |s_1 \dots s_N\rangle$

  After left sweep, we can get the right canonical form.

  $\binom{|\psi\rangle}{|\tilde{\psi}\rangle} = \sum_{s_1 \dots s_N} \begin{pmatrix} C_1^{s_1} & 0 \\ 0 & \tilde{C}_1^{s_1} \end{pmatrix} B_2'^{s_2} \dots B_N'^{s_N} \; |s_1 \dots s_N\rangle$

  To get more accuracy result ($|\psi\rangle$), **we aim to let $A_i^{s_i}$ orthogonal to $\tilde{A}_i^{s_i}$ and $B_i^{s_i}$ orthogonal to $\tilde{B}_i^{s_i}$.**

  In the following discussion, suppose we have $k-1$ $|\tilde{\psi}\rangle$.

  The reduced density matrix of $k-1$ $|\tilde{\psi}\rangle$ is:

  $\tilde{\rho} = \sum_{l=1}^{k-1} \tilde{\rho}_l$, $\tilde{\rho}_l = \tilde{C}_l^\dagger \tilde{C}_l$

  At site i of $|\psi\rangle$, do SVD:

$C = USB$

Define projector:

$P = 1 - B^\dagger B$ (project to the orthogonal complement of B)

Then we project $\tilde{\rho}$ by P:

$\overline{\rho} = P\tilde{\rho}P = \overline{B}^\dagger \overline{S}^2 \overline{B}$

Thus, $B' = \begin{pmatrix} B \\ \overline{B} \end{pmatrix}, \overline{B}B^\dagger = 0$

Then, what |tilde(psi) angle.r we have to choose?

- Krylov subspace:

  Time evolution of $|\psi\rangle$ is:

  $|\psi(t + \Delta t)\rangle = e^{-i\Delta t H} \, |\psi(t)\rangle \approx \sum_{l=1}^{k-1} \frac{(-i\Delta t)^l}{l!} \hat{H}^l \, |\psi(t)\rangle$

  We use $\hat{H}^l \, |\psi(t)\rangle$ to expand the basis.

  $\Rightarrow$ Krylov subspace: $\mathrm{K}_k\left(\hat{H}, |\psi\rangle\right) = \mathrm{span}\left\{|\psi\rangle, \hat{H} \, |\psi\rangle, \hat{H}^2 \, |\psi\rangle, ...\hat{H}^{k-1} \, |\psi\rangle\right\}$

  Norm grows exponentially: $\Rightarrow \left(1 - i\tau\hat{H}\right) |\psi(t)\rangle, ...\left(1 - i\tau\hat{H}\right)^{k-1} |\psi(t)\rangle$

  And $k = 3$ is enough for accuracy.

  Why choose Krylov subspace?

  $|\psi\rangle \rightarrow |\psi\rangle$: initial product state

  $H \, |\psi\rangle \rightarrow J_\parallel \hat{S}_{1,1}\hat{S}_{1,2} \, |\psi\rangle$: include intra-leg and rung couplings

  $\hat{H}^2 \, |\psi\rangle \rightarrow J_\parallel^2 \left(\hat{S}_{1,1}\hat{S}_{1,2}\right)^2 \, |\psi\rangle$

  (maybe combine the first two terms)

- Step:

  (1)construct MPO: $1 - i\tau\hat{H}$, apply to $|\psi\rangle$ $k - 1$ times

  then get the Krylov subspace: $\left(1 - i\tau\hat{H}\right) |\psi(t)\rangle, ...\left(1 - i\tau\hat{H}\right)^{k-1} |\psi(t)\rangle$

  (2)basis extension: $\begin{pmatrix} |\psi\rangle \\ ... \\ |\,\tilde{\psi}^{k-1}\rangle \end{pmatrix} = C_1^{s_1}...B_{N-1}^{s_{\mathbb{N}-1}} \, |s_1...s_N\rangle$

  (3) do 1TDVP, sweep from left to right

4.6 Thermal Tensor Network Approach for Spin-Lattice Relaxation in Quantum Magnets [6]

Bibliography

[1] Y.-J. Liu, K. Shtengel, and F. Pollmann, "Simulating two-dimensional topological quantum phase transitions on a digital quantum computer," *Physical Review Research*, vol. 6, no. 4, Dec. 2024, doi: 10.1103/physrevresearch.6.043256.

[2] S.-H. Lin, M. P. Zaletel, and F. Pollmann, "Efficient simulation of dynamics in two-dimensional quantum spin systems with isometric tensor networks," *Phys. Rev. B*, vol. 106, no. 24, p. 245102, Dec. 2022, doi: 10.1103/PhysRevB.106.245102.

[3] J.-G. Liu, Y.-H. Zhang, Y. Wan, and L. Wang, “Variational quantum eigensolver with fewer qubits,” *Physical Review Research*, vol. 1, no. 2, Sep. 2019, doi: 10.1103/physrevresearch.1.023025.

[4] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, “Quantum circuit learning,” *Physical Review A*, vol. 98, no. 3, Sep. 2018, doi: 10.1103/physreva.98.032309.

[5] M. Yang and S. R. White, “Time-dependent variational principle with ancillary Krylov subspace,” *Physical Review B*, vol. 102, no. 9, Sep. 2020, doi: 10.1103/physrevb.102.094315.

[6] N. Xi *et al.*, “Thermal Tensor Network Approach for Spin-Lattice Relaxation in Quantum Magnets.” [Online]. Available: https://arxiv.org/abs/2403.11895

APPENDIX A Experiences and lessons

$\langle H \rangle_\theta = \langle \psi_\theta \mid H \mid \psi_\theta \rangle$

$\frac{\partial \langle H \rangle_\theta}{\partial \theta_i} =?$