Higher-order quantum transformations of Hamiltonian dynamics

Tatsuki Odake (The University of Tokyo)
Hlér Kristjánsson (Perimeter Institute)
Akihito Soeda (National Institute of Informatics)
Mio Murao (The University of Tokyo)

Presentation slides

arXiv:2303.09788

PERIMETER INSTITUTE
東京大学 THE UNIVERSITY OF TOKYO
NII 大学共同利用機関法人 情報・システム研究機構 国立情報学研究所 National Institute of Informatics

Outline

I. Motivation & main results

II. Overview of our algorithm

III. Instance of quantum functional programming

IV. Application

V. General framework of higher-order transformation of Hamiltonian dynamics

Outline

I. Motivation & main results

II. Overview of our algorithm

III. Instance of quantum functional programming

IV. Application

V. General framework of higher-order transformation of Hamiltonian dynamics

(approximate) Hamiltonian simulation

- Hamiltonian simulation is a possible application of quantum computers

Input:

$$H = \begin{pmatrix} * & \cdots & * \\ \vdots & \ddots & \vdots \\ * & \cdots & * \end{pmatrix}$$

Description of Hamiltonian
(classical)

Output:

$$U \simeq e^{-iHt}$$

Hamiltonian dynamics
(quantum)

- Many algorithms have been proposed (e.g. qDRIFT[1], QSVT based[2])

[1] E. Campbell, PRL **123**, 070503 (2019). [2] Low, Guang Hao, and Isaac L. Chuang, Quantum **3**, 163 (2019).

Problems with Hamiltonian simulation

Problem: Classical description of target Hamiltonian is required in existing methods

$$H_{\mathrm{known}} = \begin{pmatrix} * & \cdots & * \\ \vdots & \ddots & \vdots \\ * & \cdots & * \end{pmatrix} \Rightarrow U \simeq e^{-iH_{\mathrm{known}}t}$$

E.g. qDRIFT, QSVT-based

$$H_{??} \Rightarrow f(H_{??})$$

New methods needed

E.g. negative time-evolution (simulating inverse unitary) $e^{-iHt} \Rightarrow e^{+iHt}$

Our approach

Transformation of Hamiltonian is formulated as:

"Higher-order transformation on Hamiltonian dynamics"

Our approach

Transformation of Hamiltonian is formulated as:

"Higher-order transformation on Hamiltonian dynamics"

(black-box) quantum operation → quantum operation

Our approach

Transformation of Hamiltonian is formulated as:

"Higher-order transformation on Hamiltonian dynamics"

(black-box) quantum operation → quantum operation

Previous work

$$H_{\text{known}} = \begin{pmatrix} * & \cdots & * \\ \vdots & \ddots & \vdots \\ * & \cdots & * \end{pmatrix}$$

$$e^{-iH_{\text{known}}t}$$

Our work

① e^{-iHt}

?

② Function f

$$H \mapsto f(H)$$

$$e^{-if(H)t}$$

?

Result

deterministic & approximate

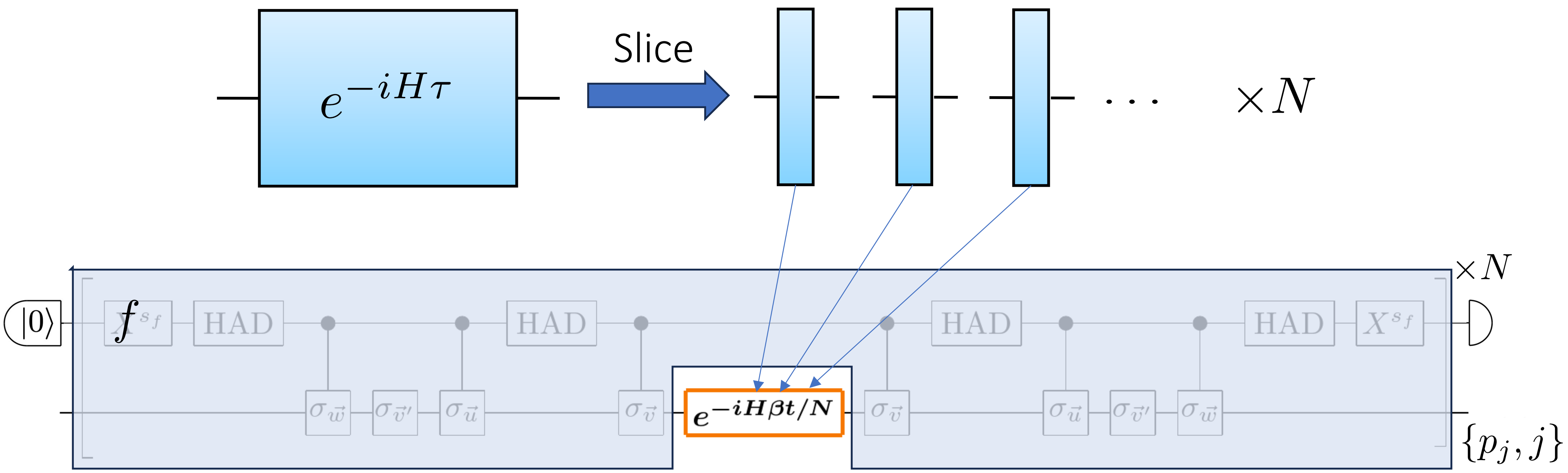
Higher-order transformation $e^{-iHt} \mapsto e^{-if(H)t}$ by linear map f of unknown Hamiltonian

Figure of merit: **Overall runtime**
(Not #queries to dynamics)

Result

Deterministic & Approximate

Higher-order transformation $e^{-iHt} \mapsto e^{-if(H)t}$ by linear map f of unknown Hamiltonian



Outline

I. Motivation & main results

II. Overview of our algorithm

III. Instance of quantum functional programming

IV. Application

V. General framework of higher-order transformation of Hamiltonian dynamics

Simulation of $e^{-if(H)t}$ (known H)

To get a clue about the higher-order algorithm, we first see how to simulate $e^{-if(H)t}$ for known H

① Decompose $f(H)$ into linear combination of simpler Hamiltonians H_j

$$f(H) = \sum_j h_j H_j$$

② Simulate RHS by qDRIFT

Extendable to unknown H

① Decomposition of f (unknown H)

We can obtain a decomposition of $f(H)$ with H-independent parameters using Pauli transfer matrix (PTM) of f

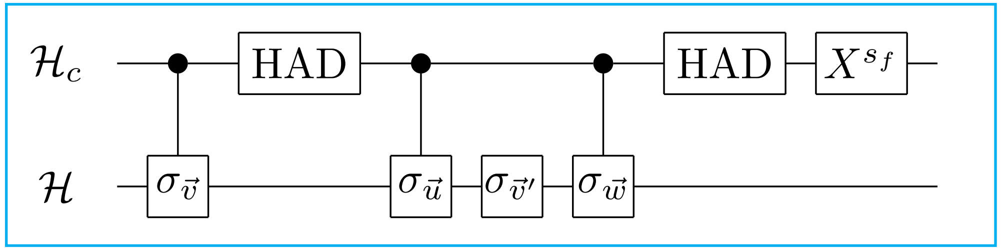
H -independent

$$\begin{pmatrix} f(H) & 0 \\ 0 & -f(H) \end{pmatrix} = \sum_j \boxed{p_j V_{f,j}} \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix} \boxed{V_{f,j}^{\dagger}}$$

Designed using functional programming

p_j : Defined using PTM of f

$V_{f,j} :=$



Pauli transfer matrix (PTM)

We describe n-qubit case:

$$H = \sum_{\vec{u}} c_{\vec{u}} \sigma_{\vec{u}} \Leftrightarrow \begin{pmatrix} 0 \\ c_{(0,\ldots,1)} \\ \vdots \\ c_{(3,\ldots,3)} \end{pmatrix} \in \mathbb{R}^{4^n}$$

$$\sigma_{\vec{v}} := \sigma_{v_1} \otimes \cdots \otimes \sigma_{v_n}$$

Pauli transfer matrix (PTM)

We describe n-qubit case:

$$H = \sum_{\vec{u}} c_{\vec{u}} \sigma_{\vec{u}} \Leftrightarrow \begin{pmatrix} 0 \\ c_{(0,\dots,1)} \\ \vdots \\ c_{(3,\dots,3)} \end{pmatrix} \in \mathbb{R}^{4^n}$$

$$\sigma_{\vec{v}} := \sigma_{v_1} \otimes \cdots \otimes \sigma_{v_n}$$

$\Rightarrow$

Hermitian preserving linear map f

$$\text{s.t. } f(\sigma_{\vec{u}}) = \sum_{\vec{w}} \gamma_{\vec{w},\vec{u}} \sigma_{\vec{w}} \Leftrightarrow$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \gamma_{(0,\dots,1),(0,\dots,1)} & \cdots & \gamma_{(0,\dots,1),(3,\dots,3)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \gamma_{(3,\dots,3),(0,\dots,1)} & \cdots & \gamma_{(3,\dots,3),(3,\dots,3)} \end{pmatrix}$$

PTM of "physically realizable linear map"

Pauli transfer matrix (PTM)

We describe n-qubit case:

$$H = \sum_{\vec{u}} c_{\vec{u}} \sigma_{\vec{u}} \Leftrightarrow \begin{pmatrix} 0 \\ c_{(0,\ldots,1)} \\ \vdots \\ c_{(3,\ldots,3)} \end{pmatrix} \in \mathbb{R}^{4^n}$$

$$\sigma_{\vec{v}} := \sigma_{v_1} \otimes \cdots \otimes \sigma_{v_n}$$

Hermitian preserving linear map f

$$\text{s.t. } f(\sigma_{\vec{u}}) = \sum_{\vec{w}} \gamma_{\vec{w},\vec{u}} \sigma_{\vec{w}} \Leftrightarrow$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \gamma_{(0,\ldots,1),(0,\ldots,1)} & \cdots & \gamma_{(0,\ldots,1),(3,\ldots,3)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \gamma_{(3,\ldots,3),(0,\ldots,1)} & \cdots & \gamma_{(3,\ldots,3),(3,\ldots,3)} \end{pmatrix}$$

PTM of "physically realizable linear map"

$$p_j := \frac{2|\gamma_{\vec{w},\vec{u}}|}{16^n \beta} \propto |\gamma_{\vec{w},\vec{u}}|$$

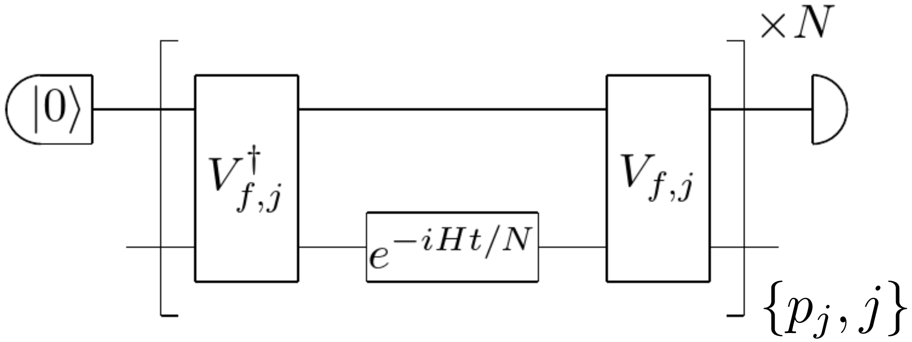
② Simulation of $f(H)$ (unknown H)

Decomposition of $f(H)$ can be used to implement $e^{-if(H)t}$ for unknown H

$$\begin{pmatrix} f(H) & 0 \\ 0 & -f(H) \end{pmatrix} = \sum_j p_j V_{f,j} \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix} V_{f,j}^\dagger$$

Hamiltonian simulation with random sampling

$$e^{-if(H)t} \simeq$$



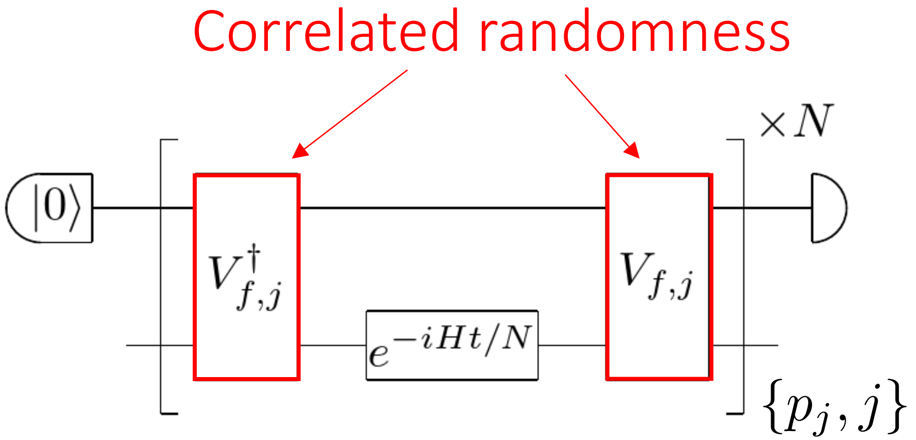
② Simulation of $f(H)$ (unknown H)

Decomposition of $f(H)$ can be used to implement $e^{-if(H)t}$ for unknown H

$$\begin{pmatrix} f(H) & 0 \\ 0 & -f(H) \end{pmatrix} = \sum_j p_j V_{f,j} \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix} V_{f,j}^{\dagger}$$

Hamiltonian simulation with random sampling

$e^{-if(H)t} \simeq$



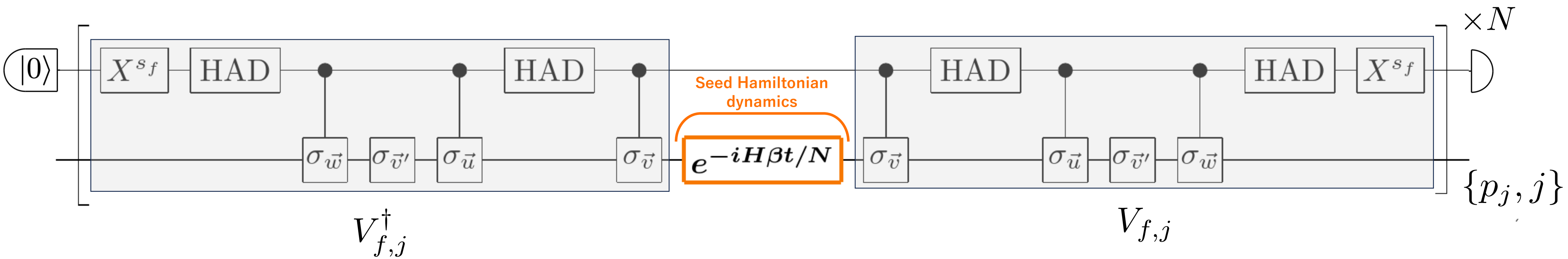
Our algorithm

We developed an algorithm applicable to arbitrary physically realizable linear map f.

Runtime: $O(\beta^2 t^2 n/\epsilon)$

$\beta :=$ (Function of PTM), ϵ : allowed error

$$\boxed{e^{-if(H)t}} \simeq$$



Outline

I. Motivation & main results

II. Overview of our algorithm

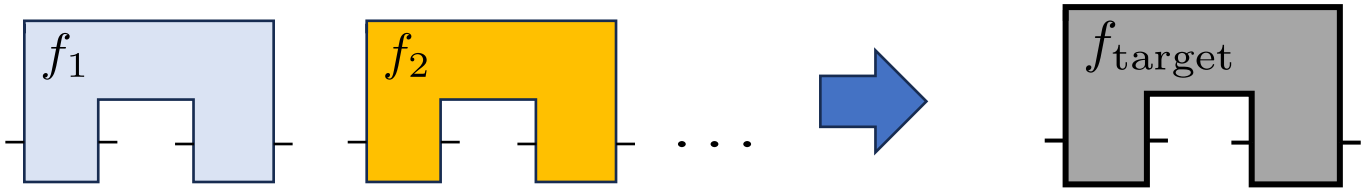
III. Instance of quantum functional programming

IV. Application

V. General framework of higher-order transformation of Hamiltonian dynamics

Quantum functional programming

Quantum functional programming:



E.g. concatenating higher-order transformations

Our algorithm: construct subroutine by concatenating seven functions

$$(\text{Subroutine function}) = g^{(7)}_{f,\vec{w},\vec{u}} \circ g^{(6)} \circ g^{(5)}_{\vec{w}} \circ g^{(4)} \circ g^{(3)}_{\vec{u}} \circ g^{(2)} \circ g^{(1)}$$

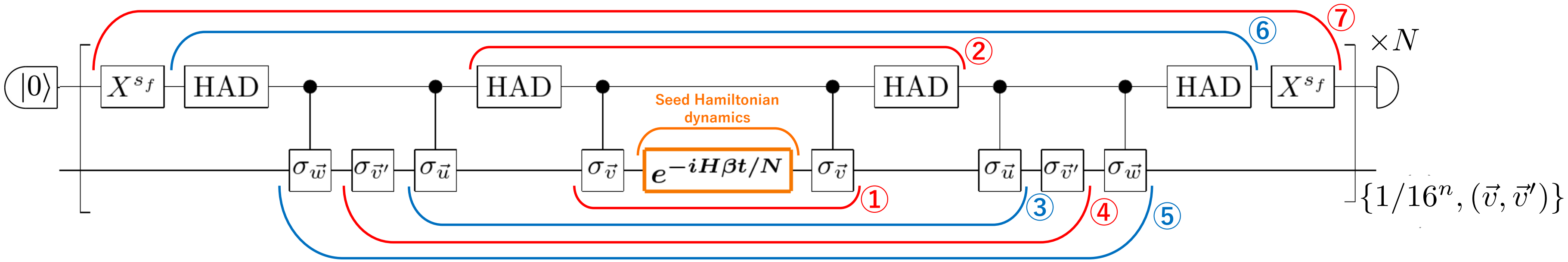
$V_{f,j}$ is naturally derived from this construction

Detail of functional programming instance

$$I \otimes H = \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix} \xrightarrow{\textcircled{1}} \begin{pmatrix} H & 0 \\ 0 & 0 \end{pmatrix} \xrightarrow{\textcircled{2}} \begin{pmatrix} H & H \\ H & H \end{pmatrix} \xrightarrow{\textcircled{3}} \begin{pmatrix} H & H\sigma_{\vec{u}} \\ \sigma_{\vec{u}}H & \sigma_{\vec{u}}H\sigma_{\vec{u}} \end{pmatrix}$$

$$\xrightarrow{\textcircled{4}} c_{\vec{u}} \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix} \xrightarrow{\textcircled{5}} c_{\vec{u}} \begin{pmatrix} 0 & \sigma_{\vec{w}} \\ \sigma_{\vec{w}} & 0 \end{pmatrix} \xrightarrow{\textcircled{6}} c_{\vec{u}} \begin{pmatrix} \sigma_{\vec{w}} & 0 \\ 0 & -\sigma_{\vec{w}} \end{pmatrix} \xrightarrow{\textcircled{7}} \mathrm{sgn}(\gamma_{\vec{w},\vec{u}})c_{\vec{u}} \begin{pmatrix} \sigma_{\vec{w}} & 0 \\ 0 & -\sigma_{\vec{w}} \end{pmatrix}$$

$$H =: \sum c_{\vec{u}}\sigma_{\vec{u}}$$



①: Controllization

②, ③: Preparing blocks

④: Block-wise tracing

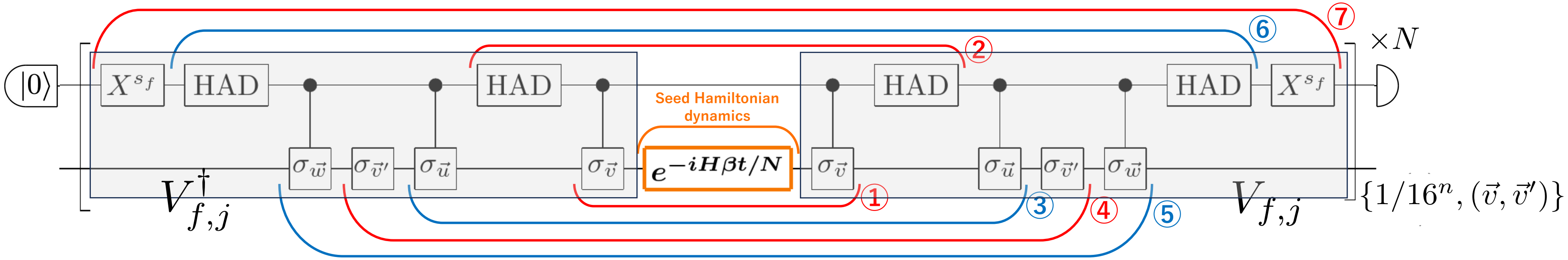
⑤, ⑥, ⑦: Basis change by Clifford

Detail of functional programming instance

$$I \otimes H = \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix} \xrightarrow{①} \begin{pmatrix} H & 0 \\ 0 & 0 \end{pmatrix} \xrightarrow{②} \begin{pmatrix} H & H \\ H & H \end{pmatrix} \xrightarrow{③} \begin{pmatrix} H & H\sigma_{\vec{u}} \\ \sigma_{\vec{u}}H & \sigma_{\vec{u}}H\sigma_{\vec{u}} \end{pmatrix}$$

$$\xrightarrow{④} c_{\vec{u}} \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix} \xrightarrow{⑤} c_{\vec{u}} \begin{pmatrix} 0 & \sigma_{\vec{w}} \\ \sigma_{\vec{w}} & 0 \end{pmatrix} \xrightarrow{⑥} c_{\vec{u}} \begin{pmatrix} \sigma_{\vec{w}} & 0 \\ 0 & -\sigma_{\vec{w}} \end{pmatrix} \xrightarrow{⑦} \mathrm{sgn}(\gamma_{\vec{w},\vec{u}}) c_{\vec{u}} \begin{pmatrix} \sigma_{\vec{w}} & 0 \\ 0 & -\sigma_{\vec{w}} \end{pmatrix}$$

$H =: \sum c_{\vec{u}} \sigma_{\vec{u}}$



①: Controllization

②, ③: Preparing blocks

④: Block-wise tracing

⑤, ⑥, ⑦: Basis change by Clifford

Description of function ① to ⑦

- Presentation slides are posted on our research group website

Slides are available here

Outline

I. Motivation & main results

II. Overview of our algorithm

III. Instance of quantum functional programming

IV. Application

V. General framework of higher-order transformation of Hamiltonian dynamics

Application

We consider two applications:

- Negative time-evolution

- Hamiltonian single parameter learning

M. Quintino et al. PRL **123**, 210502 (2019), S. Yoshida et al. arXiv: 2209.02907 (2022)

Negative time-evolution

Linear map: $f(H) = -H$ $\qquad e^{-iHt} \mapsto e^{+iHt}$

- Applicable to block encoding of unknown Hamiltonian H given by $e^{-iH\tau}$

$$e^{-iH\tau}\ (\tau > 0) \mapsto U(H) := \begin{pmatrix} H & \cdot \\ \cdot & \cdot \end{pmatrix}$$

- Runtime is exponential in n in general, but when H has a sparse support, it can be polynomial (e.g. k-local)

S. Lloyd et al. arXiv: 2104.01410 (2021)

Hamiltonian single parameter learning

Linear map: $f_{\vec{v}}(H) = c_{\vec{v}}\underbrace{Y \otimes I \otimes \cdots \otimes I}_{\text{Rotation axis (arbitrary)}}$ $\qquad e^{-iHt} \mapsto e^{-i\underline{c_{\vec{v}}}Yt} \otimes I \otimes \cdots \otimes I$

$H =: \sum c_{\vec{u}}\sigma_{\vec{u}}$

Measure by QPE

- Total evolution time: $O(1/s)$ (Heisenberg limit)
- Runtime: $O(\|H\|_{\mathrm{op}}^2 n/s^2)$ $(\mathrm{poly}(n)$ for sparse $H)$

s : Allowed RMS of error

c.f. Heisenberg limit for restricted situation [4], Runtime of full tomography of e^{-iHt} [3]:
$O(\mathrm{poly}(\exp(n), 1/s))$

[3] S. Kimmel et al. PRA **92**, 062315 (2015), [4] H. Y. Huang, et al., Phys. Rev. Lett. 130, 200403 (2020).

Outline

I. Motivation & main results

II. Overview of our algorithm

III. Instance of quantum functional programming

IV. Application

V. General framework of higher-order transformation of Hamiltonian dynamics

Higher-order transformation of Hamiltonian dynamics

Input:

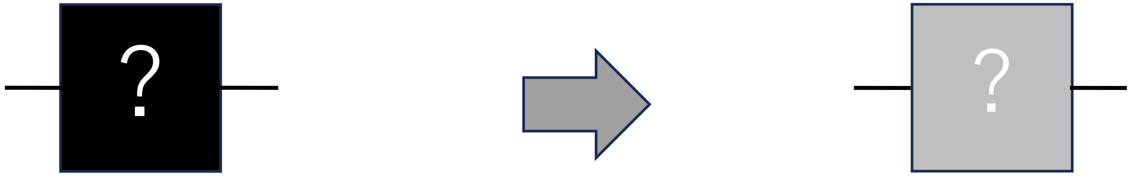
① Black-box dynamics

$e^{-iH\tau}\ (\tau > 0)$

Output:

Transformed dynamics

$e^{-if(H)t}$



② Function

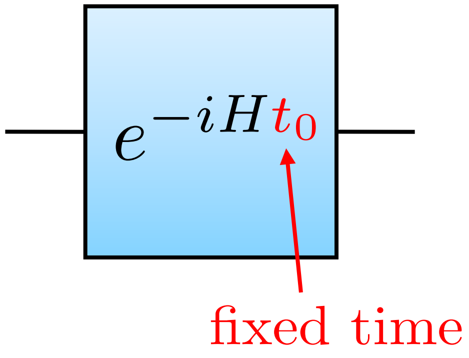
$H \mapsto f(H)$

Figure of merit: **Overall runtime**
(Not #queries to dynamics)

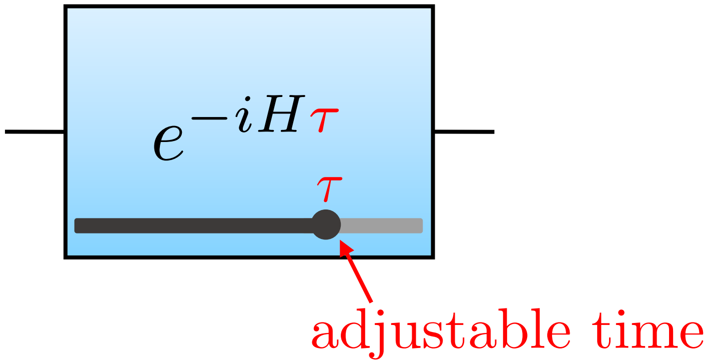
Difference with unitary input

Standard setting of higher-order transformation: black-box unitary input (e.g. inversion $U \mapsto U^\dagger$)

Unitary input



Hamiltonian dynamics input



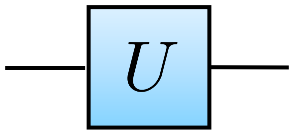
M. Quintino et al. PRL **123**, 210502 (2019), S. Yoshida et al. arXiv: 2209.02907 (2022)

Unitary input vs Hamiltonian dynamics input

Hamiltonian dynamics input has stronger simulatability

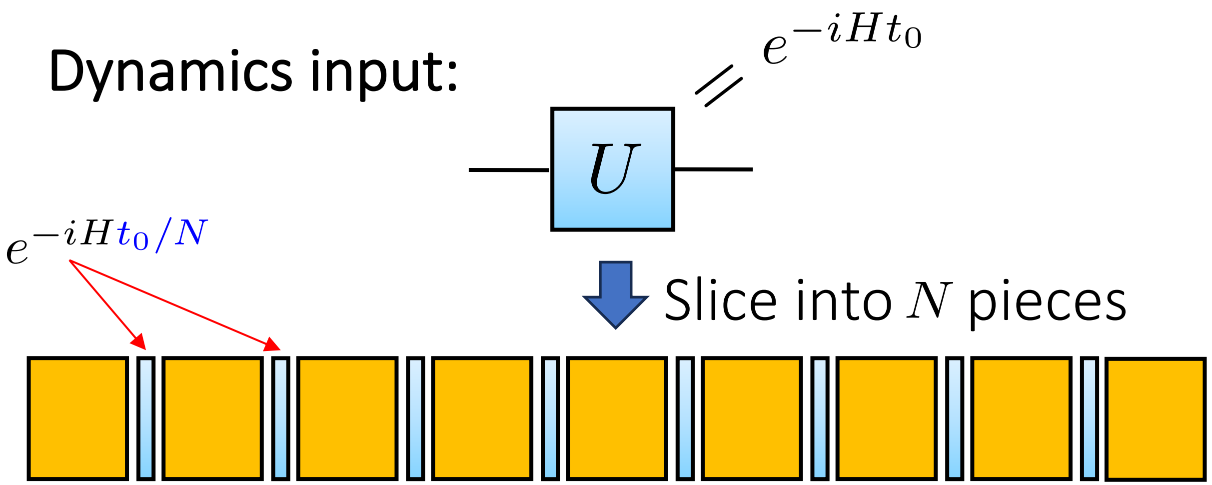
E.g. Controllization $U \mapsto \mathtt{ctrl}U$

Unitary input:



Impossible

Dynamics input:



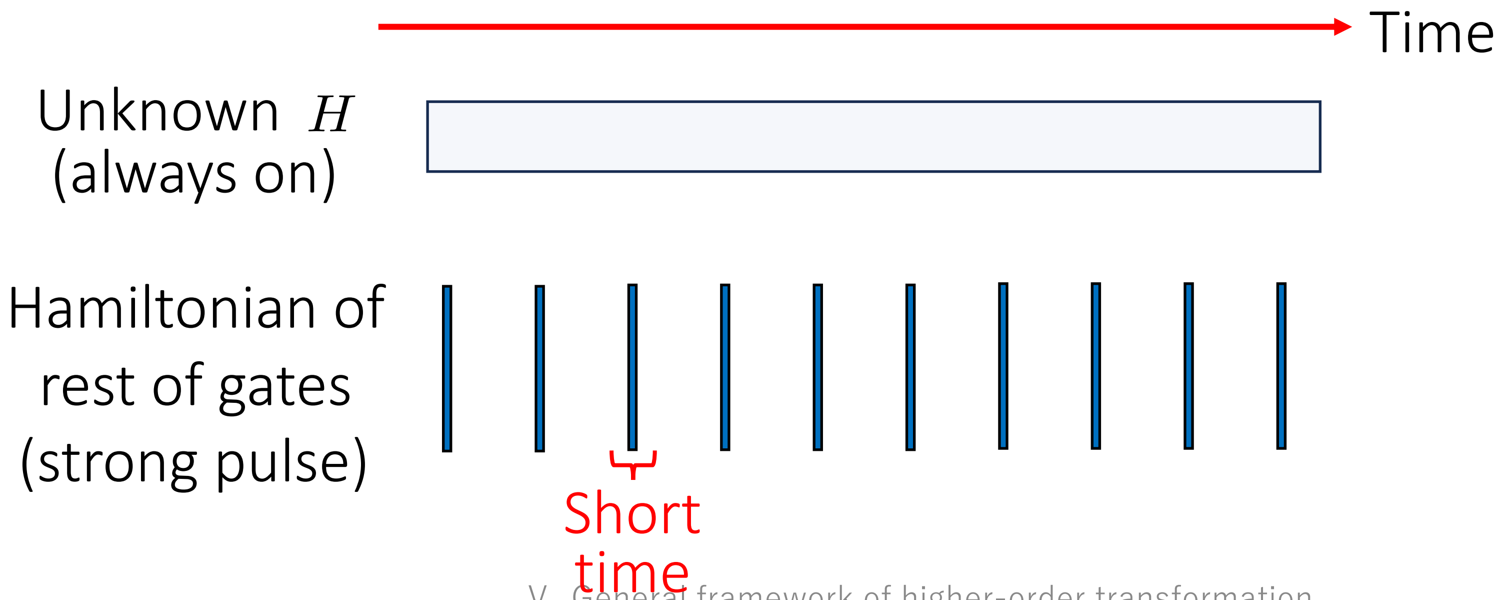
Possible

Z. Gavorova et al. arXiv:2011.10031 (2020). M. Araujo et al. New Journal of Physics 16, 093026 (2014). A. Soeda (Talk at IC- QIT, 2013). N. Friis et al. PRA **89**, 030303 (2014). Q. Dong et al. arXiv:1911.01645 (2019).

Practical implementation

The unknown dynamics can not be turned on/off in practice…

Fortunately, our algorithm can be approximately implemented if Clifford gates can be implemented in sufficiently short time



Summary

- We developed a new Hamiltonian simulation method applicable to simulation of unknown Hamiltonian
- Our algorithm can simulate $f(H)$ for an arbitrary physically realizable linear map f of Hamiltonian and unknown Hamiltonian H
- Part of our algorithm is developed by a quantum functional programming approach
- Our algorithm is an instance of higher-order transformations of Hamiltonian dynamics

Presentation slides

arXiv:2303.09788

Decomposition of linear functions

Arbitrary f can be decomposed into a sum of $\mathrm{sgn}(\gamma_{\vec{w},\vec{u}})f_{\vec{w},\vec{u}}$ with positive coefficients $|\gamma_{\vec{w},\vec{u}}|$

$$f = \sum_{\vec{u},\vec{w} \neq (0,\ldots,0)} |\gamma_{\vec{w},\vec{u}}| \cdot (\mathrm{sgn}(\gamma_{\vec{w},\vec{u}})f_{\vec{w},\vec{u}})$$

$$f_{\vec{w},\vec{u}} := \begin{array}{cc} & \vec{u} \\ \vec{w} & \begin{pmatrix} & 1 & \\ & & \\ & & \end{pmatrix} \end{array}$$

Mapping $\sigma_{\vec{u}}$ to $\sigma_{\vec{w}}$
(other input to 0)

Decomposition of linear functions

Arbitrary f can be decomposed into a sum of $\mathrm{sgn}(\gamma_{\vec{w},\vec{u}}) f_{\vec{w},\vec{u}}$ with positive coefficients $|\gamma_{\vec{w},\vec{u}}|$

$$f = \sum_{\vec{u},\vec{w} \neq (0,\ldots,0)} |\gamma_{\vec{w},\vec{u}}| \cdot (\mathrm{sgn}(\gamma_{\vec{w},\vec{u}}) f_{\vec{w},\vec{u}}) \qquad f_{\vec{w},\vec{u}} := \begin{array}{cc} & \begin{array}{ccc} & \vec{u} & \end{array} \\ \vec{w} & \begin{pmatrix} & 1 & \\ & & \\ & & \end{pmatrix} \end{array}$$

Can be simulated by summing $\mathrm{sgn}(\gamma_{\vec{w},\vec{u}}) f_{\vec{w},\vec{u}}$ by random sampling

Decomposition of linear functions

Arbitrary f can be decomposed into a sum of $\mathrm{sgn}(\gamma_{\vec{w},\vec{u}}) f_{\vec{w},\vec{u}}$ with positive coefficients $|\gamma_{\vec{w},\vec{u}}|$

$$f = \sum_{\vec{u},\vec{w} \neq (0,\ldots,0)} |\gamma_{\vec{w},\vec{u}}| \cdot (\mathrm{sgn}(\gamma_{\vec{w},\vec{u}}) f_{\vec{w},\vec{u}}) \qquad f_{\vec{w},\vec{u}} := \begin{array}{cc} & \vec{u} \\ \vec{w} & \begin{pmatrix} & 1 & \\ & & \\ & & \end{pmatrix} \end{array}$$

Subgoal: Implementing $\mathrm{sgn}(\gamma_{\vec{w},\vec{u}}) f_{\vec{w},\vec{u}} = \begin{array}{cc} & \vec{u} \\ \vec{w} & \begin{pmatrix} & \mathrm{sgn}(\gamma_{\vec{w},\vec{u}}) & \\ & & \\ & & \end{pmatrix} \end{array}$ by

Hamiltonian simulation by random sampling

Simulation of one entry of PTM

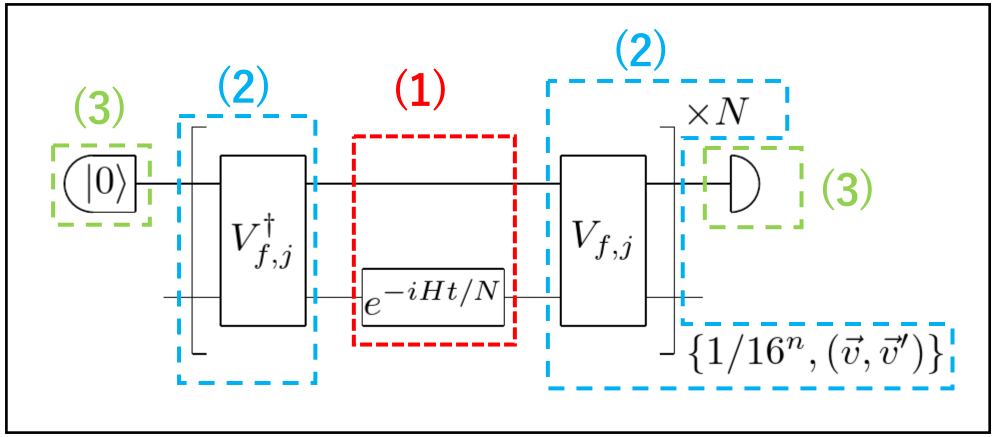
$H = \sum_{\vec{u} \neq (0,\ldots,0)} c_{\vec{u}} \sigma_{\vec{u}}$

$$H \xrightarrow{(1)} \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix} \xrightarrow{(2)} \begin{pmatrix} \mathrm{sgn}(\gamma_{\vec{w},\vec{u}}) c_{\vec{u}} \sigma_{\vec{w}} & 0 \\ 0 & -\mathrm{sgn}(\gamma_{\vec{w},\vec{u}}) c_{\vec{u}} \sigma_{\vec{w}} \end{pmatrix} \xrightarrow{(3)} \mathrm{sgn}(\gamma_{\vec{w},\vec{u}}) \underline{c_{\vec{u}}} \sigma_{\vec{w}}$$

$$\mathrm{sgn}(\gamma_{\vec{w},\vec{u}}) c_{\vec{u}} \sigma_{\vec{w}} = \mathrm{sgn}(\gamma_{\vec{w},\vec{u}}) f_{\vec{w},\vec{u}}(H)$$

(1) Adding ancilla qubit

$$\underline{I} \otimes e^{-iHt} = \exp\left[-i \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix} t \right]$$



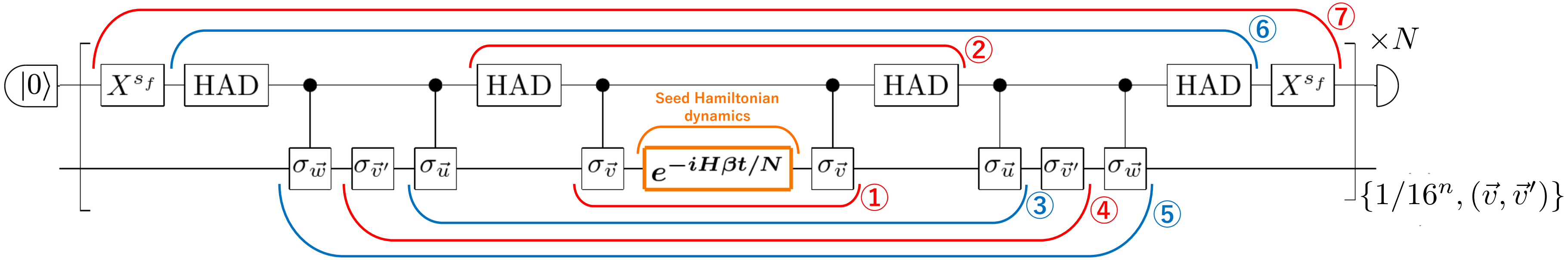
(3) Restricting ancilla to $|0\rangle$
(extracting top-left block)

Overview of Step (2)

$$I \otimes H = \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix} \xrightarrow{①} \begin{pmatrix} H & 0 \\ 0 & 0 \end{pmatrix} \xrightarrow{②} \begin{pmatrix} H & H \\ H & H \end{pmatrix} \xrightarrow{③} \begin{pmatrix} H & H\sigma_{\vec{u}} \\ \sigma_{\vec{u}}H & \sigma_{\vec{u}}H\sigma_{\vec{u}} \end{pmatrix}$$

$$\xrightarrow{④} c_{\vec{u}} \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix} \xrightarrow{⑤} c_{\vec{u}} \begin{pmatrix} 0 & \sigma_{\vec{w}} \\ \sigma_{\vec{w}} & 0 \end{pmatrix} \xrightarrow{⑥} c_{\vec{u}} \begin{pmatrix} \sigma_{\vec{w}} & 0 \\ 0 & -\sigma_{\vec{w}} \end{pmatrix} \xrightarrow{⑦} \mathrm{sgn}(\gamma_{\vec{w},\vec{u}}) c_{\vec{u}} \begin{pmatrix} \sigma_{\vec{w}} & 0 \\ 0 & -\sigma_{\vec{w}} \end{pmatrix}$$

$H =: \sum c_{\vec{u}} \sigma_{\vec{u}}$



$\mathrm{sgn}(\gamma_{\vec{w},\vec{u}}) f_{\vec{w},\vec{u}}$ is constructed by concatenating function ①,...,⑦

Function ① (controllization)

$$I \otimes H = \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix} \xrightarrow{①} \begin{pmatrix} H & 0 \\ 0 & 0 \end{pmatrix} \xrightarrow{②} \begin{pmatrix} H & H \\ H & H \end{pmatrix} \xrightarrow{③} \begin{pmatrix} H & H\sigma_{\vec{u}} \\ \sigma_{\vec{u}}H & \sigma_{\vec{u}}H\sigma_{\vec{u}} \end{pmatrix}$$

$$\xrightarrow{④} c_{\vec{u}} \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix} \xrightarrow{⑤} c_{\vec{u}} \begin{pmatrix} 0 & \sigma_{\vec{w}} \\ \sigma_{\vec{w}} & 0 \end{pmatrix} \xrightarrow{⑥} c_{\vec{u}} \begin{pmatrix} \sigma_{\vec{w}} & 0 \\ 0 & -\sigma_{\vec{w}} \end{pmatrix} \xrightarrow{⑦} \mathrm{sgn}(\gamma_{\vec{w},\vec{u}})c_{\vec{u}} \begin{pmatrix} \sigma_{\vec{w}} & 0 \\ 0 & -\sigma_{\vec{w}} \end{pmatrix}$$

Sandwiching by random $\texttt{ctrl}\sigma_{\vec{v}}$

①: $$\sum_{\vec{v}\in\{0,1,2,3\}^n} \frac{1}{4^n} \begin{pmatrix} I & 0 \\ 0 & \sigma_{\vec{v}} \end{pmatrix} \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & \sigma_{\vec{v}} \end{pmatrix}$$

$$= \underline{\begin{pmatrix} H & 0 \\ 0 & 0 \end{pmatrix}} \qquad (H \text{ : traceless})$$

Gives $\begin{pmatrix} e^{-iHt} & 0 \\ 0 & I \end{pmatrix}$ when exponentiated

Q. Dong et al. arXiv:1911.01645 (2019).

Function ① (controllization)

$$I \otimes H = \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix} \xrightarrow{①} \begin{pmatrix} H & 0 \\ 0 & 0 \end{pmatrix} \xrightarrow{②} \begin{pmatrix} H & H \\ H & H \end{pmatrix} \xrightarrow{③} \begin{pmatrix} H & H\sigma_{\vec{u}} \\ \sigma_{\vec{u}}H & \sigma_{\vec{u}}H\sigma_{\vec{u}} \end{pmatrix}$$

$$\xrightarrow{④} c_{\vec{u}} \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix} \xrightarrow{⑤} c_{\vec{u}} \begin{pmatrix} 0 & \sigma_{\vec{w}} \\ \sigma_{\vec{w}} & 0 \end{pmatrix} \xrightarrow{⑥} c_{\vec{u}} \begin{pmatrix} \sigma_{\vec{w}} & 0 \\ 0 & -\sigma_{\vec{w}} \end{pmatrix} \xrightarrow{⑦} \mathrm{sgn}(\gamma_{\vec{w},\vec{u}}) c_{\vec{u}} \begin{pmatrix} \sigma_{\vec{w}} & 0 \\ 0 & -\sigma_{\vec{w}} \end{pmatrix}$$

Sandwiching by random $\texttt{ctrl}\sigma_{\vec{v}}$

①: $$\sum_{\vec{v} \in \{0,1,2,3\}^n} \frac{1}{4^n} \begin{pmatrix} I & 0 \\ 0 & \sigma_{\vec{v}} \end{pmatrix} \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & \sigma_{\vec{v}} \end{pmatrix}$$

$$= \begin{pmatrix} H & 0 \\ 0 & 0 \end{pmatrix}$$ (H : traceless)

For Pauli $\sigma \in \{I, X, Y, Z\}^{\otimes n}$,

$$\frac{1}{4^n} \sum_{\vec{v} \in \{0,1,2,3\}^n} \sigma_{\vec{v}} \sigma \sigma_{\vec{v}} = \begin{cases} I & (\sigma = I) \\ 0 & (\text{otherwise}) \end{cases}$$

Function ②,③ (Preparing blocks)

$$I \otimes H = \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix} \xrightarrow{①} \begin{pmatrix} H & 0 \\ 0 & 0 \end{pmatrix} \xrightarrow{②} \begin{pmatrix} H & H \\ H & H \end{pmatrix} \xrightarrow{③} \begin{pmatrix} H & H\sigma_{\vec{u}} \\ \sigma_{\vec{u}}H & \sigma_{\vec{u}}H\sigma_{\vec{u}} \end{pmatrix}$$

$$\xrightarrow{④} c_{\vec{u}} \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix} \xrightarrow{⑤} c_{\vec{u}} \begin{pmatrix} 0 & \sigma_{\vec{w}} \\ \sigma_{\vec{w}} & 0 \end{pmatrix} \xrightarrow{⑥} c_{\vec{u}} \begin{pmatrix} \sigma_{\vec{w}} & 0 \\ 0 & -\sigma_{\vec{w}} \end{pmatrix} \xrightarrow{⑦} \mathrm{sgn}(\gamma_{\vec{w},\vec{u}}) c_{\vec{u}} \begin{pmatrix} \sigma_{\vec{w}} & 0 \\ 0 & -\sigma_{\vec{w}} \end{pmatrix}$$

Sandwiching by unitary (no randomness)

②: $2(\mathrm{HAD} \otimes I) \begin{pmatrix} H & 0 \\ 0 & 0 \end{pmatrix} (\mathrm{HAD} \otimes I) = \begin{pmatrix} H & H \\ H & H \end{pmatrix}$

$\times 2$ is implemented only by doubling the simulation time

③: $\begin{pmatrix} I & 0 \\ 0 & \sigma_{\vec{u}} \end{pmatrix} \begin{pmatrix} H & H \\ H & H \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & \sigma_{\vec{u}} \end{pmatrix} = \begin{pmatrix} H & H\sigma_{\vec{u}} \\ \sigma_{\vec{u}}H & \sigma_{\vec{u}}H\sigma_{\vec{u}} \end{pmatrix}$

Function ④ (block-wise tracing)

$$I \otimes H = \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix} \xrightarrow{①} \begin{pmatrix} H & 0 \\ 0 & 0 \end{pmatrix} \xrightarrow{②} \begin{pmatrix} H & H \\ H & H \end{pmatrix} \xrightarrow{③} \begin{pmatrix} H & H\sigma_{\vec{u}} \\ \sigma_{\vec{u}}H & \sigma_{\vec{u}}H\sigma_{\vec{u}} \end{pmatrix}$$

$$\xrightarrow{④} c_{\vec{u}} \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix} \xrightarrow{⑤} c_{\vec{u}} \begin{pmatrix} 0 & \sigma_{\vec{w}} \\ \sigma_{\vec{w}} & 0 \end{pmatrix} \xrightarrow{⑥} c_{\vec{u}} \begin{pmatrix} \sigma_{\vec{w}} & 0 \\ 0 & -\sigma_{\vec{w}} \end{pmatrix} \xrightarrow{⑦} \mathrm{sgn}(\gamma_{\vec{w},\vec{u}}) c_{\vec{u}} \begin{pmatrix} \sigma_{\vec{w}} & 0 \\ 0 & -\sigma_{\vec{w}} \end{pmatrix}$$

$H =: \sum c_{\vec{u}} \sigma_{\vec{u}}$

Sandwiching by random $\sigma_{\vec{v}'}$ on system

$$④: \sum_{\vec{v}' \in \{0,1,2,3\}^n} \frac{1}{4^n} (I \otimes \sigma_{\vec{v}'}) \begin{pmatrix} H & H\sigma_{\vec{u}} \\ \sigma_{\vec{u}}H & \sigma_{\vec{u}}H\sigma_{\vec{u}} \end{pmatrix} (I \otimes \sigma_{\vec{v}'})$$

$$= \begin{pmatrix} \mathrm{tr}(H)(I/2^n) & \mathrm{tr}(H\sigma_{\vec{u}})(I/2^n) \\ \mathrm{tr}(\sigma_{\vec{u}}H)(I/2^n) & \mathrm{tr}(\sigma_{\vec{u}}H\sigma_{\vec{u}})(I/2^n) \end{pmatrix}$$

$$\frac{1}{4^n} \sum_{\vec{v} \in \{0,1,2,3\}^n} \sigma_{\vec{v}} A \sigma_{\vec{v}} = \mathrm{tr}(A) \cdot \frac{I}{2^n}$$

Function ④ (block-wise tracing)

$$I \otimes H = \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix} \xrightarrow{①} \begin{pmatrix} H & 0 \\ 0 & 0 \end{pmatrix} \xrightarrow{②} \begin{pmatrix} H & H \\ H & H \end{pmatrix} \xrightarrow{③} \begin{pmatrix} H & H\sigma_{\vec{u}} \\ \sigma_{\vec{u}}H & \sigma_{\vec{u}}H\sigma_{\vec{u}} \end{pmatrix}$$

$$\xrightarrow{④} c_{\vec{u}} \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix} \xrightarrow{⑤} c_{\vec{u}} \begin{pmatrix} 0 & \sigma_{\vec{w}} \\ \sigma_{\vec{w}} & 0 \end{pmatrix} \xrightarrow{⑥} c_{\vec{u}} \begin{pmatrix} \sigma_{\vec{w}} & 0 \\ 0 & -\sigma_{\vec{w}} \end{pmatrix} \xrightarrow{⑦} \mathrm{sgn}(\gamma_{\vec{w},\vec{u}})c_{\vec{u}} \begin{pmatrix} \sigma_{\vec{w}} & 0 \\ 0 & -\sigma_{\vec{w}} \end{pmatrix}$$

$H =: \sum c_{\vec{u}}\sigma_{\vec{u}}$

Sandwiching by random $\sigma_{\vec{v}'}$ on system

$$④: \sum_{\vec{v}' \in \{0,1,2,3\}^n} \frac{1}{4^n}(I \otimes \sigma_{\vec{v}'}) \begin{pmatrix} H & H\sigma_{\vec{u}} \\ \sigma_{\vec{u}}H & \sigma_{\vec{u}}H\sigma_{\vec{u}} \end{pmatrix} (I \otimes \sigma_{\vec{v}'})$$

$$= \begin{pmatrix} \mathrm{tr}(H)(I/2^n) & \mathrm{tr}(H\sigma_{\vec{u}})(I/2^n) \\ \mathrm{tr}(\sigma_{\vec{u}}H)(I/2^n) & \mathrm{tr}(\sigma_{\vec{u}}H\sigma_{\vec{u}})(I/2^n) \end{pmatrix}$$

$$= \begin{pmatrix} 0 & c_{\vec{u}}I \\ c_{\vec{u}}I & 0 \end{pmatrix}$$

$$\mathrm{tr}(H) = \mathrm{tr}(\sigma_{\vec{u}}H\sigma_{\vec{u}}) = 0$$
$$\mathrm{tr}(H\sigma_{\vec{u}}) = \mathrm{tr}(\sigma_{\vec{u}}H) = 2^n c_{\vec{u}}$$

Function ⑤,⑥,⑦ (Basis change by Clifford)

$$I \otimes H = \begin{pmatrix} H & 0 \\ 0 & H \end{pmatrix} \xrightarrow{①} \begin{pmatrix} H & 0 \\ 0 & 0 \end{pmatrix} \xrightarrow{②} \begin{pmatrix} H & H \\ H & H \end{pmatrix} \xrightarrow{③} \begin{pmatrix} H & H\sigma_{\vec{u}} \\ \sigma_{\vec{u}}H & \sigma_{\vec{u}}H\sigma_{\vec{u}} \end{pmatrix}$$

$$\xrightarrow{④} c_{\vec{u}} \begin{pmatrix} 0 & I \\ I & 0 \end{pmatrix} \xrightarrow{⑤} c_{\vec{u}} \begin{pmatrix} 0 & \sigma_{\vec{w}} \\ \sigma_{\vec{w}} & 0 \end{pmatrix} \xrightarrow{⑥} c_{\vec{u}} \begin{pmatrix} \sigma_{\vec{w}} & 0 \\ 0 & -\sigma_{\vec{w}} \end{pmatrix} \xrightarrow{⑦} \mathrm{sgn}(\gamma_{\vec{w},\vec{u}}) c_{\vec{u}} \begin{pmatrix} \sigma_{\vec{w}} & 0 \\ 0 & -\sigma_{\vec{w}} \end{pmatrix}$$

Sandwiching by unitary (no randomness)

⑤: $(\mathtt{ctrl}\sigma_{\vec{w}})(c_{\vec{u}}X \otimes I)(\mathtt{ctrl}\sigma_{\vec{w}}) = (c_{\vec{u}}X \otimes \sigma_{\vec{w}})$

⑥: $(\mathrm{HAD} \otimes I)(c_{\vec{u}}X \otimes \sigma_{\vec{w}})(\mathrm{HAD} \otimes I) = (c_{\vec{u}}Z \otimes \sigma_{\vec{w}})$

⑦: $(X^{s_f} \otimes I)(c_{\vec{u}}Z \otimes \sigma_{\vec{w}})(X^{s_f} \otimes I) = \mathrm{sgn}(\gamma_{\vec{w},\vec{u}})(c_{\vec{u}}Z \otimes \sigma_{\vec{w}})$ $\qquad s_f := (1 - \mathrm{sgn}(\gamma_{\vec{w},\vec{u}}))/2$

1. Hamiltonian simulation with random sampling

Sum and concatenation of transformations of shape $H \mapsto \sum_j h_j U_j H U_j^\dagger$ can also be simulated using random sampling

$$f(H) := \sum_j h_j \left(U_j H U_j^\dagger\right)$$

$$g(H) := \sum_k r_k \left(V_k H V_k^\dagger\right)$$

Sum $\alpha f + \beta g \ (\alpha, \beta > 0)$

① choose f or g in prob. $\alpha/(\alpha+\beta)$ and $\beta/(\alpha+\beta)$

② choose $\begin{cases} U_j & (f \text{ is chosen}) \\ V_k & (g \text{ is chosen}) \end{cases}$ with prob. $\begin{cases} h_j/\sum_j h_j \\ r_k/\sum_k r_k \end{cases}$

Concatenation $g \circ f$

Choose $V_k U_j$ with prob. $(h_j/\sum_j h_j)(r_k/\sum_k r_k)$

2. Pauli transfer matrix (PTM)

We describe n-qubit case:

$$H = \sum_{\vec{u}} c_{\vec{u}} \sigma_{\vec{u}} \Leftrightarrow \begin{pmatrix} 0 \\ c_{(0,\ldots,1)} \\ \vdots \\ c_{(3,\ldots,3)} \end{pmatrix} \in \mathbb{R}^{4^n}$$

$$\sigma_{\vec{v}} := \sigma_{v_1} \otimes \cdots \otimes \sigma_{v_n}$$

Hermitian preserving linear map f

$$\text{s.t. } f(\sigma_{\vec{u}}) = \sum_{\vec{w}} \gamma_{\vec{w},\vec{u}} \sigma_{\vec{w}} \Leftrightarrow$$

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & \gamma_{(0,\ldots,1),(0,\ldots,1)} & \cdots & \gamma_{(0,\ldots,1),(3,\ldots,3)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \gamma_{(3,\ldots,3),(0,\ldots,1)} & \cdots & \gamma_{(3,\ldots,3),(3,\ldots,3)} \end{pmatrix}$$

PTM of “physically realizable linear map”

First row/column of PTM can w.l.o.g taken to be 0 (otherwise unphysical)