OUKA
The University of Osaka
Institutional Knowledge Archive

Title	Applications and performance analyses of quantum machine learning via fast classical simulation of a quantum computer
Author(s)	河瀬, 良亮
Citation	大阪大学, 2023, 博士論文
Version Type	VoR
URL	https://doi.org/10.18910/92207
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

The University of Osaka

Applications and performance analyses of quantum machine learning via fast classical simulation of a quantum computer

YOSHIAKI KAWASE

MARCH 2023

Applications and performance analyses of quantum machine learning via fast classical simulation of a quantum computer

A dissertation submitted to
THE GRADUATE SCHOOL OF ENGINEERING SCIENCE
OSAKA UNIVERSITY
in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY IN ENGINEERING

BY

YOSHIAKI KAWASE

MARCH 2023

Abstract

A quantum computer is a computer utilizing the properties of quantum mechanics to solve computationally hard problems for classical computers. It has been experimentally demonstrated that a quantum computer can solve a certain sampling task faster than a conventional, classical, computer. Since then, there is currently a lot of research being done to apply such a quantum computer to useful tasks such as finance, quantum chemistry, and machine learning.

Quantum machine learning is an emerging field that employs the principles of quantum computing for solving machine learning tasks such as clustering and classification. While a fully fledged fault-tolerant quantum computer provides us exponential speedup even for this purpose under certain assumptions, current quantum computers are called noisy intermediate-scale quantum (NISQ) devices, which are limited in terms of the number of qubits and gate operations and are also subject to noise. Despite these limitations, researchers are working on developing quantum machine learning algorithms that can be run on NISQ devices to demonstrate potential advantages over classical algorithms. Given quantum computing devices are still under progress, classical simulation remains a crucial approach for evaluating and optimizing the performance of quantum machine learning algorithms. Furthermore, the inherent complexity and abstract nature of quantum states, visualization techniques are crucial for the development and understanding of quantum machine learning algorithms, similar to their importance in classical machine learning.

In this thesis, we develop classical simulation techniques and visualization tools for quantum computers to accelerate the development of quantum machine learning. First, we propose a method of accelerating a classical simulation of Hamiltonian dynamics, which is employed in variaous quantum algorithms since it is most advantageous task for quantum computers. Through the utilization of commuting relations between Pauli terms in the Hamiltonian, the simulation of Hamiltonian dynamics can be significantly accelerated in comparison to a step-by-step approach implemented through simple Trotterization. This can be applied for various quantum algorithms to accelerate their simulation such as quantum circuit learning, quantum reservoir computing, and quantum approximate optimization algorithm.

Next, we develop another target, visualization technique of quantum states by projecting higher dimensional quantum information, with keeping similarity of quantum states, into two-dimensional classical data. By optimizing the parameterized quantum circuit applied to the quantum states, we learn the relation between quantum states and low-dimensional outputs so that similarity of the quantum states in the sense of the distance in high-dimensional space is kept in the low-dimensional space. For visualizing the quantum dynamics of transverse field Ising model, we successfully visualize them according to the characteristics of the quantum states. However, this method requires appropriately designing the parameterized quantum circuit and choosing observables to visualize data. This limits direct evaluation of the visualization and the performance of mapping classical data into Hilbert space via a quantum circuit. To avoid

this limitation and achieve visualization of quantum states without optimizing a quantum circuit, we propose a visualization method using quantum kernels, where the low-dimensional data consisting of the linear combination of the similarity of quantum states are optimized to keep the similarity of the quantum states. We embed a classical hand-written dataset into quantum states using a quantum feature map, which allows us visualization and classification using a clustering method in two-dimensional space. We observe that the proposed method provides a higher performance than classical kernel-based visualization method and competing state-of-the-art neural network based approaches, while only using a small number of qubits.

We believe our contributions, faster classical simulation and visualization technique, are crucial for advancing quantum computing research, particularly in the area of quantum machine learning in the future. Regarding the classical simulation, we have successfully accelerated, an order of magnitude, to simulate Hamiltonian dynamics of transverse field Ising models, which are commonly used in quantum algorithms such as quantum reservoir computing, quantum circuit learning, quantum approximate optimization algorithms and quantum annealing. Regarding visualization, we have opened up a field of quantum state visualization. The constructed visualization methods would be useful in various situations in quantum machine learning, for example, in the design of quantum feature maps that map classical information to quantum states. We have even found a practical application, visualization of the optimization process of variational algorithms, which would contribute to avoiding barren plateau issue. We hope the developed techniques here enable the evaluation of quantum algorithms for relatively large qubit systems, and allow for an in-depth examination of the effects of noise on algorithm performance. This can provide valuable insights into the potential for achieving quantum advantage and inform strategies for improving it.

Contents

1 Introduction **7**
- 1.1 Quantum computing at NISQ era . . . 7
- 1.2 Challenges on quantum machine learning . . . 8
- 1.3 Existing approaches on classical simulation and visualization technique . . . 9
- 1.4 Contributions of this thesis . . . 10

2 Preliminary **13**
- 2.1 Basics of classical simulation of quantum dynamics . . . 13
  - 2.1.1 Quantum state and quantum gates . . . 13
  - 2.1.2 Classical simulation of quantum dynamics . . . 15
- 2.2 Visualization method in classical machine learning . . . 16
  - 2.2.1 t-distributed stochastic neighbor embedding (t-SNE) . . . 17
- 2.3 Quantum machine learning methods . . . 17
  - 2.3.1 Variational quantum algorithm . . . 17
  - 2.3.2 Quantum kernel method . . . 18

3 Fast classical simulation of Hamiltonian dynamics by simultaneous diagonalization using Clifford transformation with parallel computation **21**
- 3.1 Introduction . . . 21
- 3.2 Preliminary . . . 23
  - 3.2.1 Partitioning Hamiltonian terms . . . 23
  - 3.2.2 Tableau representation of Pauli operators and Clifford operations . . . 24
- 3.3 Simultaneous diagonalization for partitioned Hamiltonian . . . 26
  - 3.3.1 Preparation . . . 27
  - 3.3.2 Partial diagonalization of Hamiltonian . . . 27
  - 3.3.3 Classical simulation of quantum dynamics with simultaneous diagonalization . . . 32
  - 3.3.4 Implementations . . . 33
- 3.4 Numerical experiments . . . 36
  - 3.4.1 Transverse field Ising model . . . 37
  - 3.4.2 Majorana fermion model . . . 38

3.5 Conclusions 40

4 Parametric t-SNE with quantum neural network 41
4.1 Introduction 41
4.2 Preliminary 42
4.2.1 Parametric t-SNE 42
4.2.2 Application of t-SNE for quantum systems 42
4.3 Parametric t-SNE with quantum circuits 43
4.4 Numerical experiments 44
4.4.1 Visualizing classical data 45
4.4.2 Visualization of quantum data 47
4.5 Conclusion and future works 51

5 Quantum Kernel t-SNE 53
5.1 Introduction 53
5.2 Kernel t-SNE 55
5.3 Quantum kernel t-SNE 55
5.3.1 Algorithm 55
5.3.2 Advantage over the previous method 56
5.4 Numerical experiments 57
5.4.1 Visualizing classical data 57
5.4.2 Visualizing quantum data 59
5.5 Conclusion 62

6 Conclusion 63

List of publications 65

A Appendix 67
A.1 Appendix for fast classical simulation of Hamiltonian dynamics . 67
A.1.1 The algorithm of searching linear independent terms . . . 67
A.1.2 Proof of our simultaneous diagonalization algorithm . . . 69
A.2 The models used in numerical experiments for visualizing classical data with quantum kernels 72

Bibliography 75

Acknowledgments 81

Chapter 1

Introduction

1.1 Quantum computing at NISQ era

Quantum computing is a field of computer science that seeks to harness the principles of quantum mechanics to perform certain types of computation more efficiently than classical computers. Research in this field aims to develop new algorithms and technologies that take advantage of the unique properties of quantum systems, such as superposition and entanglement. Some of the most promising applications of quantum computing include prime factorization [1], database search [2], solving linear systems [3], and simulation of quantum many-body systems [4]. However, due to their complex structure, these quantum algorithms that have been theoretically proven to have a quantum speedup can only be run on fully-fledged, fault-tolerant quantum computers, where quantum information is protected by quantum error correction.

In recent years, there has been active development of quantum computers by information technology companies, including Google and IBM. While quantum computers with tens to hundreds of qubits have been realized. Unfortunately, the number of qubits is currently insufficient for the realization of a fault-tolerant quantum computer. Current quantum computers, which are essentially suffering from noise, are called Noisy Intermediate-Scale Quantum (NISQ) devices [5], have been shown to have a great potential to outperform classical computers in certain sampling tasks [6]. This sampling task is just designed for benchmarking and is not useful for solving real-world problems. Researchers are now exploring the potential applications of NISQ devices for various practically important and useful tasks [5, 7, 8, 9, 10, 11, 12]. One of the promising algorithms running on NISQ devices is a variational quantum algorithm [13], expected to be useful in quantum chemistry [7], combinatorial optimization problems [8], and quantum machine learning [9]. NISQ devices, with their current limitations, may not be able to perform complex computations, but they could still prove valuable in solving specific problems that are difficult or impossible for classical computers. Research in this area is ongoing and has the potential to uncover new possibilities

for NISQ devices and bridge the gap between classical and fully-fledged fault-tolerant quantum computing.

1.2 Challenges on quantum machine learning

Quantum machine learning is a rapidly growing field that seeks to exploit the power of quantum computing for machine learning. One of the main approaches in quantum machine learning is the use of variational quantum algorithms [13], which are similar in structure to classical neural networks. These algorithms use a parameterized quantum circuit, known as an ansatz, to perform a specific task, such as classification or regression. The parameters of the ansatz are then optimized using classical optimization techniques to improve the performance of the quantum circuit. On the other hand, in the quantum kernel method [10, 11], the data is transformed into the quantum state by a quantum feature map, and the inner product is computed using quantum circuit. This allows for an efficient computation of the kernel function on a quantum computer and can lead to improved performance for certain types of data [12]. These quantum machine learning algorithms are both simple enough to be implemented on NISQ devices, which have limited number of qubits and a high level of noise. The field of quantum machine learning is still in its infancy, and many questions are yet to be answered. One of the most important issues is the barren plateau issue [14, 15, 16], which is a problem of gradient becoming exponentially small in the number of qubits before reaching local minima during optimization. This problem makes optimization hard. There is a lot of ongoing research in this field, including the development of new quantum algorithms, better understainding of their behavior and the investigation of the potential applications of quantum machine learning.

To this end, classical simulation techniques and visualization techniques play a crucial role. Classical simulation techniques are used to emulate the behavior of quantum systems on classical computers, which allows researchers to test and evaluate quantum algorithms without the need for expensive and error-prone quantum hardware. This is particularly important for quantum machine learning, as it allows researchers to understand the behavior and limitations of quantum algorithms and identify areas for improvement. Visualization techniques can be used to represent the quantum states in an intuitive way, which can facilitate the understanding of quantum algorithms and finding patterns in data and facilitate the interpretation. While we focus on applications of NISQ devices, even when fault-tolerant quantum computers are realized in the future, the classical simulation and visualization technique can help debugging quantum algorithms.

However, classical simulation and visualization of quantum states are both challenging due to exponential scaling of the required resources with the number of qubits. This is known as the curse of dimensionality, and it means that simulating even moderately sized quantum systems on classical computers quickly becomes infeasible. Furthermore, simulating the time evolution of a quantum

system, which is necessary for many quantum algorithms, is also a computationally demanding task. Visualizing quantum states is also a challenging task, as quantum states exist in high-dimensional Hilbert spaces, which can be difficult to comprehend and visualize. Additionally, quantum states, complex amplitudes, are complex mathematical objects that cannot be directly observed and must be inferred from measurements. This can lead to additional difficulties in understanding and visualizing quantum states.

1.3 Existing approaches on classical simulation and visualization technique

Variaous classical simulation softwares or platforms have been developed so far such as Qulacs [17], Qiskit [18], and Cirq [19]. However, the simulation time of classical simulation and the required memory for classical simulation exponentially increases with the number of qubits [17]. To accelerate the classical simulation, parallel computation by multiple devices [20, 21] and Graphics Processing Unit (GPU) [17, 18, 19], which has thousands of cores, is effective. Especially, the classical simulation on GPU is one order of magnitude faster than CPU [17], which would be helpful in developing and debugging quantum algorithms.

Despite its frequent appearance in quantum algorithms, classical simuation of continuous Hamiltonian dynamics remains a challenge because of the enormous amount of time steps required for simulation. More specifically, achieving high precision in classical simulation of quantum systems requires a large number of steps, such as hundreds or thousands, due to the Trotter-Suzuki decomposition [22, 23], which approximates the continuous dynamics of a Hamiltonian by breaking it into discrete time steps. This causes a serious increase in simulation time and makes performance analysis of quantum algorithms including Hamiltonian dynamics with relatively large number of qubits infeasible.

Regarding data visualization, it is used to analyze optimization problems in classical machine learning. For example, visualizing the loss landscapes and optimization trajectories is employed to analyze the effect of hyper-parameters in optimizer [24], and visualizing optimization trajectories in function space is used to analyze the effect of pre-training [25] and ensembles of deep neural networks [26]. As is also the case in classic, visualization of quantum states during optimizations can be used to gain insight into the optimization process and help to mitigate the barren plateau issue. For example, it could help to identify when the optimization process is getting stuck in poor local minima and allow researchers to avoid it. While the ground states of transverse field Ising model are clustered in two-dimensional space by dimensionality reduction technique in Ref. [27], the required memory for quantum states and the simulation cost there increases exponentially in the number of qubits since they suppose to use a classical computer for visualization. It is natural to use a quantum computer itself for visualization of quantum states.

1.4 Contributions of this thesis

In this thesis, we tackle two challenging problems, classical simulation and visualization techniques for quantum computers, to accelerate the development of quantum machine learning. Regarding classical simulation, we specifically focus on classical simulation of continuous Hamiltonian dynamics, which plays a crucial role in quantum algorithms but has high computational cost. Regarding visualization, we open up a field of research, quantum data visualization and also explore its applications. By addressing these two crucial areas, this research has the potential to significantly advance the field of quantum computing, especially quantum machine learning.

Our contributions in this thesis are as follows.

- In Chapter 3, we propose a method of accelerating a classical simulation of quantum dynamics. Utilizing the commuting relations between Pauli terms in Hamiltonian, we succeed in significantly accelerating a classical simulation of quantum dynamics compared to that by step-by-step simulation under Trotter-Suzuki approximation. This method allows us to accelerate the classical simulation of quantum algorithms, such as quantum circuit learning [9], quantum approximate optimization algorithm [8], and quantum annealing [28]. This chapter is based on the paper of [Y. Kawase and K. Fujii, arXiv:2206.11664, 2022].

- In Chapter 4, we propose a method of visualizing quantum states using a parameterized quantum circuit by mapping the quantum states in high-dimensional space into two-dimensional classical data with keeping the similarity between the quantum states. We optimize the parameterized quantum circuit applied to the quantum states to map the quantum states into two-dimensional classical data. We succeed in visualizing the quantum dynamics of transverse field Ising model according to the characteristics of the quantum states. This chapter is based on the paper of [Y. Kawase, K. Mitarai, and K. Fujii, Physical Review Research, vol. 4, 2022].

- In Chapter 5, we propose a visualization method of quantum states by quantum kernels [10, 11] instead of a quantum neural network. This is because the visualization method using a parameterized quantum circuit need to appropriately design an additional parameterized quantum circuit and choose observables to visualize data. To avoid the optimization of the additional parameterized quantum circuit, we employ quantum kernels to visualize data. By clustering quantum data, which is generated by embedding a classical hand-written dataset into quantum states using a quantum feature map, via the proposed visualization method, we find that quantum computer-based visualization achieves higher accuracy than that with a classical kernel method and is comparable to state-of-the-art neural network based approaches, while only using a small number of qubits. We also demonstrate a practical application, visualization of the

optimization process of variational algorithms, which would contribute to avoiding barren plateau issue. Note that the paper corresponding to this chapter is now writing and improving. The paper will be uploaded in arXiv.

In addition, to make this thesis self-consistent, Chapter 2 introduces the basic concepts, such as mathematical formulation of classical simulation of a quantum computer, simulation of quantum dynamics via Trotter-Suzuki decomposition, and visualization method in classical machine learning. Then, we conclude this thesis with conclusion and discussion in Chapter 6.

Chapter 2

Preliminary

In this chapter, we provide an introduction to the basics of quantum computing, including classical simulation of quantum dynamics, and visualization techniques in classical machine learning.

2.1 Basics of classical simulation of quantum dynamics

In this section, we explain the basic concepts of quantum computing and quantum dynamics. Specifically, we describe the description of quantum states and the quantum gates including how they are simulated on a classical computer. Then, we explain a classical simulation of continuous Hamiltonian dynamics via Trotter-Suzuki decomposition [22, 23].

2.1.1 Quantum state and quantum gates

In a quantum computer, a qubit is the counterpart of a bit in a classical computer. A single-qubit quantum state is represented by a vector, called a state vector,

$$|\psi\rangle = \alpha_0|0\rangle + \alpha_1|1\rangle, \quad \text{where } |0\rangle = \begin{pmatrix}1\\0\end{pmatrix}, |1\rangle = \begin{pmatrix}0\\1\end{pmatrix},$$

α_0 and α_1 are complex numbers, and these satisfy $|\alpha_0|^2 + |\alpha_1|^2 = 1$. When we measure the qubit, we get an outcome 0 with probability $|\alpha_0|^2$ and 1 with probability $|\alpha_1|^2$. The probability amplitudes of a quantum state with n qubits are described by a complex vector of 2^n dimensions:

$$|\psi\rangle = \alpha_{0\ldots00}|0\ldots00\rangle + \alpha_{0\ldots01}|0\ldots01\rangle + \cdots + \alpha_{1\ldots11}|1\ldots11\rangle, \tag{2.1}$$

where

$$|0\dots00\rangle = \begin{pmatrix}1\\0\\\vdots\\0\end{pmatrix}, |0\dots01\rangle = \begin{pmatrix}0\\1\\\vdots\\0\end{pmatrix}, \cdots, |1\dots11\rangle = \begin{pmatrix}0\\0\\\vdots\\1\end{pmatrix}.$$

An operation on qubits are represented by a unitary matrix. A single-qubit gate is represented by a two-by-two unitary matrix. When we apply it on the jth target qubit, the quantum state is updated by the following equation for all 2^{n-1} combinations of $i_k \in \{0,1\}, k = 1,\dots,j-1,j+1,\dots,n$:

$$\begin{pmatrix} \alpha'_{i_n\dots i_{j+1}0i_{j-1}\dots i_1} \\ \alpha'_{i_n\dots i_{j+1}1i_{j-1}\dots i_1} \end{pmatrix} = U_1 \begin{pmatrix} \alpha_{i_n\dots i_{j+1}0i_{j-1}\dots i_1} \\ \alpha_{i_n\dots i_{j+1}1i_{j-1}\dots i_1} \end{pmatrix}, \tag{2.2}$$

where the α'_i denotes the updated amplitudes of ith basis state and U_1 a two-by-two unitary matrix. The examples of the matrices are Pauli matrices as follows:

$$\begin{aligned} \sigma_0 = I \coloneqq \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \ \sigma_1 = X \coloneqq \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \\ \sigma_2 = Y \coloneqq \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \ \sigma_3 = Z \coloneqq \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \end{aligned} \tag{2.3}$$

Similarly, a two-qubit gate is defined by a four-by-four unitary matrix. When we apply the gate on the jth and kth qubits as the target qubits, the quantum state is updated by the following equation for all 2^{n-2} combinations of $i_l \in \{0,1\}$, where l satisfies $1 \le l \le n$ and $l \ne j,k$:

$$\begin{pmatrix} \alpha'_{i_n\dots i_{j+1}0i_{j-1}\dots i_{k+1}0i_{k-1}\dots i_1} \\ \alpha'_{i_n\dots i_{j+1}0i_{j-1}\dots i_{k+1}1i_{k-1}\dots i_1} \\ \alpha'_{i_n\dots i_{j+1}1i_{j-1}\dots i_{k+1}0i_{k-1}\dots i_1} \\ \alpha'_{i_n\dots i_{j+1}1i_{j-1}\dots i_{k+1}1i_{k-1}\dots i_1} \end{pmatrix} = U_2 \begin{pmatrix} \alpha_{i_n\dots i_{j+1}0i_{j-1}\dots i_{k+1}0i_{k-1}\dots i_1} \\ \alpha_{i_n\dots i_{j+1}0i_{j-1}\dots i_{k+1}1i_{k-1}\dots i_1} \\ \alpha_{i_n\dots i_{j+1}1i_{j-1}\dots i_{k+1}0i_{k-1}\dots i_1} \\ \alpha_{i_n\dots i_{j+1}1i_{j-1}\dots i_{k+1}1i_{k-1}\dots i_1} \end{pmatrix}, \tag{2.4}$$

where U_2 is a four-by-four unitary matrix.

Here, we describe Clifford operators (or transformations), which allows us to efficiently perform the classical simulation. The Clifford operators (or transformations) are operators that map a tensor product of Pauli matrices to another or the same tensor product of Pauli matrices. This kind of transformation includes the following quantum gates.

$$\begin{aligned} S = \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}, \quad H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \\ \mathrm{CNOT} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad \mathrm{CZ} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}. \end{aligned} \tag{2.5}$$

Table 2.1: This table shows the transformation of Pauli matrices by single-qubit gates in Clifford transformation. We can see that the Pauli matrices turn into the Pauli matrices by H and S gates. Note that we need to multiply -1 to transform the X or Y into Y.

	$U = H$	$U = S$
$UXU^\dagger$	Z	Y
$UYU^\dagger$	$-Y$	$-X$
$UZU^\dagger$	X	Z

Table 2.2: This table shows the transformation of Pauli terms by two-qubit gates in Clifford transformation. We can see that the Pauli terms are transformed into Pauli terms by CNOT and CZ gates. Note that we need to multiply -1 to transform the X_1Z_2 into Y_1Y_2.

	$U = \mathrm{CNOT}(1,2)$	$U = \mathrm{CZ}(1,2)$
$UX_1I_2U^\dagger$	X_1X_2	X_1Z_2
$UI_1X_2U^\dagger$	I_1X_2	Z_1X_2
$UX_1X_2U^\dagger$	X_1I_2	Y_1Y_2
$UY_1I_2U^\dagger$	Y_1X_2	Y_1Z_2
$UI_1Y_2U^\dagger$	Z_1Y_2	Z_1Y_2
$UY_1Y_2U^\dagger$	$-X_1Z_2$	X_1X_2
$UZ_1I_2U^\dagger$	Z_1I_2	Z_1I_2
$UI_1Z_2U^\dagger$	Z_1Z_2	I_1Z_2
$UZ_1Z_2U^\dagger$	I_1Z_2	Z_1Z_2

We show the example of transformation of the Pauli terms by the Clifford transformation in Tables 2.1 and 2.2. Note that we denote a single-qubit gate V on tth target qubit as V_t and CNOT gate on cth control qubit and tth target qubit as $\mathrm{CNOT}(c,t)$.

Since this class of operations can be efficiently simulatable on a classical computer by virtue of Gottesman-Knill theorem [29]. Clifford operations (or transformations) play a very important role in quantum computing. For example, they are used for quantum error correction to realize fault-tolerant quantum computation[2] and simultaneous measurement to reduce the number of measurements [30]. In our method, we use the Clifford transformation to partially diagonalize a Hamiltonian.

2.1.2 Classical simulation of quantum dynamics

The time evolution of a quantum state within a time-independent Hamiltonian H is described by the Schrödinger equation, and the solution is written by $|\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle$, where $|\psi(0)\rangle$ is the initial state. Since the dimensions of the Hamiltonian exponentially increases with the size of a quantum system, the simulation is intractable in general. However, the Hamiltonian can usually be

written as a sum over a polynomial number of local interactions with respect to the system size. Let H_j be a n-tensor product of Pauli operators, whose support is a finite number of qubits, with a certain coefficient. We rewrite the Hamiltonian by

$$H = \sum_{j=1}^{m} c_j H_j \qquad \text{where} \quad c_j \in \mathbb{R},\, H_j \in \{I, X, Y, Z\}^{\otimes n},$$

and, m denotes the number of terms of the Hamiltonian. We approximate the continuous Hamiltonian dynamics with the Hamiltonian H by Trotter-Suzuki decomposition [22, 23]. The first order Trotter-Suzuki decomposition is described by

$$e^{-iH\Delta t} = \Pi_{j=1}^{m} e^{-ic_j H_j \Delta t} + O(\Delta t^2),$$

where Δt denotes an infinitesimal time. Using the first order Trotter-Suzuki decomposition, the quantum state at time t can be written as

$$\begin{aligned} |\psi(t)\rangle = e^{-iHt}|\psi(0)\rangle &= \lim_{n_{TS}\to\infty} \left(\Pi_{j=1}^{m} e^{-ic_j H_j (t/n_{TS})}\right)^{n_{TS}} |\psi(0)\rangle \\ &\simeq \left(\Pi_{j=1}^{m} e^{-ic_j H_j \Delta t}\right)^{n_{TS}} |\psi(0)\rangle, \end{aligned} \tag{2.6}$$

where the n_{TS} denotes the number of Trotter steps and the $\Delta t = t/n_{TS}$ is the time step of the Trotter-Suzuki decomposition. Similarly, the second order Trotter-Suzuki decomposition is given by

$$e^{-iH\Delta t} = \left(\Pi_{j=1}^{m} e^{-ic_j H_j \Delta t/2}\right)\left(\Pi_{j=0}^{m-1} e^{-ic_{m-j} H_{m-j} \Delta t/2}\right) + O(\Delta t^3).$$

Using this equation, the time-evolution under the second order Trotter-Suzuki decomposition is described by

$$\begin{aligned} |\psi(t)\rangle &= e^{-iHt}|\psi(0)\rangle \\ &\simeq \left(\left(\Pi_{j=1}^{m} e^{-ic_j H_j (t/(2n_{TS}))}\right)\left(\Pi_{j=0}^{m-1} e^{-ic_{m-j} H_{m-j} (t/(2n_{TS}))}\right)\right)^{n_{TS}} |\psi(0)\rangle. \end{aligned}$$

The simulation of the infinitesimal time evolution operator $e^{-ic_j H_j \Delta t}$ is performed by multiplying the state vector by the unitary matrix, e.g. Eq. (2.2) and (2.4). From Eq. (2.6), the classical simulation of quantum dynamics requires a lot of steps to approximate the dynamics with higher accuracy. To reduce the classical simulation time, we propose a method of accelerating the classical simulation of Hamiltonian dynamics in Chapter 3.

2.2 Visualization method in classical machine learning

Here, we explain a visualization method in classical machine learning. Specifically, we describe a dimensionality reduction method, used in analyzing the optimization trajectories [25, 26].

2.2.1 t-distributed stochastic neighbor embedding (t-SNE)

t-distributed stochastic neighbor embedding (t-SNE)[31] is a non-parametric unsupervised machine learning method to visualize data. In concrete, t-SNE maps high-dimensional data $\boldsymbol{x}_i$ to lower dimensional data $\boldsymbol{y}_i$ by preserving the similarities between high- and low-dimensional data. The high-dimensional data similarities p_{ij} are defined using a Gaussian distribution:

$$\begin{aligned} p_{j|i} &= \frac{\exp\left(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2/2\sigma_i^2\right)}{\sum_{k\neq i}\exp\left(-\|\boldsymbol{x}_i - \boldsymbol{x}_k\|^2/2\sigma_i^2\right)}, \\ p_{ij} &= \frac{p_{j|i} + p_{i|j}}{2N}, \\ p_{ii} &= 0, \end{aligned} \tag{2.7}$$

where N denotes the number of data, and σ_i is a hyper-parameter set by a user from the efficient number of neighborhoods called perplexity $\mathrm{Perp}_i = 2^{-\sum_j p_{j|i}\log_2 p_{j|i}}$. The low-dimensional data similarities are defined using t-distribution with one degree of freedom:

$$\begin{aligned} q_{ij} &= \frac{(1 + \|\boldsymbol{y}_i - \boldsymbol{y}_j\|^2)^{-1}}{\sum_k \sum_{l\neq k}(1 + \|\boldsymbol{y}_k - \boldsymbol{y}_l\|^2)^{-1}}, \\ q_{ii} &= 0. \end{aligned} \tag{2.8}$$

In t-SNE, the initial positions of low-dimensional data are determined at random. The low-dimensional data are directly moved by minimizing the cost function C defined by Kullback-Leibler(KL) divergence between the similarities of data distributions in the high- and low- dimensional spaces:

$$C = KL(P\|Q) = \sum_i \sum_j p_{ij}\log\frac{p_{ij}}{q_{ij}}. \tag{2.9}$$

After minimizing the cost function by directly moving low-dimensional data $\boldsymbol{y}$, we can visualize data by plotting the resulting low-dimensional data. However, this model can not be applied to unseen data. We will see out-of-sample extension methods of t-SNE and expand the methods to visualize quantum states in Chapters 4 and 5.

2.3 Quantum machine learning methods

Here, we describe quantum machine learning methods of a variational quantum algorithm and a quantum kernel method, which are used in this thesis.

2.3.1 Variational quantum algorithm

A variational quantum algorithm is an algorithm that optimizes the parameters in a parameterized quantum circuit so as to minimize the cost function using

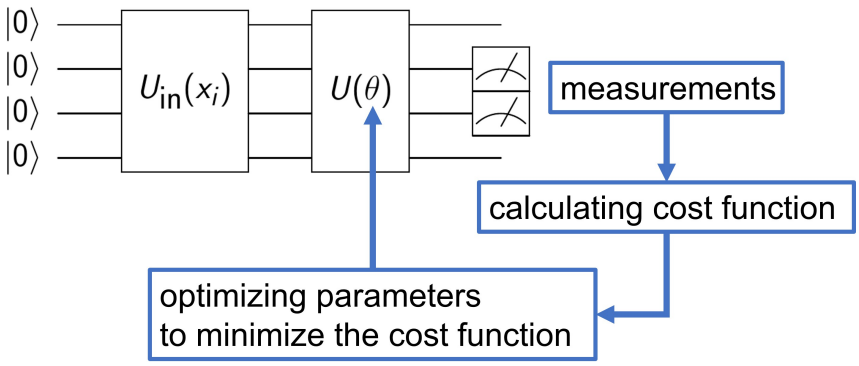


Figure 2.1: This figure shows the procedure of the variational quantum algorithm. We measure the final state of the parameterized quantum circuit. Using the measurement values, we calculate the cost function. Then we optimize the parameters in the parameterized quantum circuit.

the outputs of the parameterized quantum circuit [13]. The overview of this algorithm is described in Fig. 2.1. Specifically, this process is described as follows.

1. Measure the final state of the parameterized quantum circuit.
2. Calculate the cost function using the measurement values.
3. Optimize the parameters in the parameterized quantum circuit.

We repeat this process until we find the optimal value of the parameters.

2.3.2 Quantum kernel method

A quantum kernel method [10, 11] is a method that uses a quantum computer to estimate kernel functions, which are the similarity between data. Especially for classical data, the data is mapped via a quantum circuit to higher dimensional space. This process is similar to a classical kernel method. We can perform classification or regression tasks in the high dimensional space. The approaches to perform these tasks include direct estimation using the probabilities obtained by the measurements on quantum computer [11] and the use of support vector machines [10, 11, 12]. The basic concept of a kernel method is to consider linear models as

$$\boldsymbol{y}(\boldsymbol{x}) = \sum_{i=1}^{N} \boldsymbol{\alpha}_i k(\boldsymbol{x}, \boldsymbol{x}_i),$$

where N is the number of data and k is a kernel function. The parameters $\boldsymbol{\alpha}_i$ are often determined by matrix inversion of gram matrix, consisting of the elements

being $k(\boldsymbol{x}_j, \boldsymbol{x}_i)$, in supervised learning. Note that the gram matrix needs to satisfy the condition of a non-negative semidefinite matrix. In this thesis, we determine $\boldsymbol{\alpha}_i$ by optimizing a cost function because we tackle unsupervised machine learning problems.

Chapter 3

Fast classical simulation of Hamiltonian dynamics by simultaneous diagonalization using Clifford transformation with parallel computation

3.1 Introduction

Quantum many-body dynamics has been studied from both a practical perspective, such as its application to quantum information processing, and from a fundamental physical interest, such as quantum chaos [32], quantum information scrambling [33, 34], and discrete-time crystal [35]. While controlled quantum dynamics has been experimentally realized in various physical systems, classical simulation of quantum dynamics is still necessary to investigate detailed properties of quantum many-body dynamics and to validate experimental data. As pointed out by R. Feynman, classical simulation of quantum many-body dynamics is in general hard for classical computers, which is the reason why sophisticated classical simulation methods, such as exact diagonalization, full vector simulation, and tensor network approximation, have been developed. An exact diagonalization incurs high memory cost and execution time because the dimensions of the quantum system exponentially increase with the number of qubits. Moreover, for a time dependent Hamiltonian, diagonalization does not allow us to calculate the time evolution. A simulation based on tensor network can efficiently simulate a low-depth quantum circuit or quantum computation

with relatively small entanglement growth [36]. A quantum circuit to simulate highly complex quantum many-body dynamics with a large depth may not satisfy such conditions. Alternatively, we can use the full vector simulation, where quantum dynamics is simulated by updating a quantum state with infinitesimal time evolution operators obtained by Trotter-Suzuki decomposition [23]. Once we have enough memory to represent a quantum state, we can simulate the time evolution by updating the quantum state using a sparse matrix, without any diagonalization. This approach is most advantageous, as long as the system size stays within the limitation by memory, since the overhead scales linearly in time.

However, in practice, the Trotter-Suzuki decomposition requires at least hundreds or thousands of steps to achieve sufficient accuracy for simulation, which results in a long simulation time and makes this approach impractical for a large system even within the memory size. To settle this, we employ parallel computing architectures to accelerate simulation. One of the most ubiquitous parallel computing architectures is Graphics Processing Units (GPU), which has thousands of threads. In Ref. [37], the simulation time of the time evolution of quantum spin dynamics using the Trotter-Suzuki decomposition is 30 times faster by GPU than by CPU. However, this approach is only applicable for a simple Hamiltonian and cannot be applied for a general Hamiltonian, including an arbitrary set of Pauli strings. On the other hand, GPU-based simulation has proven its usefulness in simulating discretized quantum time evolution for quantum circuits in quantum computing [17, 18], while it is not optimized for simulating continuous quantum time evolution.

In this Chapter, by focusing on Hamiltonian dynamics of quantum many-body systems, we propose a general method to accelerate simulations and show that real-time dynamics of quantum many-body systems can be simulated in realistic time for 30 qubit-scale (approximately 10^9 dimensional state vector) quantum dynamics, where the size of the single GPU memory permits. To this end, the bottleneck is the huge number of steps originated by discretization via the Trotter-Suzuki decomposition for continuous time evolution, as we mentioned above. To relax this bottleneck, we diagonalize a part of a Hamiltonian so that multiple updates on a quantum state are done at once with minimal memory access. Since we do not diagonalize the whole Hamiltonian, the diagonalization can be done efficiently and systematically.

More specifically, in our method, Hamiltonian is partitioned into groups of mutually commuting Pauli terms, and each group is transformed by a unitary transformation consisting of Clifford gates to a diagonal matrix. Interestingly, exactly the same task is being investigated as a way to improve the efficiency of quantum algorithms. More precisely, the variational quantum eigensolver [38], which is one of the most promising quantum algorithms that work on near-term quantum computing devices, estimates an expectation value of Hamiltonian using a quantum computer. There, simultaneous diagonalization of mutually commuting Pauli terms has been employed to reduce the measurement complexity [30]. Furthermore, a similar technique is also applied to reduce the CNOT gates in a simulation of Hamiltonian dynamics on actual quantum de-

vices [39]. All these are intended to reduce quantum computational overhead, including quantum gates and measurements.

In this work, we apply and optimize the Hamiltonian partitioning technique for the purpose of accelerating classical simulation of Hamiltonian dynamics by simultaneously diagonalizing the mutually commuting Pauli terms. A matrix exponential of the diagonal matrix corresponding to mutually commuting Pauli terms is easily calculated and applied to a quantum state at once with minimum memory access. Note that without this simultaneous diagonalization, the time evolution operator must be applied for each term in the Hamiltonian with accessing the state vector each time. This simultaneous diagonalization allows us to reduce the number of steps required to simulate the Hamiltonian dynamics and overhead for accessing the exponentially large dimensional state vector. Thereby, the simulation time of the Hamiltonian dynamics is much reduced in total even though an additional computational cost is required to perform the unitary transformation for diagonalization.

To verify our algorithm and analyze the performance, we perform numerical experiments. In our numerical experiments, we simulate the time evolution under the Hamiltonian of transverse field Ising model and SYK Majorana fermion model [40, 41]. The former, the transverse field Ising model, is a prototypical model of quantum spin systems and is also widely employed for quantum information processing such as quantum annealing [28] and quantum approximate optimization algorithm [8]. The latter is another prototypical model for a highly complex quantum many-body system, which exhibits quantum informational scrambling [42]. We implement the proposed method for these Hamiltonians and compare the simulation time with straightforward Trotter-Suzuki decomposition simulated by fast and general purpose quantum computer simulator, Qulacs using both CPU and GPU [17]. We confirm that the simulation is accelerated with our proposed method regardless of the architectures. The simulation time of former model is 10 times faster, and that of the latter model is two times faster by our proposed method even when the number of qubits is 30. Furthermore, if the number of qubits is much smaller such as 15 qubits, 30 and 10 times faster, respectively.

3.2 Preliminary

3.2.1 Partitioning Hamiltonian terms

In the simulation of Hamiltonian dynamics with Trotter-Suzuki decomposition as Eq. (2.6), we multiply the state vector by a unitary matrix for each Trotter step. In the multiplication, the state vector must be read multiple times from the main memory if the size of the state vector is larger than that of the cache. Since the simulation time includes the time for the multiplication and memory access, we accelerate the simulation by reducing the number of memory access. To this end, once we read the amplitudes, we simultaneously multiply the exponentiation of multiple Hamiltonian terms. More precisely, we partition the

whole Hamiltonian terms into mutually commuting terms, and simultaneously time-evolving each group of the partitioned Hamiltonian with diagonalization.

Interestingly the same task is also done in a different context to reduce the number of measurements in variational quantum eigensolver (VQE) [30]. More precisely, the expectation value of the whole Hamiltonian terms in the group can be calculated from that of the part of the terms by diagonalizing the group of mutually commuting terms. Similarly, we diagonalize groups of mutually commuting terms and simultaneously time-evolve the terms in each group to accelerate the classical simulation. Since the simulation time depends on the number of groups, it is important to decrease the number of groups to accelerate the simulation. However, finding the partition of a given Hamiltonian with the minimum number of groups is an NP-hard problem [43]. To avoid this, we obtain the approximation of the partition by a greedy method.

In concrete, the procedure of the greedy method is described in Algorithm 1. First, we pick up a Pauli term of a given Hamiltonian that is not grouped yet and defined a new group starting from it. Next, from those that are not yet grouped, we search for a Pauli term that is mutually commuting with all terms in this group and add it to this group until there is no further commuting term. We repeat this procedure and obtain groups of mutually commuting Pauli terms. The computational cost of this greedy process is $O(m^2)$.

Algorithm 1 Implementation of greedily searching mutually commuting groups

```
m ← the number of Hamiltonian terms
G = {} : an empty set of groups
for i = 1 to m do
  the ith Hamiltonian term is regarded as a group.
  for j = i + 1 to m do
    if j(> i)th Hamiltonian term is mutually commuting with all the terms
    in the group and has not already included in any other groups then
      the term is added to the group.
    end if
  end for
  add the group to the set G
end for
return G
```

3.2.2 Tableau representation of Pauli operators and Clifford operations

In order to perform a faster classical simulation, we have to find the Clifford transformation that diagonalizes the mutually commuting Pauli terms efficiently. To this end, we make a tableau consisting of the binary representation of Pauli terms, and search for the Clifford transformation by applying it to the tableau systematically. Once we find the Clifford transformation, we can also

know the diagonalized terms. Here we explain the binary representation of Pauli terms and how the Clifford transformation acts on them.

First we describe the binary representation of Pauli terms in the Hamiltonian. Suppose the terms in the jth group of mutually commuting terms are written by

$$P_j = \sum_{k=1}^{m_j} c_{j,k} P_{j,k}, \qquad \text{where } c_{j,k} \in \mathbb{R}, P_{j,k} \in \{I, X, Y, Z\}^{\otimes n}, \tag{3.1}$$

m_j is the number of terms in jth group. Since $Y = iXZ$ holds, each Pauli term $P_{j,k}$ is also written by

$$P_{j,k} = i^{s_k} \otimes_{l=1}^{n} X^{x_{k,l}} Z^{z_{k,l}}, \qquad \text{where } s_k \in \{0,1,2,3\}, x_{k,l} \in \{0,1\}, z_{k,l} \in \{0,1\}. \tag{3.2}$$

We represent the terms in $P_{j,k}$ by $x_{k,l}$ and $z_{k,l}$ as the length $2n$ of bit string $x_{k,1} \dots x_{k,n} z_{k,1} \dots z_{k,n}$. For example, $X_1 Y_2 Z_3$ is represented by 110011. By arranging the bit strings, we can make a tableau consisting of m_j rows and $2n$ columns as

$$\left[\begin{array}{ccc|ccc} x_{1,1} & \cdots & x_{1,n} & z_{1,1} & \cdots & z_{1,n} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ x_{m_j,1} & \cdots & x_{m_j,n} & z_{m_j,1} & \cdots & z_{m_j,n} \end{array}\right].$$

Note that we say a set of binary strings is linearly independent if each of the binary strings in the set cannot be represented by exclusive-OR between the other binary strings. The coefficient i^{s_k} is caused by Pauli Y. From Tables 2.1 and 2.2, the coefficient whose original Pauli term has Pauli Y turns into $+1$ or -1 by Clifford transformation. So, in the classical simulation, we only need to track the sign. We add the column in $2n+1$th column of the tableau, and initialize the elements r_k with $r_k = 0$. In this way, we obtain the tableau

$$\left[\begin{array}{ccc|ccc|c} x_{1,1} & \cdots & x_{1,n} & z_{1,1} & \cdots & z_{1,n} & r_1 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ x_{m_j,1} & \cdots & x_{m_j,n} & z_{m_j,1} & \cdots & z_{m_j,n} & r_{m_j} \end{array}\right], \tag{3.3}$$

where $r_k \in \{0,1\}$. We call the first n columns of the tableau as X block and the following n columns of the tableau as Z block.

Next, we explain how to update the tableau under the action of each elementary Clifford transformation [39, 29]. We summarize them in Table 3.1, and depict the image of the transformation in Fig. 3.1. The binary tableau is updated by each Clifford gate according to the following procedure:

1. *H gate*
   If we apply H gate on tth qubit, we update r_k by $r_k = r_k \oplus x_{k,t} z_{k,t}$. Then we update the X block and Z block by swapping $x_{k,t}$ and $z_{k,t}$ for all $k = 1, \dots, m_j$.

Table 3.1: This table shows the Clifford operations on the binary tableau. Each $x_{k,l}$ and $z_{k,l}$ indicates the element in the kth row and lth column of the X block and Z block, respectively. In the table, $\oplus$ denotes exclusive-OR.

gate	sign	operation
H(t)	$r_k = r_k \oplus x_{k,t} z_{k,t}$	swap $x_{k,t}$ and $z_{k,t}$
S(t)	$r_k = r_k \oplus x_{k,t} z_{k,t}$	$z_{k,t} = z_{k,t} \oplus x_{k,t}$
CNOT(c,t)	$r_k = r_k \oplus x_{k,c} z_{k,t} (x_{k,t} \oplus z_{k,c} \oplus 1)$	$x_{k,t} = x_{k,t} \oplus x_{k,c}$ and $z_{k,c} = z_{k,c} \oplus z_{k,t}$
CZ(c,t)	$r_k = r_k \oplus x_{k,c} x_{k,t} (z_{k,t} \oplus z_{k,c})$	$z_{k,t} = z_{k,t} \oplus x_{k,c}$ and $z_{k,c} = z_{k,c} \oplus x_{k,t}$

2. *S gate*
   If we apply S gate on tth qubit, we update r_k by $r_k = r_k \oplus x_{k,t} z_{k,t}$. Then we update the Z block by $z_{k,t} = z_{k,t} \oplus x_{k,t}$ for all $k = 1, \ldots, m_j$.

3. *CNOT gate*
   If we apply CNOT gate on cth control qubit and tth target qubit, we update r_k by $r_k = r_k \oplus x_{k,c} z_{k,t} (x_{k,t} \oplus z_{k,c} \oplus 1)$. Then we update the X block and Z block by $x_{k,t} = x_{k,t} \oplus x_{k,c}$ and $z_{k,c} = z_{k,c} \oplus z_{k,t}$ for all $k = 1, \ldots, m_j$.

4. *CZ gate*
   If we apply CZ gate on cth control qubit and tth target qubit, we update r_k by $r_k = r_k \oplus x_{k,c} x_{k,t} (z_{k,t} \oplus z_{k,c})$. Then we update the Z block by $z_{k,t} = z_{k,t} \oplus x_{k,c}$ and $z_{k,c} = z_{k,c} \oplus x_{k,t}$ for all $k = 1, \ldots, m_j$.

In the above procedure, we denote the exclusive-OR as $\oplus$. These operations can be derived from the Clifford operations on Pauli terms in Tables 2.1 and 2.2. Using these elementary operations on the tableau, we employ them to construct a Clifford transformation to diagonalize Pauli terms. Note that "diagonal" here means that the tableau only contains Pauli Z components after the transformation, which means that the corresponding exponentiation is also diagonal in the computational basis. A concrete construction, which is optimized for classical simulation in the proposed method, will be presented in the next section.

3.3 Simultaneous diagonalization for partitioned Hamiltonian

Here we explain our main idea to accelerate the classical simulation of quantum dynamics by partial diagonalization. In the following, we describe our algorithm of partial diagonalization.

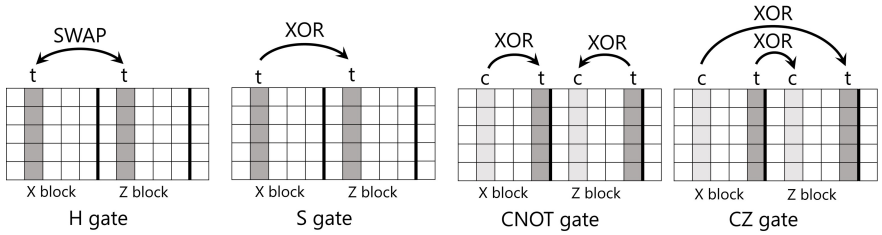


Figure 3.1: This figure shows the Clifford operations on the binary tableau of X and Z blocks. In the figure, each t and c represents target and control qubits. We represent the tth column with dark gray and cth column with ligth gray.

3.3.1 Preparation

First, we need to partition the given Hamiltonian terms into mutually commuting terms. As we mentioned in Sec. 3.2.1, we take a greedy approach to do the partitioning. Then, we transform the partitioned Hamiltonian into a binary representation tableau as in Sec. 3.2.2. We take a maximum size of the set of linearly independent binary strings. To search the set, we zero out the lower-triangular matrix of the tableau by applying exclusive-OR between rows. The detail of the algorithm is described in Appendix A.1.1. As a result, we obtain a set of linearly independent Pauli strings in P_i. Note that the binary strings which are obtained from the procedure described in Appendix A.1.1 are not directly used in the following process, since they are not the same as the original mutually commuting Pauli strings P_i via the transformation. The binary tableau consists of n rows and $(2n+1)$ columns. If the rank of the tableau is less than n, we fill the rest of the rows in the tableau with zeros. We use this tableau to find a series of Clifford transformations to simultaneously diagonalize mutually commuting terms P_i.

3.3.2 Partial diagonalization of Hamiltonian

Here we explain our algorithm for finding a series of Clifford transformations to simultaneously diagonalize the mutually commuting terms. Since the diagonalized terms are represented by Pauli Z and I, we transform the binary tableau so that all the elements in the X block are zero. Namely, we will search a series of Clifford transformations to zero out all the elements in the X block. The outline of our algorithm is described in Algorithm 2 and the proof of our algorithm is given in Appendix A.1.2.

(i) *Maximizing the rank of the X block*

We will maximize the rank of the X block by applying H gates. The procedure is described in Algorithm 3. Using this algorithm, we compare the rank of the X block with that applied by H gate on the ith target qubit. If the latter rank is larger than the former, we decide to apply the H gate

on the ith target qubit. When calculating the rank, we use an algorithm similar to the algorithm of searching linearly independent binary strings as in Appendix A.1.1. In the procedure, we need to consider the X block only instead of the whole tableau. We repeat these operations for $i = 1$ to $i = n$.

(ii) *Zeroing out the upper triangular matrix of the X block*

We will zero out the upper triangular matrix of the X block by CNOT gates. The process is described in Algorithm 4. If the ith diagonal element of the X block is zero, we search and swap the row with the $j(> i)$th row whose ith column element is one. If the search and swap are performed, the element of the ith row with the ith column is one. Then, if the element of the $k(> i)$th column in the ith row represents one, we zero out it with $\mathrm{CNOT}(i, k)$. We repeat these operations for $i = 1$ to $i = n$.

(iii) *Zeroing out the Z block*

We will zero out the Z block by CZ gates and S gates. More concretely, CZ gates are used to zero out the non-zero elements in the non-diagonal elements of the Z block and, S gates are used to zero out the non-zero elements in the diagonal elements of the Z block. We describe the procedure in Algorithm 5. If the $j(\neq i)$th column of the ith row in the Z block is one, $\mathrm{CZ}(i, j)$ is applied to zero out the element. Then, if both the ith diagonal elements of the X block and the Z block are ones, we apply S gate on the ith qubit. We repeat this process for $i = 1$ to $i = n$, then the Z block becomes a zero matrix.

(iv) *Turning the Pauli X components into the Pauli Z components*

By the previous process, the non-zero elements only exist in the X block. We transform the Pauli X components into Pauli Z components by H gates. More precisely, if the elements of the ith column in the X block have a non-zero element, we apply H gate on the ith qubit. We repeat it for $i = 1$ to $i = n$, and then the tableau only has the Pauli Z components. The process is described in Algorithm 6.

In this way, we can simultaneously diagonalize mutually commuting terms in the partitioned Hamiltonian. Note that to reduce the number of operations in the above procedure (ii) and (iii), we omit the operations if all the terms on ith qubit in the group consist of Pauli Z and I or Pauli X and I components. This is because if all the terms on ith qubit in the group consist of Pauli Z and I, the terms have already partially diagonalized. If all the terms on ith qubit in the group consist of Pauli X and I, those can be partially diagonalized by applying H_i.

Here we discuss the computational cost of our algorithm. In preprocessing of searching for a set of linearly independent binary strings by Algorithm 7, this process costs $O(nm_j)$. Step (i) also includes a process of searching for linearly independent binary strings in each iteration to calculate the rank of

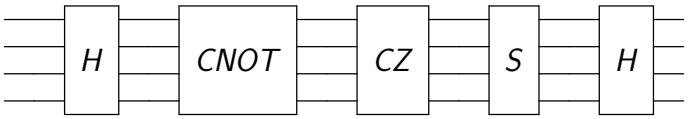


Figure 3.2: This figure shows the quantum circuit made by our algorithm. The quantum circuit consists of a series of H, CNOT, CZ, S, and H gates. In detail, the first H gates are used in the procedure (i) in our proposed method to maximize the rank of X block. The CNOT gates are used to zero out the upper triangular matrix of the X block in the procedure (ii). The CZ and S gates are used to zero out the Z block in the procedure (iii). Since these gates do not depend on the order, these gates can be arranged in a series of CZ gates and S gates. The last H gates are used to transform the Pauli X to Z in the procedure (iv).

X block, so the cost is $O(n^3)$. Since the cost of steps (ii) and (iii) depend on the number of elements in X block and Z block, respectively, the cost is $O(n^2)$. The cost of the last step (iv) depends on the number of qubits, so the cost is $O(n)$. Therefore, the computational cost of our algorithm is $O(n^3 + nm_j)$ for a single group. As a result, the total cost of our algorithm for all groups is $O(n^3 n_g + nm)$, where n_g denotes the number of groups. This total cost is much less than that of updating a quantum state because the cost of updating a quantum state exponentially increases with the number of qubits, as we will explain in Sec. 3.3.4. Let us consider an example of a fully connected transversal Ising model. From $n_g = 2$ and $m = O(n^2)$, the cost for the preprocess is $O(n^3)$ and that for searching Clifford transformations by tableau is $O(n^3)$, so the total cost is also $O(n^3)$. Note that to make the simulation faster, we perform the swap and XOR operations using decimal integers, representing binary strings in columns.

Algorithm 2 Finding the Clifford transformation of simultaneously diagonalizing mutually commuting terms

1: **for** each group of mutually commuting terms **do**
2: Search the linearly independent binary strings in the group P_j.
3: Make a binary representation tableau with the linearly independent binary strings.
4: (i) Maximizing the rank of the X block by H gates
5: (ii) Zeroing out upper triangular matrix of the X block by CNOT gates
6: (iii) Zeroing out the Z block by CZ and S gates
7: (iv) Turning the Pauli X components into the Pauli Z components by H gates
8: **end for**

Algorithm 3 Maximizing the rank of the X block

```
for all i such that 1 ≤ i ≤ n do
    r ← the rank of the X block
    r' ← the rank of the X block if we apply H(i)
    if r < r' then
        we apply H(i)
    end if
end for
```

Algorithm 4 Zeroing out upper triangular matrix of the X block

```
for all i such that 1 ≤ i ≤ n do
    if the element of ith row and ith column in the X block is zero then
        if we find the element of j(> i)th row and ith column in the X block
        is one then
            we swap the ith row with the jth row.
        end if
    end if
    for all j such that i < j ≤ n do
        if the element of the ith row and jth column is one then
            we apply CNOT(i, j).
        end if
    end for
end for
```

The Clifford transformation made by our algorithm consists of a series of H, CNOT, CZ, S, and H gates, as shown in Fig. 3.2. In concrete, the H gates act on different target qubits, and the S gates also. The CNOT gates are arranged in ascending order of control qubits. In addition, the CNOT gates with the same control qubits are arranged in ascending order of target qubits. The CZ gates are also arranged in the same manner. The simulation of applying Clifford transformation can be faster by simultaneously applying the CNOT gates with the same control qubits, CZ gates, and S gates, respectively. These gates can be implemented in the local operation, such as diagonal operators and the operation of swapping the two amplitudes. We will explain our implementation in Sec. 3.3.4.

Let us consider a concrete example of diagonalizing mutually commuting terms as

$$\begin{aligned} H = {} & Y_1 I_2 X_3 I_4 + I_1 X_2 I_3 Y_4 + X_1 Z_2 Y_3 Z_4 + Y_1 Z_2 X_3 Z_4 \\ & + X_1 I_2 Y_3 I_4 + I_1 Y_2 I_3 X_4 + Z_1 X_2 Z_3 Y_4 + Z_1 Y_2 Z_3 X_4. \end{aligned} \tag{3.4}$$

In this Hamiltonian, one of the sets of linearly independent terms is $Y_1 I_2 X_3 I_4$, $I_1 X_2 I_3 Y_4$, $X_1 Z_2 Y_3 Z_4$ and $Y_1 Z_2 X_3 Z_4$. These binary representations are 10101000, 01010001, 10100111, and 10101101, respectively. We make the tableau of these

Algorithm 5 Zeroing out the Z block

for all i such that $1 \leq i \leq n$ **do**
  for all j such that $1 \leq j \leq n$ **do**
    if the element of the ith row and the $j(\neq i)$th column in Z block is one **then**
      we apply $\mathrm{CZ}(i, j)$.
    end if
  end for
  if the element of ith row and ith column in the Z block is one **then**
    we apply $S(i)$.
  end if
end for

Algorithm 6 Turning the Pauli X components into the Pauli Z components

for all i such that $1 \leq i \leq n$ **do**
  if the elements of the ith column in the X block has a non-zero element **then**
    we apply $H(i)$.
  end if
end for

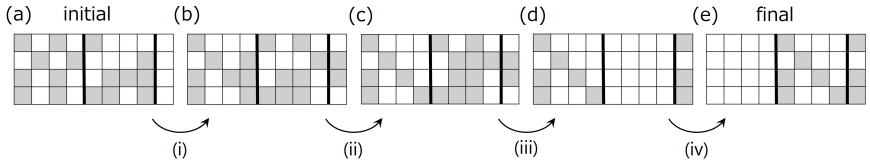


Figure 3.3: This figure shows the process of simultaneous diagonalization. The initial tableau (a) shows the binary representation of $Y_1I_2X_3I_4, I_1X_2I_3Y_4, X_1Z_2Y_3Z_4$ and $Y_1Z_2X_3Z_4$. The binary representation of these terms are 10101000, 01010001, 10100111, and 10101101, respectively. The white and gray cells represent zero and one in the table, respectively. After step (i) of maximizing the rank of X block with applying Hadamard gate on 3rd and 4th qubit, the tableau becomes (b). In step (ii), to zero out the upper triangular matrix of the X block, $\mathrm{CNOT}(2, 4)$, and $\mathrm{CNOT}(3, 4)$ are applied. The tableau goes (c). To zero out the Z block in step (iii), $\mathrm{CZ}(1, 3)$, $\mathrm{CZ}(2, 3)$, $\mathrm{CZ}(2, 4)$, $S(1)$, and $S(2)$ are applied. The tableau becomes (d). Finally, after applying H gates on all qubits in step (iv), the terms are diagonalized, as in (e).

terms as Fig. 3.3 (a). In the figure, we describe zero as a white cell and one as a gray cell. The rest of Fig. 3.3 shows how the binary tableau is transformed under the procedures above. In concrete, in the procedure (i), to maximize the rank of the X block, $H(3)$ and $H(4)$ are applied. Next, CNOT$(2,4)$ and CNOT$(3,4)$ are applied to zero out the upper triangular matrix of the X block in the procedure (ii). Then, CZ$(1,3)$, CZ$(2,3)$, and CZ$(2,4)$ are applied to zero out the non-diagonal element of the Z block, and $S(1)$ and $S(2)$ are applied to zero out the diagonal element of the Z block in the procedure (iii). Finally, the Pauli X components are transformed into Pauli Z components by $H(1)$, $H(2)$, $H(3)$, and $H(4)$ in the procedure (iv). The transformation C is written by the following equation

$$\begin{aligned} C &= (H_1H_2H_3H_4)(S_1S_2)(\mathrm{CZ}(1,3)\mathrm{CZ}(2,3)\mathrm{CZ}(2,4)) \\ &\quad (\mathrm{CNOT}(3,4)\mathrm{CNOT}(2,4))(H_3H_4) \end{aligned}$$

Thus, the Clifford transformation consists of a series of H, CNOT, CZ, S, and H gates. By this transformation, the Hamiltonian written in Eq. (3.4) is diagonalized as

$$H = C^\dagger(-Z_1 + Z_2 - Z_1Z_3 - Z_1Z_4 - Z_1Z_3Z_4 + Z_2Z_4 + Z_2Z_3Z_4 + Z_2Z_3)C.$$

We will explain how such diagonalization can accelerate classical simulation in the following sections.

3.3.3 Classical simulation of quantum dynamics with simultaneous diagonalization

In this section, we describe how to perform classical simulation by our proposed method. We diagonalize the Hamiltonian terms by the Clifford transformation C_j, which has been found in the previous argument, for a jth group of mutually commuting terms P_j as

$$P_j = \sum_{k=1}^{m_j} c_{j,k} C_j^\dagger (-1)^{r_{j,k}} \Lambda_{j,k} C_j, \qquad \text{where } r_{j,k} \in \{0,1\}, \Lambda_{j,k} \in \{I,Z\}^{\otimes n}.$$

In the above equation, $r_{j,k}$ is the r_k in the binary representation tableau of jth group, and $\Lambda_{j,k}$ is a diagonal operator of kth term in jth group. By the Clifford transformation, the whole Hamiltonian H is described by

$$\begin{aligned} H &= \sum_{j=1}^{m} c_j H_j = \sum_{j=1}^{n_g} P_j = \sum_{j=1}^{n_g}\sum_{k=1}^{m_j} c_{j,k} P_{j,k} \\ &= \sum_{j=1}^{n_g} C_j^\dagger \left(\sum_{k=1}^{m_j} (-1)^{r_{j,k}} c_{j,k} \Lambda_{j,k} \right) C_j =: \sum_{j=1}^{n_g} C_j^\dagger \Lambda_j C_j, \end{aligned} \tag{3.5}$$

where Λ_j is a diagonal operator of jth group and n_g is the number of groups of mutually commuting terms. Note that each diagonalized operator $\Lambda_{j,k}$ is

obtained by applying the Clifford operation to the binary tableau consisting of all the terms in the group.

To perform classical simulation, we use Trotter-Suzuki decomposition as in Sec. 2.1.2. Let us Δt as an infinitesimal time, an infinitesimal time-evolution by first-order Trotter-Suzuki decomposition is described by

$$\begin{aligned} e^{-iH\Delta t} &= \Pi_{j=1}^{n_g}\Pi_{k=1}^{m_j} e^{-ic_{j,k}P_{j,k}\Delta t} + O(\Delta t^2) \\ &= \Pi_{j=1}^{n_g} C_j^\dagger \left(\Pi_{k=1}^{m_j} e^{-i(-1)^{r_{j,k}}c_{j,k}\Lambda_{j,k}}\right) C_j + O(\Delta t^2). \end{aligned}$$

So, the quantum state at time t time-evolved under the Hamiltonian is described by

$$\begin{aligned} |\psi(t)\rangle = e^{-iHt}|\psi(0) &= \lim_{n_{TS}\to\infty}\left(\Pi_{j=1}^{n_g} e^{-iP_j(t/n_{TS})}\right)^{n_{TS}} |\psi(0)\rangle \\ &\simeq \left(\Pi_{j=1}^{n_g} C_j^\dagger e^{-i\Lambda_j\Delta t} C_j\right)^{n_{TS}} |\psi(0)\rangle. \end{aligned} \tag{3.6}$$

Similarly, an infinitesimal time-evolution by second-order Trotter-Suzuki decomposition is described by

$$\begin{aligned} e^{-iH\Delta t} &= \left(\Pi_{j=1}^{n_g}\Pi_{k=1}^{m_j} e^{-ic_{j,k}P_{j,k}\Delta t/2}\right) \\ &\quad \left(\Pi_{j=0}^{n_g-1}\Pi_{k=0}^{m_j-1} e^{-ic_{n_g-j,m_j-k}P_{n_g-j,m_j-k}\Delta t/2}\right) + O(\Delta t^3) \\ &= \left(\Pi_{j=1}^{n_g} C_j^\dagger \Pi_{k=1}^{m_j} e^{-i(-1)^{r_{j,k}}c_{j,k}\Lambda_{j,k}\Delta t/2} C_j\right) \\ &\quad \left(\Pi_{j=0}^{n_g-1} C_{n_g-j}^\dagger \Pi_{k=1}^{m_j} e^{-i(-1)^{r_{n_g-j,k}}c_{n_g-j,k}\Lambda_{n_g-j,k}\Delta t/2} C_{n_g-j}\right) + O(\Delta t^3). \end{aligned}$$

In this way, we can simulate time-evolution under second-order Trotter-Suzuki decomposition. This equation implies that the simulation time under the second-order Trotter-Suzuki decomposition is about twice that under the first-order Trotter-Suzuki decomposition. Since whether the use of first-order or second-order Trotter-Suzuki decomposition depends on the Trotter error, in terms of confirming the effect of acceleration of the simulation by our algorithm, it does not matter what order of Trotter-Suzuki decomposition we use. For simplicity, we employ first-order Trotter-Suzuki decomposition in our numerical experiments. The partial diagonalization of the Hamiltonian allows us to update the quantum state by Clifford transformation and the exponentiation of diagonal operators. We can accelerate the classical simulation, if the simulation cost for the Clifford transformation is much smaller than that reduced by diagonalization of the commuting Pauli terms for minimizing the memory accesses. As we will see below, implementations of Clifford transformations and diagonal operators are further optimized to reduce the computational cost.

3.3.4 Implementations

To perform the classical simulation, we use fast quantum circuit simulator, "qulacs"[17]. In "qulacs," each single gate of H, S, CNOT, and CZ gates is

implemented, but these implementations are not optimal for our method. In our method, the same kind of gates appears continuously, such as consecutive S gates, CZ gates, CNOT gates with the same control qubit, and the exponentiation of diagonal operators in Λ_j in Eq. (3.5). So, we implement them with the same concept of implementation in "qulacs."

1. *CNOT gates*

   Here we explain how to update a state vector by the consecutive CNOT gates with the same control qubit. Since the update of the state vector by these operations is similar to the update by a single CNOT gate, we will first describe the update of the quantum state by a single CNOT gate. The matrix of CNOT gate is written in Eq. (2.5). The implementation is realized by swapping the amplitudes between the $|10\rangle$ and $|11\rangle$. The control qubits of both basis states are one and the relation between the target qubits of the basis states is bit-flipped. This is easily expanded to the multiple CNOT gates with the same control qubit. Let us consider the simulation of applying $\mathrm{CNOT}(c, t_1)$, $\mathrm{CNOT}(c, t_2)$, ..., and $\mathrm{CNOT}(c, t_k)$ gates to the quantum state. The implementation is to swap the amplitudes between basis states whose cth qubits are both one and bitstrings consisting of $t_1, t_2, \ldots, t_k$ th qubits have a relation of bit-flip. More concretely, let us denote the bitstring of basis state as $b_n \ldots b_1$, then those bitstrings consisting of target qubits are written by $b_{t_1} b_{t_2} \ldots b_{t_k}$ and $(b_{t_1} \oplus 1)(b_{t_2} \oplus 1) \ldots (b_{t_k} \oplus 1)$.

2. *S and CZ gates*

   Here we explain the implementation of simultaneously applying S and CZ gates. First, we describe an implementation of applying multiple S gates. As we described in Eq. (2.5), S gate is a diagonal matrix. Let us consider the simulation of applying $S(t_1)$ and $S(t_2)$. We take a bitstring consisting of the target qubits t_1 and t_2 of the basis states. By counting the ones in the bitstring, we can calculate the coefficient. For each basis state $a_{b_n \ldots b_1} |b_n \ldots b_1\rangle$, where $a_{b_n \ldots b_1} \in \mathbb{C}, b_j \in \{0, 1\}$, the amplitudes $a_{b_n \ldots b_1}$ are updated by multiplying $i^{b_{t_1} + b_{t_2}}$. Similarly, we can simultaneously apply more than two S gates to the quantum state. Let us denote the target qubits t_k, we update the amplitudes $a_{b_n \ldots b_1}$ by multiplying $i^{\sum_k b_{t_k}}$. Next, we explain an implementation of applying multiple CZ gates. As we described in Eq. (2.5), the CZ gate is also a diagonal matrix. The implementation of the single CZ gate on a control qubit c and a target qubit t is to multiply -1 to the amplitudes whose basis states are both ones on the control qubit c and target qubit t. When applying multiple CZ gates, for each basis state, we compute the coefficient over all CZ gates by the same operation as a single CZ gate in advance. In concrete, by denoting the control and target qubits of jth CZ gate as c_j and t_j, and the bits of the control and target qubits of the basis state as b_{c_j} and b_{t_j}, the coefficient is written by $(-1)^{\sum_j b_{c_j} b_{t_j}}$. Then we update each amplitude by multiplying it. In this way, we can reduce the number of

memory access to the state vector. Since the implementation of S and CZ gates is based on counting the number of ones in the bitstring consisting of target qubits and control qubits of basis states, we can simultaneously apply all S and CZ gates to each basis state by calculating the coefficients and then applying them to the amplitudes.

3. *diagonal operator* Λ_j

   Here we explain how to efficiently update the quantum state by the diagonal operator Λ_j. By the partial diagonalization, the terms in the partitioned Hamiltonian are written by the n-fold tensor products of Pauli Z and I. The simulation of a single diagonal operator $(-1)^{r_1} c_1 \otimes_{j=1}^{n} Z^{z_j}$, where $r_1, z_j \in \{0,1\}, c_1 \in \mathbb{R}$ is performed as follows. For each basis state $a_{b_n \dots b_1} |b_n \dots b_1\rangle$, where $a_{b_n \dots b_1} \in \mathbb{C}, b_j \in \{0,1\}$, we count the ones in the bit string of basis state whose indexes are the same as the target qubit indexes of Pauli Zs in the operator, such as $\sum_{j=1}^{n} b_j z_j$. We multiply the amplitudes $a_{b_n \dots b_1}$ by the coefficient $e^{-i\Delta t c_1 (-1)^{r_1 + \sum_{j=1}^{n} b_j z_j}}$. Similarly, the simulation of exponentiation of multiple diagonal operators is to calculate the coefficients for all the terms in the partitioned Hamiltonian, and then read the amplitudes and multiply the amplitudes by the coefficients. In concrete, let us consider the diagonal operator Λ_j in Eq. (3.5), such as

$$\Lambda_j = \sum_{k=1}^{m_j} (-1)^{r_{j,k}} c_{j,k} \left(\otimes_{l=1}^{n} Z^{z_{k,l}} \right), \quad \text{where } z_{k,l} \in \{0,1\}.$$

   We multiply the amplitudes by

$$e^{-i\Delta t \sum_{k=1}^{m_j} c_{j,k} (-1)^{r_{j,k} + \sum_{l=1}^{n} z_{k,l} b_l}}$$

   for each basis state $a_{b_n \dots b_1} |b_n \dots b_1\rangle$, where $b_l \in \{0,1\}$, for $l = 1, \dots, n$ and $a_{b_n \dots b_1} \in \mathbb{C}$.

Using these implementations, we reduce the number of memory access to the state vector. The number of memory access is $O(n2^n)$ by H gates, $O(2^n)$ by S and CZ gates, $O(n2^n)$ by CNOT gates, and $O(2^n)$ by an exponentiation of diagonal operator Λ_j. The total cost is $O(n2^n)$ for each group, and then $O(n_g n 2^n)$ for all groups. On the other hand, the simulation cost for applying each Pauli term in the Hamiltonian independently is $O(m2^n)$ with m being the number of Pauli terms. Therefore, our proposed method works well if the number of groups n_g is $1/n$ times smaller than that of the terms m. For example, in the case of a fully connected transversal Ising model, $n_g = 2$ and $m = O(n^2)$ and hence we can expect acceleration by the proposed method. For more complicated Hamiltonian with fermions, whether or not we can obtain any acceleration highly depends on the number of groups and cache efficiency, since the effect of the acceleration decays with the number of groups and the limited size of cache causes delay by the memory access to obtain coefficients or amplitudes.

Note that it is easy to parallelize the calculation of multiplying the state vector by diagonal matrices as S gate, CZ gate and exponentiation of diagonal operator Λ_j because the calculation can be performed independently on each basis state. Similarly, the calculation of multiple CNOT gates with the same control qubit depends only on pairs of two basis states, so it is also easy to perform the simulation in parallel. By the parallelization, the simulation is performed in $O(n_g n\lceil 2^n/n_{\text{thread}}\rceil)$, where n_{thread} denotes the number of threads and $\lceil 2^n/n_{\text{thread}}\rceil$ represents the number of cycles of the parallel execution. Similarly, the simulation for applying each Pauli term in the Hamiltonian is performed in $O(m\lceil 2^n/n_{\text{thread}}\rceil)$. So, the simulation time will be reduced by a factor of $O(nn_g/m)$.

3.4 Numerical experiments

In this section, we perform numerical experiments to validate the acceleration of classical simulation by our proposed method. We perform the classical simulation of Hamiltonian dynamics under the Trotter-Suzuki decomposition with the time step $\Delta t = t/m = 0.01$ according to Eq. (2.6) and Eq. (3.6). We compare the simulation time of one Trotter step with and without our proposed method for two models: the transverse field Ising model and SYK Majorana fermion model [40, 41]. The former model, transverse field Ising model, is used in quantum algorithms such as quantum annealing [28] and quantum approximate optimization algorithm [8]. The latter model, SYK Majorana fermion model, is a prototypical model for condensed matter physics describing a strongly correlated electron system. It shows interesting phenomena such as quantum chaos and quantum information scrambling [42, 44].

To perform the numerical experiments, we partition the given Hamiltonian into groups of mutually commuting terms, and find the Clifford transformations to diagonalize each group. The former Hamiltonian, transverse field Ising model, is easily partitioned with the minimum number of groups. Also, the Clifford transformation of simultaneously diagonalizing each group is easily found. On the other hand, the partition of the latter Hamiltonian, SYK Majorana fermion model, with the minimum number of groups and Clifford transformation for them are not easily found by hand, so we have to systematically find them with the algorithm as we described in Sec. 3.3.2. By the partitioned Hamiltonian and the Clifford transformation, we update the quantum state by simultaneously applying terms in each group to accelerate the simulation.

It is important whether the coefficients in the groups can be stored in the cache. In the case of the transverse field Ising model, the number of terms in the Hamiltonian is 465 at 30 qubits. Since our CPU has 32 kB of the L1 cache, which corresponds to four thousand floating points with double precision, the L1 cache can store all coefficients. Therefore, the simulation is expected to be performed fast. In the case of the Majorana fermion models, the number of terms at 28 qubits is about $367,000$. These terms with double precision floating points occupy 3 MB, which is more than the size of the L2 cache (1

MB) but less than that of the L3 cache (11 MB). So, the simulation is also expected to be performed fast. The effect of the acceleration is less than in the former case because the access to the L1 cache is faster than the L3 cache. For GPU, the L2 cache has about 6 MB, so the coefficients of the models can be stored in the L2 cache. Since GPU has large bandwidth and thousands of cores, it is expected to perform simulation faster than CPU. Note that the amplitudes of the exponentially large state vector are also stored in the cache during the simulation. If the amplitudes are not stored in the cache, the data must be obtained through the memory access, and then it causes an increase in simulation time.

To confirm the effect of the acceleration, we compare the simulation time between the two methods for each model. One of them is to apply exponentiation of each Hamiltonian term one by one as Eq. (2.6). The other one is to simultaneously apply exponentiation of terms in each group as Eq. (3.6). We call the former method a baseline method and the latter method a proposed method. In these simulations, we use the quantum circuit simulator "qulacs" [17] with our additional implementation described in Sec. 3.3.4, such as fast Clifford transformations and diagonal operations. Note that the implementation for CPU is parallelized by SIMD (Single Instruction Multiple Data) and OpenMP. The results are described in the following section.

3.4.1 Transverse field Ising model

The first model we will simulate is the transverse field Ising model, whose Hamiltonian is given by

$$H = \sum_{i<j} J_{ij} Z_i Z_j + \sum_i h_i X_i,$$

where the J_{ij} and h_i are randomly chosen from the uniform distribution between -1 and 1. The Hamiltonian term $J_{ij}Z_iZ_j$ is already diagonalized, and the term h_iX_i can be diagonalized by H_i. As we mentioned above, we perform the simulation with two methods: multiplying the exponentiation of each Hamiltonian term one by one and multiplying the exponentiation of each group of partitioned Hamiltonians. The simulation time with both methods on CPU and GPU is shown in Fig. 3.4.

First, we describe the simulation time on CPU. From Fig. 3.4, up to 16 qubits, the simulation time of the baseline method is proportional to the number of terms, and that of the proposed method is proportional to the number of groups. The increase of simulation time over 16 qubits is more than that below 16 qubits. Since the size of the state vector with 16 qubits is the same as that of the L2 cache, the increase in simulation time is caused by the memory access to the L3 cache or main memory to read the amplitudes of state vector when updating the quantum state. The increase of the simulation time over 20 qubits is much more than that from 16 to 20 qubits. Since the L3 cache can not include all amplitudes of the state vector over 20 qubits, the exponential increase of the simulation time may be caused by both the calculation of updating the state

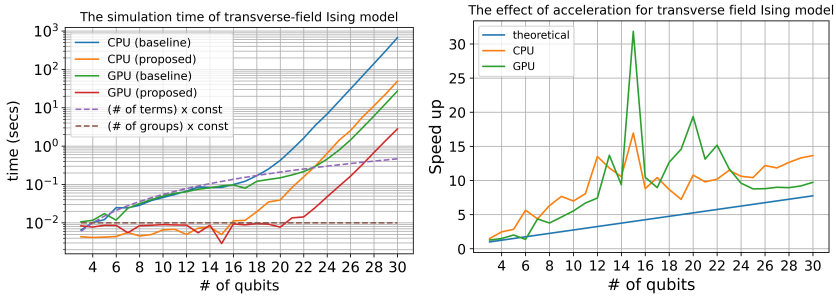


Figure 3.4: The left figure shows the simulation time of transverse-field Ising model on CPU and GPU. The right figure shows the speed-up between the baseline and the proposed method. The theoretical line represents $m/(nn_g)$.

vector and the memory access to main memory.

Next, we describe the simulation time on GPU. From Fig. 3.4, below 22 qubits, the simulation time of the baseline method is proportional to the number of terms and that of the proposed method is proportional to the number of groups. Since the size of the L2 cache is less than the size of the state vector with 19 qubits, it is expected to delay the simulation over 19 qubits. However, the simulation time below 22 qubits does not appear to be affected much by the cache size. Above 22 qubits, the simulation time eventually increases exponentially with the number of qubits because of both the memory access and the calculation of exponential large state vector.

In terms of the computational cost, from $m = O(n^2), n_g = 2$, the effect of acceleration is expected by $O(m/(nn_g)) = O(n)$. The right figure of Fig. 3.4 shows the effect of acceleration between the baseline and proposed method on CPU and GPU. Actually, while the cache is effective, the acceleration is much better than expected. For example, at 15 qubits, since the amplitudes and coefficients can be stored in the L2 cache, the simulation is accelerated 30 times faster on GPU and 15 times faster on CPU by our proposed method. At 30 qubits, we have succeeded in performing the simulation 10 times faster on both CPU and GPU.

3.4.2 Majorana fermion model

The second model we will simulate is the Majorana fermion model. The Hamiltonian of the SYK (Sachdev-Ye-Kitaev) Majorana fermion model is given by

$$
\begin{aligned}
H &= \sum_{ijkl} J_{ijkl} \hat{c}_i \hat{c}_j \hat{c}_k \hat{c}_l, \\
\hat{c}_{2i-1} &= Z_1 \dots Z_{i-1} X_i,\ \hat{c}_{2i} = Z_1 \dots Z_{i-1} Y_i, \\
\{\hat{c}_i, \hat{c}_j\} &= 2\delta_{ij},
\end{aligned}
$$

where J_{ijkl} follows a complex Gaussian distribution, and $J_{ijkl} = -J_{jikl} = -J_{ijlk} = J^*_{lkji}$ holds. As we mentioned above, the groups of mutually commuting terms in the Hamiltonian are not easily found. To find them, we use our algorithm in Sec. 3.3.2. Then, we measure the simulation time for the two methods on CPU and GPU. The results of the simulation are shown in the left figure of Fig. 3.5.

First, we describe the simulation time on CPU. Similar to the results of the transverse-field Ising model in Sec. 3.4.1, below 16 qubits, the simulation time of the baseline method is proportional to the number of terms and that of the proposed method is proportional to the number of groups. Above 16 qubits, the increase in simulation time is larger than that below 16 qubits. Since the size of the L2 cache is the same as that of the state vector with 16 qubits, the access to the L3 cache or main memory to read the amplitudes of state vector causes the increase in the simulation time. Moreover, the size of the state vector at 20 qubits is larger than that of the L3 cache. Actually, the simulation time gradually shifts from an increase proportional to the number of terms or groups to an exponential increase with the number of qubits. Above 22 qubits, the simulation time for both the baseline method and the proposed method exponentially increases with the number of qubits.

Next, we describe the simulation time on GPU. The simulation time of the baseline method is proportional to the number of terms and that of the proposed method is proportional to the number of groups. Since the L2 cache can not store the whole state vector above 19 qubits and need to access the global memory to read the amplitudes, the increase in simulation time on GPU above 19 qubits is larger than that below 19 qubits. Above 19 qubits, the increase in simulation time gradually transitions from an increase proportional to the number of terms or groups to an exponential increase with the number of qubits. Above 22 qubits, the simulation time for both the baseline method and the proposed method exponentially increases with the number of qubits.

In terms of the computational cost, from the right figure in Fig. 3.5, the expected acceleration effect is about twice faster by our algorithm because of $m/(nn_g) \simeq 2$. Around the 15 qubits, the effect of the acceleration is about eight times faster by our proposed method. Then, the acceleration effect gradually decreases with the number of qubits, and, over 22 qubits, eventually is almost the same as the expectation by $m/(nn_g) \simeq 2$. Since the size of the state vector at 16 qubits is the same as L2 cache and that at 20 qubits is larger than the size of L3 cache, the acceleration depends on cache efficiency. At 25 qubits, we confirm that our proposed method performs the simulation twice faster than the baseline method for both CPU and GPU. The effect of acceleration is smaller than the result of the transverse-field Ising model. It would be caused by how much the number of groups can be reduced.

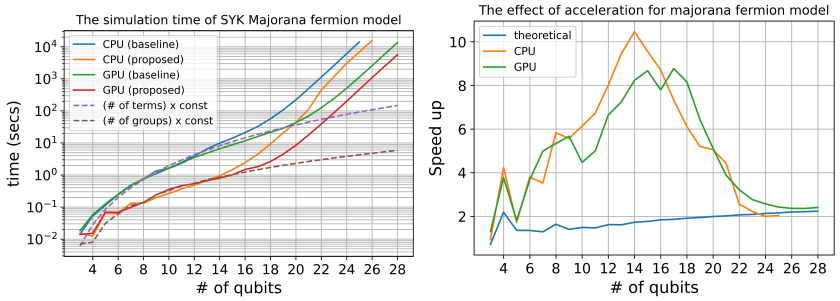


Figure 3.5: The left figure shows the simulation time of SYK Majorana fermion model on CPU and GPU. The right figure shows the speed-up between the baseline and the proposed method. The theoretical line represents $m/(nn_g)$.

3.5 Conclusions

In this Chapter, we propose a method of accelerating the classical simulation of Hamiltonian dynamics. Specifically, we partition the Hamiltonian terms into groups of mutually commuting terms, and then simultaneously diagonalize the terms in the groups by Clifford transformation. To verify the effect of acceleration by simultaneous diagonalization, we perform numerical experiments for two models: the transverse-field Ising model and the Majorana fermion model. In our numerical experiments, we confirmed that the simulation is performed faster by simultaneous diagonalization than by time-evolving Hamiltonian terms one by one. We also confirmed the acceleration on both CPU and GPU. In other words, we confirmed the acceleration is independent of the architecture.

Here we explore possible applications of our proposed method. The simulation of quantum annealing [28] and quantum approximate optimization algorithm (QAOA) [8] can be accelerated by our algorithm. Both algorithms are based on Hamiltonian dynamics of Ising models with transversal fields or other driving Hamiltonians and hence can be accelerated easily. On the other hand, the fermionic Hamiltonians are often employed for quantum simulation or variational quantum eigensolver to solve quantum many-body systems like condensed matter physics and quantum chemistry. As the number of qubits in quantum computers continues to grow, fast classical simulations are becoming increasingly important to know if these algorithms work properly and/or what parameters to choose to improve their performance. Our proposed method will make a significant contribution to accelerating the development of such quantum algorithms.

Chapter 4

Parametric t-SNE with quantum neural network

4.1 Introduction

Visualization of high-dimensional data is an important subfield of machine learning. It allows us to intuitively interpret the data and understand possible patterns in them. As visualization often involves mapping of the original data to a low, typically two or three, dimensional space, the techniques for visualizations are also useful for compression of data or preprocessing before applying other machine learning techniques. Prototypical examples of visualization techniques include t-stochastic neighbor embedding (t-SNE) [45].

Such techniques have been proven to be useful also for machine learning of quantum states. Ref. [46] applied various visualization methods to detect quantum phase transitions in the Hubbard model, where they generated states by quantum Monte Carlo simulations. They have found that t-SNE is the most promising technique in this type of the application. Ref. [47] applied t-SNE to visualize quantum states represented as a matrix product states. They successfully visualized quantum phase transitions in spin models such as transverse field Ising model.

However, the application of machine learning techniques running on classical computers for this purpose is intrinsically limited to the case where the target quantum states have efficient classical representations. The use of quantum computers can provide an advantage in widening the range of quantum states that can be used as inputs. In fact, in Ref. [48], identification of topological quantum phase has been proposed by using a clustering algorithm using a quantum computer. This approach is motivated by the fact that there is a family of quantum states that are useful for machine learning of classical data but cannot be efficiently represented by classical computers [12].

Here, we propose to use quantum neural networks (QNNs), which use a parametric quantum circuit to construct a machine learning model, for visu-

alization of quantum states. Our proposal is based on parametric t-SNE [49], which is a visualization technique where we employ neural networks to map high-dimensional data into low-dimensional space. The mapping is optimized so as to keep the similarity of the data points, which is defined from the distance in the respective spaces, unchanged. Our idea is to use QNNs instead of the classical neural network. This allows us to directly use quantum states as inputs, which may be useful for studying complex quantum systems and certain machine learning problems that are hard classically. The similarity of the quantum states can be defined from fidelity(-like) measures. Our method optimizes the parameters in a QNN so that the respective quantum states are mapped to low-dimensional points, which are defined as expectation values of certain observables at the output of the QNN, while maintaining the similarity among the points.

We also conduct numerical verification of our proposed method. First, we use Iris flower dataset [50] to test if it can be successfully applied to classical data. Second, we visualize the quantum states time-evolved under the transverse-field Ising Hamiltonian. In the visualization of quantum data, we could not effectively generate a visualization with distinct clusters when using default cost function usually utilized for parametric t-SNE. To deal with this problem, we introduce a hyper-parameter into the cost function to adapt the scaling of the low-dimensional data. With these successful demonstrations, we believe that the proposed method would be a powerful tool to visualize and analyze both classical and quantum datasets.

4.2 Preliminary

4.2.1 Parametric t-SNE

Parametric t-SNE [49] is the visualization method that can be applied to unseen data using a neural network. Let us denote weights of a neural network as $\boldsymbol{w}$, the map of neural networks $\boldsymbol{y} = f(\boldsymbol{x}|\boldsymbol{w})$ is trained to minimize the cost function C in Eq. (2.9). In numerical experiments, we use parametric t-SNE to compare the accuracy of our proposed method.

4.2.2 Application of t-SNE for quantum systems

Recently, t-SNE has also been applied to the field of quantum physics by Yuan *et al.* [47]. They considered the visualization of quantum phase transitions by applying t-SNE to (approximate) ground states $|\psi_i\rangle$ of transverse field Ising model and Heisenberg model. Their approach is to use,

$$p_{j|i} = \frac{\exp\left(-d(\psi_i, \psi_j)^2/2\sigma_i^2\right)}{\sum_{k\neq i}\exp\left(-d(\psi_i, \psi_k)^2/2\sigma_i^2\right)}, \tag{4.1}$$

instead of Eq. (2.7), where the distance $d(\psi_i, \psi_j)$ is defined by the negative logarithmic fidelity

$$d(\psi_i, \psi_j) = -\log\left(|\langle\psi_i|\psi_j\rangle|\right). \tag{4.2}$$

It has been shown that this approach can successfully visualize and identify quantum phase transitions of one-dimensional spin chains. In Ref. [47], they also considered the visualization of classical data by using the so-called "quantum feature map" via t-SNE. We share the idea with Ref. [47] in the sense of visualization of quantum states via t-SNE. However, our approach differs essentially from Ref. [47] in that a parameterized quantum model is employed to yield low dimensional data of parameteric t-SNE.

4.3 Parametric t-SNE with quantum circuits

In this work, we propose using quantum circuits to construct the parametric model $f(x|\theta)$ to generate low-dimensional data. The procedure of our proposed method is shown in Fig. 4.1. More concretely, using a parameterized unitary circuit $U(x, \theta)$ that depends on both of a d-dimensional input x and trainable parameters θ, we generate a d'-dimensional datapoint y by expectation values of d' observables $\{O_\mu\}_{i=1}^{d'}$. Then, we minimize the cost function defined in Eq. (2.9) by optimizing the parameters θ. The concrete algorithm for classical inputs is as follows:

1. Compute p_{ij} for all pairs from $\{x_i\}_{i=1}^N$.
2. For all $\{x_i\}_{i=1}^N$ and $\{O_\mu\}_{\mu=1}^{d'}$, evaluate $y_\mu(x_i, \theta) = \langle 0| U^\dagger(x_i, \theta) O_\mu U(x_i, \theta) |0\rangle$ using a quantum computer.
3. Compute q_{ij} for all pairs from $y_\mu(x_i, \theta)$ on a classical computer.
4. Compute $C(\{y_\mu(x_i, \theta)\}_{i=1}^N)$ on a classical computer.
5. Update θ so that C is minimized.

After the convergence, we expect the above protocol to find a quantum circuit $U(x, \theta)$ which can map x_i to a low-dimensional space while preserving the similarity based on distances among the data.

Note that the choice of $U(x, \theta)$ and O_μ affects the outputs y_i. First, we explain how to choose observables. We choose local observables to alleviate a vanishing gradient problem, which makes optimization hard. This is because the gradient is less likely to disappear using local observables than global observables [14]. Moreover, if we suppose $U(x, \theta) = U(\theta)U_{\text{in}}(x)$ as in Fig. 4.1, our quantum model means that quantum states representing data are measured on bases optimized by a parameterized quantum circuit $U(\theta)$. Let us define $|\psi_{\text{in},i}\rangle = U_{\text{in}}(x_i)|0\rangle$, the output values $y_\mu(x_i, \theta)$ are described by $\langle\psi_{\text{in},i}|U(\theta)^\dagger O_\mu U(\theta)|\psi_{\text{in},i}\rangle$. This means that the encoded data $|\psi_{\text{in},i}\rangle$ is measured by the observables $U(\theta)^\dagger O_\mu U(\theta)$. If the parameterized quantum circuit is deep

enough to transform the observables appropriately, the choice of observables O_μ is not likely to have much effect on results. In our numerical experiments, we choose local observables acting on a single qubit. Next, we comment on the design of a quantum circuit. In our numerical experiments, we use hardware-efficient ansatz, which is widely used in the study of quantum machine learning. However, this choice is only for proof-of-concept numerical demonstration, and it is an important future work to explore what types of quantum circuits are suited for our proposed algorithm. The desirable design of the quantum circuit for quantum machine learning is actively studied [51, 52]. At least, we can say that linear data encoding and transformation are preferred if the data is linearly separable. Otherwise, we have to encode data with non-linearity, e.g., using data re-uploading [52].

An interesting extension of the above protocol is to take a quantum dataset consisting of quantum states $\{|\psi_i\rangle\}_{i=1}^N$ as the input data. We depict the protocol in Fig. 4.1 (b). Here we can think of several ways to define the similarity of quantum data in a high-dimensional space. One possible way is to generate high-dimensional classical data by measuring a set of observables $\{P_\nu\}_{\nu=1}^d$. Then their expectation values

$$\{\langle P_\nu\rangle\}_{\nu=1}^d \tag{4.3}$$

are used as high dimensional data. Another possibility is to use a distance function of two quantum states defined via fidelity

$$d(|\psi_i\rangle, |\psi_k\rangle) = \sqrt{1 - |\langle\psi_i|\psi_k\rangle|^2}. \tag{4.4}$$

Then similarity of two quantum states are defined as follows:

$$p_{j|i} = \frac{\exp\left(-d(|\psi_i\rangle, |\psi_j\rangle)^2/2\sigma_i^2\right)}{\sum_{k\neq i}\exp\left(-d(|\psi_i\rangle, |\psi_k\rangle)^2/2\sigma_i^2\right)}. \tag{4.5}$$

This choice enables us to readily calculate $d(\psi_i, \psi_k)$ on a quantum computer using standard techniques for overlap measurement such as swap test [53].

On the other hand, for the low-dimensional space, we can use $U(\theta)$ that only depends on the trainable parameters θ at step 2 in the classical input case since we do not have classical input x. For the cost function C, we can use the same formulation as above, that is, we measure $y_\mu(\psi_i, \theta) = \langle\psi_i| U^\dagger(\theta) O_\mu U(\theta) |\psi_i\rangle$ for a certain set of observables $\{O_\mu\}$ and compute $C(\{y_\mu(\psi_i, \theta)\}_{i=1}^N)$.

Note that if we employ a quantum feature map,

$$|\psi_i\rangle = U_{\rm in}(x_i)|0\rangle, \tag{4.6}$$

the similarity of classical input data $\{x_i\}$ in high dimensional space can also be calculated via the fidelities of the quantum states.

4.4 Numerical experiments

Here, we perform the numerical simulations of the proposed method. First, let us describe the tools used in the experiments. We use qulacs [17] to simulate a

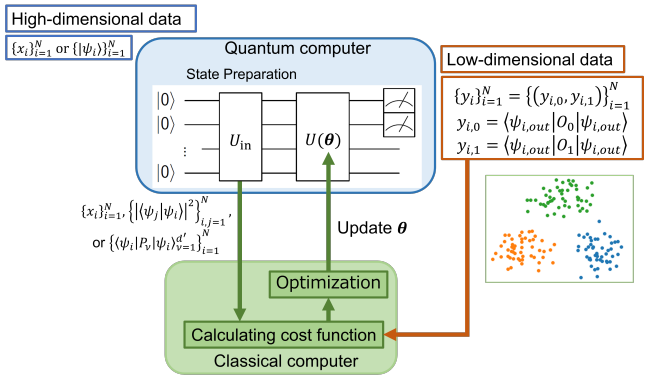


Figure 4.1: This figure shows the procedure of visualizing classical or quantum data. Specifically, we visualize classical and quantum data by mapping the high-dimensional data to the low-dimensional one. We prepare a quantum state by a parametrized quantum circuit for classical data or by time-evolution under a Hamiltonian for quantum data. The mapping is performed by a parametrized quantum circuit trained by minimizing a cost function so as to maintain the similarity of surrounding points. The similarity of high-dimensional data is defined by either expectation values, fidelities, or classical data. The similarity of low-dimensional data is determined by expectation values, and we plot them in a two-dimensional plane.

quantum circuit. We make a python wrapper to work with PyTorch[54] to use the loss function and optimizer. Our optimization is performed by Sharpness-Aware Minimization (SAM) [55, 56] with Adam[57] as the base optimizer. The overview of SAM is to consider the cost to which added the l2-regularization term is added, find the gradient at the point where that cost is the highest in the neighborhood, and descend from the current point according to that gradient.

Note that in this section, we denote X_i as the Pauli X acting on ith qubit, and also Z_i as the Pauli Z acting on ith qubit.

4.4.1 Visualizing classical data

Application of parametric t-SNE with quantum circuits

We visualize the Iris flower dataset [50] with our proposed method. The dataset contains three classes and consists of four features. We normalize each feature between -1 and $+1$. The construction of our ansatz $U(x, \theta)$ and observables $\{O_\mu\}$ for the low dimensional space are as follows. Let us define $R_y(\theta) = e^{i\theta\hat{Y}/2}$ and

$$U_{\text{in}}(x_i) = R_y(x_{i,1}) \otimes R_y(x_{i,2}) \otimes \ldots \otimes R_y(x_{i,d}),$$

where $x_{i,j}$ denotes the jth element of ith data x_i. Also, let us define U_1 and U_2 as shown in Fig. 4.2. U_1 and U_2 are commonly used in hardware efficient

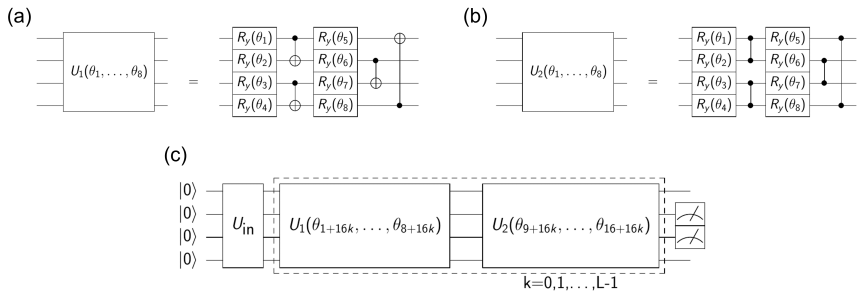


Figure 4.2: The quantum circuit to visualize data: We input data by U_{in}, and alternately act U_1 and U_2 on the quantum state. The quantum circuit U_1 and U_2 is shown in (a) and (b), respectively.

ansatz. We use the combination of them as $U(x,\theta)$, defined by We construct $U(x,\theta)$ as,

$$U(x,\theta) = \prod_{k=0}^{L-1} \left[U_2(\theta_{9+16k:16+16k})U_1(\theta_{1+16k:8+16k})\right] U_{\text{in}}(x), \tag{4.7}$$

where $\theta_{i:j}$ denotes $(j-i+1)$ dimensional vector containing ith to jth elements of θ. Specifically, we set the circuit depth $L = 4$. As for the output, we set $d' = 2$ and $\{O_\mu\} = \{X_2, X_3\}$ to visualize the Iris dataset on a two-dimensional plane. As we explained in Sec. 4.3, we choose local observables to alleviate the vanishing gradient problem.

We perform the simulation under the above settings and plot the results in Fig. 4.3. This figure shows that the data is clustered for each type of Iris flower. The left figure (a) shows the visualization by the classical machine learning method of t-SNE, and the middle figure (b) by the proposed method with similarity based on Euclidean distance. In both two cases, the data are clustered by Iris flower varieties.

Visualization with infidelity distance measure

Next, we show that the proposed method can be correctly visualized by calculating the similarity of quantum states. To this end, we will use, as a test case, a set of quantum states generated by a quantum feature map from the Iris dataset, before running the method on a truly physically meaningful quantum state. We perform the parametric t-SNE using quantum features $|\psi_i\rangle$ defined as

$$|\psi_i\rangle = U_{\text{in}}(x_i)\,|0\rangle\,, \tag{4.8}$$

and the cost function associated with the distance measure defined in Eq. (4.4). We show the result in Fig. 4.3 (c), which implies that this protocol also works.

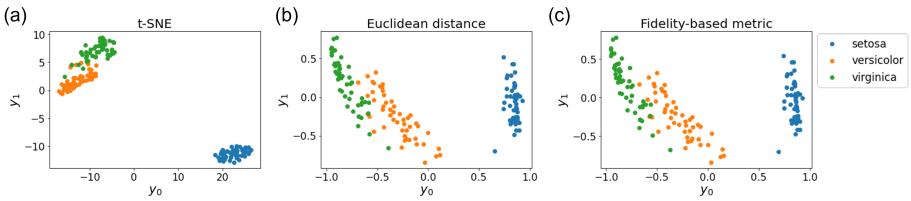


Figure 4.3: The figure (a) shows the visualization of the Iris flower dataset by the classical machine learning method of t-SNE, the figure (b) by the proposed method with the similarity based on Euclidean distance between high-dimensional data, and the figure (c) by the proposed method with the similarity based on the fidelity based metric between high-dimensional data.

4.4.2 Visualization of quantum data

Visualization based on observables

In this section, we perform the visualization of quantum states. As an example, let us consider the time-dependent two-body transverse field Ising model, which is employed in quantum annealing [28]. We prepare quantum states time-evolved under the following Hamiltonian:

$$H = (1 - t/\tau) \sum_i h_i X_i + (t/\tau) \sum_{i<j} J_{ij} Z_i Z_j, \tag{4.9}$$

where τ denotes the total simulation time of the Hamiltonian. We perform the simulation under the Trotter decomposition[22, 58], defined by

$$\begin{aligned} e^{-i(H_1+H_2)t} &= \lim_{m\to\infty} (e^{-iH_1 t/m} e^{-iH_2 t/m})^m \\ &\simeq (e^{-iH_1 t/m} e^{-iH_2 t/m})^m. \end{aligned}$$

We set the number of qubits as four, $\tau = 40$, time step $\Delta t := \tau/m = 0.01$, $h_i = -1$, and J_{ij} as uniform random number between -1 and 0 or 0 and $+1$. We prepare 200 random Hamiltonians. The half of them have all positive J_{ij}, corresponding to antiferromagnetic (AFM), and the rest have all negative J_{ij}, corresponding to ferromagnetic (FM).

The dynamics is performed as follows. Since we prepare the ground state of $\sum_i h_i X_i$ as the initial state, the adiabatic evolution results in a ground state $(|0\rangle^{\otimes n} + |1\rangle^{\otimes n})/\sqrt{2}$ of $\sum_{i<j} J_{ij} Z_i Z_j$, which does not depend on the specific values of J_{ij}. On the other hand, anti-ferromagnetic Hamiltonians have different ground states depending on the values of J_{ij}, and therefore the visualization results in a scattered pattern.

Here we visualize the quantum state at every 1000 Trotter steps ($t = 10, 20, 30, 40$) by making the models for each step. As explained before, we consider the two methods to compute similarity of the quantum states in high-dimensional space.

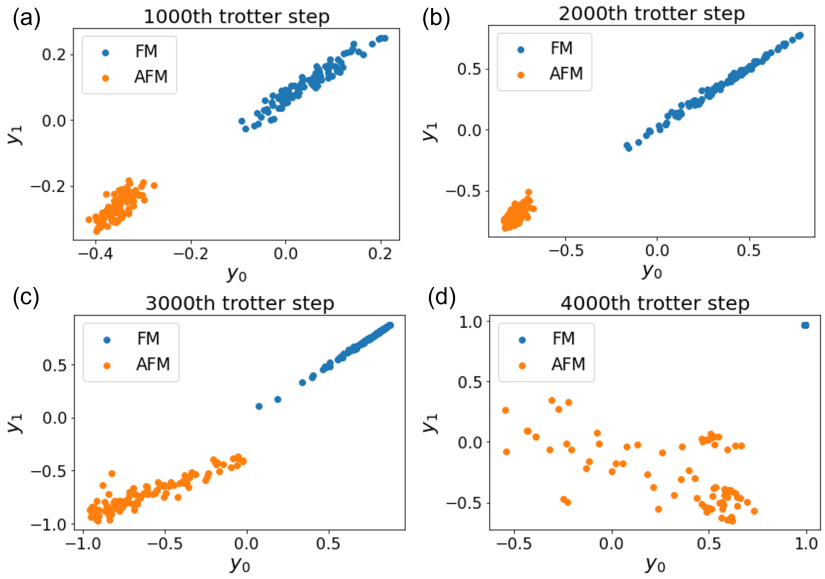


Figure 4.4: These figures illustrate the process of a time-dependent transverse-field two-body Ising model when the calculation of the similarity in high-dimensional data is based on the observables. The figures (a), (b), (c), and (d) visualize the quantum states of the 1000th($t = 10$), 2000th($t = 20$), 3000th($t = 30$), and 4000th ($t = 40$) trotter steps by making models at each step, respectively. The orange or blue points correspond to the cases where all J_{ij} in the Hamiltonian is positive (AFM) or negative (FM).

The first method is to consider the expectation values of input quantum states as high-dimensional data

$$x_i = (\langle\psi_{i,\mathrm{in}}|X_1|\psi_{i,\mathrm{in}}\rangle, \langle\psi_{i,\mathrm{in}}|X_2|\psi_{i,\mathrm{in}}\rangle, \ldots, \langle\psi_{i,\mathrm{in}}|X_n|\psi_{i,\mathrm{in}}\rangle),$$

where n denotes the number of qubits. Then we calculate the similarity between the high-dimensional data points as we explained in Sec. 4.2.1. The low-dimensional data $\{y_i\}$ is defined similarly by the following equation using a certain constant value a:

$$y_i = a(\langle\psi_{i,\mathrm{out}}|X_2|\psi_{i,\mathrm{out}}\rangle, \langle\psi_{i,\mathrm{out}}|X_3|\psi_{i,\mathrm{out}}\rangle). \tag{4.10}$$

The constant value a is a hyper-parameter to adjust the scale of low-dimensional data. We set the hyper-parameter $a = 1$ and circuit depth $L = 8$. In this numerical experiment, we visualize the quantum states for every 1000 trotter steps. The result is shown in Fig. 4.4. In this figure, we can see that two clusters corresponding to the sign of the coupling constants are formed.

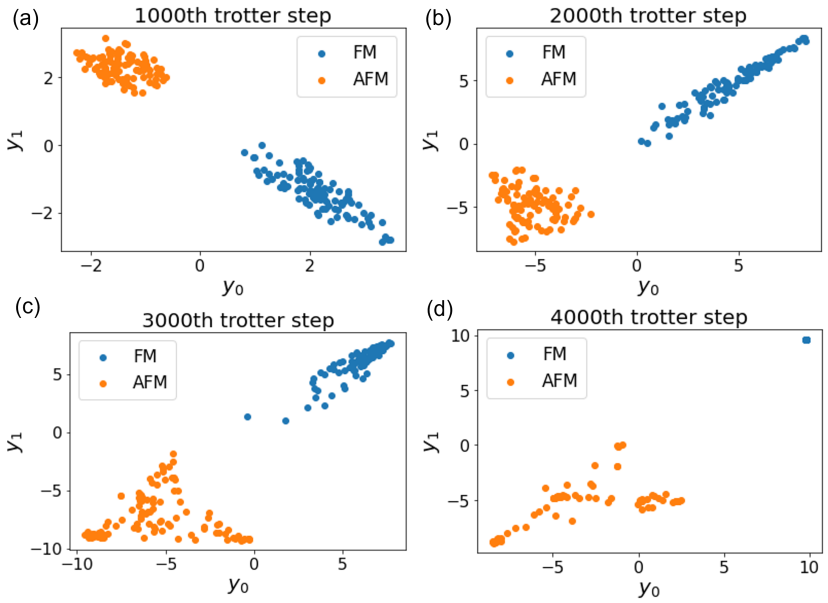


Figure 4.5: These figures illustrate the process of a time-dependent transverse-field two-body Ising model when the calculation of the similarity in high-dimensional data is based on the fidelity. The figures (a), (b), (c) and (d) visualize the quantum states of the 1000th, 2000th, 3000th and 4000th trotter step, respectively. The orange and blue points correspond to the cases where J_{ij} is positive or negative, respectively.

Visualization with infidelity metric

The second method is to calculate the similarity of the data in high-dimensional space via infidelity of two different quantum states. The similarity between low-dimensional data is calculated in the same way as the previous case, except for the setting of hyper-parameter $a = 10$. Fig. 4.5 shows the process of the dynamics for every 1000 trotter steps. This figure also shows the two clusters corresponding to the sign of the coupling constants.

The effect of multiplying a constant value

When we visualize the quantum data, we sometimes get undesirable visualization figures, which can be avoided by appropriately tuning the hyper-parameter a. For example, we plot three examples of wrong choices of a in Fig. 4.6. In Fig. 4.6 (a), the data is arranged in an almost straight line, despite the two-dimensional plot. In figures (b) and (c), it is impossible to determine that these data belong to different classes without the colors in the plots, since each class is colored to make the figures easier to understand. One possible reason is that the cost function optimization is not sufficient to converge to a local minimum.

We examine whether the optimization converges near a local minimum. To this end, we visualize the loss contours and optimization trajectories in a two-dimensional plane. In Ref. [24], they develop a method to visualize loss landscape and optimization trajectories with the loss contours. They use the visualization method to investigate the relationship between loss landscape and trainability or generalization for neural networks. Before we describe the detailed method, let us define θ_i as a vector of trainable parameters at ith epoch. We consider the matrix M consisting of $(\theta_i - \theta_n)$ for all i, where the n is the last epoch. Specifically, the M is written by

$$M = [\theta_0 - \theta_n; \theta_1 - \theta_n; \ldots; \theta_{n-1} - \theta_n]. \tag{4.11}$$

The matrix M is transformed by principal component analysis [59], and the 1st and 2nd principal component vectors are used as the axis of the visualization. Along the two principal component vectors, the loss contours and optimization trajectories are plotted on a two-dimensional plane. We use this visualization method to tune the hyper-parameter a and to confirm our optimization works appropriately.

We show the visualization of the loss landscapes in Fig. 4.6 (d), (e), and (f). From these figures, the optimization is sufficiently done to reach the local minima. Despite the cost function being well optimized, we have not been able to reflect enough features on the low-dimensional data to discriminate different clusters. This is attributed to the wrong design of the cost function, where the similarity is not appropriately defined. For example, in Fig. 4.6 (a), the parameter a for adjusting the scale is multiplied by one instead of by 10 in Fig. 4.5 (a). In Fig. 4.6 (b) and (c), both high-dimensional and low-dimensional data are multiplied by 10 in similarity calculation, instead of by one in Fig. 4.4 (c) and (d). These results suggest that when calculating similarity, it is necessary to adjust the scale of the high- and low- dimensional data so that the cost function adequately reflects the characteristics of the given data.

To be more concrete, the need to adjust the scale of the high- and low-dimensional data similarities may occur in larger systems since high- and low-dimensional data similarities may not reflect surrounding points. First, we refer to the high-dimensional data similarities. Since high-dimensional data similarities are likely to become smaller in simulating a larger quantum system, it is more difficult to reflect the features of higher-dimensional data into low-dimensional data. We can alleviate this difficulty by taking an n-square root of the fidelities to include more information about the surrounding points. This change affects the cost function, and the high-dimensional data similarities are easily reflected in the low-dimensional data. Next, we refer to the low-dimensional data similarities. If the low-dimensional data move a small region, optimizing the cost function is difficult because similarities between dissimilar points can not have small values. We deal with this difficulty by multiplying observables by a constant value. For example, let us consider Student t-distribution without the normalization factor as $u = 1/(1 + v^2)$. If v ranges from -1 to 1, the range of u is $[0.5, 1]$. In this case, optimization of cost function

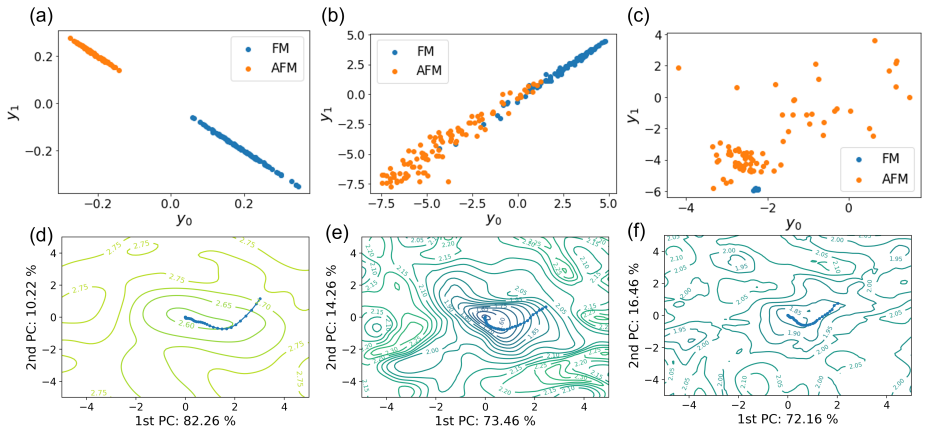


Figure 4.6: These are examples of undesirable visualizations. In figure (a), although the plot is two-dimensional, it is plotted in a one-dimensional manner. In figures (b) and (c), the different classes cannot be identified unless they are plotted in different colors. Figures (d), (e), and (f) are the loss contours and optimization trajectories of Figures (a), (b), and (c), respectively. In all three cases, the parameters are optimized.

may be difficult because the lower bound causes dissimilar points can not have small low-dimensional data similarities. We deal with this difficulty by multiplying observables by a constant value. If we multiply v by 10, the range of u becomes $[0.0099, 1]$. The lower bound of low-dimensional data similarities becomes smaller by multiplying a constant value. As a result, dissimilar points can take lower values of low-dimensional data similarities. In this way, the smaller lower bound of low-dimensional data similarities facilitates distinguishing dissimilar points in optimizing the cost function.

4.5 Conclusion and future works

In this work, we have visualized classical and quantum data by a quantum neural network. Specifically, we employed a parameterized quantum circuit as a quantum model to generate low-dimensional data, which is trained so that the similarity of the high-dimensional data is maintained. In the quantum case, the similarity of the quantum states is calculated in two ways: measuring multiple observables to obtain high-dimensional classical data from quantum states, and calculating distance of two quantum states directly from fidelity.

We performed numerical simulations of two-dimensional visualizations of Iris dataset as the case of the classical inputs and quantum states evolved under Hamiltonian dynamics as the case of the quantum inputs. It was found that the proposed method worked well for both classical and quantum inputs, with appropriate low-dimensional visualization. Specifically, in the case of quantum

data, it is difficult to visualize the data well unless the constant factor is multiplied by the similarity of the low-dimensional data. This is probably because the similarity between the quantum states takes too small value. While we treated this constant factor as a hyper-parameter, it can be trainable parameters as a_i for each low-dimensional data y_i for further improvement. At least, by doing so, the performance of non-parametric t-SNE can be guaranteed in the proposed model with a two-qubit trivial circuit. This implies that the proposed method is expected to work on a relatively small quantum device. For more complex data visualization, we can improve the representation capability of quantum circuits.

One of the other possibilities using our proposed method is to compress quantum data by defining the low-dimensional data by quantum states and the similarity by a fidelity-based metric. While real quantum data, such as a set of outputs from a quantum algorithm, live in a large Hilbert space, such quantum data can be mapped into a smaller Hilbert space with a fewer number of qubits with keeping their similarity. Then quantum machine learning algorithms can be further applied to such a compressed quantum data set. Further investigation in this direction is an intriguing future issue. The other future task is to construct a model including a decoder, which decodes data from compressed data, such as Variational AutoEncoder [60] in classical machine learning. It may be possible to create a quantum state with certain desired properties from the classical data in the middle layer.

Chapter 5

Quantum Kernel t-SNE

5.1 Introduction

Dimensionality reduction techniques such as principal component analysis, autoencoder [61], and variational autoencoder [60] are used to reduce the computation resources or abstract the essential information from data. A popular and important application of dimensionality reduction is data visualization by mapping high-dimensional data into a two-dimensional space. The visualized data is useful in finding trends and understanding the characteristics of data that are difficult to see directly from the raw high-dimensional data. In data visualization of classical machine learning, principal component analysis and t-distributed stochastic neighbor embedding (t-SNE) [45] are popular choices. For example, they can be used to visualize loss landscapes [24] and optimization processes [25, 26] to gain intuitive insights into complex optimization processes in classical machine learning.

In quantum computing, there is a similar demand to obtain an intuitive insight of cost function landscapes and parameter optimization processes. More concretely, variational quantum algorithms [13], such as variational quantum eigensolver [7, 38], quantum approximate optimization algorithm [8], and quantum neural networks [9, 62], are promising routes to use noisy intermediate-scale quantum devices for practically important problems. However, there is difficulty in designing a parameterized quantum circuit for the variational ansatz as different tasks require different expressibility [63, 64]. Furthermore, once a parameterized quantum circuit is fixed, there exist challenges in parameter optimization such as the vanishing gradient problem [14, 15, 16] and being trapped in local minima [65]. Quantum data visualization [66] can be used to approach the above issues in variational quantum algorithms. Indeed, visualizing loss landscape and optimization trajectories by using 1st and 2nd principal component vectors have been performed in quantum machine learning [67], as in classical machine learning [24]. However, this approach only visualizes the changes in variational parameters, and does not reflect direct information about

the quantum states during the optimization process.

To visualize quantum states, we have proposed a variational quantum algorithm to visualize classical and quantum data using a quantum neural network [66], sharing the idea with t-SNE. Specifically, we map quantum states representing high-dimensional data onto a two-dimensional plane by optimizing a parameterized quantum circuit so that the similarities among the original data and the low-dimensional data are preserved. However, the method requires many accesses to a quantum computer to optimize the design and the parameters of the parameterized quantum circuit applied to quantum states. If one requires a variational quantum algorithm for visualizing the quantum circuit itself, its usage for performance evaluation or enhancement through visualization becomes impractical due to the aforementioned issues.

To resolve these issues and realize faster and easier quantum data visualization we here propose a visualization method using quantum kernels [10, 11]. Quantum kernel method is a kernel method of using inner products between quantum states representing data to perform machine learning tasks such as regression or classification. Specifically, we employ quantum kernels instead of classical kernels in kernel t-SNE [68, 69, 70], which is a method of out-of-sample extension of t-SNE by classical kernels. We can perform visualization by optimizing parameters to minimize the cost function of t-SNE only on a classical computer after evaluating all pairs of overlaps between quantum states. Our proposed method requires fewer accesses to a quantum computer in most cases than the previous quantum neural network approach [66]. Moreover, since a parametrized quantum circuit applied to quantum states is not required, we can visualize quantum states more directly by our proposed method than by the previous method [66]. This allows us to investigate the performance of a quantum feature map, a map realized by a quantum circuit to encode classical data.

To verify our proposed method, we first visualize quantum states generated through a quantum feature map from scikit-learn hand-written digits data set [50] in a two-dimensional plane. To quantitatively evaluate models, we apply k-nearest neighbor algorithm to the low-dimensional data and compare the accuracy of the classification. We confirm the prediction by our proposed method is more accurate than by a quantum neural network. Moreover, we find that our method achieves the accuracy mostly equivalent to the kernel t-SNE using Gaussian kernels. While this task is useful, for example, to evaluate the performance of quantum feature maps, our goal here is to visualize quantum data themselves.

Secondly, we use the method for visualizing quantum states generated during a quantum computational task. Specifically, we visualize the optimization trajectories of the variational quantum eigensolver (VQE) algorithm applied to transverse field Ising models, which we could not perform by the previous method [66]. We find the trends of optimization trajectories based on quantum states during the training process. This visualization clarifies the ansatz's dependence on the parameter initialization and whether or not a part of a trajectory is shared with the other ansatz's trajectory. Such an application for

visualizing optimization process of variational quantum algorithms is helpful in improving the optimization strategies by changing the initial parameters or quantum circuit design with monitoring their effect.

5.2 Kernel t-SNE

Kernel t-SNE is the variant of t-SNE that can be applied to unseen data [68, 69]. The map of kernel t-SNE from high-dimensional data $\boldsymbol{x}$ to d-dimensional data (low-dimensional data) $\boldsymbol{y}$ is represented by

$$\boldsymbol{y}(\boldsymbol{x}) = \sum_{i=1}^{N} \boldsymbol{\alpha}_i k(\boldsymbol{x}_i, \boldsymbol{x}), \tag{5.1}$$

where N is the number of training data, k is a kernel function, and $\boldsymbol{\alpha}_i \in \mathbb{R}^d$. In classical machine learning, we often employ a Gaussian kernel $k(\boldsymbol{x_i}, \boldsymbol{x}) = \exp\left(-\|\boldsymbol{x_i} - \boldsymbol{x}\|^2/(2\sigma'^2)\right)$. The parameters $\boldsymbol{\alpha}_i$ are determined by minimizing the t-SNE cost function [68, 69]. This method is expected to accelerate the mapping of all data points by learning the mapping by the subset. In this paper, we use this method to accelerate the visualization in terms of the number of access to a quantum computer. Specifically, we propose to use a quantum kernel which is defined through an inner product of quantum features as a kernel function. This allows us to visualize both classical and quantum data without optimizing a parametrized quantum circuit in contrast to our previous work [66]. Note that we construct the mapping by training all data points to perform a proof-of-concept demonstration.

5.3 Quantum kernel t-SNE

Here, we describe our proposed visualization method which uses quantum kernels and its advantages over the previous one [66].

5.3.1 Algorithm

The outline of our proposed method is shown in Fig. 5.1. Here we consider learning the features of N of classical or quantum training data. Specifically, we regard $|\psi_i\rangle$ as ith quantum data or as a quantum state where ith classical data $\boldsymbol{x}_i$ is encoded via a quantum feature map. First, we calculate the high-dimensional data similarities written in Eq. (2.7), using fidelities between quantum states estimated on a quantum computer. Then, we calculate the low-dimensional data according to Eq. (5.1) employing the following kernel function:

$$k(|\psi_i\rangle, |\psi\rangle) = |\langle\psi_i|\psi\rangle|^2. \tag{5.2}$$

The procedure of our proposed method is described as follows:

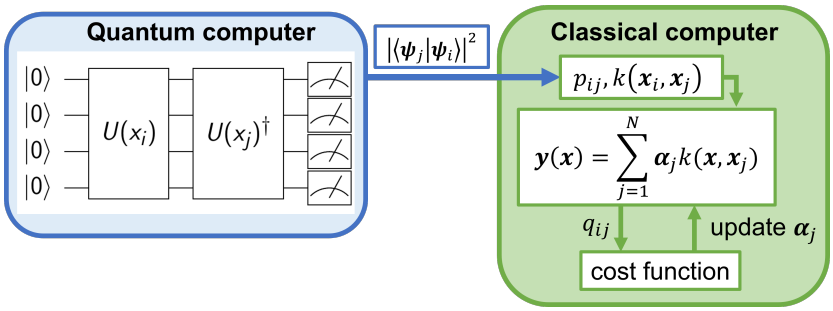


Figure 5.1: The outline of our proposed method. After the fidelities between quantum states representing data are evaluated for all data pairs on a quantum computer, the rest of the optimization process can be performed entirely on a classical computer. The optimization minimizes the cost function by updating α_j. After the optimization, we visualize data by plotting the low-dimensional data $\boldsymbol{y}$.

1. Calculate the high-dimensional data similarities p_{ij} for all data pairs.
2. Calculate the low-dimensional data $\boldsymbol{y}(\boldsymbol{x})$.
3. Calculate the low-dimensional data similarities q_{ij} and cost function C.
4. Update $\boldsymbol{\alpha}_i$ to minimize cost function C.

We can visualize data by plotting the low-dimensional data $\boldsymbol{y}$ using the optimized $\{\boldsymbol{\alpha}_i\}$.

5.3.2 Advantage over the previous method

In this section, we explain the advantages of our proposed method compared to the parametric t-SNE with a quantum neural network [66]. Specifically, we compare the number of accesses to a quantum computer. Let us define p as the number of parameters in a quantum circuit and m as the number of iterations required to optimize the parameters.

First, we compare the number of accesses when visualizing classical data. In our proposed method, we need to evaluate $O(N^2)$ fidelities to calculate the kernel function during training. For the parametric t-SNE with a quantum neural network [66], we need to use $O(Np)$ accesses for each iteration to evaluate the cost function and its gradient. So, $O(Npm)$ accesses are required for the whole optimization. The number of accesses to a quantum computer in the previous method depends on the number of data, iterations, and parameters while the proposed method only depends on the number of data. In [71], they use a quantum neural network with 15 or 60 parameters and optimize the parameters with 400 iterations to classify 360 data with two digits. In this case, the cost for accessing a quantum computer by the quantum neural network is $O(Nmp) = O(10^6)$ and that by our quantum kernel method would be

Table 5.1: This table shows the number of access to a quantum computer. In the table, N, m, and p denote the number of data, iterations, and parameters in a parametrized quantum circuit.

	classical data	quantum data
parametric t-SNE with QNN [66]	$O(Nmp)$	$O(N^2) + O(Nmp)$
proposed method	$O(N^2)$	$O(N^2)$

$O(N^2) = O(10^5)$, so our proposed method can efficiently perform the classification task. More generally, if we assume that classifying ten digits by a quantum neural network needs $O(100)$ of parameters and iterations to optimize the parameters, our proposed method is more efficient than a quantum neural network when the number of data is less than 10^4 since $O(N^2) < O(Nmp) = O(N \times 10^4)$. Datasets with a size of less than 10^4 data points are prevalent, such as the iris flower dataset and the handwritten digit dataset in scikit-learn [50]. Our proposed method is thus advantageous in terms of the number of accesses to a quantum computer when compared to quantum neural network approaches.

Next, we compare the number of accesses when visualizing quantum data. We need $O(N^2)$ accesses in the proposed method. On the other hand, the previous method [66], needs $O(N^2)$ accesses to evaluate fidelities to calculate high-dimensional data similarities. Additionally, we need $O(Np)$ accesses for each iteration to evaluate the cost funtion and its gradient. In total, $O(N^2)+O(Nmp)$ accesses is required for the whole optimization. Therefore, the proposed method is more efficient for the visualization of quantum data than the previous work [66] without any assumptions in contrast to the case of classical data. We summarize the results in Table 5.1.

5.4 Numerical experiments

In this section, first, we perform numerical experiments to validate our proposed method by visualizing classical data. To quantitatively evaluate the generalization performance, we apply k-nearest neighbor algorithm to low-dimensional data. Then, we present visualization of optimization trajectories in a variational quantum eigensolver. From the visualization, we explore the characteristics of ansatzes' optimization trajectories.

5.4.1 Visualizing classical data

Here, we visualize hand-written digits data set provided in scikit-learn [50] as an example of visualizing classical data. Specifically, we split 80% of the given data as training data and the remaining 20% as test data. We employ the quantum feature map defined by a quantum circuit described in Fig. 5.2. Specifically, we use 12 qubits and the quantum circuit consisting of 48 single-qubit rotation gates for data encoding, 12 Hadmard gates and 12 CZ gates.

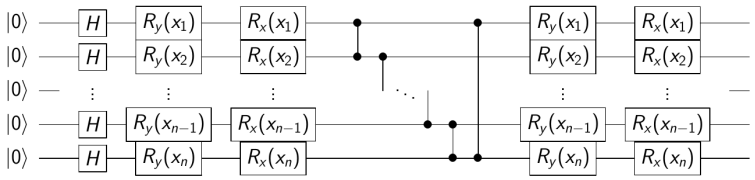


Figure 5.2: The encoding circuit for the hand-written digits data set. In the figure, x_i is the ith principal component of a data $\boldsymbol{x}$.

Table 5.2: The accuracy of 5-fold cross-validation by k-nearest neighbor algorithm

	$k=1$	$k=10$	$k=20$
[proposed method] quantum kernel	0.792	0.824	0.825
[t-sne(gauss)] classical gaussian kernel	0.788	0.816	0.814
[t-sne(nn1)] parametric t-SNE	0.787	0.836	0.838
[t-sne(nn3)] parametric t-SNE	**0.870**	**0.884**	**0.883**
[t-sne(qnn)] quantum neural network	0.678	0.749	0.745

As a preprocess, we reduce the dimension of data to 12 by principal component analysis. Additionally, we normalize each principal component from 0 to $\pi/2$ to input data through the angles of quantum gates. As shown in Fig. 5.2, we input the i-th principal component x_i as angles of single-qubit rotations acting on qubit i. We optimize parameters $\boldsymbol{\alpha}_l$ with Adam [57]. Note that we set the perplexity to 30 and the initial values of $\boldsymbol{\alpha}$ to uniform random values between 0 and 1. We show the resulting low-dimensional data in Fig. 5.3. The left figure visualizes the training data, and the right one visualizes the test data. From these figures, we can see that the clusters are formed by the data belonging to each class representing digits.

Next, to quantitatively evaluate the generalization performance of the model, we classify the low-dimensional data points by k-nearest neighbor algorithm and compare the 5-fold cross-validation accuracy with four conventional models. Specifically, the models to be compared are a kernel t-SNE with Gaussian kernels (t-sne(gauss)), a parametric t-SNE with a quantum neural network (t-sne(qnn)), a parametric t-SNE with a neural network with a single-hidden layer (t-sne(nn1)), and that with three hidden layers (t-sne(nn3)). We describe their details in Appendix A.2. The accuracies are shown in Table 5.2. The accuracy of t-sne(qnn) is the worst for all values of k. It is because we could not use many parameters due to the long optimization time it causes. From Table 5.2, our model achieves almost the same accuracy with t-sne(gauss), but results in a lower accuracy than t-sne(nn1) and t-sne(nn3).

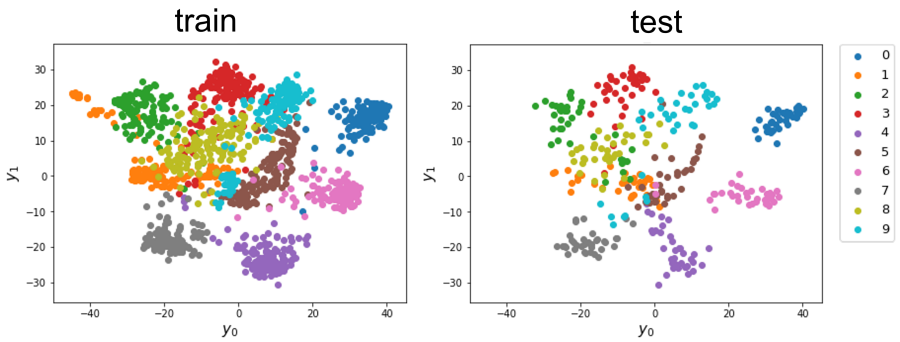


Figure 5.3: These figures visualize the hand-written digits data set, where we split 80% of the given data as training data and the remaining 20% as test data. The left figure visualizes the training data and the right one visualizes the test data.

5.4.2 Visualizing quantum data

Since we verified our model in the previous section, in this section, we visualize quantum data as a practical example that is useful in understanding the optimization trajectories and the characteristics of ansatzes by visualization. In Ref. [67], they visualize loss landscape and optimization trajectories on 1st and 2nd principal component vectors. In our method, we visualize optimization trajectories based on quantum states. This method allows us to visualize the trajectories even if the trajectories do not exist in the plane consisting of 1st and 2nd principal component vectors. In this numerical experiment, we visualize the optimization process of finding the ground states of transverse field Ising model:

$$H = J\sum_{i=1}^{n-1} Z_i Z_{i+1} + h\sum_{i=1}^{n} X_i,$$

where we set $n = 12, J = -2.0, h = -0.50$. This model has two degenerate ground states $|\psi_+\rangle$ and $|\psi_-\rangle$. To find the ground states by a quantum variational eigensolver, we use hardware-efficient ansatz [7, 64] and alternating layered ansatz [14, 64], which are respectively shown in Figures 5.4 and 5.5. We set the depth $d = 8$ and initialize the parameters with uniform random variables between 0 and $\pi/2$. We use five different initial parameters each for hardware efficient ansatz and alternating layered ansatz. The parameters are optimized by BFGS algorithm implemented in scipy with 100 iterations. The quantum state at each iteration of the optimization is visualized by our kernel t-SNE method. The optimization of the parameters in Eq. (5.1) is performed by Adam [57] setting the perplexity to 10 and the initial values of $\boldsymbol{\alpha}$ to uniform random values between 0 and 0.01.

We show the result in Fig. 5.6. The initial states exist around the origin because Eq. (5.1) returns values that are close to zeros when a quantum state has

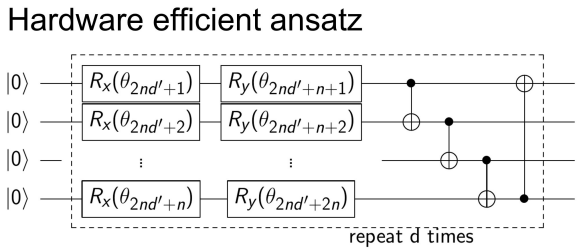


Figure 5.4: This figure shows hardware efficient ansatz, which we use in our numerical experiments. This ansatz consists of single qubit rotation gates and CNOT gates. We set $n = 12$ and $d = 8$. We repeat the block as $d' = 0, 1, \ldots, d-1$.

low similarity to all other states in the data unless α_l is large. They move away from the origin as the optimizations proceed. The alternating layered ansatz converges to two different states $(|\psi_+\rangle + |\psi_-\rangle)/\sqrt{2}$ and $(|\psi_+\rangle - |\psi_-\rangle)/\sqrt{2}$ depending on their initial parameters. From Fig. 5.6, we observe the trajectories of alt0, alt1, alt3, and alt4 in Fig. 5.6 are almost the same. A possible reason is that our alternating layered ansatz has poor expressibility because alternating layered ansatz requires a deep quantum circuit in each block of U_1 and U_2 to improve its expressibility [64]. On the other hand, the hardware efficient ansatz converges to $|\psi_+\rangle$ or $|\psi_-\rangle$. The trajectories to $|\psi_+\rangle$ have varieties. This means that the optimization trajectories of hardware efficient ansatz highly depend on the initial values and may have high expressibility. Another possible analysis is that the trajectory of hea2 transits the quantum states satisfying $|\psi(p)\rangle \coloneqq \sqrt{p}\,|\psi_+\rangle + \sqrt{1-p}\,|\psi_-\rangle$, where $2/3 \leq p \leq 1$. This can be guessed from Fig. 5.6 and the fidelities of quantum states, which reflects the distance of them, because of the fidelities $|\langle\psi(0.66)|\psi_{\mathrm{hea2},40}\rangle|^2 = 0.980$, $|\langle\psi(0.70)|\psi_{\mathrm{hea2},50}\rangle|^2 = 0.990$, $|\langle\psi(0.80)|\psi_{\mathrm{hea2},60}\rangle|^2 = 0.995$, $|\langle\psi(0.90)|\psi_{\mathrm{hea2},70}\rangle|^2 = 0.994$, $|\langle\psi(0.99)|\psi_{\mathrm{hea2},80}\rangle|^2 = 0.998$, $|\langle\psi(1.00)|\psi_{\mathrm{hea2},90}\rangle|^2 = 0.998$, and $|\langle\psi(1.00)|\psi_{\mathrm{hea2},100}\rangle|^2 = 0.999$, where we denote $|\psi_{\mathrm{hea2},i}\rangle$ as the quantum states of hea2 at ith iteration.

Our visualization method is useful in analyzing the optimization trajectories based on the quantum states. From the visualization, we find the characteristics of ansatzes' trajectory, especially for the dependence of initialization. Additionally, we can guess the trajectory by the visualization without calculation of fidelities. This may be useful in limiting the range of random initialization. Note that our visualization method is different from the visualization method in 1st and 2nd principal component vectors, used to visualize trajectories and loss landscapes around the reached local minimum [24, 67]. Specifically, the principal component approach is appropriate to analyze the trajectories around the reached local optimum. However, the approach can not know the quantum states corresponding to the parameters. In this regard, our approach can provide insight into the optimization trajectories based on quantum states.

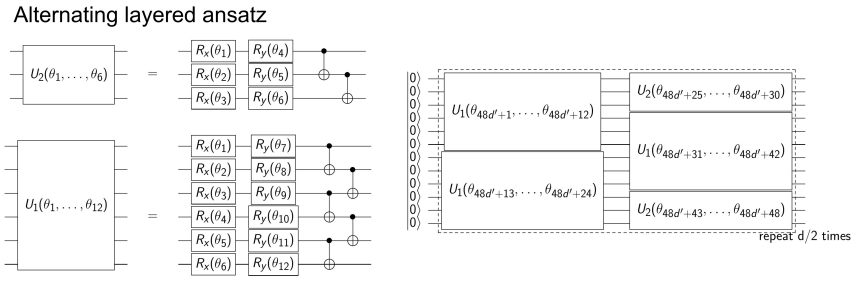


Figure 5.5: These figures show alternating layered ansatz, which we used in our numerical experiments. This ansatz consists of two quantum circuits of U_1 and U_2. In our numerical experiments, we set the parameters $n = 12$ and $d = 8$. We repeat the U_1 and U_2 as $d' = 0, 1, \ldots, d-1$.

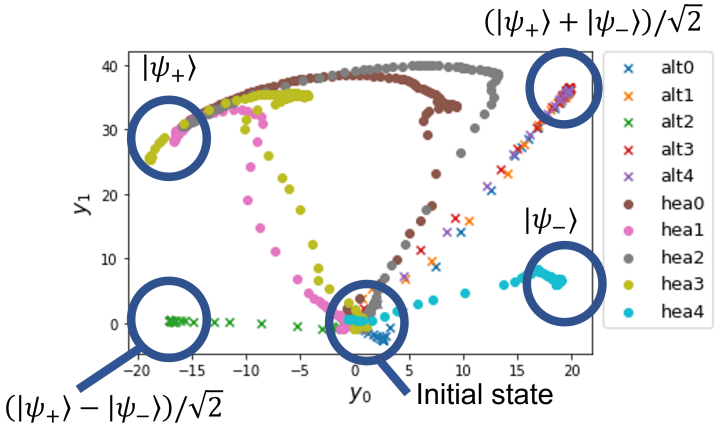


Figure 5.6: This figure shows the optimization process of a variational quantum algorithm to find the ground states of transverse field Ising model. The ansatzes to find the ground states are hardware efficient ansatz and alternating layered ansatz, shown in Figs. 5.4 and 5.5. Here, we visualize five trajectories that start from different initial values for each ansatz and label them as hea$i$ and alt$i$.

5.5 Conclusion

In this chapter, we propose a data visualization method using quantum kernels. We confirm that our proposed method can visualize both classical and quantum data by numerical experiments.

For classical data, we visualize a hand-written digits data set and apply k-nearest neighbor algorithm to the low-dimensional data to validate our proposed method. The accuracy of our proposed method is almost the same as a kernel t-SNE with Gaussian kernels, but lower than a parametric t-SNE with a neural network. Since our quantum feature map in Sec. 5.4.1 may not be the best choice of quantum circuit architecture in terms of the prediction accuracy, we can improve the quantum feature map by a data re-uploading circuit [52, 51] and a parameterized quantum circuit.

For quantum data, we visualize the optimization process of a variational quantum algorithm as an example use case. From the visualization in our numerical experiments, we find the characteristics of each ansatz's optimization trajectory. Specifically, the trajectories of hardware efficient ansatz have varieties depending on the initial values. The visualization gives us a prediction of the trajectories, which would be useful in improving the quantum circuit architecture or changing the initialization. On the other hand, the trajectories of alternating layered ansatz are almost the same. This may be because our design of alternating layered ansatz in each block of U_1 and U_2 is too simple. If we employ it with a deep quantum circuit, the trajectories may have more varieties like hardware efficient ansatz. The visualization of trajectories may give us an insight into avoiding barren plateaus or achieving better solutions by improving the initialization strategy or design of a quantum circuit not only for variational quantum eigensolver but also for variational quantum algorithms.

One interesting usage of our visualization method is in quantum architecture search. By visualizing the quantum states that accompany the addition or removal of quantum gates, we may construct better quantum circuits. Our visualization method would contribute to developing quantum machine learning algorithms by improving the design of a quantum circuit and the optimization strategies.

Chapter 6

Conclusion

We developed methods of fast classical simulation of quantum dynamics and visualization of quantum states, both of which are helpful in developing and evaluating quantum machine learning algorithms. Classical simulation and visualization techniques are essential tools for advancing quantum machine learning research, as they allow for the efficient development, testing and understanding of quantum algorithms, which is crucial for the advancement of this field.

First, we have proposed a method of accelerating the classical simulation of quantum dynamics by simultaneous diagonalization of mutually commuting Hamiltonian terms in Chapter 3. We succeeded in accelerating the classical simulation of quantum dynamics of transverse field Ising model and SYK majorana fermion model. In our numerical experiments, our proposed method provides approximately ten times acceleration compared with a general purpose quantum circuit simulator. This method allows us to accelerate the classical simulation of quantum algorithms, e.g., quantum circuit learning and quantum annealing.

Next, we have proposed methods of visualizing quantum states, which are out-of-sample extension methods of t-SNE by a quantum neural network and quantum kernels. The concept of these methods is that we make a map of the quantum states as high-dimensional data into a two-dimensional plane by optimizing the parameters with keeping the similarity between the quantum states. In Chapter 4, we have proposed a method of visualizing quantum states by a parameterized quantum circuit. This method allows us to visualize quantum dynamics by applying and optimizing a parameterized quantum circuit to quantum states. However, this method is required to optimize the parameters and the design of a parameterized quantum circuit applied to quantum states for visualization, which may cause optimization problems of barren plateaus and existence of many local minima, as we mentioned. This limits the direct evaluation of the visualization and quantum feature maps. To avoid optimizing a parameterized quantum circuit, we have proposed a visualization method using quantum kernels in Chapter 5. This method allows us to evaluate quantum feature maps since we can visualize quantum states encoded classical data without applying and optimizing a parameterized quantum circuit to them. In our numerical ex-

periments, we visualize handwritten digits data set in a two-dimensional space and apply k-nearest neighbor algorithm to the two-dimensional data to evaluate our proposed method by classification accuracy. Since this method can visualize quantum states faster than a quantum neural network presented in Chapter 4, we succeeded in visualizing optimization trajectories of variational quantum eigensolver. Our visualization method is expected to help find a better design of quantum circuit, which avoids barren plateaus or reaches a better local minimum.

Our proposed methods of accelerating a classical simulation of quantum dynamics and the visualization of quantum states are important to develop quantum computing research, especially in quantum machine learning in the future. Specifically, our proposed fast classical simulation of quantum dynamics allows us to accelerate the simulation of transverse field Ising model approximately ten times faster than a general purpose quantum circuit simulator. This acceleration is also available in accelerating classical simulation of quantum algorithms, such as quantum circuit learning, quantum approximate optimization algorithms, and quantum annealing, since they employ Ising type dynamics. It is still unclear whether these quantum algorithms offer any significant advantage over their classical counterparts. The proposed faster classical simulation techniques presented in this thesis can address this issue by providing researchers with the ability to rapidly test and iterate on different circuit designs. This is crucial for identifying the best circuit architecture that can perform a specific task and for determining whether a quantum advantage exists.

In terms of visualization, we have opened up a field of visualization of quantum states. Our proposed visualization method by quantum kernels is useful in evaluating and improving the design of quantum feature maps through the classification accuracy of two-dimensional data. One notable application is in quantum architecture search, where the visualization of the search space and the trajectory of quantum states can aid in identifying the best circuit architecture and improving the optimization strategy. Additionally, visualization of optimization strategies can aid in identifying the best initialization methods to avoid barren plateaus and achieve better local minima. Another intriguing research direction is to use the visualization method for data compression since visualization methods are based on dimensionality reduction techniques. It would be effective if a Hilbert space is too large to learn the patterns in the data. We hope that the techniques developed in this thesis provide valuable insights into the potential for achieving quantum advantage in quantum machine learning paving the way for future breakthroughs in this field.

List of publications

- Y. Kawase, K. Mitarai, and K. Fujii, "Parametric t-stochastic neighbor embedding with quantum neural network," Phys. Rev. Res., vol. 4, 2022.
- Y. Kawase and K. Fujii, "Fast Classical Simulation of Hamiltonian Dynamics by Simultaneous Diagonalization Using Clifford Transformation with Parallel Computation", arXiv:2206.11664, 2022.
- Y. Kawase, K. Mitarai, and K. Fujii, "Quantum Kernel t-Distributed Stochastic Neighbor Embedding" is now writing and will be uploaded to arXiv.
- Y. Suzuki, Y. Kawase, et al., "Qulacs: a fast and versatile quantum circuit simulator for research purpose." Quantum, vol. 5, 2021.

Appendix A

Appendix

A.1 Appendix for fast classical simulation of Hamiltonian dynamics

A.1.1 The algorithm of searching linear independent terms

Here we explain the algorithm for finding a set of linear independent terms from the mutually commuting terms. The algorithm is described in Algorithm 7.

Let us consider an example of searching linear independent terms of $H = ZYXI + ZXYI + IYXZ + IXYZ + YIZX + YZIX + XIZY + XZIY$. First, we transform the terms into binary strings and make a tableau as we described in Sec. 3.2.2. Let us denote the i_r as the row index and i_c as the column index in the tableau. We start from the 1st column and 1st row of the tableau. If the i_c th column element in i_r th row is zero and we find the $j(> i_r)$th row whose i_c th column element is one, we swap the jth and i_rth rows. Otherwise, we increase the i_c to i_{c+1} and find the row whose i_cth column element is one again. In Fig. A.1 (1), we swap the 1st and 5th row, because the 1st column of the 1st row is zero and that of the 5th row is one. Then, we search for the $k(> i_r)$th rows whose i_c th column elements have one. If we find them, we take exclusive-OR between i_r th row and the kth rows. In Fig. A.1 (2), since the 1st column element of the $6, 7$, and 8th rows is one, we apply exclusive-OR between the 1st row and $6, 7$, and 8th rows. We increment i_r and i_c to $i_r + 1$ and $i_c + 1$, respectively. We repeat it until $i_r = m_l$ or $i_c = 2n$, as in Fig. A.1 (3), (4), and (5). The terms corresponding to the rest of the rows are the linear independent terms. In Fig. A.1 (6), since the rest of the terms corresponds to the initial 5th, 2nd, 3rd, and 4th rows, the linearly independent terms are $YIZX$, $ZXYI$, $IYXZ$, and $IXYZ$.

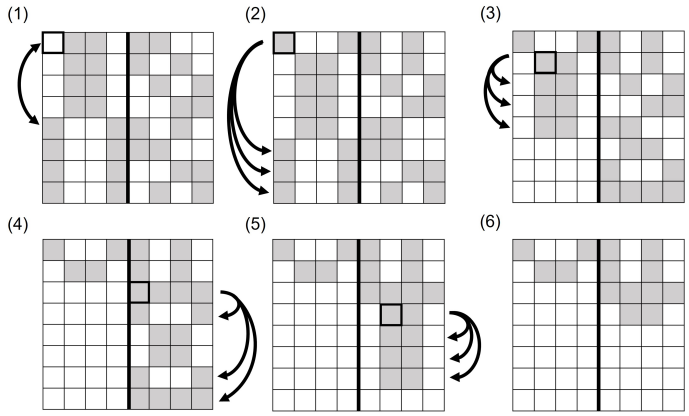


Figure A.1: This figure shows an example of the process of searching the linear independent vectors in the binary tableau consisting of mutually commuting terms.

Algorithm 7 Implementation of searching linear independent vectors

Require: A tableau of binary representation of mutually commuting terms

1: Let us denote m_l as the number of terms and n as the number of qubits.
2: Let us denote i_r as i_r th rows (terms) in the tableau and i_c as the i_c th columns in the tableau.
3: $i_r \leftarrow 1, i_c \leftarrow 1$.
4: **while** $i_r \leq m_l$ and $i_c \leq 2n$ **do**
5: **if** the element of i_cth column in the i_rth row is 0 **then**
6: We search for the $j(> i_r)$th row whose element of i_cth column is 1.
7: If we find such a row, we swap the i_rth row with the jth row.
8: Otherwise we increment i_c as $i_c \leftarrow i_c + 1$ and go back to line 4.
9: **end if**
10: **if** the element of i_cth column in the i_rth row is 1 **then**
11: We delete the ones in the i_cth column of $k(> i_r)$th row by exclusive-OR between the i_rth row and the kth row.
12: $i_r \leftarrow i_r + 1$, $i_c \leftarrow i_c + 1$.
13: **end if**
14: **end while**
15: **return** the corresponding terms whose elements in the final binary tableau have at least one non-zero element.

A.1.2 Proof of our simultaneous diagonalization algorithm

In this section, we prove our partial diagonalization algorithm in Sec. 3.3.2 as follows:

(i) deriving the condition of mutually commuting terms

First, we derive the condition of mutually commuting terms. From the definition of Pauli terms $P_{j,k}$ in Eq. (3.2), the commutation relation between $P_{j,k}$ and $P_{j,k'}$ is described by

$$[P_{j,k}, P_{j,k'}] = \{(-1)^{\sum_{l=1}^{n} z_{k,l}x_{k',l}} - (-1)^{\sum_{l=1}^{n} x_{k,l}z_{k',l}}\} i^{s_k+s_{k'}} \bigotimes_{l=1}^{n} X^{x_{k,l}+x_{k',l}} Z^{z_{k,l}+z_{k',l}}.$$

So, if the terms commute, the following relation holds:

$$\sum_{l=1}^{n} z_{k,l}x_{k',l} + x_{k,l}z_{k',l} \equiv 0 \pmod 2. \tag{A.1}$$

(ii) proving that the rank of X block maximized by H gates equals to that of tableau.

According to Algorithm 7, we turn the first r_x of diagonal elements into ones, where r_x denotes the rank of X block, and zero out lower triangular matrix elements in X block. These elements are described by

$$x_{k,k} = 1, \qquad \forall k \in \{1, \ldots, r_x\}, \tag{A.2}$$

$$x_{k,l} = 0, \qquad \forall k \in \{1, \ldots, m_j\}, l \in \{1, \ldots, \min(k-1, r_x)\}. \tag{A.3}$$

Note that the first r_x of diagonal elements are not always ones in general, but it does not lose generality by supposing adequately choosing the qubit number.

Next, we explain

$$x_{k,l} = 0, \qquad \forall k \in \{r_x + 1, \ldots, m_j\}, l \in \{r_x + 1, \ldots, n\} \tag{A.4}$$

is satisfied using proof by contradiction. If $x_{k,l} = 1$ exists such that $r_x + 1 \leq k \leq m_j$ and $r_x + 1 \leq l \leq n$ are satisfied, these rows are linearly independent for all first r_x of rows. So, this is contradicted with the rank of X block being r_x. Therefore, Eq. (A.4) is hold. From Eq. (A.2), Eq. (A.3), and Eq. (A.4), the current tableau is shown in Fig. A.2 (a).

Let us apply H gates to $(r_x + 1), \ldots, n$ th columns. Now, the elements of $(r_x + 1)$th to nth columns and $(r_x + 1)$th to m_jth rows in Z block are all zeros, i.e.

$$z_{k,l} = 0, \qquad \forall k \in \{r_x + 1, \ldots, m_j\}, \forall l \in \{r_x + 1, \ldots, n\}. \tag{A.5}$$

Let us suppose the rank of tableau is $r_x + r_z$. By applying XOR operations and swapping rows, we can set the $(r_x + 1), \ldots, (r_x + r_z)$ th diagonal elements in X block to ones, and the lower triangular elements of $(r_x + 1), \ldots, (r_x + r_z)$th columns in X block can be zeroed out. We obtain

$$x_{k,k} = 1, \qquad \forall k \in \{r_x + 1, \ldots, r_x + r_z\} \tag{A.6}$$

$$x_{k,l} = 0, \qquad \forall k \in \{r_x + 1, \ldots, m_j\}, \forall l \in \{r_x + 1, \ldots, \min(k-1, r_x + r_z)\}. \tag{A.7}$$

After this process, the elements in $(r_x + r_z + 1)$th to nth columns below $(r_x + r_z + 1)$th rows in X block are all zeros since the rank of tableau is $(r_x + r_z)$. So, we have

$$x_{k,l} = 0, \qquad \forall k \in \{r_x + r_z + 1, \ldots, m_j\}, \forall l \in \{r_x + r_z + 1, \ldots, n\}. \tag{A.8}$$

Here we prove the rest of the elements below $(r_x + r_z + 1)$th rows are also zeros:

$$z_{k,1} = \cdots = z_{k,r_x} = 0, \qquad \forall k \in \{r_x + r_z + 1, \ldots, m_j\}. \tag{A.9}$$

From Eq. (A.1), Eq. (A.3), Eq. (A.5), Eq. (A.7), and Eq. (A.8), we have

$$\sum_{l=1}^{r_x} x_{k,l} z_{k',l} \equiv 0 \pmod 2, \qquad \forall k \in \{1, \ldots, r_x + r_z\}, \forall k' \in \{r_x + r_z + 1, \ldots, m_j\}. \tag{A.10}$$

Using Eq. (A.2) and Eq. (A.3), and assigning $k = r_x, (r_x - 1), \ldots, 1$ to Eq. (A.10), we get

$$\begin{aligned} z_{k',r_x} &\equiv 0 \pmod 2, \\ z_{k',r_x-1} + x_{r_x-1,r_x} z_{k',r_x} &\equiv 0 \pmod 2, \\ &\vdots \\ z_{k',1} + \sum_{l=2}^{r_x} x_{k,l} z_{k',l} &\equiv 0 \pmod 2. \end{aligned}$$

From these relations, we obtain Eq. (A.9). Therefore, the rank of X block maximized by H gates is $r_x + r_z$, which is the same as the rank of tableau, and the upper bound is n. Now, the tableau is described in Fig. A.2 (b).

(iii) explaining upper triangular elements in X block can be zeroed out by CNOT gates.

Since the rank of X block is maximized by the previous process, the first $r_x + r_z$ of binary strings are linearly independent. So, we can turn the first $r_x + r_z$ of diagonal elements into ones and the upper triangular elements into zeros.

(iv) proving that Z block can be zeroed out by S and CZ gates.

Here we consider zeroing out elements in Z block using S and CZ gates. We can zero out the elements in 1st row of Z block according to our Algorithm 5. In concrete, since $x_{1,1} = 1$ and $x_{1,l} = 0, \forall l \in \{2, \ldots, n\}$ are hold, we can zero out $z_{1,1}$ by $S(1)$ and $z_{1,l}$ by $\mathrm{CNOT}(1, l)$. Similarly, we repeatedly use S gates to zero out the diagonal elements and CZ gates to zero out non-diagonal elements up to $r_x + r_z$th row. To prove the elements in Z block can be zeroed out by S and CZ gates, here we consider zeroing out the elements of kth row in Z block under the assumption that we have zeroed out the elements from 1st to $k-1$th row in Z block. Since all terms in a group mutually commute, we hold

$$\sum_{l=1}^{n} z_{k,l} x_{k',l} + x_{k,l} z_{k',l} \equiv 0 \pmod 2, \qquad \forall k' \in \{1, \ldots, k-1\}. \tag{A.11}$$

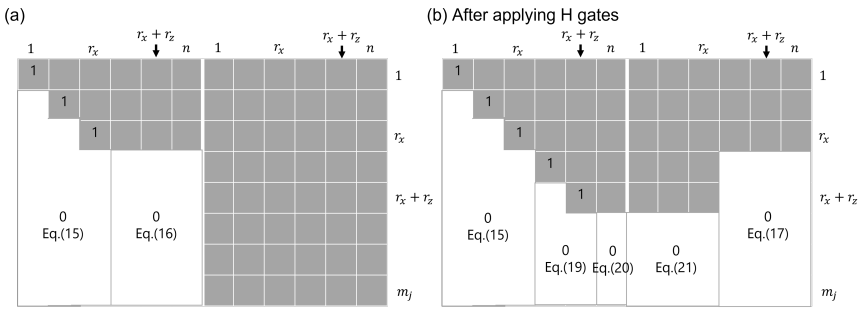


Figure A.2: The left figure shows binary tableau after the first r_x of diagonal elements are turned into ones. At this point, the lower triangular elements in X block and the elements below r_x+1th row in X block are zeros. The right figure shows binary tableau after applying H gates to $r_x+1,\ldots,n$th qubits, and by XOR operations and swapping rows, the first r_x+r_z of diagonal elements are turned into ones.

Since we have zeroed out elements from 1st to $(k-1)$th row in Z block, i.e. $z_{k',l}=0, \forall k' \in \{1,\ldots,k-1\}, \forall l \in \{1,\ldots,n\}$, we get

$$\sum_{l=1}^{n} z_{k,l}x_{k',l} \equiv 0 \pmod 2, \qquad \forall k' \in \{1,\ldots,k-1\}. \tag{A.12}$$

Since the upper triangular elements in X block are zeros, i.e. $x_{k',l}=0, \forall k' \in \{1,\ldots,k-1\}, \forall l \in \{k'+1,\ldots,n\}$, we obtain

$$\sum_{l=1}^{k'} z_{k,l}x_{k',l} \equiv 0 \pmod 2, \qquad \forall k' \in \{1,\ldots,k-1\}. \tag{A.13}$$

Since the diagonal elements in X block are ones, by assigning $k'=1,\ldots,k-1$, we get

$$\begin{aligned} z_{k,1} &\equiv 0 \pmod 2, \\ z_{k,1}x_{2,1} + z_{k,2} &\equiv 0 \pmod 2, \\ &\vdots \\ \sum_{l=1}^{k-2}(z_{k,l}x_{k-1,l}) + z_{k,k-1} &\equiv 0 \pmod 2. \end{aligned}$$

From these equations and $z_{k,k'}$ being binary integer, we obtain

$$z_{k,k'}=0, \qquad \forall k' \in \{1,\ldots,k-1\}. \tag{A.14}$$

Since we have Eq. (A.14) and we can zero out the rest of elements in kth row by S and CZ gates, we can zero out all the elements in Z block by iteratively zeroing out the elements up to r_x+r_zth row.

(v) turning the X components into Z components by H gates

Since the elements in Z block are all zeros, we can turn the X components into Z components by H gates.

A.2 The models used in numerical experiments for visualizing classical data with quantum kernels

Here, we explain our models used in numerical experiments. Since we described our proposed method in Sec. 5.3.1, we explain the other models below.

- Kernel t-SNE with Gaussian kernels
  As we explained in Sec. 5.2, the kernel t-SNE is a method of mapping high-dimensional data to low-dimensional space using kernel functions so that the similarities between low-dimensional data preserve those between high-dimensional data. Specifically, we adopt the kernel function in Eq. (5.1) as a Gaussian kernel $k(\boldsymbol{x_l}, \boldsymbol{x}) = \exp\left(-\|\boldsymbol{x_l} - \boldsymbol{x}\|^2/2\right)$. Then, we optimize the $\boldsymbol{\alpha}_l$ to minimize the cost function Eq. (2.9).

- Parametric t-SNE
  As we explained in Sec. 4.2, parametric t-SNE with a neural network is a method of mapping high-dimensional data to low-dimensional space using a neural network in such a way that the similarities between low-dimensional data are almost equal to those between high-dimensional data. First, we explain the architecture of a single-hidden layer neural network used in the numerical experiments. As shown in Fig. A.3, the architecture consists of an input layer, a hidden layer, the activation function of ReLU, and an output layer. We set the number of units in the hidden layer to 1800, which is approximately the same as the number of data and thus to the number of parameters used in kernel t-SNE. Next, we explain the architecture of a three-hidden layer neural network used in the numerical experiments. As shown in Fig. A.4, the architecture consists of an input layer, hidden layers, the activation function of ReLU, dropout layers with the rate of 0.25, and an output layer. Note that after we reduce the dimensions of the data to 12 by the principal component analysis, we standardize each feature to have a mean of zero and a variance of one.

- Parametric t-SNE with a quantum neural network
  As we explained in Sec. 4.3, parametric t-SNE with a quantum neural network is a method of mapping high-dimensional data to low-dimensional space using a quantum neural network so that the similarities between low-dimensional data preserve those between high-dimensional data [66]. As shown in Fig. A.5, the architecture of the quantum neural network consists of data encoding by the circuit described in Fig. 5.2 and unitary transformation including parameterized circuits. Specifically, we repeat

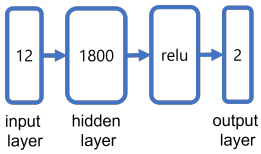


Figure A.3: This figure shows the architecture of the single-hidden layer neural network used in Sec. 5.4.1. The model consists of an input layer, a hidden layer, the activation function of ReLU, and an output layer.

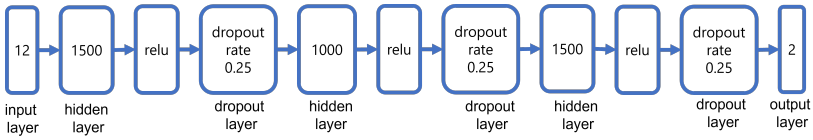


Figure A.4: This figure shows the architecture of the three-hidden layer neural network used in Sec. 5.4.1. The model consists of an input layer, hidden layers, the activation function of ReLU, dropout layers with the rate 0.25, and an output layer.

a circuit block consisting of U_{in} and U_{ent}, alternatively d times, where U_{in} and U_{ent} are described in Fig. A.5. We input data through U_{in}, and transform them by CZ and R_y gates. Note that we normalize each feature of the data to fit within $[0.0, 0.5]$ so that the principal component vectors are ranged within $[-1.0, 1.0]$. Then, We optimize the angles $\boldsymbol{\theta}$ in R_y gates in U_{ent}. Let us denote the map of the quantum neural network as $U(\boldsymbol{x}_i, \boldsymbol{\theta})$, the low-dimensional data are described by

$$y_{i,0}(\boldsymbol{x}_i, \boldsymbol{\theta}) = \sum_{j=1}^{n/2} \beta_j \langle 0| U^\dagger(\boldsymbol{x}_i, \boldsymbol{\theta}) X_j U(\boldsymbol{x}_i, \boldsymbol{\theta}) |0\rangle ,$$

$$y_{i,1}(\boldsymbol{x}_i, \boldsymbol{\theta}) = \sum_{j=n/2+1}^{n} \beta_j \langle 0| U^\dagger(\boldsymbol{x}_i, \boldsymbol{\theta}) X_j U(\boldsymbol{x}_i, \boldsymbol{\theta}) |0\rangle ,$$

where we optimize both β_j and $\boldsymbol{\theta}$ to minimize the cost function.

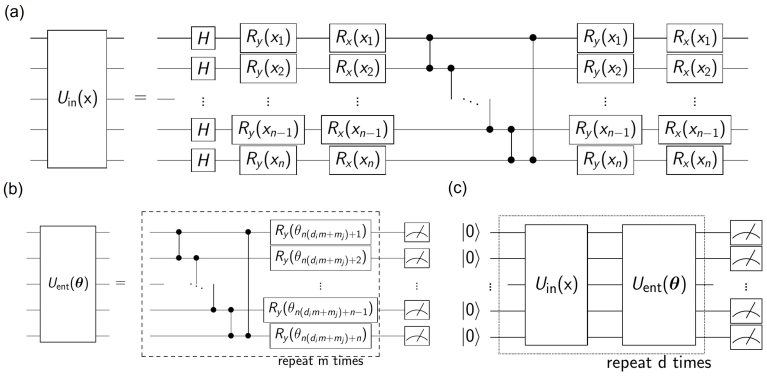


Figure A.5: This figure shows the architecture of the quantum neural network used in Sec. 5.4.1. We encode data by the same quantum circuit as in Fig. 5.2, and transform them by $U_{\text{ent}}(\boldsymbol{\theta})$ consisting of CZ and R_y gates. The angles in the R_y gates in $U_{\text{ent}}(\boldsymbol{\theta})$ are parameters to be optimized. We alternatively repeat $U_{\text{in}}(\boldsymbol{x})$ and $U_{\text{ent}}(\boldsymbol{\theta})$ d times.

Bibliography

[1] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[2] M.A. Nielsen and I.L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[3] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.

[4] Seth Lloyd. Universal quantum simulators. *Science*, 273(5278):1073–1078, 1996.

[5] John Preskill. Quantum computing in the nisq era and beyond. *Quantum*, 2:79, 2018.

[6] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[7] Abhinav Kandala, Antonio Mezzacapo, Kristan Temme, Maika Takita, Markus Brink, Jerry M Chow, and Jay M Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242–246, 2017.

[8] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[9] Kosuke Mitarai, Makoto Negoro, Masahiro Kitagawa, and Keisuke Fujii. Quantum circuit learning. *Physical Review A*, 98(3):032309, 2018.

[10] Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.

[11] Maria Schuld and Nathan Killoran. Quantum machine learning in feature hilbert spaces. *Physical review letters*, 122(4):040504, 2019.

[12] Yunchao Liu, Srinivasan Arunachalam, and Kristan Temme. A rigorous and robust quantum speed-up in supervised machine learning. *Nature Physics*, 17(9):1013–1017, 2021.

[13] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021.

[14] Marco Cerezo, Akira Sone, Tyler Volkoff, Lukasz Cincio, and Patrick J Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature communications*, 12(1):1–12, 2021.

[15] Jarrod R McClean, Sergio Boixo, Vadim N Smelyanskiy, Ryan Babbush, and Hartmut Neven. Barren plateaus in quantum neural network training landscapes. *Nature communications*, 9(1):1–6, 2018.

[16] Samson Wang, Enrico Fontana, Marco Cerezo, Kunal Sharma, Akira Sone, Lukasz Cincio, and Patrick J Coles. Noise-induced barren plateaus in variational quantum algorithms. *Nature communications*, 12(1):1–11, 2021.

[17] Yasunari Suzuki, Yoshiaki Kawase, Yuya Masumura, Yuria Hiraga, Masahiro Nakadai, Jiabao Chen, Ken M Nakanishi, Kosuke Mitarai, Ryosuke Imai, Shiro Tamiya, et al. Qulacs: a fast and versatile quantum circuit simulator for research purpose. *Quantum*, 5:559, 2021.

[18] Matthew Treinish et al. Qiskit/qiskit: Qiskit 0.39.4, December 2022.

[19] Cirq Developers. Cirq, December 2022. See full list of authors on Github: https://github .com/quantumlib/Cirq/graphs/contributors.

[20] Gian Giacomo Guerreschi, Justin Hogaboam, Fabio Baruffa, and Nicolas PD Sawaya. Intel quantum simulator: A cloud-ready high-performance simulator of quantum circuits. *Quantum Science and Technology*, 5(3):034007, 2020.

[21] Satoshi Imamura, Masafumi Yamazaki, Takumi Honda, Akihiko Kasagi, Akihiro Tabuchi, Hiroshi Nakao, Naoto Fukumoto, and Kohta Nakashima. mpiqulacs: A distributed quantum computer simulator for a64fx-based cluster systems. *arXiv preprint arXiv:2203.16044*, 2022.

[22] Hale F Trotter. On the product of semi-groups of operators. *Proceedings of the American Mathematical Society*, 10(4):545–551, 1959.

[23] Masuo Suzuki. General theory of fractal path integrals with applications to many-body theories and statistical physics. *Journal of Mathematical Physics*, 32(2):400–407, 1991.

[24] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. *Advances in neural information processing systems*, 31, 2018.

[25] Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. Why does unsupervised pre-training help deep learning? In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 201–208. JMLR Workshop and Conference Proceedings, 2010.

[26] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

[27] Yuan Yang, Zheng-Zhi Sun, Shi-Ju Ran, and Gang Su. Visualizing quantum phases and identifying quantum phase transitions by nonlinear dimensional reduction. *Physical Review B*, 103(7):075106, 2021.

[28] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse ising model. *Physical Review E*, 58(5):5355, 1998.

[29] Scott Aaronson and Daniel Gottesman. Improved simulation of stabilizer circuits. *Physical Review A*, 70(5):052328, 2004.

[30] Pranav Gokhale, Olivia Angiuli, Yongshan Ding, Kaiwen Gui, Teague Tomesh, Martin Suchara, Margaret Martonosi, and Frederic T Chong. Minimizing state preparations in variational quantum eigensolver by partitioning into commuting families. *arXiv preprint arXiv:1907.13623*, 2019.

[31] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[32] Lata Kh Joshi, Andreas Elben, Amit Vikram, Benoît Vermersch, Victor Galitski, and Peter Zoller. Probing many-body quantum chaos with quantum simulators. *Physical Review X*, 12(1):011018, 2022.

[33] Brian Swingle, Gregory Bentsen, Monika Schleier-Smith, and Patrick Hayden. Measuring the scrambling of quantum information. *Physical Review A*, 94(4):040302, 2016.

[34] Kevin A Landsman, Caroline Figgatt, Thomas Schuster, Norbert M Linke, Beni Yoshida, Norman Y Yao, and Christopher Monroe. Verified quantum information scrambling. *Nature*, 567(7746):61–65, 2019.

[35] Jiehang Zhang, PW Hess, A Kyprianidis, P Becker, A Lee, J Smith, G Pagano, I-D Potirniche, Andrew C Potter, A Vishwanath, et al. Observation of a discrete time crystal. *Nature*, 543(7644):217–220, 2017.

[36] Benjamin Villalonga, Sergio Boixo, Bron Nelson, Christopher Henze, Eleanor Rieffel, Rupak Biswas, and Salvatore Mandra. A flexible high-performance simulator for verifying and benchmarking quantum circuits implemented on real hardware. *npj Quantum Information*, 5(1):1–16, 2019.

[37] Axel D Dente, Carlos S Bederián, Pablo R Zangara, and Horacio M Pastawski. Gpu accelerated trotter-suzuki solver for quantum spin dynamics. *arXiv preprint arXiv:1305.0036*, 2013.

[38] Alberto Peruzzo, Jarrod McClean, Peter Shadbolt, Man-Hong Yung, Xiao-Qi Zhou, Peter J Love, Alán Aspuru-Guzik, and Jeremy L O ' brien. A variational eigenvalue solver on a photonic quantum processor. *Nature communications*, 5(1):1–7, 2014.

[39] Ewout van den Berg and Kristan Temme. Circuit optimization of hamiltonian simulation by simultaneous diagonalization of pauli clusters. *arXiv preprint arXiv:2003.13599*, 2020.

[40] Subir Sachdev and Jinwu Ye. Gapless spin-fluid ground state in a random quantum heisenberg magnet. *Physical review letters*, 70(21):3339, 1993.

[41] Alexei Kitaev. A simple model of quantum holography, 2015.

[42] Eiki Iyoda and Takahiro Sagawa. Scrambling of quantum information in quantum many-body systems. *Physical Review A*, 97(4):042330, 2018.

[43] Vladyslav Verteletskyi, Tzu-Ching Yen, and Artur F Izmaylov. Measurement optimization in the variational quantum eigensolver using a minimum clique cover. *The Journal of Chemical Physics*, 152(12):124114, 2020.

[44] Juan Maldacena and Douglas Stanford. Remarks on the sachdev-ye-kitaev model. *Physical Review D*, 94(10):106002, 2016.

[45] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[46] Kelvin Ch'ng, Nick Vazquez, and Ehsan Khatami. Unsupervised machine learning account of magnetic transitions in the hubbard model. *Physical Review E*, 97(1):013306, 2018.

[47] Yuan Yang, Zheng-Zhi Sun, Shi-Ju Ran, and Gang Su. Visualizing quantum phases and identifying quantum phase transitions by nonlinear dimensional reduction. *Phys. Rev. B*, 103:075106, Feb 2021.

[48] Ken N Okada, Keita Osaki, Kosuke Mitarai, and Keisuke Fujii. Identification of topological phases using classically-optimized variational quantum eigensolver. *arXiv preprint arXiv:2202.02909*, 2022.

[49] Laurens Van Der Maaten. Learning a parametric embedding by preserving local structure. In *Artificial intelligence and statistics*, pages 384–391. PMLR, 2009.

[50] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[51] Seth Lloyd, Maria Schuld, Aroosa Ijaz, Josh Izaac, and Nathan Killoran. Quantum embeddings for machine learning. *arXiv preprint arXiv:2001.03622*, 2020.

[52] Adrián Pérez-Salinas, Alba Cervera-Lierta, Elies Gil-Fuster, and José I Latorre. Data re-uploading for a universal quantum classifier. *Quantum*, 4:226, 2020.

[53] Harry Buhrman, Richard Cleve, John Watrous, and Ronald de Wolf. Quantum fingerprinting. *Phys. Rev. Lett.*, 87:167902, Sep 2001.

[54] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[55] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[56] davda54. unofficial implementation of sharpness-aware minimization algorithm for pytorch, 2020.

[57] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[58] Masuo Suzuki. Generalized trotter's formula and systematic approximants of exponential operators and inner derivations with applications to many-body problems. *Communications in Mathematical Physics*, 51(2):183–190, 1976.

[59] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin philosophical magazine and journal of science*, 2(11):559–572, 1901.

[60] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[61] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[62] Edward Farhi and Hartmut Neven. Classification with quantum neural networks on near term processors. *arXiv preprint arXiv:1802.06002*, 2018.

[63] Sukin Sim, Peter D Johnson, and Alán Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12):1900070, 2019.

[64] Kouhei Nakaji and Naoki Yamamoto. Expressibility of the alternating layered ansatz for quantum computation. *Quantum*, 5:434, 2021.

[65] Lennart Bittel and Martin Kliesch. Training variational quantum algorithms is np-hard. *Physical Review Letters*, 127(12):120502, 2021.

[66] Yoshiaki Kawase, Kosuke Mitarai, and Keisuke Fujii. Parametric t-stochastic neighbor embedding with quantum neural network. *Phys. Rev. Res.*, 4:043199, Dec 2022.

[67] Manuel S Rudolph, Sukin Sim, Asad Raza, Michal Stechly, Jarrod R McClean, Eric R Anschuetz, Luis Serrano, and Alejandro Perdomo-Ortiz. Orqviz: Visualizing high-dimensional landscapes in variational quantum algorithms. *arXiv preprint arXiv:2111.04695*, 2021.

[68] Andrej Gisbrecht, Wouter Lueks, Bassam Mokbel, and Barbara Hammer. Out-of-sample kernel extensions for nonparametric dimensionality reduction. In *ESANN*, 2012.

[69] Kerstin Bunte, Michael Biehl, and Barbara Hammer. A general framework for dimensionality-reducing data visualization mapping. *Neural Computation*, 24(3):771–804, 2012.

[70] Andrej Gisbrecht, Alexander Schulz, and Barbara Hammer. Parametric nonlinear dimensionality reduction using kernel t-sne. *Neurocomputing*, 147:71–82, 2015.

[71] Yuxuan Du, Min-Hsiu Hsieh, Tongliang Liu, Shan You, and Dacheng Tao. Learnability of quantum neural networks. *PRX Quantum*, 2(4):040337, 2021.

Acknowledgments

I would like to thank Prof. Fujii for supporting my doctoral program. Specifically, he gave me valuable advice and constructive discussions for improving our paper and research.

I also would like to thank Prof. Mitarai for his advice to improve our paper. The advice was useful in improving the advantages of our proposed method, detailed discussions, and numerical experiments.

I would like to thank Prof. Yamamoto in Keio University for his questions and comments on my presentation in QIQB seminar. Specifically, he told me that the data visualization would be nice if it could tell the difficulty of quantum machine learning. Thanks to his comments, I decided to approach the difficulty of optimization in quantum machine learning by visualizing data.

I would like to thank Prof. Kitagawa, Prof. Yamamoto, and Prof. Mizukami for reviewing my thesis.

I thank the students, alumni, and staff in Fujii Lab. Especially, Mr. Kubo and Mr. Shirai also work in quantum machine learning, and we often hold a meeting together. Their work and discussions are interesting, and I received a good stimulus for my research. Also, I thank Mr. Takahira and Mr. Yoshida for their comments on my presentation. Additionally, I thank Mr. Hakkaku for having a good time despite the many restrictions caused by the Corona Disaster.

I thank the staff at Health and Counseling Center at Osaka UNIVERSITY for supporting my mental health.