東京大学 THE UNIVERSITY OF TOKYO

8. Classical Simulation (Clifford Circuit, Tensor Network)

2024/06/07

Yoshiaki Kawase

The University of Tokyo
Special Lectures in Information Science II
Introduction to Near-Term Quantum Computing
情報科学科特別講義Ⅱ/量子計算論入門
2024年度の計画

Path to the Utility era in Quantum Computing

The goal of this course is to learn how to implement utility-scale applications on a quantum computer. To achieve the goal, the course covers from the basics of quantum information to recent advances of quantum algorithms for noisy quantum devices as well as circuit optimization and error mitigation techniques. The course also introduces how to implement quantum algorithms using open-source framework of quantum computing and real quantum device with more than 127 qubits. The course is intended to help students understand the potential and limitations of currently available quantum devices.

Schedule: Every Friday from 16:50 to 18:20 (except May 15 (Wed), May 30(Thu))

Notes: All lectures will be held in person. Recording also will be available for reviewing.

Course Schedule 2024

Date	Lecture Title	Lecturer	Date	Lecture Title	Lecturer
4/5	Invitation to the Utility era	Tamiya Onodera	6/7	Classical simulation (Clifford circuit, tensor network)	Yoshiaki Kawase
4/19	Quantum Gates, Circuits, and Measurements	Kifumi Numata	6/14	Quantum Hardware	Masao Tokunari
4/26	LOCC (Quantum teleportation/superdense coding/Remote CNOT)	Kifumi Numata/ Atsushi Matsuo	6/21	Quantum circuit optimization (transpilation)	Toshinari Itoko
5/10	Quantum Algorithms: Grover's algorithm	Atsushi Matsuo	6/28	Quantum noise and quantum error mitigation	Toshinari Itoko
5/15 (Wed)	Quantum Algorithms: Phase estimation	Kento Ueda	7/5	Quantum Utility I (127Qubit GHZ)	Kifumi Numata
5/24	Quantum Algorithms: Variational Quantum Algorithms (VQA)	Takashi Imamichi	7/12	Quantum Utility II (Utility paper implementation)	Tamiya Onodera
5/30 (Thu)	Quantum simulation (Ising model, Heisenberg, XY model), Time evolution (Suzuki Trotter, QDrift)	Yukio Kawashima	7/19	Quantum Utility III (Krylov subspace expansion)	Yukio Kawashima

The topics of today's class

Studying three types of classical simulation:

- State Vector

- Tensor Network
  Matrix Product State

- Stabilizer
  Clifford Circuit, a circuit consisting of H, S and CNOT gates

State vector simulation

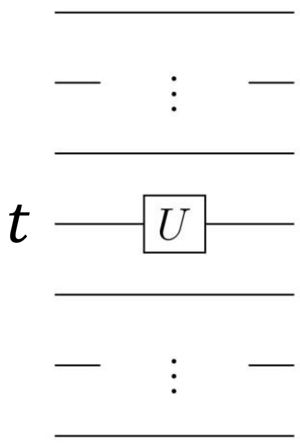
An n-qubit quantum state is represented by

$$|\psi\rangle = \sum_{\sigma_1=0}^{1} \cdots \sum_{\sigma_n=0}^{1} c_{\sigma_n \cdots \sigma_1} |\sigma_n \cdots \sigma_1\rangle = \begin{pmatrix} c_{0\ldots0} \\ \vdots \\ c_{1\ldots1} \end{pmatrix},$$

where c_i is a complex number and $\sum_{i=0}^{2^n-1} |c_i|^2 = 1$ is satisfied.

Since the length of state vector is 2^n,
the time and space complexity increase exponentially with the number of qubits.

Applying a single qubit gate (state vector simulation)



A single qubit gate can be written by

$$\boldsymbol{U_t} = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix}$$
$$= u_{00}|0\rangle\langle 0| + u_{10}|1\rangle\langle 0| + u_{01}|0\rangle\langle 1| + u_{11}|1\rangle\langle 1|,$$

where $\boldsymbol{U}_t\boldsymbol{U}_t^\dagger = \boldsymbol{U}_t^\dagger\boldsymbol{U}_t = I$ is satisfied.
Here, we apply a single qubit gate $\boldsymbol{I} \otimes \cdots \otimes \boldsymbol{I} \otimes \boldsymbol{U}_t \otimes \boldsymbol{I} \otimes \cdots \otimes \boldsymbol{I}$ to a quantum state:

$$\boldsymbol{I} \otimes \cdots \otimes \boldsymbol{I} \otimes \boldsymbol{U}_t \otimes \boldsymbol{I} \otimes \cdots \otimes \boldsymbol{I}\,|\psi\rangle = \sum_{\boldsymbol{\sigma_1,\cdots,\sigma_t,\cdots,\sigma_n}} (\boldsymbol{I} \otimes \cdots \otimes \boldsymbol{I} \otimes \boldsymbol{U}_t \otimes \boldsymbol{I} \otimes \cdots \otimes \boldsymbol{I}\,)\boldsymbol{c_{\sigma_n\cdots\sigma_t\cdots\sigma_1}}|\boldsymbol{\sigma_n \cdots \sigma_t \cdots \sigma_1}\rangle$$

$$= \sum_{\boldsymbol{\sigma_1,\cdots,\sigma_{t-1},\sigma_{t+1},\cdots,\sigma_n}} \left(\sum_{\boldsymbol{\sigma_t=0}}^{\boldsymbol{1}} \boldsymbol{U}_t \boldsymbol{c_{\sigma_n\cdots\sigma_t\cdots\sigma_1}}|\boldsymbol{\sigma_t}\rangle \right) |\boldsymbol{\sigma_n \cdots \sigma_{t-1}\sigma_{t+1} \cdots \sigma_1}\rangle$$

We need $O(2^n)$ to update all the elements of the state vector.

Examples of applying a single qubit gate (state vector simulation)

For 1 qubit system, a quantum state is updated by

$$\boldsymbol{U_t}|\psi\rangle = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} u_{00}c_0 + u_{01}c_1 \\ u_{10}c_0 + u_{11}c_1 \end{pmatrix}.$$

Similarly, for 2 qubit system (the right most qubit is the target qubit), a quantum state is updated by

$$\boldsymbol{U_1}|\psi\rangle = \begin{pmatrix} \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix} \begin{pmatrix} c_{00} \\ c_{01} \end{pmatrix} \\ \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix} \begin{pmatrix} c_{10} \\ c_{11} \end{pmatrix} \end{pmatrix} = \begin{pmatrix} u_{00}c_{00} + u_{01}c_{01} \\ u_{10}c_{00} + u_{11}c_{01} \\ u_{00}c_{10} + u_{01}c_{10} \\ u_{10}c_{10} + u_{11}c_{11} \end{pmatrix}.$$

In general, we need to calculate the following formula 2^{n-1} times:

$$\begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix} \begin{pmatrix} c_{\sigma_n \dots \sigma_{t+1} 0 \sigma_{t-1} \dots \sigma_1} \\ c_{\sigma_n \dots \sigma_{t+1} 1 \sigma_{t-1} \dots \sigma_1} \end{pmatrix} \quad \forall \sigma_1, \cdots, \sigma_{t-1}, \sigma_{t+1}, \cdots, \sigma_n \in \{0,1\}$$

$$\uparrow$$

$$\sigma_t$$

Applying CNOT gate (using a state vector)

CNOT gate can be written by

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X.$$

$$\text{swap} \quad \begin{pmatrix} c_{00} \\ c_{01} \\ c_{10} \\ c_{11} \end{pmatrix}$$

If we apply a CNOT gate to a n-qubit quantum state $|\psi\rangle$ with cth control qubit and tth target qubit, the quantum state becomes

$$\begin{aligned} &(I_1 \otimes \cdots \otimes I_{c-1} \otimes |0\rangle\langle 0| \otimes I_{c+1} \otimes \cdots \otimes I_{t-1} \otimes I_t \otimes I_{t+1} \otimes \cdots \otimes I_n \\ &+ I_1 \otimes \cdots \otimes I_{c-1} \otimes |1\rangle\langle 1| \otimes I_{c+1} \otimes \cdots \otimes I_{t-1} \otimes X_t \otimes I_{t+1} \otimes \cdots \otimes I_n)|\psi\rangle \\ &= \sum_{\boldsymbol{\sigma_1, \cdots, \sigma_c, \cdots, \sigma_t, \cdots, \sigma_n}} \boldsymbol{c}_{\boldsymbol{\sigma_n \cdots \sigma_c \cdots \underset{t\text{th}}{(\sigma_c \oplus \sigma_t)} \cdots \sigma_1}} |\boldsymbol{\sigma_n \cdots \sigma_c \cdots \sigma_t \cdots \sigma_1}\rangle\,. \end{aligned}$$

So, we need $O(2^n)$ to update all the elements of the quantum state.

Examples of applying CNOT gate (using a state vector)

For 2 qubit system, when we apply a CNOT gate with 2nd qubit as control qubit and 1st qubit as target qubit to a quantum state, the quantum state is updated by

$$\boldsymbol{CX_{21}}|\psi\rangle = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} c_{00} \\ c_{01} \\ c_{10} \\ c_{11} \end{pmatrix} = \begin{pmatrix} c_{00} \\ c_{01} \\ c_{11} \\ c_{10} \end{pmatrix}.$$

just swap c_{10} and c_{11}

Similarly, for 3qubit system, it is updated by

$$\boldsymbol{CX_{21}}|\psi\rangle = \begin{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} c_{000} \\ c_{001} \\ c_{010} \\ c_{011} \end{pmatrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} c_{100} \\ c_{101} \\ c_{110} \\ c_{111} \end{pmatrix} \end{pmatrix} = \begin{pmatrix} c_{000} \\ c_{001} \\ c_{011} \\ c_{010} \\ c_{100} \\ c_{101} \\ c_{111} \\ c_{110} \end{pmatrix}.$$

swap c_{010} and c_{011}

swap c_{110} and c_{111}

Examples of applying CNOT gate (using a state vector)

In general, we need to calculate the following formula 2^{n-2} times:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} c_{\sigma_n \ldots 0 \ldots 0 \ldots \sigma_1} \\ c_{\sigma_n \ldots 0 \ldots 1 \ldots \sigma_1} \\ c_{\sigma_n \ldots 1 \ldots 0 \ldots \sigma_1} \\ c_{\sigma_n \ldots \underset{\underset{c}{\uparrow}}{1} \ldots \underset{\underset{t}{\uparrow}}{1} \ldots \sigma_1} \end{pmatrix} \quad \forall \sigma_1, \cdots, \sigma_{t-1}, \sigma_{t+1}, \cdots, \sigma_n \in \{0,1\}$$

The calculation of inner-products (using state vector)

Inner product between two quantum states is calculated by

$$\langle \psi' | \psi \rangle = \sum_{\substack{\sigma_1,\ldots,\sigma_n \\ \sigma'_1 \cdots \sigma'_n}} \langle \sigma'_n \cdots \sigma'_1 | {c'}^*_{\sigma'_n \cdots \sigma'_1} c_{\sigma_n \cdots \sigma_1} | \sigma_n \cdots \sigma_1 \rangle$$

$$= \sum_{\substack{\sigma_1,\ldots,\sigma_n \\ \sigma'_1 \cdots \sigma'_n}} {c'}^*_{\sigma'_n \cdots \sigma'_1} c_{\sigma_n \cdots \sigma_1} \, \delta_{\sigma_1, \sigma'_1} \cdots \delta_{\sigma_n, \sigma'_n}$$

$$= \sum_{\sigma_1,\ldots,\sigma_n} {c'}^*_{\sigma_n \cdots \sigma_1} c_{\sigma_n \cdots \sigma_1}$$

We need $O(2^n)$.

Disadvantage of state vector simulation

Disadvantage of state vector simulation:
The length of state vector $O(2^n)$ -> This takes exponentially large memory space.
Also, we need $O(2^n)$ times to update the state vector.

We will present a classical simulation method using Matrix Product States
to perform more efficient classical simulation.

Tensor

- Scalar: s
  (rank-0 tensor)

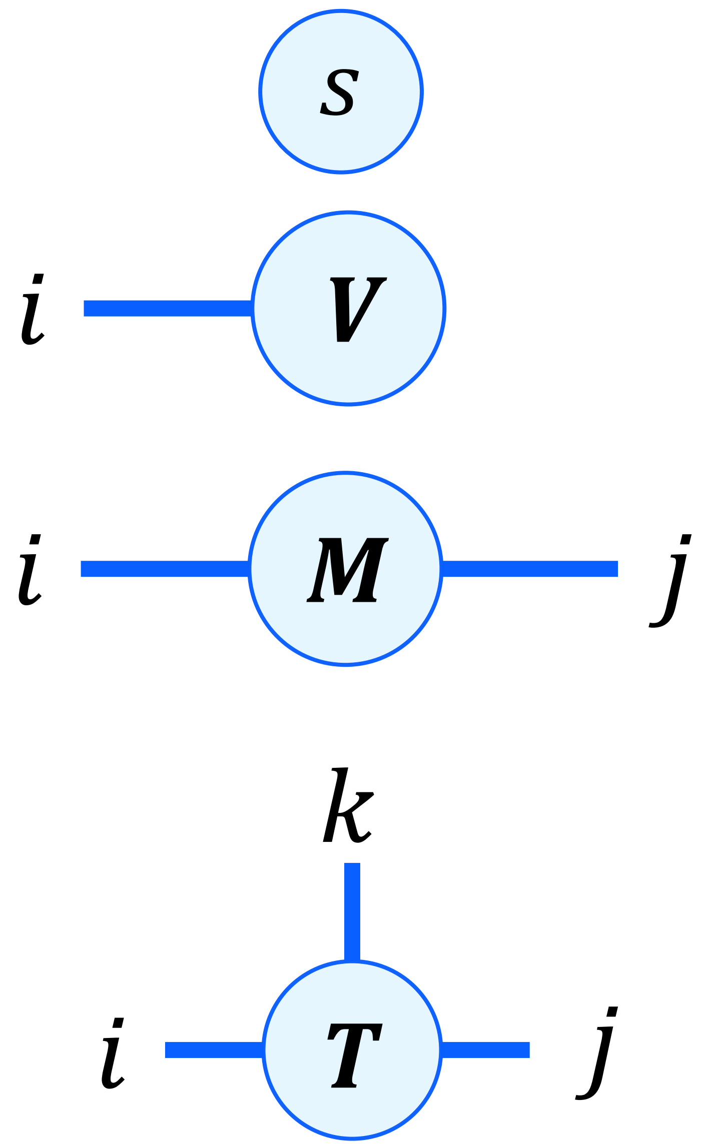
- Vector: $\boldsymbol{V} = (V_1, V_2, \dots, V_n)$
  (rank-1 tensor)

- Matrix:
  (rank-2 tensor)

$$\boldsymbol{M} = \begin{pmatrix} M_{11} & \cdots & M_{1n} \\ \vdots & \ddots & \vdots \\ M_{n1} & \cdots & M_{nn} \end{pmatrix}$$

- Rank-3 tensor: $\boldsymbol{T}_{ijk}$ or $\boldsymbol{T}_{ij}^{k}$

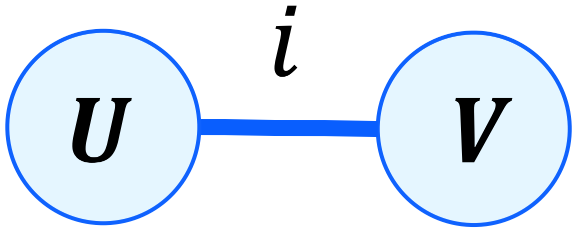
In this lecture, we regard these tensors representing the same tensor.



Tensor Contraction

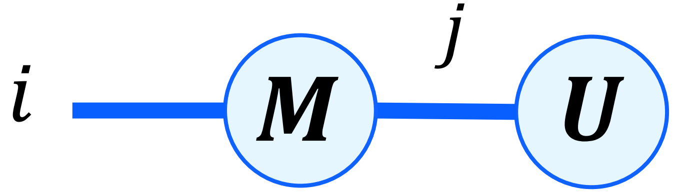
Inner product:

$$\boldsymbol{U} \cdot \boldsymbol{V} = \sum_i U_i V_i$$



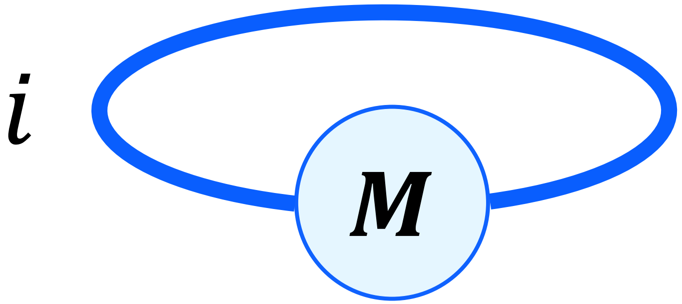
Matrix Vector Product:

$$V_{\boldsymbol{i}} = \sum_{\boldsymbol{j}} M_{\boldsymbol{ij}} U_{\boldsymbol{j}}$$



Trace:

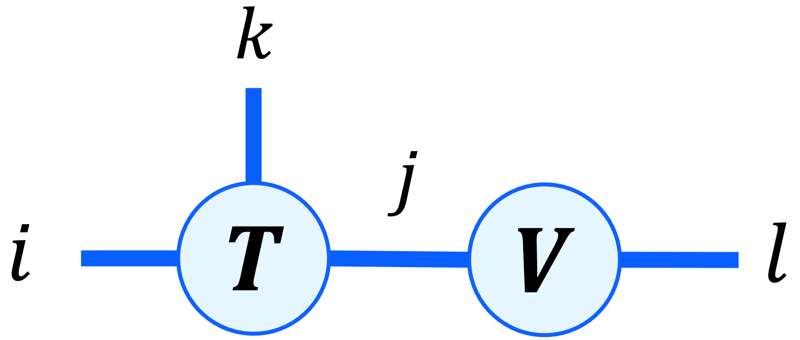
$$\mathrm{Tr}\, \boldsymbol{M} = \sum_{\boldsymbol{i}} M_{\boldsymbol{ii}}$$



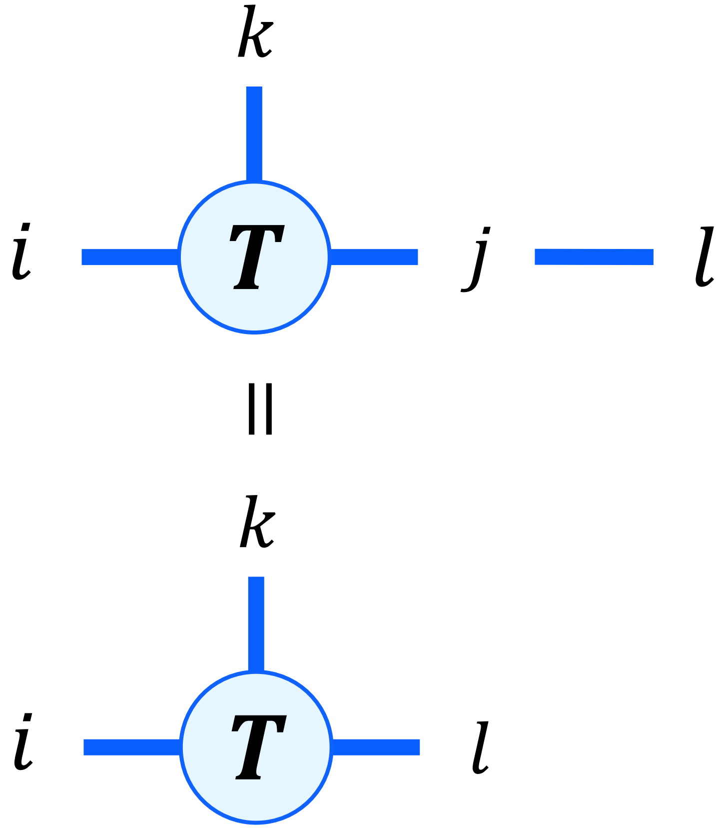
$$\boldsymbol{U} = (U_1, \dots, U_n), \boldsymbol{V} = (V_1, \dots, V_n), M = \begin{pmatrix} M_{11} & \dots & M_{1n} \\ \vdots & \ddots & \vdots \\ M_{n1} & \dots & M_{nn} \end{pmatrix}$$

Tensor Contraction

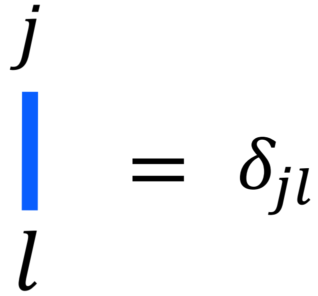
$$\sum_j T_{ijk} V_{jl}$$



$$\sum_j T_{ijk} \delta_{jl} = T_{ilk}$$

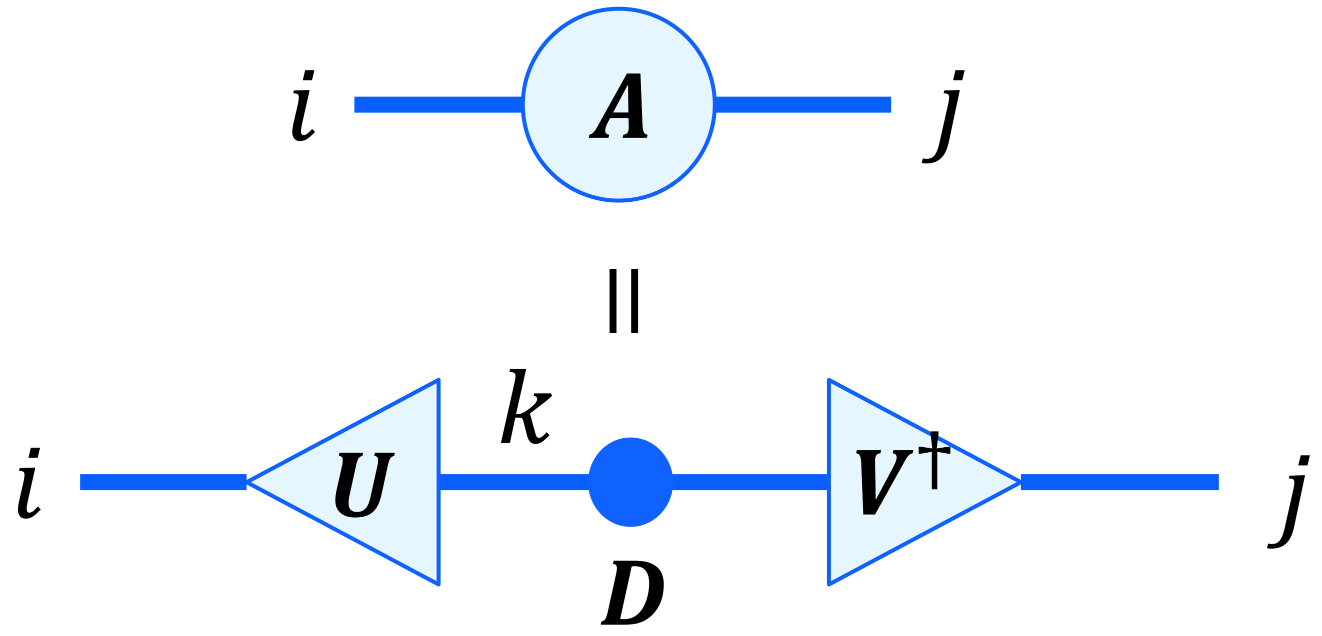


Identity matrix



Tensor Decomposition Using Singular Value Decomposition

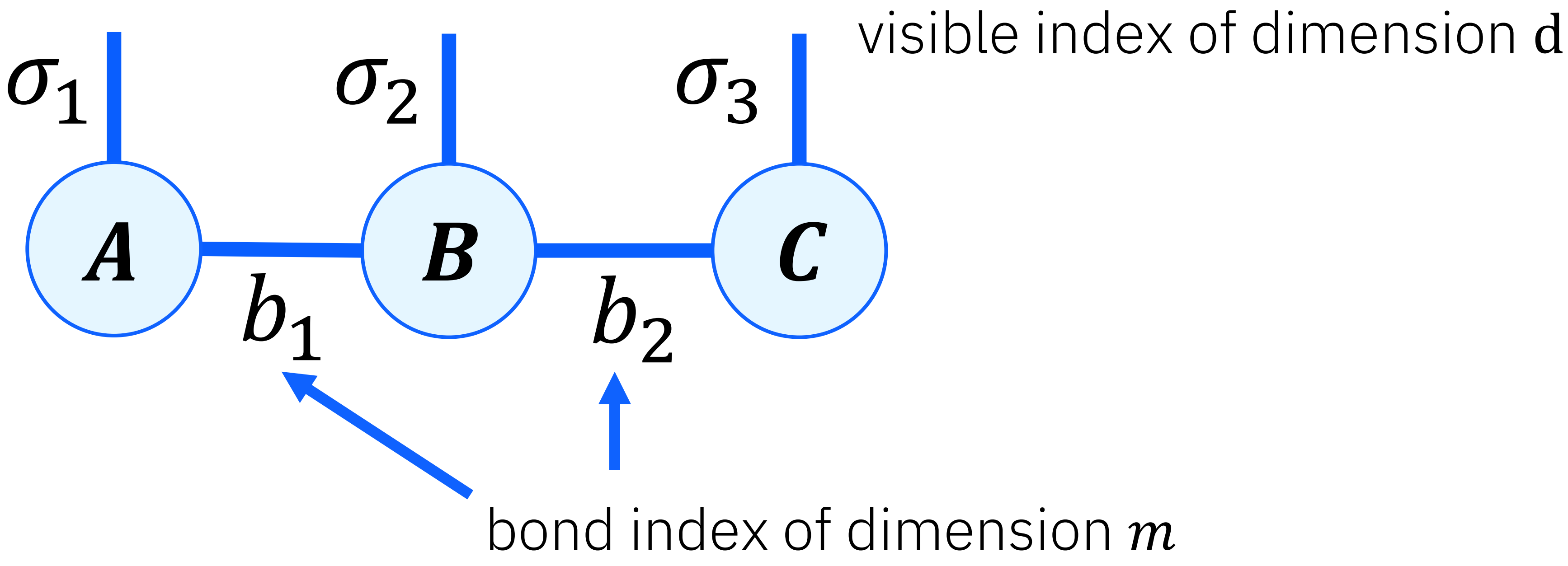
$$A_{ij} = \sum_{k=1}^{s}\sum_{l=1}^{s} U_{ik}(D_k\delta_{kl})V_{jl}^{\dagger} = \sum_{k=1}^{s} U_{ik}D_kV_{jk}^{\dagger}$$



Matrix Product State (MPS)

$$|\psi\rangle = \sum_{\substack{\sigma_1\sigma_2\sigma_3,\\ b_1 b_2}} A^{\sigma_1}_{b_1} B^{\sigma_2}_{b_1 b_2} C^{\sigma_3}_{b_2} |\sigma_1\sigma_2\sigma_3\rangle$$

This is useful for representing one-dimensional quantum systems.



Example of Matrix Product State

Representing 3 qubit GHZ state using Matrix Product State:

$$
\begin{aligned}
|\psi\rangle &= \frac{1}{\sqrt{2}}(|000\rangle + |111\rangle) \\
&= \frac{1}{\sqrt{2}}(1\ \ 0)\begin{pmatrix}1 & 0\\0 & 0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}|000\rangle + \frac{1}{\sqrt{2}}(1\ \ 0)\begin{pmatrix}1 & 0\\0 & 0\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}|001\rangle + \frac{1}{\sqrt{2}}(1\ \ 0)\begin{pmatrix}0 & 0\\0 & 1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}|010\rangle \\
&+ \frac{1}{\sqrt{2}}(1\ \ 0)\begin{pmatrix}0 & 0\\0 & 1\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}|011\rangle + \frac{1}{\sqrt{2}}(0\ \ 1)\begin{pmatrix}1 & 0\\0 & 0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}|100\rangle + \frac{1}{\sqrt{2}}(0\ \ 1)\begin{pmatrix}1 & 0\\0 & 0\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}|101\rangle \\
&+ \frac{1}{\sqrt{2}}(0\ \ 1)\begin{pmatrix}0 & 0\\0 & 1\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix}|110\rangle + \frac{1}{\sqrt{2}}(0\ \ 1)\begin{pmatrix}0 & 0\\0 & 1\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}|111\rangle
\end{aligned}
$$

$$
|\psi\rangle = \sum_{\sigma_1,\sigma_2,\sigma_3,b_1,b_2} A_{b_1}^{\sigma_1} B_{b_1,b_2}^{\sigma_2} C_{b_2}^{\sigma_3} |\sigma_1\sigma_2\sigma_3\rangle\,,
$$

where $\sigma_1 \in \{0,1\}, \sigma_2 \in \{0,1\}, \sigma_3 \in \{0,1\}$,

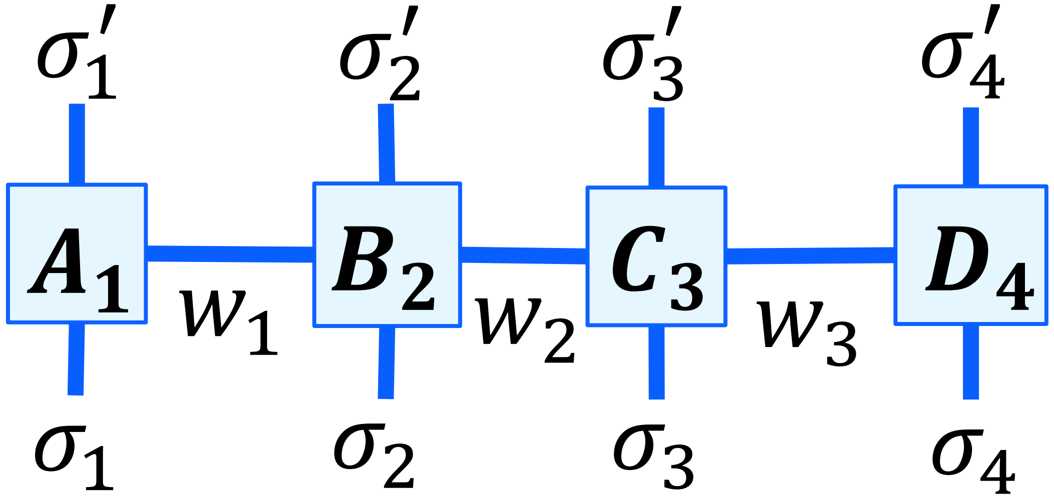
$$
A_{b_1}^0 = (1\ \ 0), A_{b_1}^1 = (0\ \ 1), B_{b_1,b_2}^0 = \begin{pmatrix}1/\sqrt{2} & 0\\0 & 0\end{pmatrix}, B_{b_1,b_2}^1 = \begin{pmatrix}0 & 0\\0 & 1/\sqrt{2}\end{pmatrix}, C_{b_2}^0 = \begin{pmatrix}1\\0\end{pmatrix}, C_{b_2}^1 = \begin{pmatrix}0\\1\end{pmatrix}
$$

Matrix Product Operator

Matrix Product Operator is written by

$$\hat{O} = \sum_{\substack{\sigma_1,\sigma_2,\sigma_3,\sigma_4 \\ \sigma_1',\sigma_2',\sigma_3',\sigma_4' \\ w_1,w_2,w_3}} A_{w_1}^{\sigma_1,\sigma_1'} B_{w_1,w_2}^{\sigma_2,\sigma_2'} C_{w_2,w_3}^{\sigma_3,\sigma_3'} D_{w_3}^{\sigma_4,\sigma_4'}.$$

Matrix Product Operator is used for applying quantum gates and evaluating expectation values.



Applying a single qubit gate (using MPS)

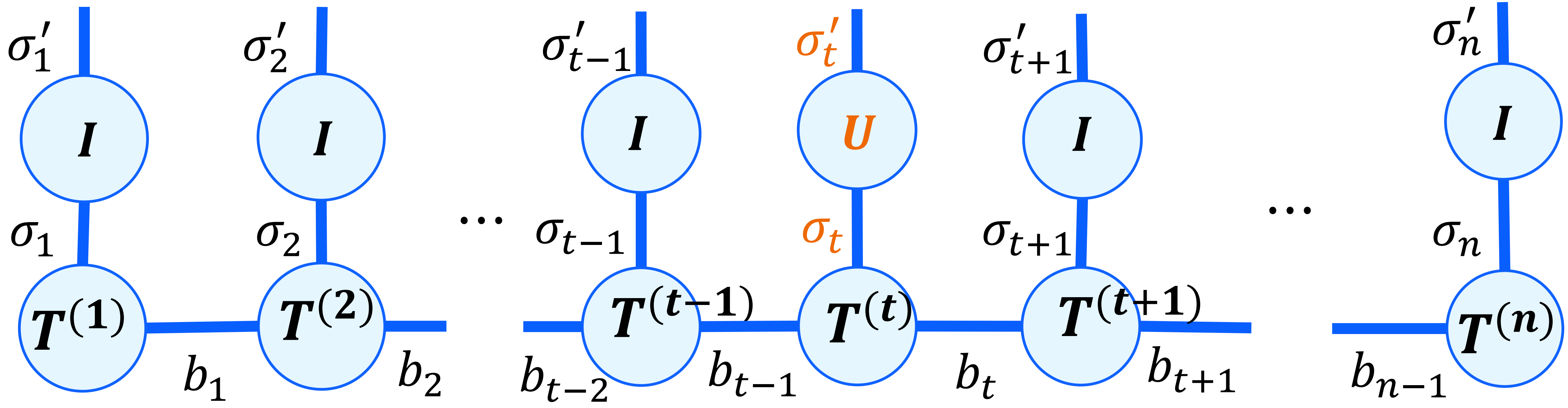
A single qubit gate can be represented by

$$\boldsymbol{U}^{\sigma_t,\sigma_t'} = u_{00}|0\rangle\langle 0| + u_{10}|1\rangle\langle 0| + u_{01}|0\rangle\langle 1| + u_{11}|1\rangle\langle 1| = \sum_{\sigma_t'=0}^{1}\sum_{\boldsymbol{\sigma_t=0}}^{1} u_{\boldsymbol{\sigma_t'\sigma_t}}|\boldsymbol{\sigma_t'}\rangle\langle\boldsymbol{\sigma_t}|\,.$$

A quantum state applied by a single qubit gate $\boldsymbol{I}\otimes\cdots\otimes\boldsymbol{I}\otimes\boldsymbol{U}_t\otimes\boldsymbol{I}\otimes\cdots\otimes\boldsymbol{I}$ is described by

$$(\boldsymbol{I}\otimes\cdots\otimes\boldsymbol{I}\otimes\boldsymbol{U}_t\otimes\boldsymbol{I}\otimes\cdots\otimes\boldsymbol{I})\,|\psi\rangle$$

$$= \sum_{\sigma_1,\ldots,\sigma_n,\sigma_1',\ldots,\sigma_n' b_1,\ldots,b_{n-1}} \left(\boldsymbol{I}^{\sigma_1\sigma_1'}\ldots\boldsymbol{I}^{\sigma_{t-1}\sigma_{t-1}'}\boldsymbol{U}^{\sigma_t\sigma_t'}\boldsymbol{I}^{\sigma_{t+1}\sigma_{t+1}'}\ldots I^{\sigma_n\sigma_n'}\right) T^{(1)}{}^{\sigma_1}_{b_1} T^{(2)}{}^{\sigma_2}_{b_1,b_2}\cdots T^{(n-1)}{}^{\sigma_{n-1}}_{b_{n-2},b_{n-1}} T^{(n)}{}^{\sigma_n}_{b_{n-1}}|\sigma_1\cdots\sigma_n\rangle,$$



Applying a single qubit gate (using MPS)

$$(\boldsymbol{I} \otimes \cdots \otimes \boldsymbol{I} \otimes \boldsymbol{U}_t \otimes \boldsymbol{I} \otimes \cdots \otimes \boldsymbol{I})\,|\psi\rangle$$

$$= \sum_{\substack{\sigma_1,\ldots,\sigma_n,b_1,\ldots,b_{n-1}\\ \sigma'_1,\ldots,\sigma'_n}} \left(\boldsymbol{I}^{\sigma_1\sigma'_1} \ldots \boldsymbol{I}^{\sigma_{t-1}\sigma'_{t-1}} \boldsymbol{U}^{\sigma_t\sigma'_t} \boldsymbol{I}^{\sigma_{t+1}\sigma'_{t+1}} \ldots I^{\sigma_n\sigma'_n}\right) T^{(1)}{}^{\sigma_1}_{b_1} T^{(2)}{}^{\sigma_2}_{b_1,b_2} \cdots T^{(n-1)}{}^{\sigma_{n-1}}_{b_{n-2},b_{n-1}} T^{(n)}{}^{\sigma_n}_{b_{n-1}} \,|\sigma_1 \cdots \sigma_n\rangle,$$

$$= \sum_{\substack{\sigma'_1,\ldots,\sigma'_n\\ b_1,\ldots,b_{n-1}}} T^{(1)}{}^{\sigma'_1}_{b_1} \cdots T^{(t-1)}{}^{\sigma'_{t-1}}_{b_{t-2},b_{t-1}} \underbrace{\left(\sum_{\sigma_t=0}^{1} \boldsymbol{U}^{\sigma_t,\sigma'_t}\, \boldsymbol{T}^{(t)}{}^{\sigma_t}_{b_{t-1},b_t} |\sigma_t\rangle\right)} T^{(t+1)}{}^{\sigma'_{t+1}}_{b_t,b_{t+1}} \cdots T^{(n)}{}^{\sigma'_n}_{b_{n-1}} |\sigma'_1 \cdots \sigma'_{t-1}\sigma'_{t+1} \cdots \sigma'_n\rangle$$

We only need to update these terms.

d: the dimension of σ_i

m: the dimension of b_i

The computational complexity is $O(d^2m^2) \ll O(2^n)$.

State vector simulation (d=2)

An example of applying a single qubit gate (using MPS)

For 2 qubit system, let us consider applying $\boldsymbol{U} = \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix}$ on the 1st qubit,

$$(\boldsymbol{U} \otimes \boldsymbol{I})|00\rangle = \boldsymbol{U}|0\rangle \otimes \boldsymbol{I}|0\rangle = (u_{00}|0\rangle + u_{10}|1\rangle) \otimes (1|0\rangle + 0|1\rangle)$$

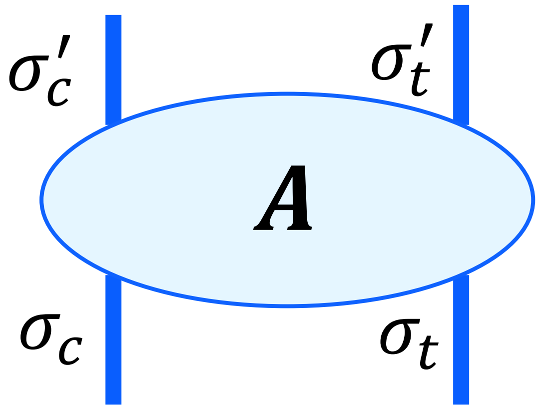
$$= \sum_{\sigma_1=0}^{1} \sum_{\sigma_2=0}^{1} \sum_{b=0}^{0} A_b^{\sigma_1} B_b^{\sigma_2} \, |\sigma_1 \sigma_2\rangle \, , \text{where } A_0^0 = u_{00}, A_0^1 = u_{10}, B_0^0 = 1, B_0^1 = 0$$

Applying CNOT gate (using MPS)

A CNOT gate is also represented by

$$(\sigma_c, \sigma_t) = (0,0), (0,1), (1,0), (1,1)$$

$$A^{\sigma_c,\sigma_t,\sigma_c',\sigma_t'} := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{array}{c} (\sigma_c', \sigma_t') \\ (0,0) \\ (0,1) \\ (1,0) \\ (1,1) \end{array}$$



Applying a CNOT gate (using MPS)

A quantum state applied by CNOT gate with cth control qubit and tth target qubit is updated by

$$(I_1 \otimes \cdots \otimes I_{c-1} \otimes |0\rangle\langle 0| \otimes I_{c+1} \otimes \cdots \otimes I_{t-1} \otimes I_t \otimes I_{t+1} \otimes \cdots \otimes I_n$$
$$+ I_1 \otimes \cdots \otimes I_{c-1} \otimes |1\rangle\langle 1| \otimes I_{c+1} \otimes \cdots \otimes I_{t-1} \otimes X_t \otimes I_{t+1} \otimes \cdots \otimes I_n)|\psi\rangle$$

$$= \sum_{\substack{\sigma'_1,\ldots,\sigma'_n \\ b_1,\ldots,b_{n-1}}} \underbrace{\left(\sum_{\sigma_c,\sigma_t} A^{\sigma_c,\sigma_t,\sigma'_c,\sigma'_t} T^{(c)}{}^{\sigma_c}_{b_{c-1},b_c} T^{(t)}{}^{\sigma_t}_{b_{t-1},b_t} |\sigma_c \sigma_t\rangle \right)}$$
$$T^{(1)}{}^{\sigma'_1}_{b_1} T^{(2)}{}^{\sigma'_2}_{b_1,b_2} \cdots T^{(c-1)}{}^{\sigma'_{c-1}}_{b_{c-2},b_{c-1}} T^{(c+1)}{}^{\sigma'_{c+1}}_{b_c,b_{c+1}} \cdots T^{(t-1)}{}^{\sigma'_{t-1}}_{b_{t-2},b_{t-1}} T^{(t+1)}{}^{\sigma'_{t+1}}_{b_t,b_{t+1}} \cdots T^{(n-1)}{}^{\sigma'_{n-1}}_{b_{n-2},b_{n-1}} T^{(n)}{}^{\sigma'_n}_{b_{n-1}} |\sigma'_1 \cdots \sigma'_{c-1} \sigma'_{c+1} \cdots \sigma'_{t-1} \sigma'_{t+1} \cdots \sigma'_n\rangle$$

We only need to update these terms.

The computational complexity is $O(d^4 m^4) \ll O(2^n)$.

State vector simulation (d=2)

Decomposition

A quantum state is represented by the following formula:

$$|\psi\rangle = \sum_{\sigma_1,\cdots,\sigma_n} c_{\sigma_1,\cdots,\sigma_n} |\sigma_1\rangle \cdots |\sigma_n\rangle\,.$$

We would like to turn it into MPS form.

First, we use singular value decomposition(SVD) to decompose the system of first qubit and the other part.

To this end, we make a matrix consisting of the coefficients $c_{\sigma_1,\cdots,\sigma_n}$, and apply SVD to this matrix:

$$\begin{pmatrix} c_{0,0,\cdots,0} & \cdots & c_{0,d-1,\cdots,d-1} \\ \vdots & \ddots & \vdots \\ c_{d-1,0,\cdots,0} & \cdots & c_{d-1,d-1,\cdots,d-1} \end{pmatrix} = UDV^\dagger$$

$$= \begin{pmatrix} u_{0,0} & \cdots & u_{0,r-1} \\ \vdots & \ddots & \vdots \\ u_{d-1,0} & \cdots & u_{d-1,r-1} \end{pmatrix} \begin{pmatrix} \lambda_0 & & \\ & \ddots & \\ & & \lambda_{r-1} \end{pmatrix} \begin{pmatrix} v_{0,0,\cdots,0} & \cdots & v_{0,d-1,\cdots,d-1} \\ \vdots & \ddots & \vdots \\ v_{r-1,0,\cdots,0} & \cdots & v_{r-1,d-1,\cdots,d-1} \end{pmatrix}$$

$$c_{\sigma_1,\cdots,\sigma_n} = \sum_{k=0}^{r-1} u_{\sigma_1,k}\, \lambda_k v_{k,\sigma_2,\cdots,\sigma_n}, \forall\, \sigma_2, \ldots \sigma_n \in \{0, \ldots, d-1\},$$

where r is the rank of diagonal matrix D.

Decomposition

$$c_{\sigma_1,\cdots,\sigma_n} = \sum_{k=0}^{r-1} u_{\sigma_1,k}\, \lambda_k v_{k,\sigma_2,\cdots,\sigma_n}, \forall\ \sigma_2, \dots \sigma_n \in \{0, \dots, d-1\},$$

We assign this expression to the initial formula:

$$|\psi\rangle = \sum_{k=0}^{r-1} \sum_{\sigma_1=0}^{d-1} u_{\sigma_1,k}\, \lambda_k |\sigma_1\rangle \sum_{\sigma_2,\cdots,\sigma_n} v_{k,\sigma_2,\cdots,\sigma_n} |\sigma_2\rangle \cdots |\sigma_n\rangle\,.$$

We repeatedly apply SVD to $v_{k,\sigma_2,\cdots,\sigma_n}$, we obtain

$$|\psi\rangle = \sum_{\substack{\sigma_1,\cdots,\sigma_n \\ k_1,\dots,k_{n-1}}} u^{(1)}_{\sigma_1,k_1} \lambda^{(1)}_{k_1} u^{(2)}_{k_1,\sigma_2,k_2} \lambda^{(2)}_{k_2} \cdots \lambda^{(n-1)}_{k_{n-1}} u^{(n)}_{\sigma_n,k_{n-1}} |\sigma_1\rangle|\sigma_2\rangle \cdots |\sigma_n\rangle\,.$$

We can rewrite a quantum state into MPS form using SVD.

The decomposition in applying a CNOT gate

$$\sum_{\sigma_c=0}^{d-1}\sum_{\sigma_t=0}^{d-1} A^{\sigma_c,\sigma_t,\sigma_c',\sigma_t'}\, T^{(c)}{}^{\sigma_c}_{b_{c-1},b_c} T^{(t)}{}^{\sigma_t}_{b_{t-1},b_t} \left|\sigma_c\sigma_t\right\rangle$$

To decompose the above formula, we need to apply SVD to a matrix consisting of $d \times d$ elements.
So, the cost of SVD is $O(d^3m^4)$, which is smaller than $O(d^4m^4)$.

An example of applying a CNOT gate

By applying CNOT gate with 1 st qubit as control qubit and 2 nd qubit as target qubit to $|+\rangle \otimes |0\rangle$, the quantum state is updated as follows:

$$(|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X)\left(\left(\frac{1}{\sqrt{2}}|0\rangle + \frac{1}{\sqrt{2}}|1\rangle\right) \otimes |0\rangle\right) = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$$

From the following equation

$$\begin{pmatrix} c_{00} & c_{01} \\ c_{10} & c_{11} \end{pmatrix} = \begin{pmatrix} 1/\sqrt{2} & 0 \\ 0 & 1/\sqrt{2} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1/\sqrt{2} & 0 \\ 0 & 1/\sqrt{2} \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

We can rewrite $\frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$ as

$$\begin{pmatrix} 1 & 0 \end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 0 \end{pmatrix}|00\rangle + \begin{pmatrix} 1 & 0 \end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix} 0 \\ 1 \end{pmatrix}|01\rangle + \begin{pmatrix} 0 & 1 \end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix} 1 \\ 0 \end{pmatrix}|10\rangle + \begin{pmatrix} 0 & 1 \end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix} 0 \\ 1 \end{pmatrix}|11\rangle$$

$$= \left(\begin{pmatrix} 1 & 0 \end{pmatrix}\frac{1}{\sqrt{2}}|0\rangle + \begin{pmatrix} 0 & 1 \end{pmatrix}\frac{1}{\sqrt{2}}|1\rangle\right) \otimes \left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}|0\rangle + \begin{pmatrix} 0 \\ 1 \end{pmatrix}|1\rangle\right)$$

the bond dimension is 2

Remind
$\sum_{\sigma_1=0}^{1}\sum_{\sigma_2=0}^{1}\sum_{b_1=0}^{1} A_{b_1}^{\sigma_1} B_{b_1}^{\sigma_2} |\sigma_1\sigma_2\rangle$

An example of applying a CNOT gate

Here, we focus on 2 nd qubit and 3 rd qubit of the quantum state

$$\left(\begin{pmatrix}1 & 0\end{pmatrix}\frac{1}{\sqrt{2}}|0\rangle + \begin{pmatrix}0 & 1\end{pmatrix}\frac{1}{\sqrt{2}}|1\rangle\right) \otimes \left(\begin{pmatrix}1\\0\end{pmatrix}|0\rangle + \begin{pmatrix}0\\1\end{pmatrix}|1\rangle\right) \otimes |0\rangle.$$

We apply a CNOT gate with 2 nd qubit as control qubit and 3 rd qubit as target qubit to the quantum state. It is updated as follows:

$$(|0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X)\left(\begin{pmatrix}1\\0\end{pmatrix}|00\rangle + \begin{pmatrix}0\\1\end{pmatrix}|10\rangle\right) = \begin{pmatrix}1\\0\end{pmatrix}|00\rangle + \begin{pmatrix}0\\1\end{pmatrix}|11\rangle$$

From the following svd results:

for $b_1 = 0$, $\begin{pmatrix}c_{00} & c_{01}\\ c_{10} & c_{11}\end{pmatrix} = \begin{pmatrix}1 & 0\\0 & 0\end{pmatrix} = \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\begin{pmatrix}1 & 0\\0 & 0\end{pmatrix}\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}$,

for $b_1 = 1$, $\begin{pmatrix}c_{00} & c_{01}\\ c_{10} & c_{11}\end{pmatrix} = \begin{pmatrix}0 & 0\\0 & 1\end{pmatrix} = \begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}\begin{pmatrix}0 & 0\\0 & 1\end{pmatrix}\begin{pmatrix}1 & 0\\0 & 1\end{pmatrix}$,

we can write $|00\rangle$ and $|11\rangle$ as

Remind
$\sum_{\sigma_1=0}^{1}\sum_{\sigma_2=0}^{1}\sum_{b_1=0}^{1} A_{b_1}^{\sigma_1} B_{b_1}^{\sigma_2}|\sigma_1\sigma_2\rangle$

$$|00\rangle = \left(\begin{pmatrix}1 & 0\end{pmatrix}|0\rangle + \begin{pmatrix}0 & 0\end{pmatrix}|1\rangle\right) \otimes \left(\begin{pmatrix}1\\0\end{pmatrix}|0\rangle + \begin{pmatrix}0\\1\end{pmatrix}|1\rangle\right)$$

when $b_1 = 0$,

$$|11\rangle = \left(\begin{pmatrix}0 & 0\end{pmatrix}|0\rangle + \begin{pmatrix}0 & 1\end{pmatrix}|1\rangle\right) \otimes \left(\begin{pmatrix}1\\0\end{pmatrix}|0\rangle + \begin{pmatrix}0\\1\end{pmatrix}|1\rangle\right)$$

when $b_1 = 1$,

An example of applying a CNOT gate

So, we can rewrite $\begin{pmatrix}1\\0\end{pmatrix}|00\rangle + \begin{pmatrix}0\\1\end{pmatrix}|11\rangle$ as

$$\left(\begin{pmatrix}1 & 0\\0 & 0\end{pmatrix}|0\rangle + \begin{pmatrix}0 & 0\\0 & 1\end{pmatrix}|1\rangle\right) \otimes \left(\begin{pmatrix}1\\0\end{pmatrix}|0\rangle + \begin{pmatrix}0\\1\end{pmatrix}|1\rangle\right).$$

As a result, we obtain

$$\frac{1}{\sqrt{2}}|000\rangle + \frac{1}{\sqrt{2}}|111\rangle$$
$$= \left(\begin{pmatrix}1 & 0\end{pmatrix}\frac{1}{\sqrt{2}}|0\rangle + \begin{pmatrix}0 & 1\end{pmatrix}\frac{1}{\sqrt{2}}|1\rangle\right) \otimes \left(\begin{pmatrix}1 & 0\\0 & 0\end{pmatrix}|0\rangle + \begin{pmatrix}0 & 0\\0 & 1\end{pmatrix}|1\rangle\right) \otimes \left(\begin{pmatrix}1\\0\end{pmatrix}|0\rangle + \begin{pmatrix}0\\1\end{pmatrix}|1\rangle\right).$$

Inner Product between two MPSs

$$\langle\psi'|\psi\rangle = \left(\sum_{\substack{\sigma_1'\cdots\sigma_n' \\ b_1'\cdots b_{n-1}'}} \langle\sigma_1'\cdots\sigma_n'| T'^{(n)}{}^{\sigma_n'\dagger}_{b_{n-1}'} T'^{(n-1)}{}^{\sigma_{n-1}'\dagger}_{b_{n-1}'b_{n-2}'} \cdots T'^{(2)}{}^{\sigma_2'\dagger}_{b_2'b_1'} T'^{(1)}{}^{\sigma_1'\dagger}_{b_1'}\right)$$

$$\left(\sum_{\substack{\sigma_1\cdots\sigma_n \\ b_1\cdots b_{n-1}}} T^{(1)}{}^{\sigma_1}_{b_1} T^{(2)}{}^{\sigma_2}_{b_1b_2} \cdots T^{(n-1)}{}^{\sigma_{n-1}}_{b_{n-2}b_{n-1}} T^{(n)}{}^{\sigma_n}_{b_{n-1}} |\sigma_1\cdots\sigma_n\rangle\right)$$

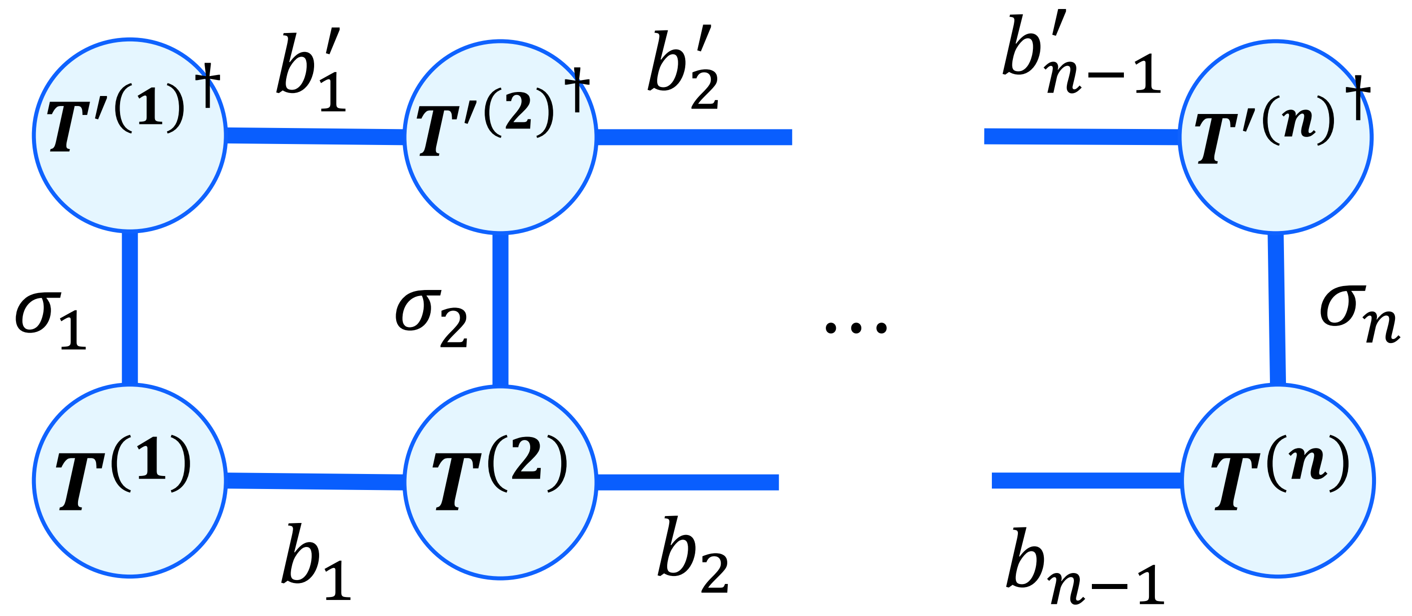
$$= \sum_{\substack{\sigma_1'\cdots\sigma_n' \\ b_1'\cdots b_{n-1}'}} \sum_{\substack{\sigma_1\cdots\sigma_n \\ b_1\cdots b_{n-1}}} T'^{(n)}{}^{\sigma_n'\dagger}_{b_{n-1}'} T'^{(n-1)}{}^{\sigma_{n-1}'\dagger}_{b_{n-1}'b_{n-2}'} \cdots T'^{(2)}{}^{\sigma_2'\dagger}_{b_2'b_1'} T'^{(1)}{}^{\sigma_1'\dagger}_{b_1'} T^{(1)}{}^{\sigma_1}_{b_1} T^{(2)}{}^{\sigma_2}_{b_1b_2} \cdots T^{(n-1)}{}^{\sigma_{n-1}}_{b_{n-2}b_{n-1}} T^{(n)}{}^{\sigma_n}_{b_{n-1}} \delta_{\sigma_1'\sigma_1} \cdots \delta_{\sigma_n'\sigma_n}$$

$$= \sum_{b_1'\cdots b_{n-1}'} \sum_{\substack{\sigma_1\cdots\sigma_n \\ b_1\cdots b_{n-1}}} T'^{(n)}{}^{\sigma_n'\dagger}_{b_{n-1}'} T'^{(n-1)}{}^{\sigma_{n-1}'\dagger}_{b_{n-1}'b_{n-2}'} \cdots T'^{(2)}{}^{\sigma_2'\dagger}_{b_2'b_1'} T'^{(1)}{}^{\sigma_1'\dagger}_{b_1'} T^{(1)}{}^{\sigma_1}_{b_1} T^{(2)}{}^{\sigma_2}_{b_1b_2} \cdots T^{(n-1)}{}^{\sigma_{n-1}}_{b_{n-2}b_{n-1}} T^{(n)}{}^{\sigma_n}_{b_{n-1}}$$

We will perform the remaining calculations by using diagrams.

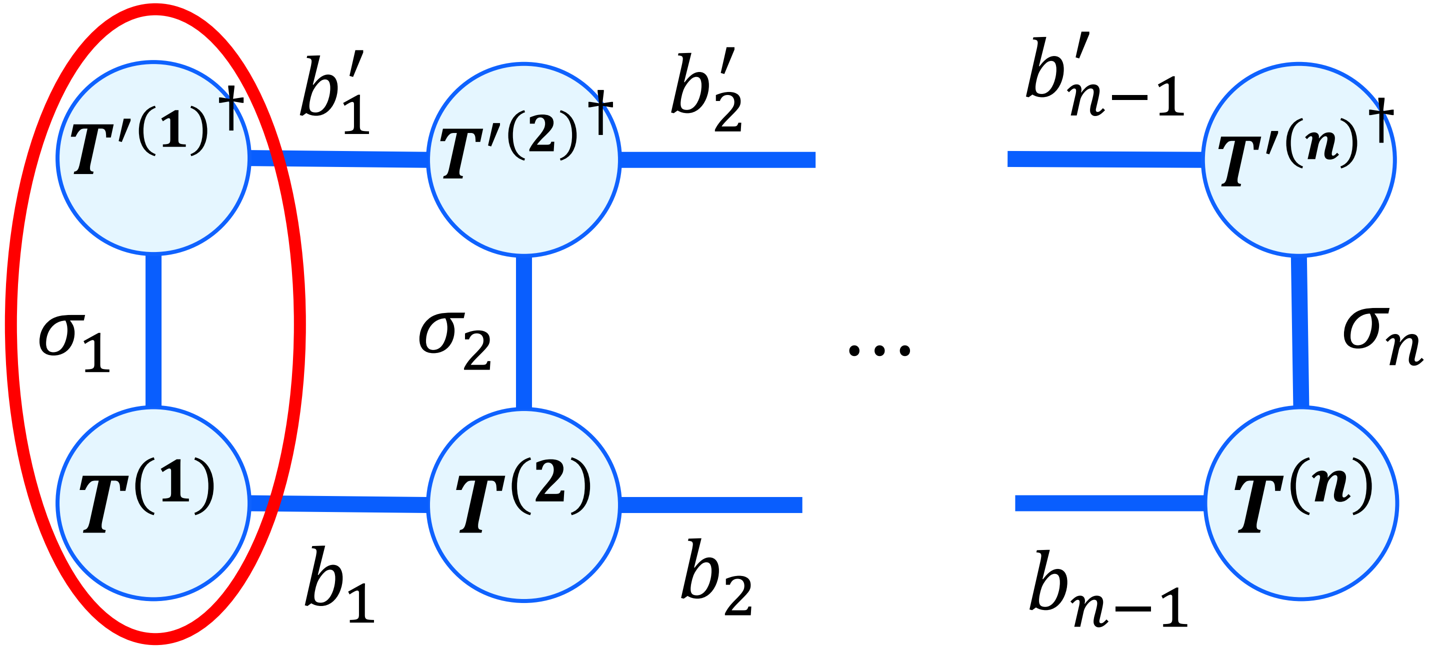
An example of calculating Inner Product using MPS

$$\sum_{b'_1\cdots b'_{n-1}} \sum_{\substack{\sigma_1\cdots\sigma_n \\ b_1\cdots b_{n-1}}} T'^{(n)}{}^{\sigma'_n \dagger}_{b'_{n-1}} T'^{(n-1)}{}^{\sigma'_{n-1}\dagger}_{b'_{n-1}b'_{n-2}} \cdots T'^{(2)}{}^{\sigma'_2\dagger}_{b'_2 b'_1} T'^{(1)}{}^{\sigma'_1\dagger}_{b'_1} T^{(1)}{}^{\sigma_1}_{b_1} T^{(2)}{}^{\sigma_2}_{b_1 b_2} \cdots T^{(n-1)}{}^{\sigma_{n-1}}_{b_{n-2}b_{n-1}} T^{(n)}{}^{\sigma_n}_{b_{n-1}}$$



An example of calculating Inner Product using MPS

$$\sum_{b_1'\cdots b_{n-1}'} \sum_{\substack{\sigma_1\cdots\sigma_n \\ b_1\cdots b_{n-1}}} T'^{(n)}{}^{\sigma_n'\dagger}_{b_{n-1}'} T'^{(n-1)}{}^{\sigma_{n-1}'\dagger}_{b_{n-1}'b_{n-2}'} \cdots T'^{(2)}{}^{\sigma_2'\dagger}_{b_2'b_1'} T'^{(1)}{}^{\sigma_1'\dagger}_{b_1'} T^{(1)}{}^{\sigma_1}_{b_1} T^{(2)}{}^{\sigma_2}_{b_1 b_2} \cdots T^{(n-1)}{}^{\sigma_{n-1}}_{b_{n-2}b_{n-1}} T^{(n)}{}^{\sigma_n}_{b_{n-1}}$$

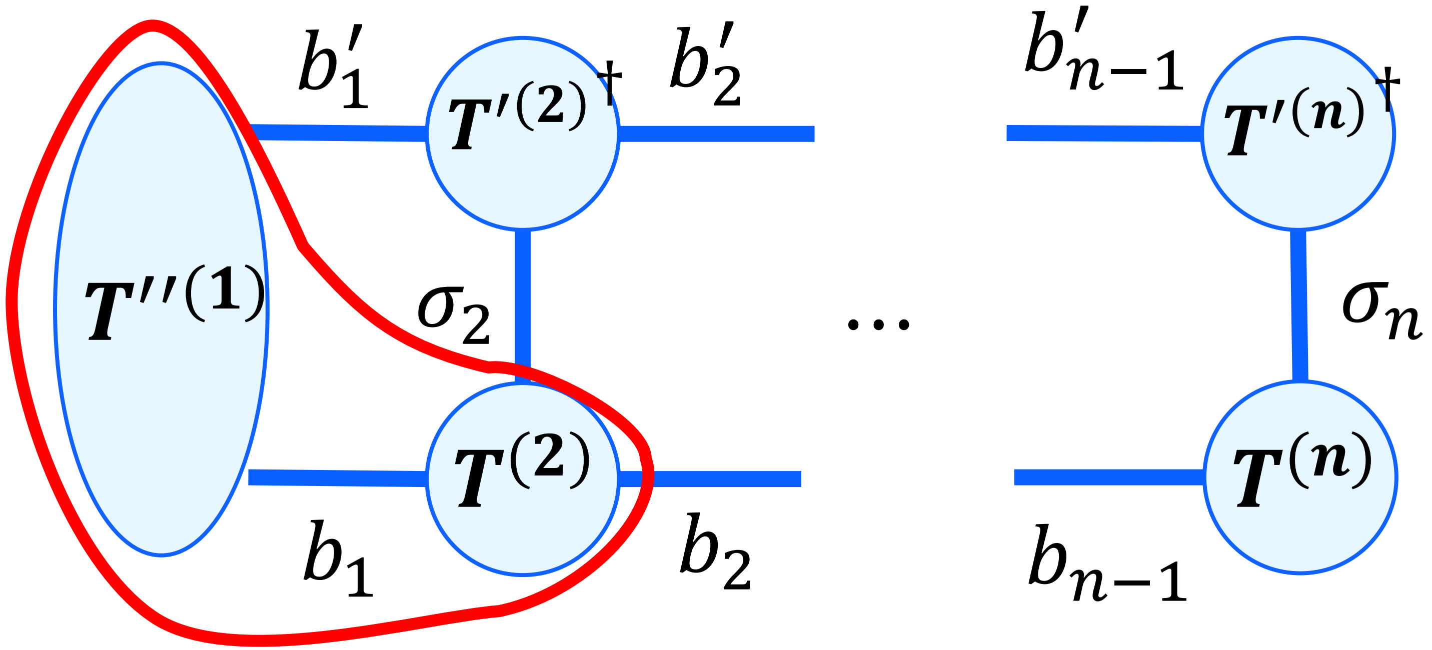


$$\sum_{\sigma_1=1}^{d} T'^{(1)\sigma_1\,\dagger}_{b_1'} T^{(1)\sigma_1}_{b_1} =: T''^{(1)}_{b_1 b_1'}$$

cost: dm^2

An example of calculating Inner Product using MPS

$$\sum_{b_1'\cdots b_{n-1}'} \sum_{\substack{\sigma_1\cdots\sigma_n \\ b_1\cdots b_{n-1}}} T'^{(n)}{}^{\sigma_n'\dagger}_{b_{n-1}'} T'^{(n-1)}{}^{\sigma_{n-1}'\dagger}_{b_{n-1}'b_{n-2}'} \cdots T'^{(2)}{}^{\sigma_2'\dagger}_{b_2'b_1'} T''^{(1)}_{b_1'b_1} T^{(2)}{}^{\sigma_2}_{b_1b_2} \cdots T^{(n-1)}{}^{\sigma_{n-1}}_{b_{n-2}b_{n-1}} T^{(n)}{}^{\sigma_n}_{b_{n-1}}$$

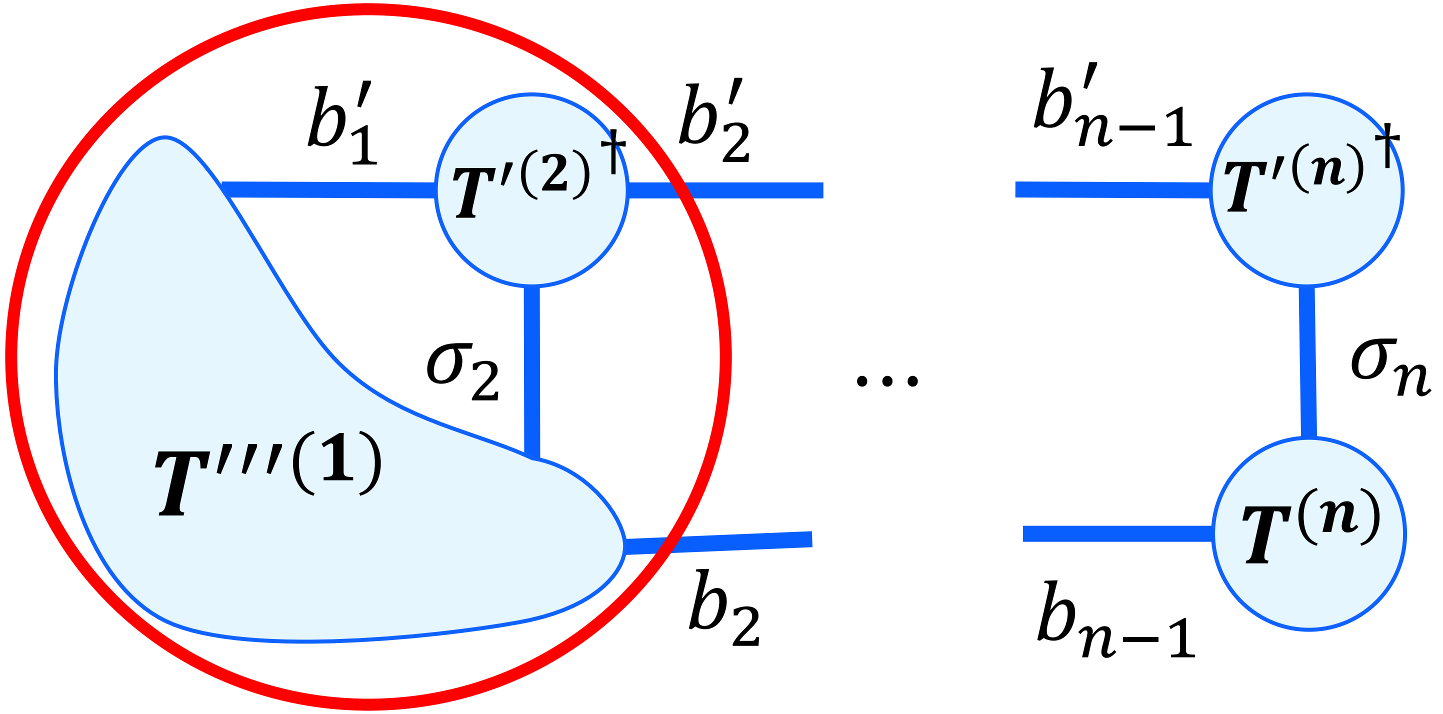


$$\sum_{b_1=1}^{m} T''^{(1)}_{b_1b_1'} T^{(2)\sigma_2}_{b_1b_2} =: T'''^{(1)\,\sigma_2}_{b_1'b_2}$$

cost: dm^3

An example of calculating Inner Product using MPS

$$\sum_{b'_1\cdots b'_{n-1}} \sum_{\substack{\sigma_1\cdots\sigma_n \\ b_1\cdots b_{n-1}}} T'^{(n)}{}^{\sigma'_n\dagger}_{b'_{n-1}} T'^{(n-1)}{}^{\sigma'_{n-1}\dagger}_{b'_{n-1}b'_{n-2}} \cdots T'^{(2)}{}^{\sigma'_2\dagger}_{b'_2 b'_1} T'''^{(1)}{}^{\sigma_2}_{b'_1 b_2} T^{(3)}{}^{\sigma_3}_{b_2 b_3} \cdots T^{(n-1)}{}^{\sigma_{n-1}}_{b_{n-2}b_{n-1}} T^{(n)}{}^{\sigma_n}_{b_{n-1}}$$



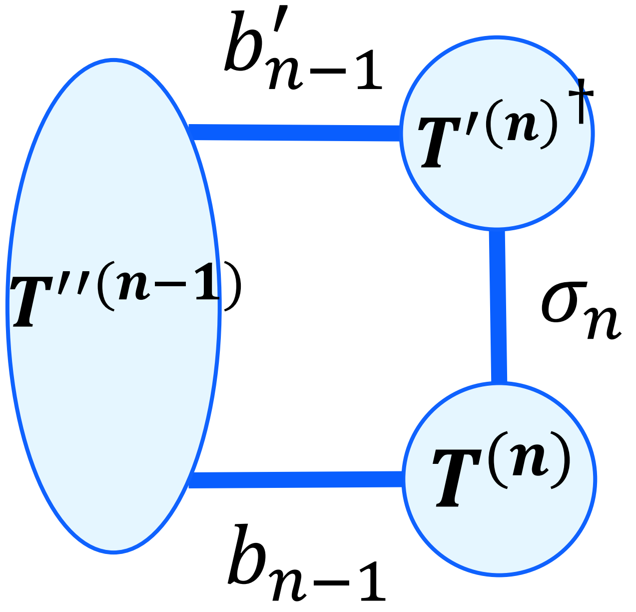
$$\sum_{\sigma_2=1}^{d} \sum_{b'_1=1}^{m} T'^{(2)\sigma_2\dagger}_{b'_1 b'_2} T'''^{(1)\,\sigma_2}_{b'_1 b_2} := T'^{(2)}_{b_2 b'_2}$$

cost: dm^3

An example of calculating Inner Product using MPS

$$\sum_{\substack{\sigma_n \\ b_{n-1}, b'_{n-1}}} T'^{(n)}{}^{\sigma'_n \dagger}_{b'_{n-1}} T''^{(n-1)}{}_{b'_{n-1} b_{n-1}} T^{(n)}{}^{\sigma_n}_{b_{n-1}}$$

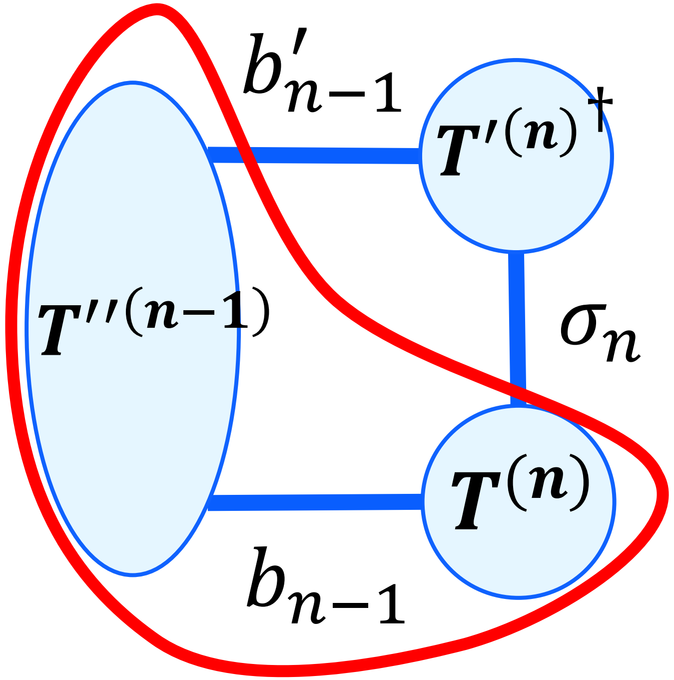
Repeat the same operations

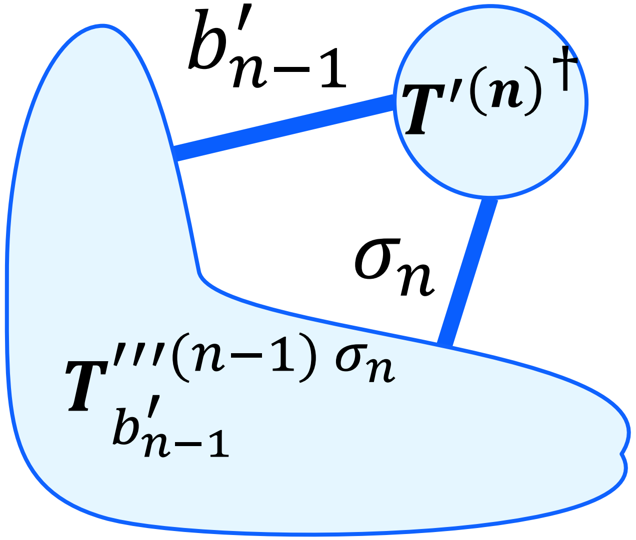


cost: $2dm^3(n-3)$

An example of calculating Inner Product using MPS

$$\sum_{\sigma_3,b_2,b_2'} C'^{\sigma_3\dagger}_{b_2'} B''_{b_2 b_2'} C^{\sigma_3}_{b_2}$$





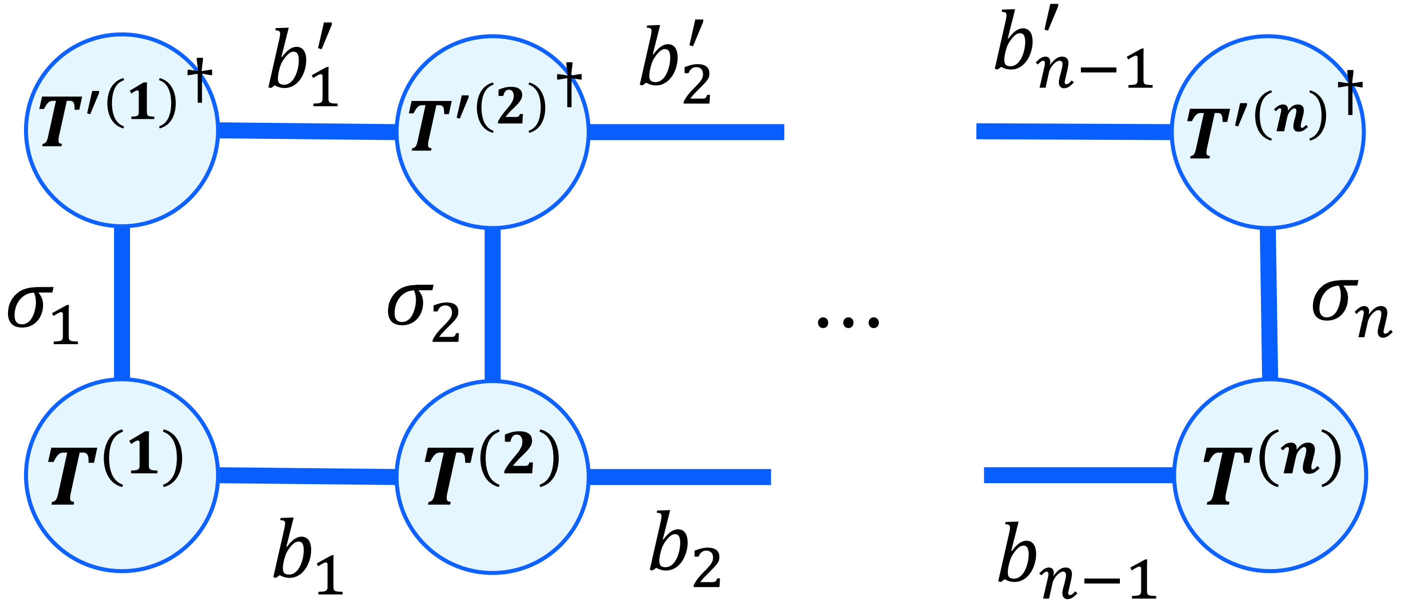
$$\sum_{b_2=1}^{m} \boldsymbol{T}''^{(n-1)}_{b_{n-1},b'_{n-1}} \boldsymbol{T}^{(n)\sigma_n}_{b_{n-1}} =: T'''^{(n-1)\sigma_n}_{b'_{n-1}}$$

cost: dm^2

$$\sum_{\sigma_3=1}^{d} \sum_{b_2'=1}^{m} \boldsymbol{T}'^{(n)\dagger} \boldsymbol{T}'''^{(n-1)\,\sigma_n}_{b'_{n-1}}$$

cost: dm

An example of calculating Inner Product using MPS



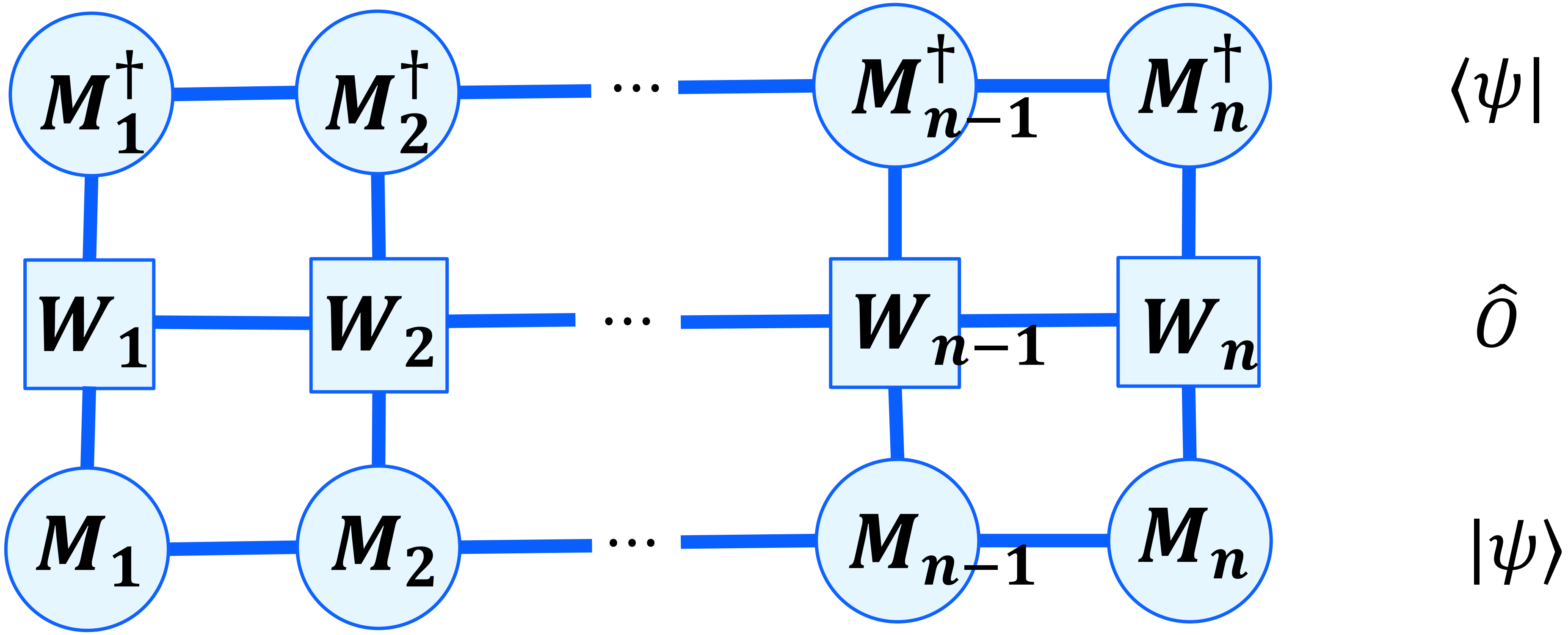
The total cost:

$$O(ndm^3) \ll O(2^n)$$

where n is the number of qubits

MPS allows us to calculate inner products more efficiently than state vector simulation.

The expectation values of MPS



We can calculate expectation values similar to calculating inner-products.
Computational complexity is

$$O\left(n(m^3kd + m^2k^2d^2)\right) \ll O(2^n).$$

MPS also efficiently calculate expectation values.

k is a typical dimension of Matrix Product Operator

Clifford Circuit

Pauli Operator and Pauli Group

– Pauli Operator

$$\boldsymbol{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \boldsymbol{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \; \boldsymbol{Y} = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \; \boldsymbol{Z} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Pauli group is defined by

$$\mathcal{P}_n = \left\{ c\, \boldsymbol{P}_1 \otimes \cdots \otimes \boldsymbol{P}_j \otimes \cdots \otimes \boldsymbol{P}_n \middle|\, c = \pm 1, \pm i \,;\, \boldsymbol{P}_j = \boldsymbol{I}, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z} \right\}.$$

Examples:

$$i\mathbf{Y} \otimes \mathbf{X} \in \mathcal{P}_2, -\boldsymbol{X} \otimes \boldsymbol{Y} \otimes \boldsymbol{Z} \in \mathcal{P}_3$$

Clifford Gates and Clifford Group

Clifford Gates

$$\mathbf{H} = \begin{pmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \end{pmatrix} \quad \mathbf{S} = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix} \qquad \mathbf{CX} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Clifford gates are elements of the following Clifford group $\mathcal{C}_n$

$$\mathcal{C}_n = \{\boldsymbol{V} \in U_n | V Q_n V^\dagger = Q_n', \forall Q_n \in \mathcal{P}_n, Q_n' \in \mathcal{P}_n\}$$

for n-qubit Unitary U_n, the elements of Pauli group Q_n'
Clifford gates map the elements of Pauli group to itself. For example,

$$\boldsymbol{HXH} = \boldsymbol{Z}, \boldsymbol{SYS}^\dagger = -\boldsymbol{X}, \boldsymbol{SZS}^\dagger = \boldsymbol{Z},$$
$$(\boldsymbol{CX})(\boldsymbol{I} \otimes \boldsymbol{Z})(\boldsymbol{CX}) = \boldsymbol{Z} \otimes \boldsymbol{Z}.$$

Stabilizer state

A stabilizer state is used to efficiently simulate the Clifford circuit.
A stabilizer state is a quantum state with +1 eigen value for a Pauli term.

$$X|+\rangle = |+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}, Y|i\rangle = |i\rangle = \frac{|0\rangle + i|1\rangle}{\sqrt{2}}, Z|0\rangle = |0\rangle$$

Pauli X, Y, and Z stabilize $|+\rangle, |i\rangle$, and $|0\rangle$, respectively.

A stabilizer state can be uniquely determined by n stabilizer operators.

	$+XX$	$-XX$
$+\boldsymbol{ZZ}$	$\frac{\vert 00\rangle + \vert 11\rangle}{\sqrt{2}}$	$\frac{\vert 00\rangle - \vert 11\rangle}{\sqrt{2}}$
$-\boldsymbol{ZZ}$	$\frac{\vert 01\rangle + \vert 10\rangle}{\sqrt{2}}$	$\frac{\vert 01\rangle - \vert 10\rangle}{\sqrt{2}}$

Binary representation of Pauli terms

We introduce a binary representation of Pauli terms
to efficiently simulate Clifford circuit.

We transform $\boldsymbol{P}_1 \otimes \cdots \otimes \boldsymbol{P}_i \otimes \cdots \otimes \boldsymbol{P}_n$ into $x_1 \cdots x_i \cdots x_n z_1 \cdots z_i \cdots z_n$,
where $\boldsymbol{P}_i \in \{\boldsymbol{I}, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z}\}$, $\mathrm{x_i} \in \{0,1\}$, $z_i \in \{0,1\}$ for $\mathrm{i} = 1, \ldots, n$.

$$\begin{aligned}
\boldsymbol{P}_i = \boldsymbol{I} &\Rightarrow x_i = 0, z_i = 0 \\
\boldsymbol{P}_i = \boldsymbol{X} &\Rightarrow x_i = 1, z_i = 0 \\
\boldsymbol{P}_i = \boldsymbol{Y} &\Rightarrow x_i = 1, z_i = 1 \\
\boldsymbol{P}_i = \boldsymbol{Z} &\Rightarrow x_i = 0, z_i = 1
\end{aligned}$$

For example,

	$\boldsymbol{x_1 x_2 x_3 x_4}$	$\boldsymbol{z_1 z_2 z_3 z_4}$
$\boldsymbol{X_1 Y_2 Z_3 I_4}$	1100	0110

For simplicity, we ignore the phase.

Applying Clifford gates using binary representation

Applying H gate on t-th qubit

$x_{11}x_{12}\cdots x_{1t}\cdots x_{1n}$	$z_{11}z_{12}\cdots z_{1t}\cdots z_{1n}$
$\vdots$	$\vdots$
$x_{i1}x_{i2}\cdots x_{it}\cdots x_{in}$	$z_{i1}z_{i2}\cdots z_{it}\cdots z_{in}$
$\vdots$	$\vdots$
$x_{n1}x_{n2}\cdots x_{nt}\cdots x_{nn}$	$z_{n1}z_{n2}\cdots z_{nt}\cdots z_{nn}$

swap

$$x_{it}^{new} = z_{it}, \mathrm{z}_{\mathrm{it}}^{\mathrm{new}} = \mathrm{x}_{\mathrm{it}}, \text{ for } i = 1, \dots, n$$

Applying S gate on t-th qubit

$x_{11}x_{12}\cdots x_{1t}\cdots x_{1n}$	$z_{11}z_{12}\cdots z_{1t}\cdots z_{1n}$
$\vdots$	$\vdots$
$x_{i1}x_{i2}\cdots x_{it}\cdots x_{in}$	$z_{i1}z_{i2}\cdots z_{it}\cdots z_{in}$
$\vdots$	$\vdots$
$x_{n1}x_{n2}\cdots x_{nt}\cdots x_{nn}$	$z_{n1}z_{n2}\cdots z_{nt}\cdots z_{nn}$

Apply exclusive-OR

$$z_{it}^{new} = z_{it} \oplus x_{it}, \text{ for } i = 1, \dots, n$$

We can apply ***H*** or ***S*** gate by updating the binary table in $O(n)$.

Example of applying H or S gate

$|00\rangle$ is stabilized by $+IZ$ and $+ZI$.

	x_1x_2	z_1z_2
Z_1I_2	00	10
I_1Z_2	00	01

swap

Here, we act *H* gate on 1st qubit.

X_1I_2 and I_1Z_2 stabilize $|0\rangle \otimes \frac{|0\rangle+|1\rangle}{\sqrt{2}}$.

2nd 1st

	x_1x_2	z_1z_2
X_1I_2	10	00
I_1Z_2	00	01

$\oplus$

Next, we act *S* gate on 1st qubit.

Y_1I_2 and I_1Z_2 stabilize $|0\rangle \otimes \frac{|0\rangle+i|1\rangle}{\sqrt{2}}$.

2nd 1st

	x_1x_2	z_1z_2
Y_1I_2	10	10
I_1Z_2	00	01

Applying Clifford gates using binary representation

Applying CNOT gate on c-th control and t-th target qubit

$x_{11} \cdots x_{1c} \cdots x_{1t} \cdots x_{1n}$	$z_{11} \cdots z_{1c} \cdots z_{1t} \cdots z_{1n}$
$\vdots$	$\vdots$
$x_{i1} \cdots x_{ic} \cdots x_{it} \cdots x_{in}$	$z_{i1} \cdots z_{ic} \cdots z_{it} \cdots z_{in}$
$\vdots$	$\vdots$
$x_{n1} \cdots x_{nc} \cdots x_{nt} \cdots x_{nn}$	$z_{n1} \cdots z_{nc} \cdots z_{nt} \cdots z_{nn}$

Applying exclusive-OR　　　Applying exclusive-OR

$$x_{it}^{new} = x_{it} \oplus x_{ic}, \text{ for } i = 1, \dots, n$$
$$z_{ic}^{new} = z_{ic} \oplus z_{it}, \text{ for } i = 1, \dots, n$$

We can apply **CNOT** gate by updating the binary table in $O(n)$

Example of Applying CNOT gate

$|0\rangle \otimes \frac{|0\rangle + i|1\rangle}{\sqrt{2}}$ is stabilized by $\boldsymbol{Y_1 I_2}$ and $\boldsymbol{I_1 Z_2}$.

2nd 1st

	$x_1 x_2$	$z_1 z_2$
$Y_1 I_2$	10	10
$I_1 Z_2$	00	01

$\oplus$ $\oplus$

Here, we apply CNOT gate on 1st qubit as control and 2nd qubit as target qubit.

	$x_1 x_2$	$z_1 z_2$
$Y_1 X_2$	11	10
$Z_1 Z_2$	00	11

$\boldsymbol{Y_1 X_2}$ and $\boldsymbol{Z_1 Z_2}$ stabilize $\frac{|00\rangle + i|11\rangle}{\sqrt{2}}$.