Noisy intermediate-scale quantum computation and its application

February 2022

Graduate School of Science and Technology, Keio University

Kohei Nakaji

A Thesis for the Degree of Ph.D. in Engineering

Noisy intermediate-scale quantum computation and its application

February 2022

Graduate School of Science and Technology, Keio University

Kohei Nakaji

Abstract

The quantum computer is expected to solve problems that the classical computer cannot find the answer to in a realistic time. While the development of quantum computers is accelerating, the near-term quantum computer (NISQ) that we will obtain in the next few decades is far from ideal; the number of qubits is small, and noise significantly affects the computational results.

This thesis aims to propose algorithms maximizing the functionality of NISQ and assess their feasibility. First, we propose the faster amplitude estimation (FAE) algorithm as an efficient quantum amplitude estimation algorithm (QAE) in NISQ. QAE is the algorithm to estimate the amplitude of a quantum state with good accuracy. Various quantum algorithms utilize QAE as their submodule. However, a well-known QAE algorithm using phase estimation is not executable in NISQ due to many noisy two-qubit gates. Recently proposed QAE algorithms successfully reduce the number of two-qubit gates, but the computational complexity upper bound is not given or loose. FAE proposed in this thesis both achieve the significant reduction of the number of two-qubit gates and the tight upper bound of the computational complexity.

Second, we propose the algorithm called the quantum semi-supervised generative adversarial network (qSGAN), which is the variant of the variational quantum algorithm (VQA). VQA is the most promising algorithm for solving real-world problems utilizing NISQ. VQA is the method to train a quantum circuit that generates a quantum state minimizing the expectation value of an observable. There are wide applications of VQA, such as the minimum eigenvalue problem of a Hamiltonian. Quantum generative adversarial network (QGAN) is a variant of VQA, which is proposed as the quantum counterpart of the classical machine learning algorithm called generative adversarial network (GAN). Even though GAN is successfully applied to the semi-supervised learning problems, there were no methods to apply QGAN to the semi-supervised learning problems. In this thesis, we propose qSGAN, the first application of QGAN to semi-supervised learning problems. We also demonstrate the performance of the algorithm.

Finally, we discuss the scalability issue of VQA. In VQA, we train the quantum circuit by updating the parameters embedded in the quantum circuit; if we find the optimal parameters, we obtain the optimal quantum circuit. We usually fix the structure of the quantum circuit; the structure often used is the so-called hardware efficient ansatz (HEA), which has enough expressive power to generate an arbitrary quantum state. However, it is recently shown that VQA with HEA suffers from the scalability issue called the barren plateau phenomenon; namely, as the number of qubits in HEA increases, the training of HEA becomes exponentially harder. The theoretically guaranteed solution to the issue is changing the quantum circuit's structure from HEA to the alternating layered ansatz (ALT). Still, even though VQA with ALT successfully avoids the barren plateau phenomenon under a specific condition, ALT may not have the expressive power that HEA has. If ALT cannot generate a wide variety of the quantum state, we cannot use it in VQA. In this thesis, with a proper definition of the expressibility found in literature, we show that ALT has almost the same expressive power as HEA. As a result, we show that VQA with ALT is a promising architecture that can avoid scalability issues and find solutions to various problems with NISQ.

Contents

List of Figures

Acknowledgements

First of all, I would like to express my gratitude to all the people who have helped me over the past two years of the doctoral program. In particular, I appreciate my advisor Prof. Naoki Yamamoto for the continuous support of my Ph.D. study and related research for his immense knowledge, motivation, and patience. Despite this challenging infectious disease situation, he effectively guided me in the research and writing whenever I asked him for help. I am sure that he is the best advisor and mentor for my Ph.D. study. I am also very grateful to him for starting and managing the Keio Quantum Computing Center in the first place. For a long time, I had regrets about not researching quantum computing even when I was working in the industry. Without the environment of the Keio Quantum Computing Center, I would not have been able to start my quantum computing research at all.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Tatsuhiko Koike, Prof. Shu Tanaka, and Prof. Hiroshi Watanabe. Their careful reading and insightful comments highly raised this thesis's quality and readability. Also, their difficult questions stimulated me to widen my research from diverse points of view.

Additionally, my sincere thanks go to the Keio quantum computing center members. I would like to thank my collaborators for inspiring discussions: Naoki Mitsuda, Tamiya Onodera, Rudy Raymond, Takahiko Sato, Yohichi Suzuki, Tomoki Tanaka, Hiroyuki Tezuka, and Shumpei Uno.

My research achievements, including this thesis, are also owed to those who guided me before I attended the doctoral program. People in the elementary particle physics laboratory at the University of Tokyo, where I got my master's degree, taught me the basics of the research act. People in SmartNews Inc., especially the founder, Mr. Kaisei Hamamoto, helped me obtain my software engineering skill, basic knowledge of machine learning, and startup spirits. Also, I would like to thank Prof. Naoaki Okazaki at the Tokyo Institute of Technology, an expert in natural language processing and an adviser of SmartNews, and Dr. Rikiya Takahashi, a colleague and an expert in machine learning — the discussion and the collaboration with them were so inspiring, which made me consider coming back to the academic research field. People in Boston Consulting Group helped me develop various capabilities such as critical thinking, effective communication skills, etc. More than anything, the friends I met there have constantly stimulated me to devote myself to big challenges.

Finally, I declare that I could not pursue the research without the help of my family including my parents. In particular, my partner Mayumi supported me in every way to concentrate on my research. She was also a mental guru with her positive thinking, who always ignited my inner motivation. My seven-year-old son Soichiro was also a big supporter of mine. I would never forget that he cared about me whenever I was in trouble. Thank you.

Kohei Nakaji
Keio University
Feburary 2022

Chapter 1

Introduction

Richard Feynman once shared the idea of a quantum computer that works according to the principles of quantum mechanics in 1982 [1]. Forty years later, we are now within reach of a quantum computer that applies to real-world problems. A decade ago, quantum computing was a secret art of a few researchers. In the past decade, the industry has been racing to develop quantum computers, including big companies like IBM, Google, and Microsoft and startups like IonQ and Rigetti.

A quantum computer is a device to perform calculations by sequentially changing the state of quantum bits (qubits). A quantum state represents the state of each qubit with two binary bases, $|0\rangle$ and $|1\rangle$. A quantum state with n qubits are represented by a quantum state in Hilbert space with 2^n bases: $|i_1, i_2, ..., i_n\rangle$ $(i_j = \pm 1)$. Each unitary operation called a quantum gate corresponds to each computational process; the sequence of the quantum gates is said to be the quantum circuit. We perform quantum computation by transforming a quantum state by a quantum circuit. In the case of a classical computer, the scale of the computation increases linearly with each additional bit. In contrast, in the case of a quantum computer, the size of the Hilbert space, which corresponds to the scale of the computation, increases exponentially as the number of qubits increases. By taking advantage of the characteristics, rich range of applications of quantum computers has been proposed, e.g., algorithms for prime factoring (Shor's algorithm [2]) quantum chemistry [3–6], finance [7–9], and machine learning [10–15].

However, quantum computers developed in the next few decades are predicted to be far from the ideal quantum computer. The biggest problems are the small number of qubits and the considerable noise. First, the number of qubits available for the computation may be increased only up to $O(10) \sim O(100)$ in decades, which spoils superiority over the classical computer; for example, the prime factoring problem which Shor's algorithm with $O(100)$ qubits can solve, is also solved by classical computers in realistic time. Another issue is the significant noise. Currently, the error rate of a quantum gate that operates on two qubits is as substantial as $O(10^{-3})$. In other words, if the number of the quantum gates to be operated is $O(1000)$, the calculation results will be wholly unreliable, which makes the execution of most quantum algorithms impossible. Even though the above problems will be gradually improved, they are still predicted to remain significant limitations. The near-term noisy quantum computer with $O(10) \sim O(100)$ qubits available for the computation is called a noisy intermediate-scale quantum computer (NISQ) [16].

Researchers in the quantum computing community have been actively studying how to utilize NISQ rather than waiting for the development of the ideal quantum computers for decades. The most promising direction for maximizing the functionality of NISQ is reducing the number of quantum gates to solve problems.

An important example of significantly reducing the number of gates is the *quantum amplitude estimation* algorithm (QAE). QAE is the algorithm to estimate the amplitude of a quantum state with good accuracy; namely, the goal of the algorithm is estimating the value of ξ in the following equation: $|\Psi\rangle \equiv \mathcal{A}|0\rangle^{\otimes n}|0\rangle = \xi|\tilde{\Psi}_1\rangle|1\rangle + \sqrt{1-\xi^2}|\tilde{\Psi}_0\rangle|0\rangle$, where $|\Phi_0\rangle$ and $|\Phi_1\rangle$ are n-qubit states, $\xi \in [0, 1]$, and $\mathcal{A}$ is an quantum operator. Various quantum algorithms include QAE as its submodule. For example, the quantum Monte Carlo integration [17] embeds the integration result into ξ, and we need to extract the value of ξ from the quantum state. In practical applications, executing $\mathcal{A}$ is often costly, thus reducing the number of calling $\mathcal{A}$ while estimating ξ with required accuracy is the heart of the problem. There is a well-known approach to estimate ξ by using the phase estimation algorithm [18, 19]. By the quantum algorithm using the phase estimation, the estimation error scales as $O(1/N_{\mathrm{shot}})$, where N_{shot} is the number of calling $\mathcal{A}$. However, the phase estimation requires many two-qubit gates, which are noisy and intractable in the near-term devices. The attempt to reduce the number of gates is firstly made by the literature [20]. Literature [20] shows that the number of gates is reduced in their proposed algorithm using the maximum likelihood estimation. However, the theoretical proof of the error scaling $O(1/N_{\mathrm{shot}})$ is not given while they give some numerical evidence for it. As subsequent works, [21] give algorithms where the error scaling is theoretically proven. Among these works, the theoretical upper bounds of the error in [21] is larger than $10^6/N_{\mathrm{shot}}$, while in the faster amplitude estimation (FAE) [22], which is the contribution of the author of this thesis, the constant factor is improved as $O(10^3/N_{\mathrm{shot}})$. The detail of the FAE algorithm is described in this thesis.

Another important example of quantum algorithms realizable with fewer gates is the *variational quantum algorithm* (VQA) [23, 24]. In VQA, we usually use the quantum circuit with a specific structure where parameters are embedded; the structure often used is the so-called hardware efficient ansatz (HEA) [25], which has enough expressive power to generate an arbitrary quantum state. Given the parameters $\boldsymbol{\theta}$ and the unitary operation corresponding to the quantum circuit as $U(\boldsymbol{\theta})$, the goal of VQA is finding the optimal parameters $\boldsymbol{\theta}_*$ so that the generated state by the quantum circuit $|\phi(\boldsymbol{\theta}_*)\rangle = U(\boldsymbol{\theta}_*)|\phi_0\rangle$ with $|\phi_0\rangle$ as an initial state independent of the parameters, minimizes the expectation value of an observable $\mathcal{H}$: $\langle\phi(\boldsymbol{\theta}_*)|\mathcal{H}|\phi(\boldsymbol{\theta}_*)\rangle$. For example, if the problem we want to solve is finding the lowest eigenstate of a Hamiltonian, then $\mathcal{H}$ is the Hamiltonian and $|\phi(\boldsymbol{\theta}_*)\rangle$ is the lowest eigenstate. We iteratively update the parameters; for each iteration we compute the expectation value $C(\boldsymbol{\theta}) = \langle\phi_0|U^\dagger(\boldsymbol{\theta})\mathcal{H}U(\boldsymbol{\theta})|\phi_0\rangle$ by a quantum computer and compute the new parameters, which is embedded into the quantum circuit in the next iteration, by a classical computer. The algorithm is expected to be realized with a small number of quantum gates since a classical computer assists the calculation, and therefore, is suitable for NISQ.

Various variants of VQA have also been proposed recently. One promising variant is the quantum generative adversarial network (QGAN) [26–40], which is the counterpart of the generative adversarial network (GAN) [41]. GAN is a classical machine learning algorithm with many applications [42, 43]. In general, GAN consists of two adversarial components, typically a generator (generating fake data) and a discriminator (discriminating real or fake data), and their adversarial training yield an outperforming system over the one trained solely. In original GAN, the generator and the discriminator are typically implemented by the deep neural network, but in QGAN, the generator and/or the discriminator are implemented by the quantum system. GAN is most successfully applied to semi-supervised learning [44–51] (SSL). The first work combining QGAN and SSL is done by the author of the thesis in [52], which proposes the algorithm named the quantum semi-supervised adversarial network (qSGAN). We will describe the detail of qSGAN in this thesis.

Even though VQA and its variants are arguably the most important quantum algorithms realizable in NISQ, it is shown that VQA with HEA has a severe issue called *barren plateau* phenomenon [53]. Normally, parameters $\boldsymbol{\theta}$ are updated to the direction of the gradient vector $\{\partial C(\boldsymbol{\theta})/\partial\theta_j\}$. However, the literature [53] shows that the norm of the gradient becomes exponentially small: $O(1/2^n)$ with n as the number of qubits. As a result, we need $O(2^n)$ measurements of $|\phi(\boldsymbol{\theta})\rangle$ for updating the parameters to the proper direction, which is intractable when $n \geq O(100)$. As a solution to the barren plateau phenomenon, the literature [54] proposes the method utilizing the quantum circuit called alternating layered ansatz (ALT) instead of HEA. ALT is a quantum circuit $U(\boldsymbol{\theta})$ that has a specific structure different from HEA. The structure is the key to avoiding the barren plateau issue; by using the detail of the structure, it is theoretically shown that the barren plateau issue is avoided when using ALT under a specific condition. However, there is one fatal caveat in the method. Specifically, due to the restricted structure of ALT, it is unclear whether or not ALT has sufficient expressive power (expressibility) for generating a rich class of states, which contains the optimal state minimizing $C(\boldsymbol{\theta})$. If it does not have enough expressive power, we cannot adopt the method proposed in [54] as the solution for the barren plateau issue or the scalability issue. Conversely, analyzing the expressibility of ALT is critical to assessing the scalability of VQA. In [55], the author of this thesis analyzes the expressibility of ALT and other ansatzes by utilizing the expressibility measure [56]. The method and the results of the expressibility analysis are described in this thesis.

The goal of this thesis is to develop algorithms maximizing the functionality of NISQ and assess their feasibility. More precisely, we summarize the contribution of this thesis as follows.

- The author of the thesis proposes the quantum amplitude estimation algorithm named the faster amplitude estimation (FAE), whose state-of-the-art performance is theoretically guaranteed (Chapter 4).
- The author of the thesis proposes a variant of the variational quantum algorithm called the quantum semi-supervised generative adversarial network (qSGAN) for solving the semi-supervised learning task, which is one of the most important tasks in machine learning (Chapter 5).
- The author of the thesis analyzes the expressibility of ALTs through theoretical and numerical analysis. The results show that a particular class of ALTs has sufficient expressive power to generate arbitrary quantum states. Furthermore, it is shown that there exists a class of ALTs that can both provide sufficient expressive power and solve the barren plateau phenomenon (Chapter 6).

Additionally, we review the basics of quantum computation and NISQ comprehensively. We hope this thesis is also helpful for those unfamiliar with quantum computation and NISQ.

The rest of this thesis is structured as follows. In Chapter 2, we review the basics of quantum computation. We review the feature of NISQ in Chapter 3. VQA and the issues of VQA, including the barren plateau phenomenon, are also discussed in the same chapter. We show the QAE algorithm tailored for NISQ in Chapter 4. The variant of the variational quantum algorithm for the semi-supervised learning tasks is discussed in Chapter 5. In Chapter 6, the possible solutions for the barren plateau issue, including the one using ALT, are introduced and, we analyze the expressibility of ALT in the same section. Finally, we conclude this thesis and show outlooks in Chapter 7.

Chapter 2

Quantum computation

In this chapter, we introduce the basics of quantum computation. We assume the knowledge of quantum mechanics in this thesis. We review the building block of quantum computation, such as the quantum bit and the quantum gate. Next, we briefly review how to include the effect of noise in the quantum computation framework, which is inevitable for understanding the next chapter. Finally, we show some essential quantum algorithms. For a more detailed review of the quantum computation, please see the literature [19].

2.1 Building-blocks of the quantum computation

2.1.1 Quantum bit (qubit)

In the classical computation, the information is stored in bits that take the value of either 0 or 1. To the contrary, in the quantum computation, the information is stored in the quantum bits named *qubits*. Each qubit retains information in a quantum state, whose basis are $|0\rangle$ and $|1\rangle$. Namely, the quantum state that corresponds to a state of a qubit can be written as

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{2.1}$$

where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$.

The computation process is done by using multiple qubits. Given the number of qubits as n, when each qubit is independently operated, the n-qubit state is written in the form of

$$|\phi_1\rangle \otimes |\phi_2\rangle \otimes \cdots \otimes |\phi_n\rangle = (\alpha_1|0\rangle + \beta_1|1\rangle) \otimes (\alpha_2|0\rangle + \beta_2|1\rangle) \otimes \cdots \otimes (\alpha_n|0\rangle + \beta_n|1\rangle), \tag{2.2}$$

where $\alpha_j, \beta_j \in \mathbb{C}(\forall j)$, $|\alpha_j|^2 + |\beta_j^2| = 1(\forall j)$, and $\otimes$ denotes the symbol for the tensor product. In general, the computational process includes the interactions between qubits, and the n-qubit quantum state cannot be written in the form of n tensor products; the general n-qubit state is written as

$$|\phi\rangle = \sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \cdots \sum_{i_n=0}^{1} \alpha_{i_1 i_2,\cdots i_n} |i_1\rangle \otimes |i_2\rangle \otimes |i_n\rangle, \tag{2.3}$$

where $\alpha_{i_1 i_2,\cdots i_n} \in \mathbb{C}$ and $\sum_{i_1=0}^{1} \sum_{i_2=0}^{1} \cdots \sum_{i_n=0}^{1} |\alpha_{i_1 i_2,\cdots i_n}|^2 = 1$. In the following we simplify the notation '$|i_1\rangle \otimes |i_2\rangle \otimes |i_n\rangle$' to $|i_1 i_2 \cdots i_n\rangle$. The basis $\{|i_1 i_2, \cdots i_n\rangle\}$ is said to be the *computational basis*. Even though we can arbitrarily choose the basis for expanding $|\phi\rangle$, the computational basis is special because, as we see later, measurements in quantum computation are performed in the computational basis.

In the classical case, n bits can store at most n binary data. Remarkably, n-qubits can store 2^n complex numbers in the amplitudes of the quantum state. Thus, ideally, even one hundred qubits quantum state store $O(2^{100}) = O(10^{30})$ bit data, which is intractable by any classical devices. There are various ways of implementing qubits. For example, ion trap qubits [57], and superconducting qubits [58] have been actively studied and developed.

2.1.2 Quantum gate

Normally, the quantum state of qubits is prepared as $|0\rangle^{\otimes n}$ (n tensor products of $|0\rangle$). From the state, all the transformations other than the measurements are ideally done by unitary operators; given a quantum state that we want to build as $|\phi\rangle$, the state is built by using a unitary operator U as

$$|\phi\rangle = U|0\rangle^{\otimes n}. \tag{2.4}$$

By measuring $|\phi\rangle$, we obtain computational results.

The unitary operation is built by single-qubit gates and two-qubit gates. A single-qubit gate is a unitary operation that operates on a single-qubit. Suppose that we write an arbitrary single-qubit quantum state $\alpha|0\rangle + \beta|1\rangle$ in a vector form $\begin{pmatrix}\alpha\\ \beta\end{pmatrix}$, where the first row corresponds to the basis $|0\rangle$ and the second row corresponds to the basis $|1\rangle$. One of the example of single-qubit gates is the Hadamard gate

$$H = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}. \tag{2.5}$$

The Hadamard gate H transforms an arbitrary single-qubit state as

$$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} \to \begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \frac{1}{\sqrt{2}}\begin{pmatrix}\alpha+\beta\\ \alpha-\beta\end{pmatrix}. \tag{2.6}$$

In the quantum state form,

$$H(\alpha|0\rangle + \beta|1\rangle) = \begin{pmatrix}|0\rangle & |1\rangle\end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \frac{1}{\sqrt{2}}(\alpha+\beta)|0\rangle + \frac{1}{\sqrt{2}}(\alpha-\beta)|1\rangle. \tag{2.7}$$

As another example, the X gate is defined by

$$X = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}, \tag{2.8}$$

which transforms an arbitrary single-qubit quantum state as

$$\begin{pmatrix}\alpha\\ \beta\end{pmatrix} \to \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \begin{pmatrix}\beta\\ \alpha\end{pmatrix}. \tag{2.9}$$

In the quantum state form,

$$X(\alpha|0\rangle + \beta|1\rangle) = \begin{pmatrix}|0\rangle & |1\rangle\end{pmatrix}\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}\begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \beta|0\rangle + \alpha|1\rangle. \tag{2.10}$$

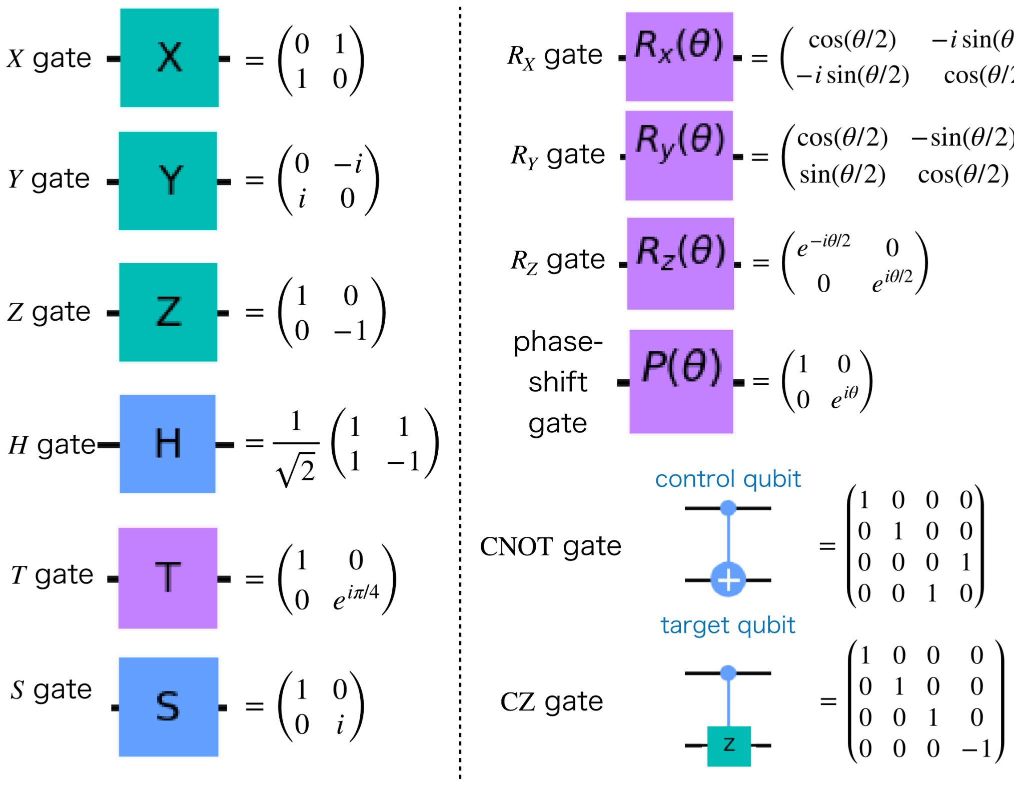
Since the X gate invert the bit value, it is called the NOT gate.

The two-qubit gate is a unitary operation that operates on a control-qubit and a target-qubit. If the state of the control qubit is $|0\rangle$, no operations are performed to the target-qubit, while if that is $|1\rangle$, an operation is operated to the target qubit. Suppose that we write an arbitrary two-qubit state as $\begin{pmatrix}\alpha_{00}\\ \alpha_{01}\\ \alpha_{10}\\ \alpha_{11}\end{pmatrix}$, where α_x denotes the coefficient for the basis $|x\rangle$. One of the example of two-qubit gates is controlled-not (CNOT) gate

$$\mathrm{CNOT} = \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}, \tag{2.11}$$

which transforms an arbitrary state as

$$\begin{pmatrix}\alpha_{00}\\ \alpha_{01}\\ \alpha_{10}\\ \alpha_{11}\end{pmatrix} \to \begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 1\\ 0 & 0 & 1 & 0\end{pmatrix}\begin{pmatrix}\alpha_{00}\\ \alpha_{01}\\ \alpha_{10}\\ \alpha_{11}\end{pmatrix} = \begin{pmatrix}\alpha_{00}\\ \alpha_{01}\\ \alpha_{11}\\ \alpha_{10}\end{pmatrix}. \tag{2.12}$$



X gate
X
$= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$
Y gate
Y
$= \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$
Z gate
Z
$= \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$
H gate
H
$= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$
T gate
T
$= \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}$
S gate
S
$= \begin{pmatrix} 1 & 0 \\ 0 & i \end{pmatrix}$
R_X gate
$R_x(\theta)$
$= \begin{pmatrix} \cos(\theta/2) & -i\sin(\theta/2) \\ -i\sin(\theta/2) & \cos(\theta/2) \end{pmatrix}$
R_Y gate
$R_y(\theta)$
$= \begin{pmatrix} \cos(\theta/2) & -\sin(\theta/2) \\ \sin(\theta/2) & \cos(\theta/2) \end{pmatrix}$
R_Z gate
$R_z(\theta)$
$= \begin{pmatrix} e^{-i\theta/2} & 0 \\ 0 & e^{i\theta/2} \end{pmatrix}$
phase-shift gate
$P(\theta)$
$= \begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$
control qubit
CNOT gate
target qubit
$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$
CZ gate
Z
$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix}$

Figure 2.1: Examples of quantum gates.

In the quantum state form,

$$\mathrm{CNOT}(\alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle) = \begin{pmatrix} |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha_{00} \\ \alpha_{01} \\ \alpha_{10} \\ \alpha_{11} \end{pmatrix} \quad (2.13)$$

$$= \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|11\rangle + \alpha_{11}|10\rangle. \quad (2.14)$$

We see that $|10\rangle$ becomes $|11\rangle$ and $|11\rangle$ becomes $|10\rangle$, while $|00\rangle$ and $|01\rangle$ are unchanged. Similarly, with G as an arbitrary single-qubit operation, controlled-G gate transforms the target-qubit by G when the control qubit is $|1\rangle$ while no operations are operated to the target-qubit if the control-qubit is $|0\rangle$. In Fig. 2.1, we show representative examples of the quantum gates with their matrix forms.

Quantum circuit

The sequence of the above-defined quantum gates is called a 'quantum circuit.' Quantum circuits are often visualized in the following manner. Suppose that the number of qubits is n, we write vertically aligned n-horizontal lines; each line corresponds to each qubit. We write quantum gates from left to right in the order of the operation. Operations that can be simultaneously executable can be lined up vertically. Each gate is written as in the images in Fig. 2.1. Each single-qubit gate is appended in the line (qubit) where it performs. Each controlled-G gate is written over the two lines where it performs; a dot is depicted in the control bit, the G is shown in the target bit, and the vertical line is displayed between the control qubit and the target qubit. As an example, we visualize the following operations in three qubits in Fig. 2.2:

1. Operate H gate to the first qubit.
2. Operate Y gate to the second qubit.

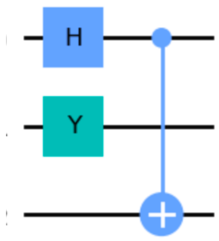


Figure 2.2: An example of the quantum circuit.

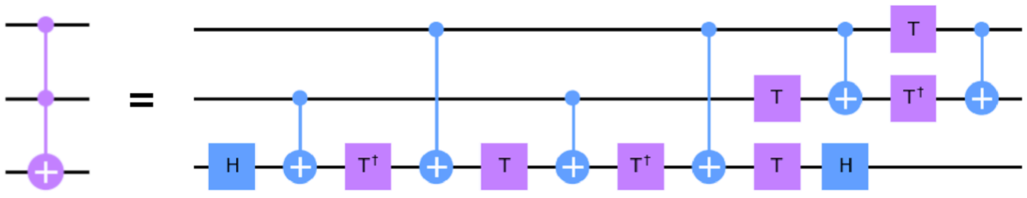


Figure 2.3: A decomposition of the Toffoli gate.

3. Operate CNOT gate to the first and the third qubits with the first qubit as the controlled qubit.

Universal quantum gates

The other unitary operations are built from the single-qubit and the two-qubit gates. For example, the controlled-NOT operation with two control qubits, which performs NOT to the target qubit only if the state of control qubits are $|11\rangle$, can be built as shown in Fig. 2.3; the three-qubit operation is called the Toffoli gate. Similarly, controlled operations with multiple control qubits can be built from single-qubit and two-qubit gates.

The swap operation is also an important operation implementable with multiple gates. The swap operation $\mathrm{Swap}(i,j)$ swaps the i-th qubit and the j-th qubit in the sense that each basis state $|k_1k_2\cdots k_i\cdots k_j\cdots k_n\rangle$ is transformed as

$$\mathrm{Swap}(i,j)|k_1k_2\cdots k_i\cdots k_j\cdots k_n\rangle = |k_1k_2\cdots k_j\cdots k_i\cdots k_n\rangle. \tag{2.15}$$

The swap gate is implementable by using three CNOT gates, which is shown in Fig. 2.4.

Beyond those operations, it is well-known that certain combinations of the single-qubit and the two-qubit gates can approximate any unitary operations to arbitrary accuracy. If combinations of a set of gates can approximate any unitary operations to arbitrary precision, the gate set is said to be an *universal quantum gates*. One example of the universal gate set is $\{\mathrm{CNOT}, \mathrm{H}, \mathrm{S}, \mathrm{T}\}$ [19].

Note that even though the limited gate set can be universal, it generally requires an exponential number of gates to implement an arbitrary unitary operation. Thus, for reducing the number of gates, we do not need to restrict ourselves to use only the universal gate set; instead, we use various gates such as the ones listed in Fig. 2.1 to implement desired unitary operations.

2.1.3 Measurement

We need measurements to extract the final result of the computation since we cannot efficiently extract the quantum state itself. Recall that in quantum mechanics, the measurements are described by using the positive operator-valued measure (POVM). POVM is a set of positive

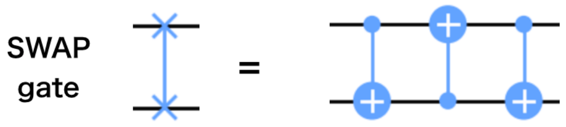


Figure 2.4: The implementation of the swap gate.

semi-definite operator $\{E_j\}_{j=1}^J$ that satisfies

$$\sum_{j=1}^{J} E_j = I. \tag{2.16}$$

Each index j corresponds to each measurement outcome. Given ρ as the density matrix corresponds to the final state, the probability that j-th outcome is obtained is given by $\mathrm{Tr}(\rho E_j)$. Since E_j is positive semi-definite, there exists an operator $M_j = \sqrt{E_j}$, namely, $M_j = \sum_k \sqrt{\lambda_k^j}|e_k\rangle\langle e_k|$ with $\{\lambda_j^k\}$ and $\{|e_k\rangle\}$ as eigenvalues and eigenvectors of E_j. By using M_j, we can write the state after observing j-th outcome as

$$\rho_j = \frac{M_j \rho M_j^\dagger}{\mathrm{Tr}(\rho E_j)}. \tag{2.17}$$

For example, in a two-qubit system, let us define the following POVM:

$$\begin{aligned} E_1 &= |00\rangle\langle 00| \\ E_2 &= |01\rangle\langle 01| \\ E_3 &= |10\rangle\langle 10| + |11\rangle\langle 11|, \end{aligned} \tag{2.18}$$

then

$$M_1 = E_1, M_2 = E_2, M_3 = E_3. \tag{2.19}$$

If the quantum state is $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, the corresponding density matrix is given by

$$\rho = \frac{1}{2}(|00\rangle + |11\rangle)(\langle 00| + \langle 11|), \tag{2.20}$$

and therefore, the probabilities that the measurement outcomes corresponding to E_1, E_2, E_3 are

$$\mathrm{Tr}(E_1\rho) = \frac{1}{2},\ \mathrm{Tr}(E_2\rho) = 0,\ \mathrm{Tr}(E_3\rho) = \frac{1}{2} \tag{2.21}$$

respectively. Also, the density matrix after E_1 is measured, is given by

$$\rho_1 = \frac{\frac{1}{2}|00\rangle\langle 00|}{\frac{1}{2}} = |00\rangle\langle 00|, \tag{2.22}$$

and similarly, the density matrix after E_3 is measured is

$$\rho_3 = \frac{\frac{1}{2}|11\rangle\langle 11|}{\frac{1}{2}} = |11\rangle\langle 11|. \tag{2.23}$$

In the quantum computation, measurements are performed in *the computational basis*, where the POVM is given by $\{|i_1 i_2 \cdots i_n\rangle\langle i_1 i_2 \cdots i_n|\}$ $(i_1, i_2 \cdots = 0 \text{ or } 1)$. Still, it should be noted that the measurement regarding the other POVMs is possible by rotating the final state ρ. The quantum circuit corresponding to the computational basis measurement of a qubit is drawn as in Fig. 2.5. In Fig. 2.5, the Hadamard gate is firstly operated to one qubit, and then the state is measured on the computational basis.

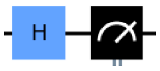


Figure 2.5: Computational measurement

2.2 Noisy quantum operations

Up to this point, we have proceeded with our discussion by implicitly assuming that no noise is introduced into the calculation results. However, the calculation is subject to noise in reality, which distorts the computational results. In the presence of noise, quantum state transitions would be performed by quantum operations rather than unitary transformations. In general, the quantum operation $\mathcal{E}$ is the function that maps density matrix ρ as:

$$\rho' = \mathcal{E}(\rho). \tag{2.24}$$

In noiseless case, the map $\mathcal{E}$ is

$$\mathcal{E}(\rho) = U\rho U^\dagger, \tag{2.25}$$

with an unitary operator U.

Normally, we assume that noise affects quantum states after each quantum gate is operated to the state. Namely, given the unitary quantum operation and the noise quantum operation corresponding to k-th gate as $\mathcal{U}_k$ and $\mathcal{E}_k$, the quantum operation for performing K gates to the initial state ρ_{in} can be written as

$$\mathcal{E}(\rho_{\text{in}}) = \mathcal{E}_K \circ \mathcal{U}_K \circ \cdots \mathcal{E}_2 \circ \mathcal{U}_2 \circ \mathcal{E}_1 \circ \mathcal{U}_1(\rho), \tag{2.26}$$

where $A \circ B$ denotes the composition function of A and B. Note that given an unitary operator corresponding to k-th gate as U_k, the unitary quantum operation $\mathcal{U}_k$ can be written as

$$\mathcal{U}_k(\rho) = U_k \rho U_k^\dagger. \tag{2.27}$$

Operator-sum representation

There is various ways to represent the function $\mathcal{E}$, one of which is the operator-sum representation. In the operator sum representation, the map is expanded as

$$\mathcal{E}(\rho) = \sum_{j=0} K_j \rho K_j^\dagger. \tag{2.28}$$

As long as the map does not change the trace of the density matrix, we obtain

$$\mathrm{Tr}\left(\sum_{j=0} K_j \rho K_j^\dagger\right) = 1, \tag{2.29}$$

for all density matrix ρ. Therefore,

$$\sum_{j=0} K_j K_j^\dagger = I. \tag{2.30}$$

It is well known that this representation can describe a wide variety of noise properties.

Examples of noisy quantum operations

Phase flip

The phase flip is the quantum operation that can be written as

$$K_0 = \sqrt{1-p}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \ K_1 = \sqrt{p}\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{2.31}$$

in the operator-sum representation with p as a real number satisfying $0 \leq p < 1$. For example, given an arbitrary single-qubit pure state as $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, the noise quantum operation to the quantum state is

$$\mathcal{E}(|\psi\rangle\langle\psi|) = (1-p)|\psi\rangle\langle\psi| + p|\psi'\rangle\langle\psi'|, \tag{2.32}$$

where

$$|\psi'\rangle = \alpha|0\rangle - \beta|1\rangle. \tag{2.33}$$

Therefore, the phase flip quantum operation flips the sign of the amplitude regarding the quantum state $|1\rangle$ with the probability p.

Amplitude damping

The amplitude dumping is the quantum operation with the operator-sum representation:

$$K_0 = \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{pmatrix},\ K_1 = \begin{pmatrix} 0 & \sqrt{p} \\ 0 & 0 \end{pmatrix}. \tag{2.34}$$

The effect of the noise quantum operation to an arbitrary single-qubit state is

$$\begin{aligned}\mathcal{E}(|\psi\rangle\langle\psi|) &= \begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{pmatrix}\begin{pmatrix} |\alpha|^2 & \alpha\beta^* \\ \alpha^*\beta & |\beta|^2 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{pmatrix} + \begin{pmatrix} 0 & \sqrt{p} \\ 0 & 0 \end{pmatrix}\begin{pmatrix} |\alpha|^2 & \alpha\beta^* \\ \alpha^*\beta & |\beta|^2 \end{pmatrix}\begin{pmatrix} 0 & 0 \\ \sqrt{p} & 0 \end{pmatrix} \\ &= \begin{pmatrix} |\alpha|^2 + p|\beta|^2 & \sqrt{1-p}\ \alpha\beta^* \\ \sqrt{1-p}\ \alpha^*\beta & (1-p)|\beta|^2 \end{pmatrix}.\end{aligned} \tag{2.35}$$

Depolarization

The depolarization is the quantum operation with the operator-sum representation:

$$K_0 = \sqrt{1 - \frac{3p}{4}} I, K_1 = \sqrt{p}X, K_2 = \sqrt{p}Y, K_3 = \sqrt{p}Z. \tag{2.36}$$

The resulting quantum operation is written as

$$\mathcal{E}(\rho) = (1-p)\rho + \frac{p}{2}I. \tag{2.37}$$

.

Fault torrent quantum computation

Even if each p in the above examples is small, the effect of noise increases as the number of gates increase, which makes the computational results unreliable. For the reliable computation, device developers are trying to reduce the effect of noise in each gate.

However, to obtain reliable computational results, we do not need to reduce the noise to exact zero because there is a technique called the *fault torrent computation.* In the fault torrent computation, information that can be expressed by one qubit is redundantly embedded into multiple qubits. Even when some errors occur during the computation, the error-correcting function corrects the quantum state, and as a result, reliable quantum computation is possible. For more detail, please see a recent review [59].

2.3 Quantum algorithms

Now let us review a quantum algorithm for demonstrating the power of quantum computation. The quantum algorithm we review here is the *quantum phase estimation* [18, 19], which is the essential quantum algorithm that has various applications such as prime factoring. The quantum phase estimation requires the quantum Fourier transform (QFT) [19] as its submodule. Thus, we first review QFT and next review the quantum phase estimation in the following.

2.3.1 Quantum Fourier transform

The Fourier transform is a useful mathematical tool for classical computing. The quantum counterpart of the transform is said to be the *quantum Fourier transform* (QFT). Specifically, QFT is the unitary opertion U_{QFT} that transforms an arbitrary computational basis state $|j\rangle$ as

$$U_{\mathrm{QFT}}|j\rangle = \frac{1}{\sqrt{N}}\sum_{k=0}^{N-1} e^{2\pi i kj/N}|k\rangle, \tag{2.38}$$

where $N = 2^n$ with n as the number of qubits. The indices j and k are the decimal representation of binary bits; for example $|4\rangle \equiv |100\rangle$ in three-qubit state. If we build the transformation U_{QFT}, we can transform an arbitrary n-qubit state $|\psi\rangle = \sum_j \alpha_j|j\rangle$ as

$$\begin{aligned} U_{\mathrm{QFT}}|\psi\rangle &= \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1} \alpha_j U_{\mathrm{QFT}}|j\rangle \\ &= \sum_{k=0}^{N-1} y_k|k\rangle, \end{aligned} \tag{2.39}$$

where

$$y_k = \sum_{j=0}^{N-1} e^{2\pi i kj}\alpha_j. \tag{2.40}$$

We can easily check that the inverse operation of U_{QFT} works as follows:

$$U_{\mathrm{QFT}}^{\dagger}|k\rangle = \frac{1}{\sqrt{N}}\sum_{\ell=0}^{N-1} e^{-2\pi i k\ell/N}|\ell\rangle. \tag{2.41}$$

Implementation

As a preparation, we write the binary representations of $|j\rangle$ and $|k\rangle$ as $|j_1 j_2\cdots j_n\rangle$ and $|k_1 k_2\cdots k_n\rangle$ respectively. The important building blocks for the Fourier transform is the following phase shift gate

$$R_k \equiv P(2\pi i/2^k) = \begin{pmatrix} 1 & 0 \\ 0 & e^{2\pi i/2^k} \end{pmatrix}. \tag{2.42}$$

Let us also write the controlled-R_k operation as $CR_k(x, y)$, where x is the index for the controlled qubit, and y is the index for the target qubit.

The process of QFT is separated into n-stages. Let us write $|\psi(s)\rangle$ as the quantum state after s stage, where $|\psi(0)\rangle = |j\rangle$. In s-th stage, the quantum state is transformed as

$$|\psi(s)\rangle = U_s^f|\psi(s-1)\rangle, \tag{2.43}$$

where

$$U_s^f = \left(\prod_{k=2}^{n-s+1} CR_k(s+k-1, s)\right)(I_{s-1}\otimes H\otimes I_{n-s-1}) \tag{2.44}$$

For example, in the first stage,

$$|\psi(1)\rangle = CR_n(n,1)\cdots CR_3(3,1)CR_2(2,1)(H\otimes I_{n-1})|\psi(0)\rangle. \tag{2.45}$$

With this definition of the operation in each stage, the following theorem holds:

Theorem 1. *Suppose that the transformation at n-th stage is given by* (2.43)*, and* $|\psi(0)\rangle = |j\rangle$*. Then for* $1 \le s \le n$*,*

$$|\psi(s)\rangle = \frac{1}{\sqrt{2^n}}\left(\prod_{a=1}^{s}\left(|0\rangle + e^{2\pi i 0.j_a j_{a+1}\cdots j_n}|1\rangle\right)\right)|j_{s+1}\cdots j_n\rangle, \tag{2.46}$$

where the product is defined by

$$\prod_{a=1}^{s}\left(|0\rangle + e^{2\pi i 0.j_a j_{a+1}\cdots j_n}|1\rangle\right) = \left(|0\rangle + e^{2\pi i 0.j_1 j_2\cdots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_2\cdots j_n}|1\rangle\right)\cdots\otimes\left(|0\rangle + e^{2\pi i 0.j_s j_{s+1}\cdots j_n}|1\rangle\right). \tag{2.47}$$

In the theorem, we follow the convention in [19] that writes a number $\sum_{a=1}^{n} 2^{-a} j_a$ as its binary representation $0.j_1 j_2 \cdots j_n$. For example 0.11 in binary representation denotes $1/2 + 1/4 = 3/4$ in decimal and 0.101 denotes $1/2 + 1/8 = 5/8$ in decimal.

Proof. We inductively prove the theorem. First, we prove the theorem in case of $s = 1$. The Hadamard gate operates to the initial state $|\psi(0)\rangle$ as

$$H|\psi(0)\rangle = (H|j_1\rangle)|j_2\cdots j_n\rangle = \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.j_1}|1\rangle\right)|j_2\cdots j_n\rangle. \tag{2.48}$$

Then, the operations of $CR_k(k,1)$ recursively add the phase to $|1\rangle$ state in the first qubit as

$$\begin{aligned}
CR_2(2,1)(H|j_1\rangle)|j_2\cdots j_n\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.j_1}e^{2\pi i j_2/2^2}|1\rangle\right)|j_2\cdots j_n\rangle\\
&= \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2}|1\rangle\right)|j_2\cdots j_n\rangle,\\
CR_3(3,1)CR_2(2,1)(H|j_1\rangle)|j_2\cdots j_n\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2}e^{2\pi i j_3/2^3}|1\rangle\right)|j_2\cdots j_n\rangle\\
&= \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2 j_3}|1\rangle\right)|j_2\cdots j_n\rangle,\\
&\vdots\\
|\psi(1)\rangle = CR_n(n,1)\cdots CR_2(2,1)(H|j_1\rangle)|j_2\cdots j_n\rangle &= \frac{1}{\sqrt{2}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2\cdots j_n}|1\rangle\right)|j_2\cdots j_n\rangle,
\end{aligned} \tag{2.49}$$

which concludes the proof of the case when $s = 1$.

From the induction hypothesis, (2.46) holds for $s = t$ ($t < n$). Then, the operation of the Hadamard gate to the $t+1$-th qubit of $|\psi(s)\rangle$ can be written as

$$\begin{aligned}
I_s \otimes H \otimes I_{n-1-s}|\psi(s)\rangle &= |\phi(t)\rangle(H|j_{t+1}\rangle)|j_{t+2}\cdots j_n\rangle,\\
&= |\phi(t)\rangle(|0\rangle + e^{2\pi i 0.j_{t+1}}|1\rangle)|j_{t+2}\cdots j_n\rangle,
\end{aligned} \tag{2.50}$$

where

$$|\phi(t)\rangle \equiv \frac{1}{\sqrt{2^t}}\prod_{a=1}^{t}\left(|0\rangle + e^{2\pi i 0.j_a j_{a+1}\cdots j_n}|1\rangle\right). \tag{2.51}$$

Similar as (2.49), $CR_k(k, t+1)$ recursively append the phase to $|1\rangle$ state in the $t+1$-th qubit as

$$\begin{aligned}
CR_2(2,t+1)(H|j_{t+1}\rangle)|j_{t+2}\cdots j_n\rangle &= \frac{1}{\sqrt{2}}|\phi(t)\rangle\left(|0\rangle + e^{2\pi i 0.j_{t+1}}e^{2\pi i j_{t+2}/2^2}|1\rangle\right)|j_{t+2}\cdots j_n\rangle\\
&= \frac{1}{\sqrt{2}}|\phi(t)\rangle\left(|0\rangle + e^{2\pi i 0.j_{t+1} j_{t+2}}|1\rangle\right)|j_{t+2}\cdots j_n\rangle,\\
CR_3(3,t+1)CR_2(2,t+1)(H|j_1\rangle)|j_{t+2}\cdots j_n\rangle &= \frac{1}{\sqrt{2}}|\phi(t)\rangle\left(|0\rangle + e^{2\pi i 0.j_{t+1} j_{t+2}}e^{2\pi i j_{t+3}/2^3}|1\rangle\right)|j_{t+2}\cdots j_n\rangle\\
&= \frac{1}{\sqrt{2}}|\phi(t)\rangle\left(|0\rangle + e^{2\pi i 0.j_{t+1} j_{t+2} j_{t+3}}|1\rangle\right)|j_{t+2}\cdots j_n\rangle,\\
&\vdots\\
CR_n(n,t+1)\cdots CR_2(2,t+1)(H|j_1\rangle)|j_2\cdots j_n\rangle &= \frac{1}{\sqrt{2}}|\phi(t)\rangle\left(|0\rangle + e^{2\pi i 0.j_{t+1} j_{t+2}\cdots j_n}|1\rangle\right)|j_{t+2}\cdots j_n\rangle.
\end{aligned} \tag{2.52}$$

Thus, Eq. (2.46) holds for $s = t + 1$, which concludes the proof of the theorem. □

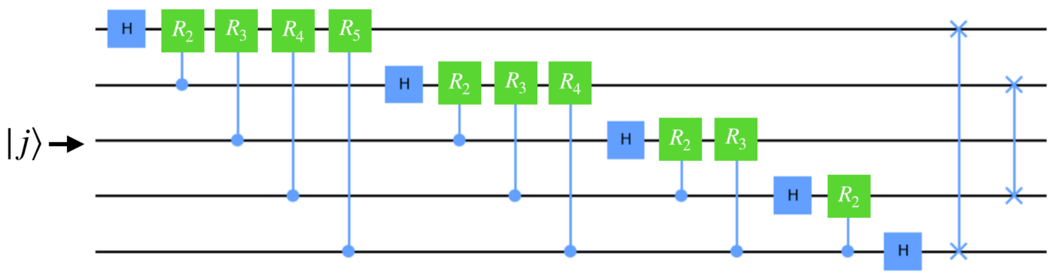


Figure 2.6: The quantum circuit for U_{QFT} when the number of qubits is five.

As a result of the n-stage operations, we obtain

$$|\psi(n)\rangle = \frac{1}{\sqrt{2^n}}\left(|0\rangle + e^{2\pi i 0.j_1 j_2\cdots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_2\cdots j_n}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right). \tag{2.53}$$

Given an operation that swaps j-th qubit with $n-j$-th qubit for all $j(< n/2)$ as $SWAP$,

$$|\psi'(n)\rangle = SWAP|\psi(n)\rangle = \frac{1}{\sqrt{2^n}}\left(|0\rangle + e^{2\pi i 0.j_n}|1\rangle\right)\cdots\left(|0\rangle + e^{2\pi i 0.j_2\cdots j_n}|1\rangle\right)\left(|0\rangle + e^{2\pi i 0.j_1 j_2\cdots j_n}|1\rangle\right). \tag{2.54}$$

The coefficient of the basis $|k\rangle = |k_1 k_2\cdots k_n\rangle$ in $|\psi'(n)\rangle$ is given by

$$\begin{aligned}\langle k_1 k_2\cdots k_n|\psi'(n)\rangle &= \frac{1}{\sqrt{2^n}}e^{i\gamma},\\ \gamma &= 2\pi i\left(k_1 0.j_n + k_2 0.j_{n-1}j_n + k_3 0.j_{n-2}j_{n-1}j_n + \cdots k_n 0.j_1\right)\\ &= \frac{2\pi i}{N}\left[k_1 2^{n-1}j_n + k_2(2^{n-1}j_{n-1} + 2^{n-2}j_n) + \cdots k_n(2^{n-1}j_1 + 2^{n-2}j_2 + \cdots + j_n)\right]\\ &= \frac{2\pi i}{N}\sum_{a=0}^{n-1}\left(2^a j_{n-a}k - 2^n\sum_{b=1}^{a} k_{a-b+1}2^{b-1}\right)\\ &= \frac{2\pi i jk}{N} + 2\pi i N_1,\end{aligned} \tag{2.55}$$

with N_1 as an integer. Thus, $|\psi'(n)\rangle$ is the quantum Fourier transform of the basis $|j\rangle$ given in (2.38), and

$$U_{\mathrm{QFT}} = (SWAP)U_n^f U_{n-1}^f\cdots U_1^f. \tag{2.56}$$

In Fig. 2.6, we show the quantum circuit for implementing U_{QFT} when the number of qubits is five.

2.3.2 Quantum phase estimation

Given an unitary operator as U and its eigenstate as $|u\rangle$, the following holds

$$U|u\rangle = e^{2\pi i\gamma}|u\rangle \tag{2.57}$$

in general, where γ is a phase that takes a real number ($0 \le \gamma < 1$). The quantum phase estimation is an algorithm to obtain the value of the phase γ. First, we assume that we can prepare the eigenstate $|u\rangle$, and later we relax the condition. We also assume that Controlled-U^{2^j} ($j = 0, 1, \cdots, n$) operation is implementable.

In the phase estimation, we prepare the following state as

$$|\phi(0)\rangle = (H^{\otimes q}|0\rangle^{\otimes n})|u\rangle = \frac{1}{\sqrt{2^q}}\left(|0\rangle + |1\rangle\right)^{\otimes q}|u\rangle. \tag{2.58}$$

There are two stages for the phase estimation: the first stage is the sequencial operations of the controlled-U^{2^j} gates, and the second stage is the operation $U_{\mathrm{QFT}}^\dagger$. Suppose that we write the controlled-U^{2^j} operation as $CU_j(x)$, where x is the index of the control qubit and U^{2^j} is operated to the last w qubits with w as the number of qubits in $|u\rangle$. Conversely, $CU_j(x)$ operates U^{2^j} to the last w qubits only if the x-th qubit is $|1\rangle$. Then, the output of the first stage $|\phi(1)\rangle$ is written as

$$|\phi(1)\rangle = CU_{q-1}(1)\cdots CU_2(q-2)CU_1(q-1)CU_0(q)|\phi(0)\rangle. \tag{2.59}$$

Each controlled-U^{2^j} operation sequentially append phases to the first n-qubits as follows:

$$\begin{aligned}
&CU_0(q)|\phi(0)\rangle\\
&= \frac{1}{\sqrt{2^q}}(|0\rangle + |1\rangle)^{\otimes q-1}(|0\rangle|u\rangle + |1\rangle U^{2^0}|u\rangle)\\
&= \frac{1}{\sqrt{2^q}}(|0\rangle + |1\rangle)^{\otimes q-1}(|0\rangle + e^{2\pi i 2^0\gamma}|1\rangle)|u\rangle\\
&CU_1(q-1)CU_0(q)|\phi(0)\rangle\\
&= \frac{1}{\sqrt{2^q}}(|0\rangle + |1\rangle)^{\otimes q-2}(|0\rangle + e^{2\pi i 2^1\gamma}|1\rangle)(|0\rangle + e^{2\pi i 2^0\gamma}|1\rangle)|u\rangle\\
&\vdots\\
&CU_{q-1}(1)\cdots CU_2(q-2)CU_1(q-1)CU_0(q)|\phi(0)\rangle\\
&= \frac{1}{\sqrt{2^q}}(|0\rangle + e^{2\pi i 2^{q-1}\gamma}|1\rangle)(|0\rangle + e^{2\pi i 2^{q-2}\gamma}|1\rangle)\cdots(|0\rangle + e^{2\pi i 2^0\gamma}|1\rangle)|u\rangle\\
&= \frac{1}{\sqrt{2^q}}\sum_{k=0}^{2^q-1} e^{2\pi i k\gamma}|k\rangle|u\rangle.
\end{aligned} \tag{2.60}$$

The second stage is the operation of $U_{\mathrm{QFT}}^\dagger$ to the first q-qubits. By utilizing (2.41), we obtain

$$\begin{aligned}
|\phi(2)\rangle = U_{\mathrm{QFT}}^\dagger \otimes I_w|\phi(1)\rangle &= \frac{1}{\sqrt{2^q}}\sum_{k=0}^{2^q-1} e^{2\pi i k\gamma}(U_{\mathrm{QFT}}^\dagger|k\rangle)|u\rangle\\
&= \frac{1}{2^q}\sum_{k=0}^{2^q-1}\sum_{\ell=0}^{2^q-1} e^{2\pi i k(\gamma-\ell/2^q)}|\ell\rangle|u\rangle.
\end{aligned} \tag{2.61}$$

If $\gamma = N_\gamma/2^q$ with some integer $0 \le N_\gamma < N$,

$$\sum_{k=0}^{N-1} e^{2\pi i k(\gamma-\ell/2^q)} = 2^q\delta_{N_\gamma \ell}, \tag{2.62}$$

and therefore,

$$|\phi(2)\rangle = \sum_{\ell=0}^{2^q-1}\delta_{N_\gamma\ell}|\ell\rangle|u\rangle = |N_\gamma\rangle|u\rangle. \tag{2.63}$$

Thus, the measurement result of the first n-qubit always returns N_γ, which gives γ. Note that even when γ cannot be written in the form of $N_\gamma/2^q$, it is known that we obtain the nearest integer to $2^q\gamma$ with high probability. The quantum circuit for the phase estimation in case that $q = 5$ is written in Fig. 2.7.

The number of gates required for implementing the quantum phase estimation is $O(2^q)$ as long as we need to implement U^{2^q} as the 2^q repetition of the operation U. In some algorithms, such as the prime factoring, U^{2^q} is implementable by utilizing $O(q^3)$ gates [2]. In this case, the total number of gates for implementing the quantum phase estimation is also $O(q^3)$ since for implementing $U_{\mathrm{QFT}}^\dagger$, only $O(q^2)$ gates are needed.

In the above, we assume that we can prepare the eigenstate of U as $|u\rangle$. Even without the assumption, the same approach estimates the phases of an arbitrary initial state. Suppose that

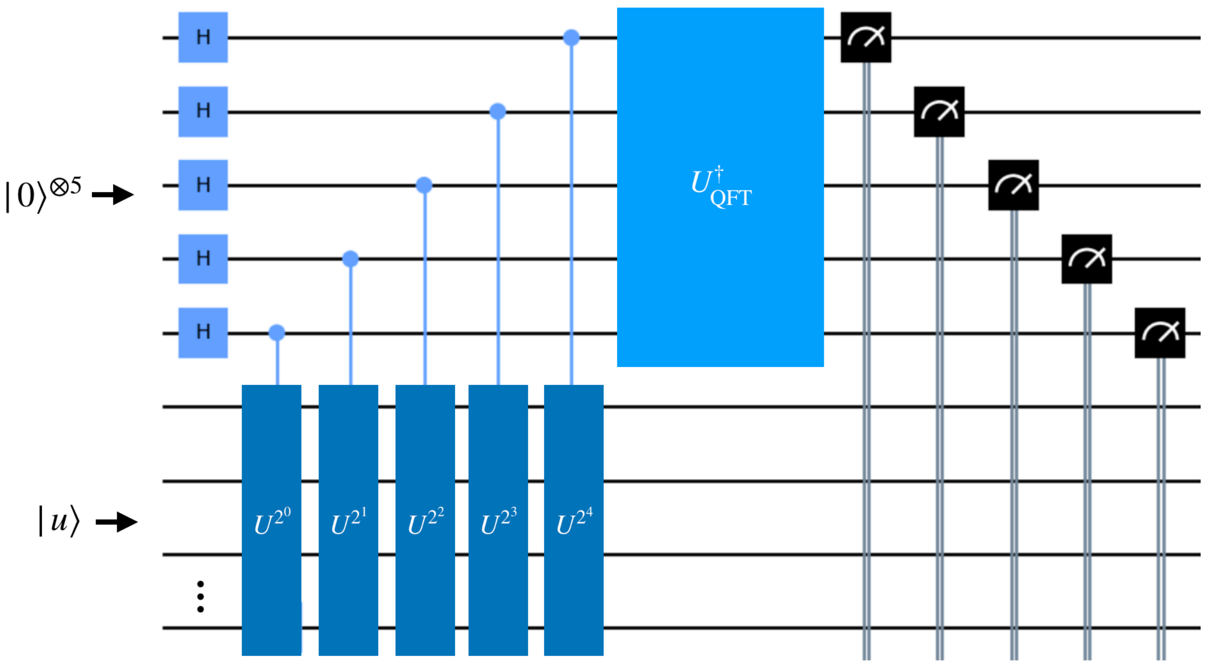


Figure 2.7: The quantum circuit for the quantum phase estimation when $q = 5$.

the set of the eigenstates of U is given by $\{|\phi_j\rangle\}$, and the corresponding eigenvalues are given by $\{\gamma_j\}$. If we prepare $|\phi\rangle$ as the initial state, it can be expanded as

$$|\phi\rangle = \sum_{j=1}^{2^w} x_j |\phi_j\rangle. \tag{2.64}$$

Given the unitary operation for the phase estimation as U_{PE},

$$U_{\text{PE}}|0\rangle^{\otimes n}|\phi\rangle = \sum_{j=1}^{2^w} x_j |N\gamma_j\rangle|\phi_j\rangle, \tag{2.65}$$

here we assume that $2^q\gamma_j$ is an integer. Thus, by the computational measurement of the first n-qubits, we obtain the value of γ_j and its eigenstate $|\phi_j\rangle$ with the probability $|x_j|^2$.

Finding the ground state energy of a Hamiltonian

One of the application of the phase estimation is finding the ground state of a given Hamiltonian $\mathcal{H}$. More precisely, the problem is finding the following $|\phi_0\rangle$ and E_0:

$$\mathcal{H}|\phi_0\rangle = E_0|\phi_0\rangle, \tag{2.66}$$

where E_0 is the minimum eigenvalue of $\mathcal{H}$. The problem can be rewritten in the following form:

$$e^{i\mathcal{H}t}|\phi_0\rangle = e^{iE_0t}|\phi_0\rangle. \tag{2.67}$$

Thus, if we can prepare $|\phi_0\rangle$ and implement $e^{i\mathcal{H}t}$ by the quantum circuit efficiently, we can solve the problem by using the quantum phase estimation. In reality, we need careful treatment for the initial state preparation and the implementation of $e^{i\mathcal{H}t}$.

As for the initial state preparation, if we prepare a state $|0\rangle^{\otimes q}|\phi\rangle = |0\rangle^{\otimes q}\sum_{j=0}^{2^w} x_j|\phi_j\rangle$, then the probability that we obtain the eigenstate with the minimum energy as a result of the measurement is $|x_0|^2$ after the phase estimation. Thus, for getting the target state with high probability, we need to prepare the state $|\phi\rangle$ that has enough overlap with $|\phi_0\rangle$, which depends on the problems; for example, in quantum chemistry, the adiabatic state preparation is a method to prepare a good initial state [60].

An important technique to efficiently implement $e^{i\mathcal{H}t}$ is the one using the Lie-Trotter-Suzuki decomposition [61]. Hamiltonians we often have interests, such as the ones in the quantum chemistry, can be written in the form of

$$\mathcal{H} = \sum_{j=1}^{M} h_j, \tag{2.68}$$

where $e^{ih_j t}$ is implementable with polynomial number of qubits (see [62]). The first order of the Lie-Trotter-Suzuki decomposition of $e^{i\mathcal{H}t}$ is

$$e^{i\mathcal{H}t} = \left(\prod_{j=1}^{M} e^{ih_j t/S}\right)^S + O(t^2/S), \tag{2.69}$$

where we can arbitrary choose S; the first term in the right hand side is implementable by using polynomial number of circuits. If we choose S as $S = \epsilon/t^2$, then the approximation error becomes $O(\epsilon)$. We implement the first term of (2.69) instead of $e^{i\mathcal{H}t}$.

Other algorithms utilizing the phase estimation

We saw the power of the quantum phase estimation above. There are other critical applications of utilizing phase estimation.

One of the essential algorithms is the Shor's algorithm [2] for prime factoring. By taking advantage of the feature that $CU_j(\cdot)$ in the prime factoring is implementable with $O(j^3)$ gates, the computational complexity of the quantum algorithm is exponentially smaller than the classical counterpart. Also, Harrow-Hassidim-Lloyd (HHL) algorithm [63] effectively uses phase estimation to solve linear equations.

Chapter 3

Noisy intermediate-scale quantum computers and variational quantum algorithms

In the previous chapter, we reviewed some powerful quantum algorithms that outperform each classical counterpart. To execute those algorithms, we need ideal quantum computers that have $O(1000)$ qubits available for the computation, few noise, long coherence time, and the error correction functions. However, such ideal quantum computers will not be available for several decades due to the difficulty of the development. In fact, current quantum computers are far from the ideal devices; they have $O(10) \sim O(100)$ available qubits, much noise, short coherence time, and no error correction functions. Such non-ideal quantum devices are called noisy intermediate-scale quantum computers (NISQ) [16].

Researchers in the quantum computing community have been actively studying how to utilize NISQ rather than waiting for the development of the ideal quantum computers for decades. The key points of maximizing the functionality of NISQ are two-fold: (i) efficiently mitigating the errors from the computational results and (ii) solving problems with a small number of gates. The former is essential for making the final computational results reliable, while the latter is essential for reducing the noise. These two key points are targeted in many recent proposals for NISQ utilization.

Based on the above, we review the researches regarding NISQ utilization in this chapter. In Section 3.1, we will first see the limitation of NISQ. Then in Section 3.2, we review the method called *error mitigation* for mitigating the noise, which corresponds to the key point (i). Note that Section 3.2 does not directly relate with the rest of this thesis, but we add the section for completeness. In Section 3.3, we review the variational quantum algorithm (VQA) realizing quantum computation processes with fewer gates, which corresponds to the key point (ii). Even though VQA is arguably the most important NISQ algorithm, there exist critical scalability issues in VQA. We discuss the issues in Section 3.4. Possible solutions to the issues are discussed in Chapter 6.

3.1 Limitation of the near-term devices

Various types of quantum computers are now being actively developed. One of the most promising quantum computers is one with superconducting qubits; companies such as IBM, Google, Microsoft, etc., are now working on developing superconducting quantum computers. As we note above, the devices are far from the ideal device, and there are many limitations of the current devices. In the following, we see those limitations with the example of superconducting qubits.

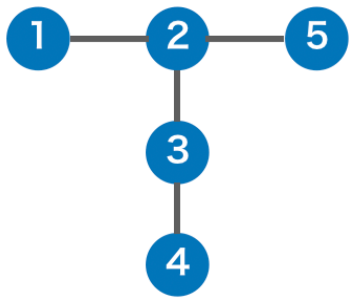


Figure 3.1: An example of the qubit connectivity in superconducting quantum computers. Each number in the figure corresponds to the label of each qubit, and the black bars between each pair of qubits show that two-qubit operations are executable in the pair.

3.1.1 Limited number of qubits

The apparent limitation of NISQ is the number of qubits. Currently, the number of qubits available for computation in quantum computers is $O(10)$, and may go to $O(100)$ in the near future. The limited number of qubits result in the two consequences.

First, the execution of many important quantum algorithms such as factorization does not have a quantum advantage over classical algorithms. Even if there is exponential speed-up, the classical computational complexity is at most $O(2^n)$, where n is the number of qubits; if n is small, $O(2^n)$ can be easily simulated by classical devices.

Second, the quantum error correction can not be used if the number of qubits is limited. The error correction protocol requires several qubits per one logical qubit for adding redundancy. Even if $O(100)$ qubits are available in a device, only O(10) qubits are used for the computation if we use the error correction, which makes the first point even worse; therefore, the execution of the quantum error correction is unrealistic in NISQ.

3.1.2 Noise

In NISQ, errors in computations and the decoherence affect the computational results in the noise. Since the noise cannot be corrected by the quantum error correction protocol mentioned above, the effect of noise is directly reflected in the computational results.

Regarding the gate error, two-qubit gates are severer than single-qubit gates; the error rate of the two-qubit gate is as large as $O(10^{-3})$ [64–66] while that of single-qubit gate is $O(10^{-4})$ [67]. Consequently, the computational result after $O(100)$ operations of CNOT gates can not be trusted at all. In addition, the measurement operations also append sizable errors, which is also a significant source of the error in the computational results.

We also need to care about the coherence time. The duration that a qubit maintains its quantum state is called the coherence time; qubits decohere by interacting with their environment and losing information about the original state. Since qubits gradually decohere, as the time for computation becomes longer, the decoherence affects more badly the computational results. Thus, the effect of decoherence also limits the number of gate operations.

3.1.3 Qubit connectivity

In NISQ, two-qubit operations are only executable in limited pairs of the qubits. If two-qubit operations are executable in a pair of qubits, the pair is said to have the connection, and the set of the pairs is called the qubit connectivity. In Fig. 3.1, we show an example of the qubit connectivity in superconducting quantum computers. Each number in the figure corresponds to the label of each qubit. The black bars between each pair of qubits show that two-qubit operations are executable in the pair.

The limited connections between the qubits increase the number of two-qubit gates. For example, suppose that the qubit connectivity is given by Fig. 3.1 and an algorithm requires a

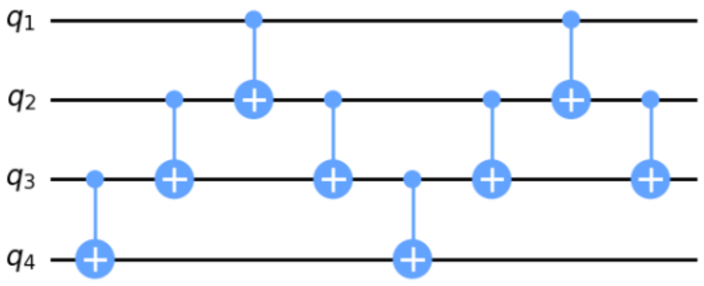


Figure 3.2: CNOT operation between q_1 and q_4 when the qubit connectivity is given by Fig. 3.1.

CNOT gate between the qubit with label 1 (q_1) and the qubit with label 4 (q_4), where q_1 is the control and q_4 is the target. Since we can not directly execute the CNOT gate between these qubits, we need to construct an equivalent operation in another way, which is done by utilizing the qubits with label 2 (q_2) sand label 3 (q_3). The resulting circuit is drawn in Fig. 3.2. We see that we use two-qubit operations only between pairs of connected qubits. Notably, the number of CNOT gates becomes eight times larger than when there is a connection between q_1 and q_4. In this way, the number of two-qubit gates considerably increases in NISQ, where the connections are limited.

3.2 Error mitigation

Here we review some of the protocols to mitigate the error from the computational results. Most error mitigation protocol aims to remove the effect of noise on the expectation value of observables rather than recovering the ideal quantum state. Namely given an ideal final state not affected by noise as ρ_{ideal}, and an observable as $\mathcal{O}$, the goal of the error mitigation is recovering the following property $\mathrm{Tr}(\rho_{\mathrm{ideal}}\mathcal{O})$ from the measurement results in a noisy environment.

3.2.1 Measurement error mitigation

One of the important source of errors is appended in the measurement. Given a set of POVM operators as $\{E_k\}_{k=1}^{N_P}$ with N_P as the number of POVM elements, the probability that the k-th element is observed is written as

$$p_k = \mathrm{Tr}(E_k\rho), \tag{3.1}$$

where ρ is the state to be measured. We assume that the probabilities $\mathbf{p} = \{p_k\}_{k=1}^{N_P}$ are distorted by noise as

$$\mathbf{p}' = X\mathbf{p}, \tag{3.2}$$

where X is an invertible matrix corresponding to the effect of noise and $\mathbf{p}'$ is the probability distribution distorted by the noise [68]. If $X = I_{N_P}$, then there is no noise in the measurement; to the contrary, large off-diagonal elements correspond to the case where the effect of noise in the measurement is large.

To obtain the ideal probability distribution $\mathbf{p}$ from distorted probability distribution $\mathbf{p}'$, we need to know the vector $\mathbf{p}$. In fact, the matrix X is obtained by the quantum detector tomography (QDT) under the assumption that the noise in the measurement of a qubit does not affect the noise in the measurement of another qubit [68]. Once X is obtained, we may compute $\mathbf{p}$ by

$$\mathbf{p} = X^{-1}\mathbf{p}'. \tag{3.3}$$

The vector $\mathbf{p}$ might be unphysical, e.g., some elements of $\mathbf{p}$ are negative. In such case, $\mathbf{p}$ is obtained by

$$\mathbf{p} = \underset{\mathbf{p}_t\in Q_P}{\arg\min}\left[\mathbf{p}' - X\mathbf{p}_t\right], \tag{3.4}$$

where Q_P is the set of physical probability distribution with N_P elements.

3.2.2 Zero noise extrapolation

There are several ways to mitigate the noise in the computational process. As one of the most powerful tools, extrapolation error mitigation techniques are proposed. Here we review some of such methods.

Richardson extrapolations

Let us assume that a quantum gate is distorted after the execution of p-th gate by the following noise channel:

$$\mathcal{E}_p = (1-\epsilon(p))\mathcal{I} + \epsilon(p)\mathcal{N}_p, \tag{3.5}$$

where $\mathcal{I}$ is the identity map, $\mathcal{N}_p$ is p-th noisy map, and $\epsilon(p)$ is the p-th noise parameter. Also, we assume that $\epsilon(p)$ takes the same value ϵ for all p. Then, given the expectation value of an observable $\mathcal{O}$ when the noise parameter is ϵ as $\langle\mathcal{O}\rangle_\epsilon$, the expectation value can be expanded to Taylor series as

$$\langle\mathcal{O}\rangle_\epsilon = \langle\mathcal{O}\rangle_0 + \sum_{k=1}^{\infty} O_k \epsilon^k. \tag{3.6}$$

In the Richardson extrapolation [69, 70], it is assumed that the noise parameter can be amplified without changing the noise channel $\mathcal{N}(\rho)$. There are several ways for error amplification. One example is appending the extra gates that ideally do not change the computational results; for example, two consequent CNOT gates to a certain pair of qubits do not change the computational result ideally, but the extra noise is appended. Another example is the re-scaling of the physical Hamiltonian for gate operations; namely, by slowing down the process of executing quantum gates, the noise for each gate is amplified.

Suppose that we measure the observable in different noise parameters: $\epsilon = \epsilon_0 < \epsilon_1 < \cdots < \epsilon_K$. Then until the K-th order, we obtain

$$\langle\mathcal{O}\rangle_{\epsilon_j} = \langle\mathcal{O}\rangle_0 + \sum_{k=1}^{K} o_k \epsilon_j^k + O(\epsilon^{K+1}), \tag{3.7}$$

where $\{o_k\}_{k=1}^K \in \mathbb{R}$. In the matrix equation form,

$$\begin{pmatrix} 1 & \epsilon_0 & \epsilon_0^2 & \cdots & \epsilon_0^K \\ 1 & \epsilon_1 & \epsilon_1^2 & \cdots & \epsilon_1^K \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \epsilon_K & \epsilon_K^2 & \cdots & \epsilon_K^K \end{pmatrix} \begin{pmatrix} \langle\mathcal{O}\rangle_0 + O(\epsilon^{K+1}) \\ o_1 \\ \vdots \\ o_K \end{pmatrix} = \begin{pmatrix} \langle\mathcal{O}\rangle_{\epsilon_0} \\ \langle\mathcal{O}\rangle_{\epsilon_1} \\ \vdots \\ \langle\mathcal{O}\rangle_{\epsilon_K} \end{pmatrix}. \tag{3.8}$$

By solving the equation, we obtain the ideal expectation value as

$$\langle\mathcal{O}\rangle_0 = \sum_{k=1}^{K} \alpha_k \langle\mathcal{O}\rangle_{\epsilon_k} + O(\epsilon^{K+1}), \tag{3.9}$$

where $\{\alpha_k\}_{k=1}^K \in \mathbb{R}$. Note that each α_k increases exponentially as K becomes larger; therefore if the number of measurements for estimating $\langle\mathcal{O}\rangle_{\epsilon_k}$ is fixed, the estimation error of $\langle\mathcal{O}\rangle_0$ increases exponentially with the value of K. Thus, we need to set K to a small value.

Exponential extrapolations

Suppose that the noise channel after the p-th gate is written by

$$\mathcal{E}_p = (1-\epsilon)\mathcal{I} + \epsilon\mathcal{N}_p \tag{3.10}$$

as in the Richardson extrapolation case. In Richardson extrapolation, we implicitly assume that the Talor expansion until K-th order well approximate the relationship between $\langle O\rangle_\epsilon$ and $\langle O\rangle_0$. However, when the number of gates is large while ϵ is small, the Talor expansion does not capture the relationship effectively, as we see in the following.

Suppose that p-th noiseless gate operation is given by $\mathcal{U}_p$, then given the number of gate as N_G, the whole noisy gate operations is written by

$$\mathcal{E} \equiv \mathcal{E}_{N_G} \circ \mathcal{U}_{N_G} \circ \cdots \circ \mathcal{E}_2 \circ \mathcal{U}_2 \circ \mathcal{E}_1 \circ \mathcal{U}_1. \tag{3.11}$$

Using (3.10), it holds that

$$\mathcal{E} = \sum_{k=0}^{N_G} \epsilon^k (1-\epsilon)^{N_G-k} \sum_{q=1}^{n_k} \mathcal{W}_q^k, \tag{3.12}$$

where $n_k = \binom{N_G}{k}$ and $\{\mathcal{W}_j^k\}_{j=1}^{n_k}$ are all the elements of the set S^k defined by

$$S^k = \left\{ \mathcal{W} | \mathcal{W} = \prod_{j=1}^{N_G} \mathcal{X}_j \mathcal{U}_j, \mathcal{X}_j = \mathcal{I} \text{ or } \mathcal{E}_j, \text{the number of } j \text{ that satisfies } \mathcal{X}_j = \mathcal{I} \text{ is } k \right\}. \tag{3.13}$$

Then, by setting $\mathcal{W}^k = \sum_{q=1}^{n_k} W_q^k / n_k$, we obtain

$$\mathcal{E} = \sum_{k=0}^{N_G} \binom{N_G}{k} \epsilon^k (1-\epsilon)^{N_G-k} \mathcal{W}^k. \tag{3.14}$$

In the limit that N_G is large while $N_G\epsilon < 1$, the Poisson limit theorem tells that

$$\binom{N_G}{k} \epsilon^k (1-\epsilon)^{N_G-k} \simeq e^{-N_G\epsilon} \frac{(N_G\epsilon)^k}{k!}, \tag{3.15}$$

and therefore

$$\mathcal{E} \simeq e^{-N_G\epsilon} \sum_{k=0}^{N_G} \frac{(N_G\epsilon)^k}{k!} \mathcal{W}^k. \tag{3.16}$$

Approximating the sum up to the first order of $N_G\epsilon$, we obtain

$$\mathcal{E} \simeq e^{-N_G\epsilon} (\mathcal{W}^0 + N_G\epsilon\mathcal{W}^1 + O((N_G\epsilon)^2)). \tag{3.17}$$

Then the expectation value of the observable $\mathcal{O}$ when the error parameter is ϵ is computed as

$$\langle \mathcal{O} \rangle_\epsilon = e^{-N_G\epsilon} \left(\langle O \rangle_0 + N_G\epsilon \mathrm{Tr}(\mathcal{O}\mathcal{W}^1(\rho)) + O((N_G\epsilon)^2) \right), \tag{3.18}$$

where ρ is the input state. If we can amplify the noise parameter to $a\epsilon$ $(a > 1)$,

$$\langle \mathcal{O} \rangle_{a\epsilon} = e^{-N_G a\epsilon} \left(\langle O \rangle_0 + N_G a\epsilon \mathrm{Tr}(\mathcal{O}\mathcal{W}^1(\rho)) + O((N_G\epsilon)^2) \right). \tag{3.19}$$

Combining (3.18) with (3.19), we obtain

$$\langle O \rangle_0 = \frac{a\langle O \rangle_\epsilon e^{N_G\epsilon} - \langle O \rangle_{a\epsilon} e^{N_G a\epsilon}}{a-1} + O((N_G\epsilon)^2). \tag{3.20}$$

The error mitigation protocol using this formula is called the exponential extrapolation [71, 72].

Note that the merit of expanding $\mathcal{E}$ as (3.17) is that we can include all order of $e^{-N_G\epsilon}$, while the Taylor series expansion of $\mathcal{E}$, which is utilized in the Richardson extrapolation only includes the first order of $e^{-N_G\epsilon}$ as

$$\mathcal{E} \simeq \mathcal{W}^0 + (-N_G\mathcal{W}^0 + N_G\mathcal{W}^1)\epsilon + O((N_G\epsilon)^2). \tag{3.21}$$

Thus, in case that we can use the approximation (3.16), it is more preferable to use the exponential extrapolation than the Richardson extrapolation.

3.3 Variational quantum algorithms

This section reviews the variational quantum algorithm (VQA) proposed to obtain the quantum advantage in NISQ [23, 24]. VQA is the method to solve tasks by expressing them in the form of the cost function minimization problems using parameterized quantum circuits. VQA is now utilized in various problems, and they share some of the concepts. The variants of VQA are also actively studied. One of the most promising variants is the quantum generative adversarial network (QGAN).

In this chapter, we firstly review the basic concepts shared with almost all VQA applications in Section 3.3.1. In Section 3.3.2 we review the applications of VQA. In Section 3.3.3, we review QGAN, which is a promissing variant of VQA.

3.3.1 Basic concepts

The building blocks of VQA are the parameterized quantum circuit (PQC), the cost function, and the optimizer. In the following, we firstly review those components one by one, and next, we see how those components are combined in VQA.

Parameterized quantum circuit

PQC is a circuit where some of the gates are parameterized by $\boldsymbol{\theta}$; we write the circuit as $U(\boldsymbol{\theta})$. How we embed parameters into the circuit depends on tasks. PQC with a specific structure is called ansatz. There are two types of ansatz used in VQA. One type is the problem agnostic ansatz, where prior knowledge is not reflected in the structure of the ansatz. Another type is the ansatz tailored for specific problems, e.g., the unitary coupled-cluster ansatz [73]. Since the problem agnostic ansatz is now actively examined due to its broad applicability, we focus on it in the subsequent section. The most typical ansatz among the problem agnostic ansatz is the hardware efficient ansatz, which we show the detail in the following.

Hardware efficient ansatz

The hardware efficient ansatz [25] consists of multiple layers of single-qubit rotation gates and the entangler that entangles all qubits. Concretely suppose that the number of layers is L and the unitary operator corresponding to the ℓ-th layer is $U_\ell(\boldsymbol{\theta^\ell})$, then the unitary operator corresponding to the hardware efficient ansatz is given by

$$U(\boldsymbol{\theta}) = \prod_{\ell=1}^{L} U_\ell(\boldsymbol{\theta}^\ell), \tag{3.22}$$

where $U_\ell(\boldsymbol{\theta}^\ell)$ can be decomposed as

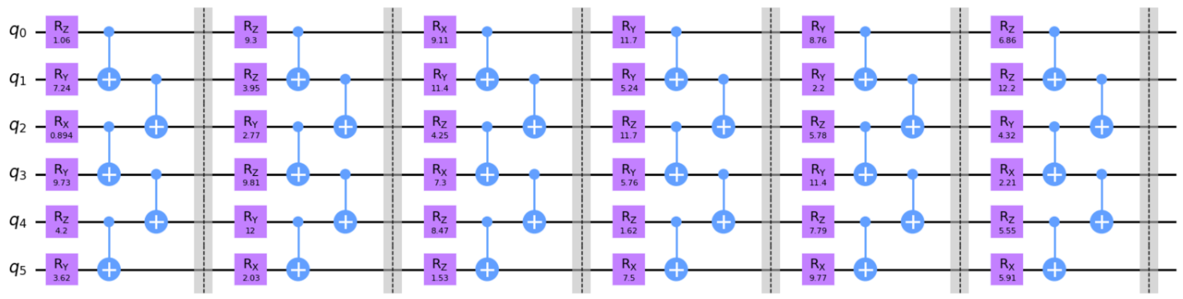
$$U_\ell(\boldsymbol{\theta}^\ell) = WV(\boldsymbol{\theta^\ell}), \tag{3.23}$$

with $V(\boldsymbol{\theta}^\ell)$ corresponding to the single-qubit gates and W corresponding to the entangler. In the single-qubit gate, the j-th parameter is embedded as $\exp(i\theta_j\sigma_j)$, where σ_j is one of the Pauli operators. In many of the settings $V(\boldsymbol{\theta}^\ell)$ is the tensor product of the single-qubit rotation gates;

$$V(\boldsymbol{\theta}^\ell) = \exp(i\theta_1^\ell\sigma_1^\ell) \otimes \exp(i\theta_2^\ell\sigma_2^\ell) \otimes \cdots \otimes \exp(i\theta_n^\ell\sigma_n^\ell), \tag{3.24}$$

where θ_j^ℓ is the j-th parameter in the ℓ-th layer and σ_j^ℓ is again one of the Pauli operators. We show an example of the structure of the hardware efficient ansatz in Fig. 3.3, which is depicted by using [74]. As we see in the figure, we do not use any knowledge to build the structure of the hardware efficient ansatz.

Note that the hardware efficient ansatz characteristics that each entangler only entangles neighboring qubits is suitable for the execution in current superconducting quantum computers where the connection between qubits is limited as in Fig. 3.1. On the contrary, we need many long-range interactions between qubits in general to implement ansatz tailored for problems; namely, we need many two-qubit operations between qubits not directly connected, which requires an unignorable amount of swap gates.


q_0 R_Z 1.06 R_Z 9.3 R_X 9.11 R_Y 11.7 R_Y 8.76 R_Z 6.86
q_1 R_Y 7.24 R_Z 3.95 R_Y 11.4 R_Y 5.24 R_Y 2.2 R_Z 12.2
q_2 R_X 0.894 R_Y 2.77 R_Z 4.25 R_Z 11.7 R_Z 5.78 R_Y 4.32
q_3 R_Y 9.73 R_Z 9.81 R_X 7.3 R_Y 5.76 R_Y 11.4 R_X 2.21
q_4 R_Z 4.2 R_Y 12 R_Z 8.47 R_Z 1.62 R_Z 7.79 R_Z 5.55
q_5 R_Y 3.62 R_X 2.03 R_Z 1.53 R_X 7.5 R_X 9.77 R_X 5.91
Figure 3.3: The structure of the hardware efficient ansatz when the number of qubits is six and the number of layers is six.

Cost function

The cost function in VQA is the function to be minimized, which is often computed by

$$C(\boldsymbol{\theta}) = \mathrm{Tr}(\rho U(\boldsymbol{\theta})^\dagger \mathcal{O} U(\boldsymbol{\theta})) \tag{3.25}$$

with an Hermitian observable $\mathcal{O}$ and an input state ρ. We update the parameters in the direction that $C(\boldsymbol{\theta})$ decreases, and after enough iterations we obtain the optimal parameter $\boldsymbol{\theta}^*$.

The cost function is chosen so that minimization of the cost function corresponds to the solution to the target problem. For example, if the problem we want to solve is finding the minimum energy of a Hamiltonian H, the observable and the input state is chosen to be $\mathcal{O} = H$ and $\rho = (|0\rangle\langle 0|)^{\otimes n}$, where n is the number of qubits. The minimum energy obtained as the solution of VQA is $C(\boldsymbol{\theta}^*) = \mathrm{Tr}(\rho U(\boldsymbol{\theta})^\dagger \mathcal{O} U(\boldsymbol{\theta}))$.

Optimizer

To find the optimal parameters $\boldsymbol{\theta}^*$ we iteratively update the parameters $\boldsymbol{\theta}$ as $\boldsymbol{\theta}(0) \to \boldsymbol{\theta}(1) \to \cdots \to \boldsymbol{\theta}(T)$, where $\boldsymbol{\theta}(k)$ are the parameters at the k-th iteration step and T is the final iteration step. The component to find the parameters $\boldsymbol{\theta}(t+1)$ from $\boldsymbol{\theta}(t)$ is called the optimizer. The gradient-based optimizers are used in many of the problems.

The gradient-based optimizers utilize the gradient vector of the cost function, which is given by $\{\partial C(\boldsymbol{\theta})/\partial\theta_j\}_{j=1}^P$ where P is the number of parameters. The gradient vector can be computed by using the following *parameter shift rule* [75]:

$$\frac{\partial C(\boldsymbol{\theta})}{\partial \theta_j} = C(\boldsymbol{\theta}^{(j+)}) - C(\boldsymbol{\theta}^{(j-)}), \tag{3.26}$$

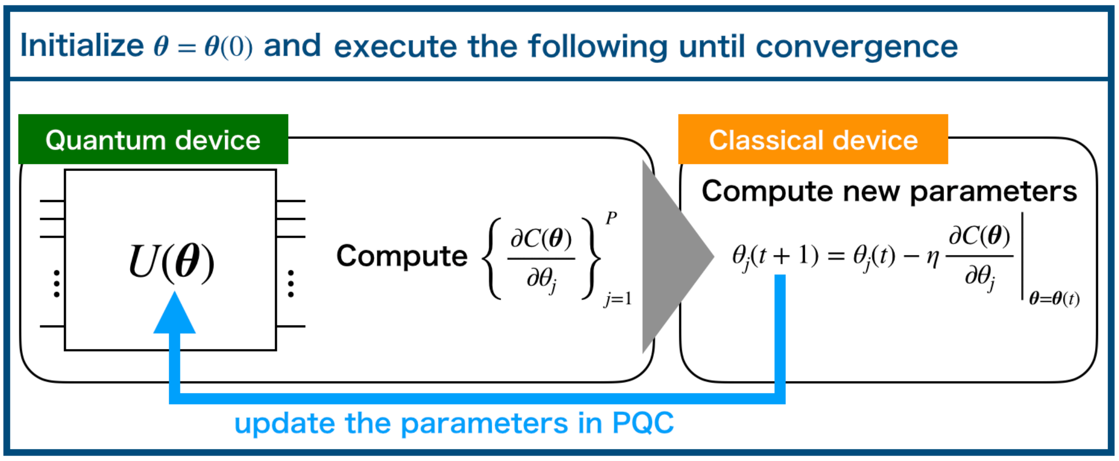
where the elements of $\boldsymbol{\theta}^{(j\pm)}$ are given by

$$\theta_k^{(j\pm)} = \begin{cases} \theta_k & (k \neq j) \\ \theta_k \pm \pi/2 & (k = j) \end{cases} \tag{3.27}$$

under the condition that the j-th parameter θ_j is embedded into the quantum circuit as $\exp(-i\theta_j\Sigma/2)$ with Σ as the Hermitian operator satisfying $\Sigma^2 = I$. In the rest of this thesis, we assume that the condition for the parameter-shift rule is satisfied because ansatz we have interests, such as the hardware efficient ansatz, satisfy the condition. In the practical situation we can not exactly estimate the value of the gradient, but by using the $O(M)$ measurements of $C(\boldsymbol{\theta}^{(j+)})$ and $C(\boldsymbol{\theta}^{(j-)})$, we can estimate the value of the gradient within $O(1/\sqrt{M})$ error with high probability.

The most important gradient-based optimizer is the stochastic gradient descendent (SGD) where each parameter is updated by the following rule:

$$\theta(t+1)_j = \theta(t)_j - \eta \left.\frac{\partial C(\boldsymbol{\theta})}{\partial \theta_j}\right|_{\boldsymbol{\theta}=\boldsymbol{\theta}(t))}, \tag{3.28}$$



Initialize $\boldsymbol{\theta} = \boldsymbol{\theta}(0)$ and execute the following until convergence
Quantum device
$U(\boldsymbol{\theta})$
Compute $\left\{ \frac{\partial C(\boldsymbol{\theta})}{\partial \theta_j} \right\}_{j=1}^{P}$
Classical device
Compute new parameters
$\theta_j(t+1) = \theta_j(t) - \eta \frac{\partial C(\boldsymbol{\theta})}{\partial \theta_j}\Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}(t)}$
update the parameters in PQC

Figure 3.4: The overview of the VQA when using the gradient descent optimizer.

where η is called learning rate that we can freely choose. There are also many variants of the SVD optimizer; for example, Adam [76], which is originally proposed for the classical machine learning, is one of the SGD optimizer.

Combining the above-discussed components, we can write down the process of VQA. Initially, we determine the ansatz and the cost function, so the minimization of the cost corresponds to the problem's solution. Then we initialize the parameters in the ansatz; there are various ways of initialization, such as random initialization. The optimizer updates the parameters according to its rule. For example, suppose we use a gradient-based optimizer. In that case, the gradient vector of the cost function is computed, and the parameters are updated for the direction of the gradient vector as in (3.28). The update of the parameters is repeated until convergence. The parameters, the value of the cost function, and the value of the quantum state generated by the circuit at the final iteration are the output of VQA. In Fig. 3.4 we show the overview of VQA when using the gradient descent optimizer.

Note that classical devices execute the computation of finding the new parameters. In contrast, quantum computers compute the cost function or the gradient vector. It should also be noted that we can choose the number of gates in the ansatz; if the number of gates is small, then the noise can be reduced, which is why VQA is expected to be executable in NISQ.

3.3.2 Applications

Variational quantum eigensolver

The variational quantum eigensolver (VQE) is an algorithm to find the minimum eigenvalue and corresponding eigenstate of a Hamiltonian. More precisely, given a Hamiltonian as $\mathcal{H}$ the goal of VQE is finding the solution that satisfies

$$\mathcal{H}|\phi\rangle = E_0|\phi\rangle, \tag{3.29}$$

where E_0 is the minimum eigenvalue of the Hamiltonian and $|\phi\rangle$ is the corresponding eigenstate. Note that finding such a solution is especially important in the quantum chemistry [3–6], where $\mathcal{H}$ is the molecule's Hamiltonian. As we can see, if the dimension of the Hilbert space is exponentially large, the problem is difficult to be solved by any classical devices.

It can be easily checked that by using the phase estimation algorithm, which we discussed in Section 2.3.2, we efficiently find the solution. However, the phase estimation algorithm requires the inverse Fourier transform, which requires many controlled operations and is intractable in NISQ.

To find the solution by NISQ, VQE is proposed as a method that does not use phase estimation. In the algorithm, the cost function $C(\boldsymbol{\theta})$ is given by the expectation value of the Hamiltonian; namely

$$C(\boldsymbol{\theta}) = \mathrm{Tr}(\rho U^\dagger(\boldsymbol{\theta})\mathcal{H}U(\boldsymbol{\theta})), \tag{3.30}$$

where $U(\boldsymbol{\theta})$ is the unitary operator corresponding to PQC and ρ is a pure input state: $\rho = |\Phi_{\mathrm{in}}\rangle\langle\Phi_{\mathrm{in}}|$. Then given the optimal parameters as $\boldsymbol{\theta}^*$, the minimum eigenvalue is given by $C(\boldsymbol{\theta}^*)$ and the corresponding eigenstate is given by $U(\boldsymbol{\theta}^*)|\Phi_{\mathrm{in}}\rangle$. From the parameter-shift rule (3.26), we can compute the gradient vector as

$$\frac{\partial C(\boldsymbol{\theta})}{\partial \theta_j} = \mathrm{Tr}(\rho U^\dagger(\boldsymbol{\theta}^{(j+)})\mathcal{H}U(\boldsymbol{\theta}^{(j+)})) - \mathrm{Tr}(\rho U^\dagger(\boldsymbol{\theta}^{(j-)})\mathcal{H}U(\boldsymbol{\theta}^{(j-)})). \tag{3.31}$$

The update of the parameters is often performed by using the gradient descendant with this gradient formula.

Quantum circuit learning

The variational quantum algorithm is also used for machine learning problems. As one of the essential variational quantum algorithms in machine learning, we review the quantum circuit learning (QCL)[77] in the following. Before going into the detail of QCL, let us review the supervised machine learning problem settings.

Supervised learning

Supervised learning is one of the most important machine learning tasks, whose goal is building a functional approximator or a classifier from a labeled training dataset. There are various real-world supervised learning applications, such as image processing, object recognition, and natural language processing.

In supervised machine learning, the training dataset denoted by $\{\mathbf{x}^k, \mathbf{y}^k\}_{k=1}^M$ is given, where $\mathbf{x}^k$ and $\mathbf{y}^k$ are the k-th input vector and the label vector respectively, and M is the number of the training data. The supervised learning is the task to find the function that correctly maps an input vector to an label vector. More precisely suppose that the training dataset is sampled from the unknown data distribution $\mathcal{D}$, then the supervised learning is the task to find the function f that minimizes the following value

$$\mathcal{C}(f) \equiv \mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}[L(f(\mathbf{x}), \mathbf{y})], \tag{3.32}$$

where $L(f(\mathbf{x})), \mathbf{y})$ is a distance measure between $f(\mathbf{x})$ and $\mathbf{y}$; therefore as $f(\mathbf{x})$ predicts the label vector correctly, the cost becomes small. An example of the distance measure is the following, which is called as the mean squared error (MSE) given by

$$L(f(\mathbf{x}), \mathbf{y}) = ||f(\mathbf{x}) - \mathbf{y}||_2^2, \tag{3.33}$$

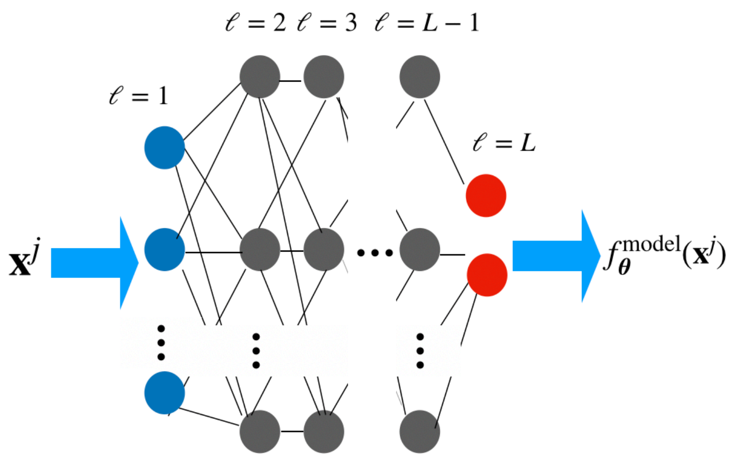
where $||\cdot||_2$ is the L^2-norm.

Normally we limit the function space to a parameterized function space $\mathcal{F} = \{g | g = f_{\boldsymbol{\theta}}^{\mathrm{model}}, \boldsymbol{\theta} \in \mathbb{R}^P\}$, where P is the number of parameters and $f_{\boldsymbol{\theta}}^{\mathrm{model}}$ is a function whose dimension of the input and output are same as the input vector and the label vector of the training dataset respectively. The choice of the function form of $f_{\boldsymbol{\theta}}^{\mathrm{model}}$ corresponds to the choice of the model, e.g. if we use a neural network with a certain structure as the model, the function form of $f_{\boldsymbol{\theta}}^{\mathrm{model}}$, and hence the function space $\mathcal{F}$ is determined. Then the problem of minimizing (3.32) is converted to the problem of minimizing the following value

$$\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}[L(f_{\boldsymbol{\theta}}^{\mathrm{model}}(\mathbf{x}), \mathbf{y})]. \tag{3.34}$$

However, we can not directly obtain the solution of (3.34) because the distribution $\mathcal{D}$ is unknown. Thus, we approximate the expectation value by using the given training dataset as

$$\mathbb{E}_{(\mathbf{x},\mathbf{y})\sim\mathcal{D}}[L(f_{\boldsymbol{\theta}}^{\mathrm{model}}(\mathbf{x})), \mathbf{y})] \approx \mathcal{L}(\boldsymbol{\theta}) \equiv \frac{1}{M}\sum_{k=1}^{M} L(f_{\boldsymbol{\theta}}^{\mathrm{model}}(\mathbf{x}^k), \mathbf{y}^k). \tag{3.35}$$



$\ell = 2$ $\ell = 3$ $\ell = L-1$
$\ell = 1$
$\ell = L$
$\mathbf{x}^j$
$f_{\boldsymbol{\theta}}^{\mathrm{model}}(\mathbf{x}^j)$

Figure 3.5: The overview of the structure of $f_{\boldsymbol{\theta}}^{\mathrm{model}}$, when the model is DNN. The blue nodes correspond to the input layer, and the red nodes correspond to the output layer.

We can update parameters by using an optimizer such as the gradient descendant (3.28) so that (3.35) is minimized. The function $\mathcal{L}(\boldsymbol{\theta})$ is called as the cost function.

As an example of $f_{\boldsymbol{\theta}}^{\mathrm{model}}$ let us briefly introduce the deep neural network (DNN), which is currently the most successful machine learning model. In DNN, the model function $f_{\boldsymbol{\theta}}^{\mathrm{model}}$ is constructed by a network of L layers. Let n_ℓ be the number of nodes (width) of the ℓ-th layer ($\ell = 1$ and $\ell = L$ correspond to the input and output layers, respectively). Then the input $\mathbf{x}^j$ is converted layer by layer to the output $f_{\boldsymbol{\theta}}^{\mathrm{model}}(\mathbf{x}^j)$ in the following manner:

$$\begin{aligned}
\boldsymbol{\alpha}^{(1)}(\mathbf{x}^j) &= \mathbf{x}^j, \\
\boldsymbol{\alpha}^{(\ell)}(\mathbf{x}^j) &= \sigma^{(\ell)}(\tilde{\boldsymbol{\alpha}}^{(\ell)}(\mathbf{x}^j)), \\
\tilde{\boldsymbol{\alpha}}^{(\ell+1)}(\mathbf{x}^j) &= \frac{1}{\sqrt{n_\ell}} W^{(\ell)} \boldsymbol{\alpha}^{(\ell)}(\mathbf{x}^j) + \xi b^{(\ell)}, \\
f_{\boldsymbol{\theta}}^{\mathrm{model}}(\mathbf{x}^j) &= \boldsymbol{\alpha}^{(L)}(\mathbf{x}^j),
\end{aligned} \tag{3.36}$$

where $W^{(\ell)} \in \mathbb{R}^{n_l \times n_{l-1}}$ is trainable weight matrix and $b^{(\ell)} \in \mathbb{R}^{n_l}$ is trainable bias vector, and $\sigma^{(\ell)}$ is the activation function in the ℓ-th layer. The choices of each $W^{(\ell)}$, $b^{(\ell)}$ and $\sigma^{(\ell)}$ determine the function structure of each DNN. We illustrate the overview of the structure of DNN in Fig. 3.5.

Note that the minimization of $\mathcal{L}(\boldsymbol{\theta})$ may lead to the phenomenon called *overfitting*. For example, suppose that the dimensions of both the input vector and the label vector is one, and the training data is sampled from the following distribution $\mathcal{D}$; the input x distributes under the uniform distribution between 0 and 3π and the corresponding label y distributes as

$$y = \sin(2x/3) + \epsilon, \tag{3.37}$$

where ϵ is sampled from the centerd Gaussian distribution with the standard deviation equals to 0.2. Then obviously (3.34) is minimzed if and only if $f_{\boldsymbol{\theta}}^{\mathrm{model}}(x) \sim \sin(2x/3)$ in $x \in [0, 3\pi]$. On the contrary there are many functions that minimize (3.35) including non-smooth function; in Fig. 3.6 we show one of $f_{\boldsymbol{\theta}}^{\mathrm{model}}(x)$ when (3.35) is almost minimized where $M = 15$ and L is MSE. We see that although $f_{\boldsymbol{\theta}}^{\mathrm{model}}(x)$ goes through most of the training data, it is not close to $\sin(2x/3)$ at all, and therefore $f_{\boldsymbol{\theta}}^{\mathrm{model}}(x)$ does not correctly predict the label unless the input x coincides with one of the training data. For avoiding this issue, the regularization term $\Omega(\boldsymbol{\theta})$, which makes $f_{\boldsymbol{\theta}}^{\mathrm{model}}(x)$ smooth, is often added to $\mathcal{L}(\boldsymbol{\theta})$ as

$$\mathcal{L}_\Omega(\boldsymbol{\theta}) \equiv \mathcal{L}(\boldsymbol{\theta}) + \Omega(\boldsymbol{\theta}) \tag{3.38}$$

and $\mathcal{L}_\Omega$ is minimized instead of $\mathcal{L}(\boldsymbol{\theta})$. One example of the regularization term is $\Omega(\boldsymbol{\theta}) = \lambda ||\boldsymbol{\theta}||_2^2$ with a positive real value λ; in this case the value of $\mathcal{L}_\Omega$ become large if the norm of the parameter

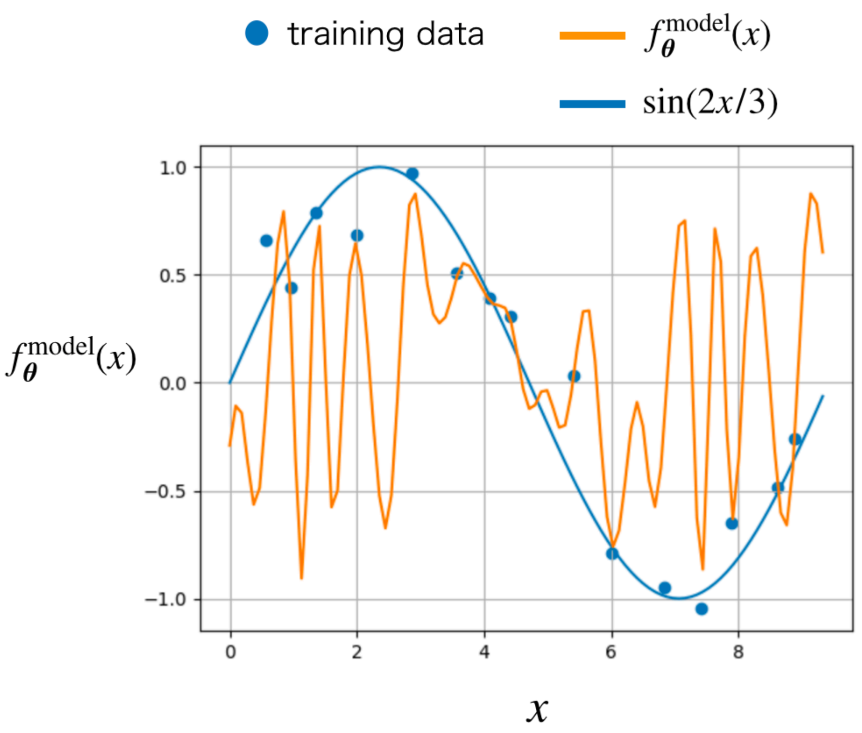


Figure 3.6: An example of overfitting in a supervised machine learning problem.

vector is large, and therefore the norm of the parameter vector tends to become small as a result of the optimization. Note that even though the strategy to use the regularization term works in some cases, it is often difficult to design the term properly.

The algorithm of the quantum circuit learning

QCL is an algorithm using the quantum circuit for the supervised learning problem. In QCL, the function $f_{\boldsymbol{\theta}}^{\text{model}}$ is defined by

$$f_{\boldsymbol{\theta}}^{\text{model}}(\mathbf{x}) \equiv \text{Tr}(\rho_{\mathbf{x}} U(\boldsymbol{\theta})^{\dagger} \mathcal{O} U(\boldsymbol{\theta})), \tag{3.39}$$

where $\mathcal{O}$ is an Hermite observable and $\rho_{\mathbf{x}} = U(\mathbf{x})|0\rangle^{\otimes n}\langle 0|^{\otimes n} U(\mathbf{x})^{\dagger}$ with $U(\mathbf{x})$ as a unitary operator that encodes an input vector $\mathbf{x}$. The choice of the encoding circuit $U(\mathbf{x})$ is arbitrary; one of the choice is

$$U(\mathbf{x}) = e^{i\sigma_z x_1} \otimes e^{i\sigma_z x_2} \otimes \cdots \otimes e^{i\sigma_z x_n}. \tag{3.40}$$

In QCL, we can compute the cost function by (3.35) and the gradient vector by

$$\begin{aligned}\frac{\partial \mathcal{L}(\boldsymbol{\theta})}{\partial \theta_j} &= \frac{1}{M} \sum_{k=1}^{M} \left.\frac{\partial L(f, \mathbf{y}^k)}{\partial f}\right|_{f=f_{\boldsymbol{\theta}}^{\text{model}}(\mathbf{x}^k)} \frac{\partial \text{Tr}(\rho_{\mathbf{x}^k} U(\boldsymbol{\theta})^{\dagger} \mathcal{O} U(\boldsymbol{\theta}))}{\partial \theta_j} \\ &= \frac{1}{M} \sum_{k=1}^{M} \left.\frac{\partial L(f, \mathbf{y}^k)}{\partial f}\right|_{f=f_{\boldsymbol{\theta}}^{\text{model}}(\mathbf{x}^k)} \left(\text{Tr}(\rho_{\mathbf{x}^k} U(\boldsymbol{\theta}^{(j+)})^{\dagger} \mathcal{O} U(\boldsymbol{\theta}^{(j+)})) - \text{Tr}(\rho_{\mathbf{x}^k} U(\boldsymbol{\theta}^{(j-)})^{\dagger} \mathcal{O} U(\boldsymbol{\theta}^{(j-)}))\right),\end{aligned} \tag{3.41}$$

where in the last line, we use the parameter-shift rule shown in (3.26). Using the gradient descendant algorithm with this gradient vector, we update the parameters so that the cost function is minimized.

In [77], it is claimed that the possible quantum advantage of QCL over classical algorithms is that the model function of QCL (3.39) has a potential power to represent more complex functions than its classical counterpart. On the contrary, subsequent studies [78] have shown that the conditions under which the quantum advantage is achieved in QCL are stringent; namely, this topic is an area of active discussion, and further research is needed in this regard.

Also, the possibility of overfitting is discussed in [77]. Concretely, it is inferred that QCL is less likely to overfit compared to the classical counterpart because the model function (3.39) is built by the unitary operator and the unitary constraint corresponds to the regularization term. Even though no counterexamples have been shown so far, theoretical evidence is needed for verifying this conjecture.

3.3.3 Quantum generative adversarial network

In this subsection, we review the quantum generative adversarial network (QGAN), which is the variant of the variational quantum algorithm, and the quantum counterpart of the generative adversarial network (GAN) [41] in classical machine learning. We first review the GAN algorithm, and next, we show the detail of QGAN algorithms.

Generative adversarial network (GAN)

GAN is originally proposed as an algorithm to train generative models. A generative model is a sampler trained by a dataset. What a generative model generates as a sampler depends on the dataset; for example, a generative model becomes an image generator if trained by an image dataset and becomes a text generator if it is trained by a document dataset. Normally the goal of the training of the generative model is that the samples generated by the model become similar to the training dataset. The generative model has various real-world applications such as the image synthesis [79–86], the text generation [87–90], and the object recognition [91–93].

The most successful algorithm for training the generative model is GAN. GAN consists of two adversarial components: a generator and a discriminator. The goal of the training is to obtain a good generator as the generative model that generates samples similar to the training dataset. The generator takes a set of random seeds as its input and transforms them into samples called fake data. The discriminator receives either the fake data or the training data (real data) and classifies them as 'fake' or 'real' exclusively. The discriminator is trained so that the true data is classified as 'real' and the fake data is classified as 'fake'. In contrast, the generator is trained so that the fake data generated by the generator is classified as 'real' by the discriminator. Namely, the generator is trained to fool the discriminator, and the discriminator is trained to identify fakes. Surprisingly we obtain a good generator as a result of this adversarial training.

The training of GAN is formulated as follows. We denote the function corresponding to the generator as $G(\mathbf{z})$ and that corresponding to the discriminator as $D(\mathbf{x})$, where $\mathbf{x}$ corresponds to the fake/real data and z corresponds to random numbers. A random number is necessary because if the generator does not take any random inputs, the generator's output is always the same; by taking the random numbers as its input, the generator can generate a wide variety of samples. The dimension of $G(\mathbf{z})$ is the same as the training data. The discriminator $D(\mathbf{x})$ takes the value between zero and one; zero means that the classification result is 'fake,' and one means that that is 'real.' The generator and the discriminator implicitly have trainable parameters $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$ respectively. Still, we do not show them for simplicity. The generator and the discriminator tend to be implemented by deep neural networks. We denote the training dataset by $X = \{\mathbf{x}^j_{\text{data}}\}_{j=1}^M$ and the corresponding random numbers as $\{\mathbf{z}^j\}_{j=1}^M$.

The cost function for the generator and its gradient is given by

$$L_G(\boldsymbol{\theta}) = -\int d\mathbf{z} \log D\left(G(\mathbf{z})\right) p(\mathbf{z}) = -\mathbb{E}_{\mathbf{z}\sim p(z)}[\log D(G(\mathbf{z}))], \tag{3.42}$$

$$\frac{\partial L_G(\boldsymbol{\theta})}{\partial \theta_k} = -\int d\mathbf{z} \frac{1}{D\left(G(\mathbf{z})\right)} \left.\frac{\partial D\left(g\right)}{\partial g}\right|_{g=G(\mathbf{z})} \frac{\partial G(\mathbf{z})}{\partial \theta_k} p(\mathbf{z}), \tag{3.43}$$

where $p(\mathbf{z})$ is the distribution of the random numbers that we can freely choose, e.g., the Gaussian distribution. In the practical training, we can not exactly compute the expectation value and therefore we approximate the value by using M samples $\{\mathbf{x}^j_{\text{fake}}\}_{j=1}^M$ and random numbers $\{\mathbf{z}^j\}_{j=1}^M$ as

$$L_G(\boldsymbol{\theta}) \simeq -\frac{1}{M}\sum_{j=1}^{M} \log D(\mathbf{x}^j_{\text{fake}}), \tag{3.44}$$

$$\frac{\partial L_G}{\partial \theta_k} \simeq -\frac{1}{M}\sum_{j=1}^{M} \frac{1}{D\left(\mathbf{x}^j_{\text{fake}}\right)} \left.\frac{D\left(g\right)}{\partial g}\right|_{g=\mathbf{x}^j_{\text{fake}}} \left.\frac{\partial G(\mathbf{z})}{\partial \theta_k}\right|_{\mathbf{z}=\mathbf{z}^j}. \tag{3.45}$$

The cost function and the gradient vector for the discriminator is given by

$$\begin{aligned}
L_D(\boldsymbol{\theta}') &= -\frac{1}{2}\int d\mathbf{z} \left[\log(1-D(G(\mathbf{z}))\right] p(\mathbf{z}) - \frac{1}{2}\int d\mathbf{x} \log D(\mathbf{x}) q(\mathbf{x}) &(3.46)\\
&= -\frac{1}{2}\left(\mathbb{E}_{\mathbf{z}\sim p(z)}[\log(1-D(G(\mathbf{z}))] + \mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})}[\log D(\mathbf{x})]\right), &(3.47)\\
\frac{\partial L_D(\boldsymbol{\theta}')}{\partial \theta'_k} &= \frac{1}{2}\int d\mathbf{z} \frac{1}{1-D(G(\mathbf{z}))}\frac{\partial D(G(\mathbf{z}))}{\partial \theta'_k} p(\mathbf{z}) - \frac{1}{2}\int d\mathbf{x} \frac{1}{D(\mathbf{x})}\frac{\partial D(\mathbf{x})}{\partial \theta'_k} q(\mathbf{x}) &(3.48)\\
&= \frac{1}{2}\mathbb{E}_{\mathbf{z}\sim \mathbf{p}(\mathbf{z})}\left[\frac{1}{1-D(G(\mathbf{z}))}\frac{\partial D(G(\mathbf{z}))}{\partial \theta'_k}\right] - \frac{1}{2}\mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})}\left[\frac{1}{D(\mathbf{x})}\frac{\partial D(\mathbf{x})}{\partial \theta'_k}\right], &(3.49)
\end{aligned}$$

where $q(\mathbf{x})$ is unknown data distribution. The expectation values are also approximated by the sum as follows:

$$L_D(\boldsymbol{\theta}') \simeq -\frac{1}{2M}\sum_{j=1}^{M}\left[\log\left(1-D(\mathbf{x}^j_{\text{fake}})\right) + \log D(\mathbf{x}^j_{\text{data}})\right], \tag{3.50}$$

$$\frac{\partial L_D(\boldsymbol{\theta}')}{\partial \theta'_k} \simeq \frac{1}{2M}\sum_{j=1}^{M}\left(\frac{1}{1-D(\mathbf{x}^j_{\text{fake}})}\frac{\partial D(\mathbf{x}^j_{\text{fake}})}{\partial \theta'_k} - \frac{1}{D(\mathbf{x}^j_{\text{data}})}\frac{\partial D(\mathbf{x}^j_{\text{data}})}{\partial \theta'_k}\right). \tag{3.51}$$

It should be noted that the gradient of $G(\cdot)$ and $D(\cdot)$ are efficiently computable by using the technique called the *back propagation* as long as we implement the generator and the discriminator by DNN discussed around the equation (3.36).

The training is the repetition of the following two steps: (1) updating the parameters $\boldsymbol{\theta}$ by the gradient descendant with the gradient vector in (3.45), and (2) updating the parameters $\boldsymbol{\theta}'$ by the gradient descendant with the gradient vector in (3.51). As a result of the training, we obtain the generator as the good generative model for generating samples similar to the dataset X; namely, the distribution of the probability that the generator generates a sample $\mathbf{x}$ becomes close to $q(\mathbf{x})$. The overview of the architecture of GAN is shown in Fig. 3.7.

Quantum generative adversarial network (QGAN)

QGAN is the algorithm whose generator and/or the discriminator are implemented by quantum circuits. Among various QGAN models, we review QGAN for training the data loading circuit (QGANDL) [29].

Loading data into the amplitudes of a quantum state, which is called as *amplitude encoding*, is an important submodule for various algorithms. However, exactly loading data into the amplitudes of a quantum state requires exponential number of quantum gates [94–100], which

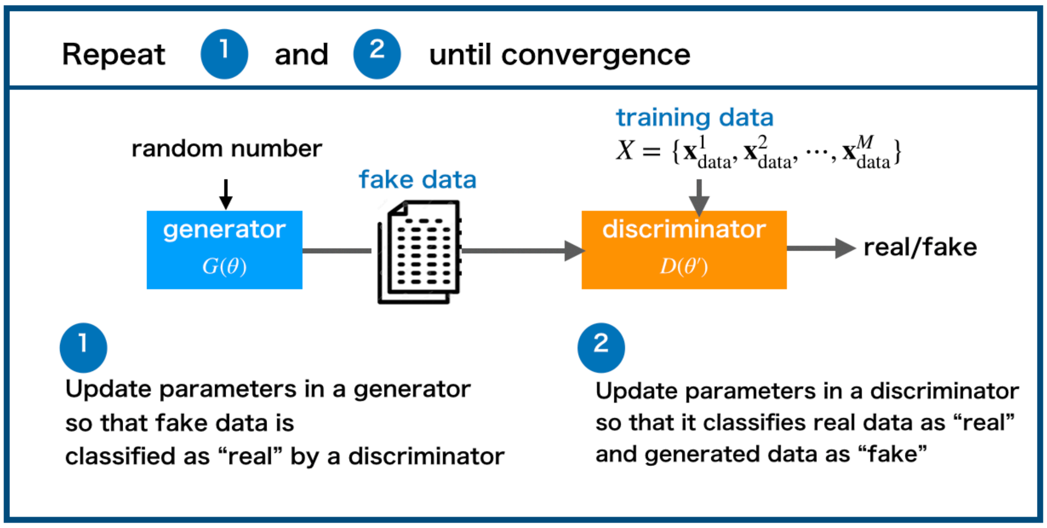

Repeat 1 and 2 until convergence
random number
generator
$G(\theta)$
fake data
training data
$X = \{\mathbf{x}_{\rm data}^1, \mathbf{x}_{\rm data}^2, \cdots, \mathbf{x}_{\rm data}^M\}$
discriminator
$D(\theta')$
real/fake
1
Update parameters in a generator so that fake data is classified as "real" by a discriminator
2
Update parameters in a discriminator so that it classifies real data as "real" and generated data as "fake"

Figure 3.7: The overview of the generative adversarial network (GAN).

is intractable in NISQ when the number of qubits is large. Methods for realizing the amplitude encoding with fewer gates are highly demanded.

The literature [29] proposes a QGANDL to approximately load the underlying probability distribution of the training data into the amplitudes of a quantum state with fewer gates. More precisely, given the discretized underlying probability distribution as $q(\mathbf{r})$, the goal of the algorithm is generating the following state:

$$|\psi\rangle \simeq \sum_{\mathbf{r}} \sqrt{q(\mathbf{r})} e^{i\alpha_{\mathbf{r}}} |\mathbf{r}\rangle, \tag{3.52}$$

where each $|\mathbf{r}\rangle$ is the computational basis, and each $\alpha_{\mathbf{r}}$ can be any real number, namely controlling each α_r is beyond the scope of the algorithm. In other words, the goal of the algorithm is approximately building a state where the probability of measuring the state $|\mathbf{r}\rangle$ is $q(\mathbf{r})$. Note that if $|\psi\rangle$ is loaded, it is applicable to practical problems such as the Monte Carlo integration.

QGANDL is proposed as an algorithm to achieve the goal by using GAN formulation, where the generator is implemented by a quantum circuit and the discriminator is implemented by a classical machine learning model such as a deep neural network. More precisely, M outputs of the generator $\{\mathbf{x}_{\rm fake}^j\}_{j=1}^M$ are obtained by measuring a state $U(\boldsymbol{\theta})|0\rangle^{\otimes n}$ in the computational basis, where $U(\boldsymbol{\theta})$ is a PQC. When generating one output, the probability that the output equals to $\mathbf{r}$ is given by

$$p_G(\boldsymbol{\theta}, \mathbf{r}) = \mathrm{Tr}\left(\rho_0 U(\boldsymbol{\theta})^\dagger |\mathbf{r}\rangle\langle\mathbf{r}| U(\boldsymbol{\theta})^\dagger\right), \tag{3.53}$$

where $\rho_0 = |0\rangle^{\otimes n}\langle 0|^{\otimes n}$. The discriminator is the same as the one in the classical counterpart. It should be noted that, unlike the classical case, the generator need not take random numbers as its input because the randomness is already included in the quantum generator; namely, the computational measurement of the quantum state returns a different result for each measurement.

Suppose that the training data is given by $X = \{\mathbf{x}_{\rm data}{}^j\}_{j=1}^M$ as in the discussion about GAN. The function for the discriminator is $D(\mathbf{x})$, and the generator is G, where the input of the generator is omitted. Then the cost function and the gradient vector for the discriminator in QGANDL are same as (3.50) and (3.51). Also the cost function for the generator is given by (3.44). To the contrary, the gradient vector of the generator in QGANDL is different from the formula (3.45). Concretely the gradient is given by

$$\frac{\partial L_G(\boldsymbol{\theta})}{\partial \theta_k} = -\frac{\partial}{\partial \theta_k}\left(\int d\mathbf{r} \log(D(\mathbf{r})) p_G(\boldsymbol{\theta}, \mathbf{r})\right) \tag{3.54}$$

$$= -\int d\mathbf{r} \log(D(\mathbf{r})) \frac{\partial p_G(\boldsymbol{\theta}, \mathbf{r})}{\partial \theta_k}, \tag{3.55}$$

$$= -\int dr \log(D(\mathbf{r})) \left(p_G(\boldsymbol{\theta}^{(k+)}, \mathbf{r}) - p_G(\boldsymbol{\theta}^{(k-)}, \mathbf{r})\right), \tag{3.56}$$

$$= -\mathbb{E}_{r\sim p_G(\boldsymbol{\theta}^{(k+)}, \mathbf{r})}\left[\log(D(\mathbf{r}))\right] + \mathbb{E}_{r\sim p_G(\boldsymbol{\theta}^{(k-)}, \mathbf{r})}\left[\log(D(\mathbf{r}))\right], \tag{3.57}$$

where in the third line, we use the parameter shift rule (3.26). Each expectation value $\mathbb{E}_{\mathbf{r}\sim p_G(\boldsymbol{\theta}^{(k\pm)}, \mathbf{r})}[\cdot]$ can be approximately computed by M samples generated by measuring the quantum circuit $U(\boldsymbol{\theta}^{(j\pm)})|0\rangle^{\otimes n}$ in the computational basis. Let us denote the j-th sample obtained from measuring $U(\boldsymbol{\theta}^{(k\pm)})|0\rangle^{\otimes n}$ as $\mathbf{x}_{\text{fake}}^{j(k\pm)}$, then

$$\frac{\partial L_G(\boldsymbol{\theta})}{\partial \theta_k} \simeq \frac{1}{M}\sum_{j=1}^{M}\left(-\log(D(\mathbf{x}_{\text{fake}}^{j(k+)})) + \log(D(\mathbf{x}_{\text{fake}}^{j(k-)}))\right). \tag{3.58}$$

The procedure to train the generator and the discriminator using the gradient vectors is the same as that in the classical counterpart. As a result of the adversarial training, the samples generated by the generator becomes close to the underlying distribution $q(\mathbf{r})$. Namely, given the state generated by PQC as $|\psi\rangle$, the following holds

$$|\langle\psi|\mathbf{r}\rangle|^2 \simeq q(\mathbf{r}), \tag{3.59}$$

which is the goal (3.52).

Recall that QGANDL only loads the probability distribution, and the phases $\alpha_{\mathbf{r}}$ is uncontrollable. On the contrary, the algorithm in [101], which is a contribution of the author of this thesis, proposes a method to load any real vectors into the amplitudes of a quantum state without using QGAN formalism.

In classical machine learning, we can find the most successful application of GAN in the semi-supervised learning tasks [44–46]. Even in the quantum setting, the quantum generative adversarial network can be applied to the semi-supervised learning tasks by utilizing the algorithm proposed by the author of this thesis. We describe the algorithm in Chapter 5.

3.4 Critical issues of the variational quantum algorithms

As seen in Section 3.3, VQA and its variants are suitable for the execution in NISQ, and there are many promising applications. However, to apply VQA to practical problems, we need to overcome several critical issues. Of particular important issue is the so-called *barren plateau* issue, which is pointed initially out by Ref. [102]. In Section 3.4.1, we firstly review the barren plateau issue, and the other issues are briefly discussed in Section 3.4.2. We discuss the possible solutions to the barren plateau issue in Chapter 6.

3.4.1 Barren plateau issue

In VQA, we need to update parameters in PQC so that the cost function is likely to decrease. In many of the problems of VQA, the new parameters are found by utilizing the gradient descent algorithm that we discuss in Section 3.3. Reference [102] theoretically shows that as the number of qubits n increases, the norm of the gradient becomes exponentially small when PQC is deep, and we randomly choose $\boldsymbol{\theta}$, which is called the barren plateau issue. More precisely, given the input state as ρ, the observable as $\mathcal{O}$, the unitary operator corresponding to PQC as $U_{\boldsymbol{\theta}}$, and the cost function as $C(\boldsymbol{\theta}) \equiv \mathrm{Tr}(\rho U_{\boldsymbol{\theta}}^{\dagger}\mathcal{O}U_{\boldsymbol{\theta}})$,

$$\left\langle\frac{\partial C}{\partial \theta_j}\right\rangle \equiv \int d\mu(U_{\boldsymbol{\theta}})\frac{\partial C}{\partial \theta_j} = 0, \quad \left\langle\left(\frac{\partial C}{\partial \theta_j}\right)^2\right\rangle \equiv \int d\mu(U_{\boldsymbol{\theta}})\left(\frac{\partial C}{\partial \theta_j}\right)^2 = O\left(\frac{1}{2^n}\right), \tag{3.60}$$

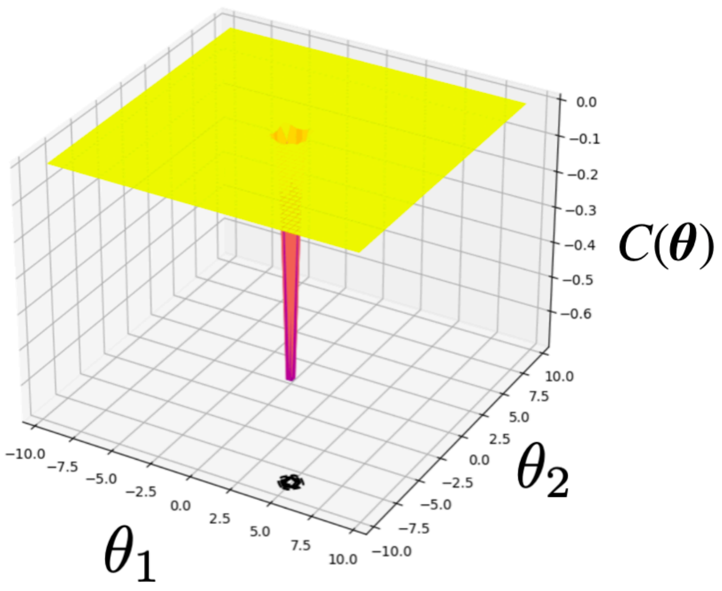


Figure 3.8: A cartoon of the cost function landscape with the cost function $C(\boldsymbol{\theta})$ corresponding to Equation (3.60).

where $d\mu(U_{\boldsymbol{\theta}})$ is the measure of the probability of the unitary operator when randomly choosing $\boldsymbol{\theta}$. The deriviation of (3.60) is shown at the end of this section. The formulas imply that the landscape of the cost function is almost everywhere flat as we show in Fig. 3.8.

The reasons why (3.60) is fatal in VQA are two-fold. First, we need an exponential number of measurements to correctly estimate the sign of $\partial C/\partial\theta_j$ for each j and each point. Thus to update the parameters to proper directions, $O(2^n)$ measurements are needed, which is intractable in practical problems with $O(100)$ qubits (recall $2^{100} \sim O(10^{30})$). Second, we need an exponential number of iterations to obtain the optimal parameters since the distance between the initial parameters and the optimal parameters are $\mathcal{O}(1)$ on average, while in each iteration, the parameters are updated only by $O(1/2^n)$ as long as we choose the learning rate as $O(1)$.

It should be noted that in [103], it is shown that the presence of noise makes the barren plateau issue severer, which is unfavorable for VQA that is expected to work in noisy devices. Also, it is shown in [104] that even if we use the gradient-free method, e.g., Nelder-Mead, Powell, and COBYLA, for the optimization, the barren plateau issue is unavoidable. That is because the gradient-free techniques need to find new points where the cost function becomes smaller as in the case of the gradient descendant, but for finding the points where the cost decrease by $O(1)$ value, $O(2^n)$ trials are necessary since the cost function landscape is flat in almost everywhere.

As we see in the above discussion, the barren plateau issue is so severe that it may kill all of the variational quantum algorithms, including the ones stated in Section 3.3. Thus many pieces of research are tackling this issue, some of which provide possible solutions. We will discuss those solutions in Chapter 6.

Derivation of the equations (3.60)

Before concluding this subsection let us show the detail derivation of (3.60). The equations (3.60) are theoretically proven under the assumption that the distribution of the unitary matrices corresponding to the PQC is unitary 2-design [105]. The unitary 2-design is the random distribution that resembles the Haar distribution, the random distribution in the Haar measure. Namely, when the distribution of the unitary operators is unitary 2-design, the following element-wise integration formulae hold:

$$\int_{\text{2design}} dU U_{ab}U^*_{cd} = \int_{\text{Haar}} dU U_{ab}U^*_{cd} = \frac{\delta_{ab}\delta_{cd}}{N}, \tag{3.61}$$

$$\begin{aligned}\int_{\text{2design}} dU U_{ab}U^*_{cd}U_{ef}U^*_{gh} = \int_{\text{Haar}} U_{ab}U^*_{cd}U_{ef}U^*_{gh} &= \frac{1}{N^2-1}\left(\delta_{ac}\delta_{bd}\delta_{eg}\delta_{fh} + \delta_{ag}\delta_{bh}\delta_{ce}\delta_{df}\right) \\ &- \frac{1}{N(N^2-1)}(\delta_{ac}\delta_{bh}\delta_{eg}\delta_{fd} + \delta_{ah}\delta_{bd}\delta_{ec}\delta_{fg}),\end{aligned} \tag{3.62}$$

where $\int_{\text{2design}} dU$ is the integration over the unitary 2-design, $\int_{\text{Haar}} dU$ is integration over the Haar distribution, and N is the dimension of the unitary matrix. The assumption seems to be valid because as the PQC becomes deeper, the distribution of the unitary operator corresponding to PQC becomes more random and closer to unitary 2-design when randomly choosing the parameters [102]. These formulae are used to prove (3.60).

The gradient of the cost function can be written as

$$\frac{\partial C}{\partial \theta_j} = \left(\mathrm{Tr}\left(\rho \frac{\partial U^\dagger_{\boldsymbol{\theta}}}{\partial \theta_j}\mathcal{O}U_{\boldsymbol{\theta}}\right) - \mathrm{Tr}\left(\rho U^\dagger_{\boldsymbol{\theta}}\mathcal{O}\frac{\partial U_{\boldsymbol{\theta}}}{\partial \theta_j}\right)\right). \tag{3.63}$$

Since each parameter is embedded as $\exp(i\Sigma_j\theta_j)$ where Σ_j is an Hermitian operator, we can write

$$U_{\boldsymbol{\theta}} = \tilde{U}_+ \exp(i\Sigma_j\theta_j)\tilde{U}_- \tag{3.64}$$

by using a parameterized circuit $\tilde{U}_+$ and $\tilde{U}_-$. Then its derivative becomes

$$\frac{\partial U_{\boldsymbol{\theta}}}{\partial \theta_j} = i\tilde{U}_+\Sigma_j \exp(i\Sigma_j\theta_j)\tilde{U}_- = iU_+\Sigma_j U_-, \tag{3.65}$$

where in the last equality we write $U_+ \equiv \tilde{U}_+$ and $U_- \equiv \exp(i\Sigma_j\theta_j)\tilde{U}_-$. By substituting (3.65) to (3.63), we obtain

$$\frac{\partial C}{\partial \theta_j} = -i\left(\mathrm{Tr}\left(\rho U^\dagger_-\Sigma_j U^\dagger_+\mathcal{O}U_+U_-\right) - \mathrm{Tr}\left(\rho U^\dagger_- U^\dagger_+\mathcal{O}U_+\Sigma_j U_-\right)\right) \tag{3.66}$$

We can prove the first formula in (3.60) when U_+ or U_- is 2-design. When U_+ is 2-design, the first term becomes

$$\begin{aligned}\int_{\text{2design}} dU_+ \mathrm{Tr}\left(\rho U^\dagger_-\Sigma_j U^\dagger_+\mathcal{O}U_+U_-\right) &= \sum_{b,c,d,e}\int_{\text{2design}} dU_+ \left[U_-\rho U^\dagger_-\Sigma_j\right]_{eb} U^\dagger_{+bc}\mathcal{O}_{cd}U_{+de} \\ &= \frac{1}{2^n}\sum_{b,c,d,e}\delta_{be}\delta_{cd}\left[U_-\rho U^\dagger_-\Sigma_j\right]_{eb}\mathcal{O}_{cd} \\ &= \frac{1}{2^n}\mathrm{Tr}(\mathcal{O})\mathrm{Tr}\left(U_-\rho U^\dagger_-\Sigma_j\right),\end{aligned} \tag{3.67}$$

where in the second equality we use (3.61). Similarly,

$$\int_{\text{2design}} dU_+ \mathrm{Tr}\left(\rho U^\dagger_- U^\dagger_+\mathcal{O}U_+\Sigma_j U_-\right) = \frac{1}{2^n}\mathrm{Tr}(\mathcal{O})\mathrm{Tr}\left(\rho U^\dagger_-\Sigma_j U_-\right). \tag{3.68}$$

Thus from (3.66),

$$\int d\mu(U_{\boldsymbol{\theta}})\frac{\partial C}{\partial \theta_j} = \int dU_- \int_{\text{2design}} dU_+ \frac{\partial C}{\partial \theta_j} = 0. \tag{3.69}$$

When U_- is 2-design the first term in (3.66),

$$\begin{aligned}\int_{\text{2design}} dU_+ \mathrm{Tr}\left(\rho U^\dagger_-\Sigma_j U^\dagger_+\mathcal{O}U_+U_-\right) &= \sum_{b,c,d,e,f}\rho_{fb}U^\dagger_{-bc}\Sigma_{jcd}\left[U^\dagger_+\mathcal{O}U_+\right]_{de}U_{-ef} \\ &= \frac{1}{2^n}\sum_{b,c,d,e,f}\delta_{bf}\delta_{ce}\rho_{fb}\Sigma_{jcd}\left[U^\dagger_+\mathcal{O}U_+\right]_{de} \\ &= \frac{1}{2^n}\mathrm{Tr}\left[\Sigma_j U^\dagger_+\mathcal{O}U_+\right],\end{aligned} \tag{3.70}$$

where in the second equality, we use (3.62). Similarly,

$$\int_{2\text{design}} dU_+ \mathrm{Tr}\left(\rho U_-^\dagger U_+^\dagger \mathcal{O} U_+ \Sigma_j U_-\right) = \frac{1}{2^n}\mathrm{Tr}\left[\Sigma_j U_+^\dagger \mathcal{O} U_+\right], \tag{3.71}$$

and therefore,

$$\int d\mu(U_{\boldsymbol{\theta}})\frac{\partial C}{\partial \theta_j} = \int dU_+ \int_{2\text{design}} dU_- \frac{\partial C}{\partial \theta_j} = 0. \tag{3.72}$$

Thus we successfully derive the first formula in (3.60).

The second formula in (3.60) can also be derived when U_+ or U_- is 2-design. From (3.66) we obtain

$$\left(\frac{\partial C}{\partial \theta_j}\right)^2 = -\left(\mathrm{Tr}\left(\rho U_-^\dagger \Sigma_j U_+^\dagger \mathcal{O} U_+ U_-\right) - \mathrm{Tr}\left(\rho U_-^\dagger U_+^\dagger \mathcal{O} U_+ \Sigma_j U_-\right)\right)^2. \tag{3.73}$$

For the case that the distribution of U_+ is 2-design, the following term can be transformed as

$$\begin{aligned}
&\int_{2\text{design}} dU_+ \left(\mathrm{Tr}\left(\rho U_-^\dagger \Sigma_j U_+^\dagger \mathcal{O} U_+ U_-\right)\right)^2 \\
&= \sum_{b_1,c_1,d_1,e_1,f_1}\sum_{b_2,c_2,d_2,e_2,f_2} \left[U_-\rho U_-^\dagger \Sigma_j\right]_{e_1 b_1} U^\dagger_{+b_1c_1}\mathcal{O}_{c_1d_1}U_{+d_1e_1}\left[U_-\rho U_-^\dagger \Sigma_j\right]_{e_2 b_2} U^\dagger_{+b_2c_2}\mathcal{O}_{c_2d_2}U_{+d_2e_2} \\
&= \sum_{b_1,c_1,d_1,e_1,f_1}\sum_{b_2,c_2,d_2,e_2,f_2} \left[U_-\rho U_-^\dagger \Sigma_j\right]_{e_1 b_1} \mathcal{O}_{c_1d_1}\left[U_-\rho U_-^\dagger \Sigma_j\right]_{e_2 b_2} \mathcal{O}_{c_2d_2} \\
&\times \frac{1}{2^{2n}-1}\left[\left(\delta_{b_1e_1}\delta_{c_1d_1}\delta_{b_2e_2}\delta_{c_2d_2} + \delta_{b_1e_2}\delta_{c_1d_2}\delta_{b_2e_1}\delta_{c_2d_1}\right) - \frac{1}{2^n}\left(\delta_{b_1e_1}\delta_{c_1d_2}\delta_{b_2e_2}\delta_{c_2d_1} + \delta_{b_1e_2}\delta_{c_1d_1}\delta_{b_2e_1}\delta_{c_2d_2}\right)\right] \\
&= \frac{1}{2^{2n}-1}\left\{(\mathrm{Tr}(\mathcal{O}))^2\left(\left(\mathrm{Tr}\left(\rho U_-^\dagger \Sigma_j U_-\right)\right)^2 + O\left(\frac{1}{2^n}\right)\right) + \mathrm{Tr}(\mathcal{O}^2)\left(\mathrm{Tr}\left[\left(\rho U_-^\dagger \Sigma_j U_-\right)^2\right] + O\left(\frac{1}{2^n}\right)\right)\right\},
\end{aligned} \tag{3.74}$$

where in the second equality we use (3.62). Similarly,

$$\begin{aligned}
&\int_{2\text{design}} dU_+ \left(\mathrm{Tr}\left(U_-^\dagger U_+^\dagger \mathcal{O} U_+ \Sigma_j U_-\right)\right)^2 \\
&= \frac{1}{2^{2n}-1}\left\{(\mathrm{Tr}(\mathcal{O}))^2\left(\left(\mathrm{Tr}\left(\rho U_-^\dagger \Sigma_j U_-\right)\right)^2 + O\left(\frac{1}{2^n}\right)\right) + \mathrm{Tr}(\mathcal{O}^2)\left(\mathrm{Tr}\left[\left(\rho U_-^\dagger \Sigma_j U_-\right)^2\right] + O\left(\frac{1}{2^n}\right)\right)\right\}.
\end{aligned} \tag{3.75}$$

The cross term can be transformed as

$$\begin{aligned}
&\int_{2\text{design}} dU_+ \mathrm{Tr}\left(\rho U_-^\dagger \Sigma_j U_+^\dagger \mathcal{O} U_+ U_-\right)\mathrm{Tr}\left(\rho U_-^\dagger U_+^\dagger \mathcal{O} U_+ \Sigma_j U_-\right) \\
&= \sum_{b_1,c_1,d_1,e_1,f_1}\sum_{b_2,c_2,d_2,e_2,f_2} \left[U_-\rho U_-^\dagger \Sigma_j\right]_{e_1 b_1} U^\dagger_{+b_1c_1}\mathcal{O}_{c_1d_1}U_{+d_1e_1}\left[\Sigma_j U_-\rho U_-^\dagger\right]_{e_2 b_2} U^\dagger_{+b_2c_2}\mathcal{O}_{c_2d_2}U_{+d_2e_2} \\
&= \sum_{b_1,c_1,d_1,e_1,f_1}\sum_{b_2,c_2,d_2,e_2,f_2} \left[U_-\rho U_-^\dagger \Sigma_j\right]_{e_1 b_1} \mathcal{O}_{c_1d_1}\left[\Sigma_j U_-\rho U_-^\dagger\right]_{e_2 b_2} \mathcal{O}_{c_2d_2} \\
&\times \frac{1}{2^{2n}-1}\left[\left(\delta_{b_1e_1}\delta_{c_1d_1}\delta_{b_2e_2}\delta_{c_2d_2} + \delta_{b_1e_2}\delta_{c_1d_2}\delta_{b_2e_1}\delta_{c_2d_1}\right) - \frac{1}{2^n}\left(\delta_{b_1e_1}\delta_{c_1d_2}\delta_{b_2e_2}\delta_{c_2d_1} + \delta_{b_1e_2}\delta_{c_1d_1}\delta_{b_2e_1}\delta_{c_2d_2}\right)\right] \\
&= \frac{1}{2^{2n}-1}\left\{(\mathrm{Tr}(\mathcal{O}))^2\left(\left(\mathrm{Tr}\left(\rho U_-^\dagger \Sigma_j U_-\right)\right)^2 + O\left(\frac{1}{2^n}\right)\right) + \mathrm{Tr}(\mathcal{O}^2)\left(\mathrm{Tr}\left(\rho^2 U_-^\dagger \Sigma_j^2 U_-\right) + O\left(\frac{1}{2^n}\right)\right)\right\},
\end{aligned} \tag{3.76}$$

where in the third line we use (3.62). Thus,

$$\begin{aligned}\int dU_- \int_{2\text{design}} dU_+ \left(\frac{\partial C}{\partial \theta_j}\right)^2 &= \frac{2\mathrm{Tr}(\mathcal{O}^2)}{2^{2n}-1}\int dU_- \left\{\mathrm{Tr}\left(\rho^2 U_-^\dagger \Sigma_j^2 U_-\right) - \mathrm{Tr}\left[\left(\rho U_-^\dagger \Sigma_j U_-\right)^2\right]\right\} \\ &\quad + O\left(\frac{\mathrm{Tr}(\mathcal{O})^2}{2^{3n}} + \frac{\mathrm{Tr}(\mathcal{O}^2)}{2^{3n}}\right) \\ &\le O\left(\frac{1}{2^n}\right),\end{aligned} \tag{3.77}$$

where in the last line, we use $\mathrm{Tr}(\mathcal{O}^2) \le O(2^n)$, $\mathrm{Tr}(\mathcal{O})^2 \le O(2^{2n})$, $\mathrm{Tr}\left(\rho^2 U_-^\dagger \Sigma_j^2 U_-\right) \le O(1)$, and $\mathrm{Tr}\left[\left(\rho U_-^\dagger \Sigma_j U_-\right)^2\right] \le O(1)$.

For the case that the distribution of U_- is 2-design, the following term can be transformed as

$$\begin{aligned}&\int_{2\text{design}} dU_- \left(\mathrm{Tr}\left(\rho U_-^\dagger \Sigma_j U_+^\dagger \mathcal{O} U_+ U_-\right)\right)^2 \\ &= \sum_{b_1,c_1,d_1,e_1,f_1}\ \sum_{b_2,c_2,d_2,e_2,f_2} \rho_{e_1 b_1} U^\dagger_{-b_1 c_1} [\Sigma_j U_+^\dagger \mathcal{O} U_+]_{c_1 d_1} U_{-d_1 e_1} \rho_{e_2 b_2} U^\dagger_{-b_2 c_2} [\Sigma_j U_+^\dagger \mathcal{O} U_+]_{c_2 d_2} U_{-d_2 e_2} \\ &= \sum_{b_1,c_1,d_1,e_1,f_1}\ \sum_{b_2,c_2,d_2,e_2,f_2} \rho_{e_1 b_1} [\Sigma_j U_+^\dagger \mathcal{O} U_+]_{c_1 d_1} \rho_{e_2 b_2} [\Sigma_j U_+^\dagger \mathcal{O} U_+]_{c_2 d_2} \\ &\times \frac{1}{2^{2n}-1}\left[\left(\delta_{b_1 e_1}\delta_{c_1 d_1}\delta_{b_2 e_2}\delta_{c_2 d_2} + \delta_{b_1 e_2}\delta_{c_1 d_2}\delta_{b_2 e_1}\delta_{c_2 d_1}\right) - \frac{1}{2^n}\left(\delta_{b_1 e_1}\delta_{c_1 d_2}\delta_{b_2 e_2}\delta_{c_2 d_1} + \delta_{b_1 e_2}\delta_{c_1 d_1}\delta_{b_2 e_1}\delta_{c_2 d_2}\right)\right] \\ &= \frac{1}{2^{2n}-1}\left\{\left(\mathrm{Tr}(\Sigma_j U_+^\dagger \mathcal{O} U_+)\right)^2 \left(1 + O\left(\frac{1}{2^n}\right)\right) + \mathrm{Tr}\left(\rho^2\right)\mathrm{Tr}\left(\left(\Sigma_j U_+^\dagger \mathcal{O} U_+\right)^2\right)\left(1 + O\left(\frac{1}{2^n}\right)\right)\right\},\end{aligned} \tag{3.78}$$

Similary,

$$\begin{aligned}&\int_{2\text{design}} dU_- \langle\Phi| U_-^\dagger U_+^\dagger \mathcal{O} U_+ \Sigma_j U_- |\Phi\rangle\langle\Phi| U_-^\dagger U_+^\dagger \mathcal{O} U_+ \Sigma_j U_- |\Phi\rangle \\ &= \frac{1}{2^{2n}-1}\left(\left(\mathrm{Tr}(\Sigma_j U_+^\dagger \mathcal{O} U_+)\right)^2 \left(1 + O\left(\frac{1}{2^n}\right)\right) + \mathrm{Tr}\left(\rho^2\right)\mathrm{Tr}\left(\left(\Sigma_j U_+^\dagger \mathcal{O} U_+\right)^2\right)\left(1 + O\left(\frac{1}{2^n}\right)\right)\right).\end{aligned} \tag{3.79}$$

The cross term can be transformed as

$$\begin{aligned}&\int_{2\text{design}} dU_- \mathrm{Tr}\left(\rho U_-^\dagger \Sigma_j U_+^\dagger \mathcal{O} U_+ U_-\right)\mathrm{Tr}\left(\rho U_-^\dagger U_+^\dagger \mathcal{O} U_+ \Sigma_j U_-\right) \\ &= \sum_{b_1,c_1,d_1,e_1,f_1}\ \sum_{b_2,c_2,d_2,e_2,f_2} \rho_{e_1 b_1} U^\dagger_{-b_1 c_1} [\Sigma_j U_+^\dagger \mathcal{O} U_+]_{c_1 d_1} U_{-d_1 e_1} \rho_{e_2 b_2} U^\dagger_{-b_2 c_2} [U_+^\dagger \mathcal{O} U_+ \Sigma_j]_{c_2 d_2} U_{-d_2 e_2} \\ &= \sum_{b_1,c_1,d_1,e_1,f_1}\ \sum_{b_2,c_2,d_2,e_2,f_2} \rho_{e_1 b_1} [\Sigma_j U_+^\dagger \mathcal{O} U_+]_{c_1 d_1} \rho_{e_2 b_2} [U_+^\dagger \mathcal{O} U_+ \Sigma_j]_{c_2 d_2} \\ &\times \frac{1}{2^{2n}-1}\left[\left(\delta_{b_1 e_1}\delta_{c_1 d_1}\delta_{b_2 e_2}\delta_{c_2 d_2} + \delta_{b_1 e_2}\delta_{c_1 d_2}\delta_{b_2 e_1}\delta_{c_2 d_1}\right) - \frac{1}{2^n}\left(\delta_{b_1 e_1}\delta_{c_1 d_2}\delta_{b_2 e_2}\delta_{c_2 d_1} + \delta_{b_1 e_2}\delta_{c_1 d_1}\delta_{b_2 e_1}\delta_{c_2 d_2}\right)\right] \\ &= \frac{1}{2^{2n}-1}\left(\left(\mathrm{Tr}(\Sigma_j U_+^\dagger \mathcal{O} U_+)\right)^2 \left(1 + O\left(\frac{1}{2^n}\right)\right) + \mathrm{Tr}\left(\rho^2\right)\mathrm{Tr}(\mathcal{O}^2 U_+^\dagger \Sigma_j^2 U_+)\left(1 + \left(\frac{1}{2^n}\right)\right)\right),\end{aligned} \tag{3.80}$$

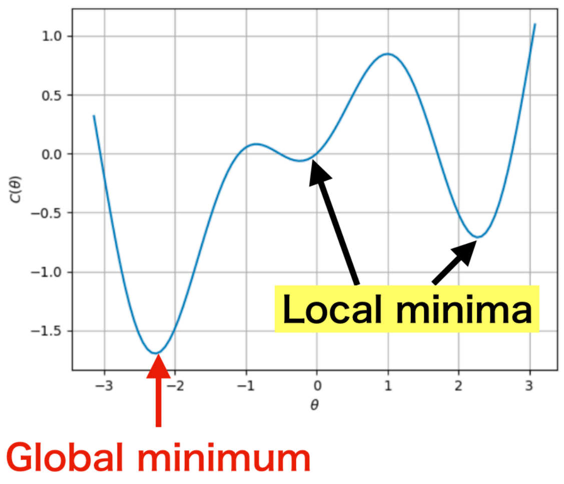


Figure 3.9: A cartoon of the local minima in one dimensional parameter space.

where in the second equality we use (3.62). Thus,

$$\begin{aligned}\int dU_+ \int_{2\text{design}} dU_- \left(\frac{\partial C}{\partial \theta_j}\right)^2 &= \frac{2}{2^{2n}-1}\int dU_+ \left(\mathrm{Tr}(\mathcal{O}^2 U_+^\dagger \Sigma_j^2 U_+) - \mathrm{Tr}\left(\left(\Sigma_j U_+^\dagger \mathcal{O} U_+\right)^2\right)\right)\\ &= O\left(\frac{1}{2^n}\right),\end{aligned} \tag{3.81}$$

where in the last line, we use $\mathrm{Tr}(\mathcal{O}^2 U_+^\dagger \Sigma_j^2 U_+) \le O(2^n)$, $\mathrm{Tr}\left(\left(\Sigma_j U_+^\dagger \mathcal{O} U_+\right)^2\right) \le O(2^{2n})$, and $\mathrm{Tr}\left(\rho^2\right) \le 1$.

3.4.2 Other issues in VQA

There are two issues other than the barren plateau issue that we need a careful treatment of when executing VQA. We discuss the two issues in the following.

The first issue, which is related to the barren plateau issue, is that the landscape of the cost function in VQA may have many local minima. A local minimum is a point that the gradient vector is zero while the value of the cost function is not minimized; the cartoon of the local minima is shown in Fig. 3.9. As long as we use the gradient descendant, the existence of the local minima is troublesome because once the optimization is trapped by the local minima, we need to restart the optimization; therefore, the more the number of local minima is, the more trials are necessary to find the global minimum that minimizes the cost function. Employing several established classical optimizers [106] or using optimization strategy [107] tailored for VQA might be a solution for the local minima issue in some cases, but none of which is currently the comprehensive solution.

The second issue is that the depth of the circuit tends to become larger for reducing the value of cost function below a certain specified value [108], which is an important issue because a deep circuit leads to much noise (recall that one of the most important motivations to introduce VQA is that VQA can be executed with a shallow circuit). There are various attempts to reduce the circuit depth in the previous literature [109–112]. Also, Refs. [113, 114] provide methods to split a large quantum circuit into several small quantum circuits. However, none of them is the perfect solution to this depth issue.

Note that even though the local minima issue and the depth issue are severe, those issues may be less critical than the barren plateau issue for certain algorithms. For example in the quantum machine learning, which we discussed in Section 3.3, does not require the perfect optimization; namely, we do not need to completely minimize the cost function as long as the resulting quantum machine learning model performs well. Thus, in such cases, it may be acceptable that the optimization is trapped by local minima and/or the circuit depth is not enough deep to reduce the cost value to near zero.

On the contrary, the barren plateau issue is critical in every variational algorithms; solving the issue is essential for the practicality of all variational algorithms. In Chapter 6, we focus on the solution to the barren plateau issue.

Chapter 4

Quantum amplitude estimation algorithm tailored for NISQ

As we show in the previous chapter, there is significant noise in NISQ, and we need to reduce the noise for building real-world applications. One approach is reducing the number of gates required for the computation. As an important example of this direction, we discussed VQA in Section 3.3. Still, there are other essential algorithms whose number of gates has been successfully reduced recently. Among those algorithms, the quantum amplitude estimation algorithm (QAE) we discuss in the following has been significantly improved regarding the gate count.

4.1 Quantum amplitude estimation

QAE is the algorithm to estimate the amplitude of a quantum state with good accuracy. Concretely, QAE is the algorithm to estimate the value of ξ in the following equation:

$$|\Psi\rangle \equiv \mathcal{A}|0\rangle^{\otimes n}|0\rangle = \xi|\tilde{\Psi}_1\rangle|1\rangle + \sqrt{1-\xi^2}|\tilde{\Psi}_0\rangle|0\rangle, \tag{4.1}$$

where $|\tilde{\Psi}_0\rangle$ and $|\tilde{\Psi}_1\rangle$ are n-qubit states, $\xi \in [0,1]$, and $\mathcal{A}$ is an quantum operator. QAE is included as a submodule in various quantum algorithms; namely, $\mathcal{A}$ is a quantum algorithm that embeds the computational result into the amplitude of the quantum state, and we need QAE to extract the computational result. One such example of the quantum algorithm is the quantum Monte Carlo integration [17], which embeds the integration result into ξ—in practical applications, executing $\mathcal{A}$ is often costly, thus reducing the number of calling $\mathcal{A}$ while estimating ξ with required accuracy is the heart of the problem.

Previous approaches to the QAE problem

A naive approach for solving the QAE problem is estimating ξ by measuring the $n+1$-th qubit on the computational basis many times. Given the number of calling $\mathcal{A}$ (i.e., the number of measurements) as N_{shot}, the estimation error of ξ is as large as $O(1/\sqrt{N_{\text{shot}}})$.

On the contrary, there is a more efficient approach to estimate ξ. Let us rewrite (4.1) as $|\Psi\rangle = \mathcal{A}|0\rangle^{\otimes n}|0\rangle = \sin\phi|\tilde{\Psi}_1\rangle|1\rangle + \cos\phi|\tilde{\Psi}_0\rangle|0\rangle$. Then, we can define an amplitude amplification operator $G \equiv \mathcal{A}(I_{n+1} - 2|0\rangle_{n+1}\langle 0|_{n+1})\mathcal{A}^\dagger(I_{n+1} - 2I_n \otimes |1\rangle\langle 1|)$ with I_{n+1} as the identity operator operates on $n+1$-qubits and $|0\rangle_{n+1}$ as the $n+1$ tensor product of $|0\rangle$. The operator G operates on $|\Psi\rangle$ as

$$G^m|\Psi\rangle = \sin((2m+1)\phi)|\tilde{\Psi}_1\rangle|1\rangle + \cos((2m+1)\phi)|\tilde{\Psi}_0\rangle|0\rangle. \tag{4.2}$$

Namely, G rotates $|\Psi\rangle$ in the two dimensional space spanned by $|\tilde{\Psi}_0\rangle|0\rangle$ and $|\tilde{\Psi}_1\rangle|1\rangle$. We can easily check that the eigenvalues of G are $\pm 2i\theta$, and the corresponding eigenstates are $|\Psi_\pm\rangle = (|\tilde{\Psi}_1\rangle|1\rangle \pm |\tilde{\Psi}_0\rangle|0\rangle)/\sqrt{2}$. Since $|\Psi\rangle = (|\Psi_+\rangle + |\Psi_-\rangle)/\sqrt{2}$.

$$G|\Psi\rangle = e^{2i\phi}|\Psi_+\rangle + e^{-2i\phi}|\Psi_-\rangle, \tag{4.3}$$

Thus, we transform the QAE problem to the one finding the eigenvalue of the operator G, which can be solved by the phase estimation algorithm introduced in Section 2.3.2. By using the phase estimation, we can estimate the value of the amplitude with the estimation error as $O(1/N_{\mathrm{shot}})$, which is the quadratic improvement from the naive approach. However, as we see in Fig. 2.7 with setting $U = G$, we need many controlled-G operations for the phase estimation, which require many noisy two-qubit gates, and intractable in the near-term devices. Quantum amplitude estimation algorithms executable with much fewer two-qubit gates but achieves the error scaling $O(1/N_{\mathrm{shot}})$ is highly demanded.

Recently, several attempts to reduce the number of gates have been made, all of which utilizes the amplitude amplification operator [20–22]. The first attempt is made by the literature [20]. Literature [20] shows that by measuring $\sin^2((2m+1)\theta)$ in (4.2) with various ms, we can reconstruct the value of the amplitude by using the maximum likelihood estimation without using the controlled-G gates. However, the theoretical proof of the error scaling $O(1/N_{\mathrm{shot}})$ is not given while they give some numerical evidence for it.

Subsequent works [21, 22] also utilize (4.2) and do not require the controlled-G gates. Due to their detailed strategy to choose m, they successfully prove the error scaling $O(1/N_{\mathrm{shot}})$. Among these works, the theoretical upper bounds of the error in [21] is larger than $10^6/N_{\mathrm{shot}}$ when firstly proposed, while in [22] the constant factor is dramatically improved as $O(10^3/N_{\mathrm{shot}})$. In the following, we describe the faster amplitude estimation (FAE) algorithm proposed by the author of this thesis in [22].

4.2 Preliminary

Before going into the detail of the FAE algorithm, let us prepare some operators and functions utilized in the algorithm. The most important technique we introduce here is the amplitude amplification operator.

Attenuation of ξ

In the algorithm of FAE, we use the assumption that ξ is less than or equals to 1/4. However, it is not necessary to impose the condition on ξ since the amplitude can be attenuated by appending an extra auxiliary qubit as follows:

$$\begin{aligned}|\Psi'\rangle \equiv \mathcal{X}|0\rangle|00\rangle &= \frac{\xi}{4}|\tilde{\Psi}_1\rangle|11\rangle + \frac{\sqrt{15}\xi}{4}|\tilde{\Psi}_1\rangle|10\rangle + \frac{\sqrt{1-\xi^2}}{4}|\tilde{\Psi}_0\rangle|01\rangle + \frac{\sqrt{15(1-\xi^2)}}{4}|\tilde{\Psi}_0\rangle|00\rangle. \\ &= \sin\theta|\tilde{\Psi}_1\rangle|11\rangle + \cos\theta|\perp\rangle, \qquad (4.4)\end{aligned}$$

where $\sin\theta \equiv \xi/4$ and $|\perp\rangle$ is a state orthogonal to $|\tilde{\Psi}_1\rangle|11\rangle$, and $\mathcal{X} = \mathcal{A}\otimes\mathcal{R}$ with $\mathcal{R}$ as a single qubit rotational gate, which operates as:

$$\mathcal{R}|0\rangle = \frac{1}{4}|1\rangle + \frac{\sqrt{15}}{4}|0\rangle. \qquad (4.5)$$

As expected, the amplitude is attenuated such that $\sin\theta \in [0, 1/4]$, and without loss of generality, we can assume

$$0 \le \theta < 0.252. \qquad (4.6)$$

Thus, instead of estimating the value of ξ directly, FAE estimates the value of θ and converts it to ξ. The condition (4.6) is utilized as the initial bound in FAE.

Amplitude amplification

FAE utilizes a variant of the amplitude amplification operator $\mathbf{Q}$ that amplifies the amplitude whose last two qubits are both 1. Concretely, the operator $\mathbf{Q}$ is defined by

$$\mathbf{Q} \equiv \mathcal{X}U_0\mathcal{X}^\dagger U_{11}, \qquad (4.7)$$

$$U_0 = (I_{n+2} - 2|0\rangle_{n+2}\langle 0|_{n+2}), \qquad (4.8)$$

$$U_{11} = (I_{n+2} - 2I_n \otimes |11\rangle\langle 11|), \qquad (4.9)$$

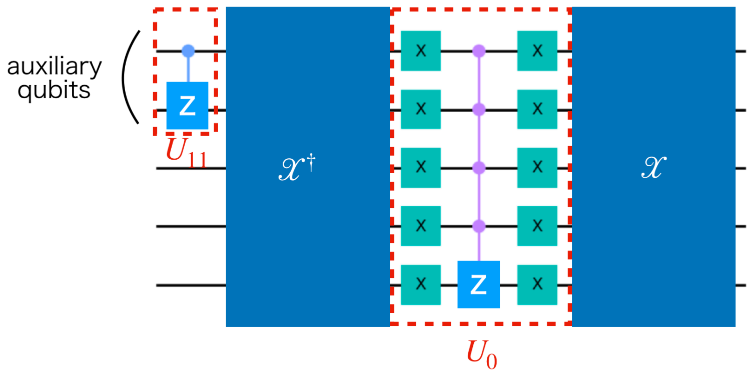


Figure 4.1: The circuit for $\mathbf{Q}$ when the number of qubits is five.

where I_n is the identity operator that operates to an n-qubit state. We show an example of the circuit for $\mathbf{Q}$ when the number of qubits is five in Fig. 4.1. The direct calculation shows

$$\mathbf{Q}^m|\Psi'\rangle = \sin(2m+1)\theta|\tilde{\Psi}_1\rangle_n|11\rangle + \cos(2m+1)\theta|\perp\rangle. \tag{4.10}$$

We get the estimates of $\cos(2(2m+1)\theta)$ by measuring the state (4.10) for multiple m. Let us define c_m as

$$c_m \equiv 1 - 2\frac{N_{11}}{N_{\text{shot}}}, \tag{4.11}$$

where N_{11} is the number of the results of the measurements in which the last two qubits in (4.10) are both one and N_{shot} is the total number of measurements of the state (4.10). The above defined c_m is a good estimate of $\cos(2(2m+1)\theta)$; the estimation error of c_m can be evaluated by using the Chernoff bound for the Bernoulli distribution, i.e., given the confidence interval of c_m as $[c_m^{\min}, c_m^{\max}]$, the bounds of the interval are computed as

$$c_m^{\max} = \min\left[1, c_m + \sqrt{\ln\left(\frac{2}{\delta_c}\right)\frac{12}{N_{\text{shot}}}}\right], \qquad c_m^{\min} = \max\left[-1, c_m - \sqrt{\ln\left(\frac{2}{\delta_c}\right)\frac{12}{N_{\text{shot}}}}\right], \tag{4.12}$$

where δ_c is the probability that the true value of c_m (i.e. $\cos(2(2m+1)\theta)$) is out of the interval.

Other functions

For later purpose, let us also define following three functions: $\mathbf{COS}(m, N_{\text{shot}})$, $\mathbf{CHERNOFF}(c_m, N_{\text{shot}}, \delta_c)$, and $\text{atan}(s, c)$. The function $\mathbf{COS}(m, N_{\text{shot}})$ returns c_m as a result of N_{shot} measurements of the amplified state (4.10). The function $\mathbf{CHERNOFF}(c_m, N_{\text{shot}}, \delta_c)$ returns the confidence interval $[c_m^{\min}, c_m^{\max}]$ of c_m, which is computed from the parameters: c_m, N_{shot}, and δ_c. The function $\text{atan}(s, c)$ is an extended arctangent function defined in the realm $c, s \in [-1, 1]$ by

$$\text{atan}(s,c) = \begin{cases} \arctan(s/c) & (c > 0) \\ \pi/2 & (c = 0, s > 0) \\ 0 & (c = 0, s = 0) \\ -\pi/2 & (c = 0, s < 0) \\ \pi + \arctan(s/c) & (c < 0, s \geq 0) \\ -\pi + \arctan(s/c) & (c < 0, s < 0). \end{cases} \tag{4.13}$$

Finally, we denote the number of calls of $\mathbf{Q}$ required for estimating θ by N_{orac}. The goal of FAE is estimating θ with required accuracy while reducing the number of N_{orac}.

4.3 Faster amplitude estimation algorithm

Now let us describe the algorithm of FAE. FAE is the method to estimate the amplitude by utilizing the amplitude amplification operator effectively. Even though the previous works [20, 21] also utilizes the amplitude amplification operator, the way of using the operator in FAE is different from those in the previous works, which leads to the major difference in the algorithms.

Suppose that $[\theta_{\text{min}}^j, \theta_{\text{max}}^j]$ as the confidence interval of θ in j-th iteration, the algorithm updates the values of θ_{max}^j and θ_{min}^j so that the width $\theta_{\text{max}}^j - \theta_{\text{min}}^j$ becomes smaller in each iteration. Users of the algorithm can choose the total iteration count ℓ; ℓ should be chosen so that the final estimation result satisfies the required accuracy. As we see later, given an acceptable error of the amplitude as ϵ, $\epsilon \sim 1/2^\ell$ holds. Therefore, it is suffice to take ℓ as $\ell \sim \log_2(1/\epsilon)$.

Algorithm 1 Faster Amplitude Estimation (δ_c and ℓ as the parameters)

```
1: #$\theta_{\text{min}}^j$ and $\theta_{\text{max}}^j$: the confidence interval of $\theta$ in $j$-th iteration.
2: Set $\theta_{\text{min}}^0$ to 0 and $\theta_{\text{max}}^0$ to 0.252.
3: Set $N_{\text{shot}}^{1st} = 1944 \ln\left(\frac{2}{\delta_c}\right)$ and $N_{\text{shot}}^{2nd} = 972 \ln\left(\frac{2}{\delta_c}\right)$.
4: Set FIRST_STAGE to true.
5: Set $j_0$ to $\ell$.
6: for $j = 1$ to $\ell$ do
7:     if FIRST_STAGE then
8:         Set $c_{2^{j-1}}$ to $\textbf{COS}(2^{j-1}, N_{\text{shot}}^{\text{1st}})$.
9:         Set $c_{2^{j-1}}^{\text{min}}, c_{2^{j-1}}^{\text{max}}$ to $\textbf{CHERNOFF}(c_{2^{j-1}}, N_{\text{shot}}^{\text{1st}}, \delta_c)$.
10:        Set $\theta_{\text{max}}^j = \arccos(c_{2^{j-1}}^{\text{min}})/(2^{j+1}+2)$ and $\theta_{\text{min}}^j = \arccos(c_{2^{j-1}}^{\text{max}})/(2^{j+1}+2)$.
11:        if $2^{j+1}\theta_{\text{max}}^j \geq \frac{3\pi}{8}$ and $j < \ell$ then
12:            Set $j_0$ to $j$.
13:            Set $\nu = 2^{j_0}(\theta_{\text{max}}^{j_0} + \theta_{\text{min}}^{j_0})$ # the estimate of $2^{j_0+1}\theta$
14:            Set FIRST_STAGE to false.
15:        end if
16:    else
17:        Set $c_{2^{j-1}}$ to $\textbf{COS}(2^{j-1}, N_{\text{shot}}^{\text{2nd}})$.
18:        Set $s_{2^{j-1}}$ to $(c_{2^{j-1}} \cos\nu - \textbf{COS}(2^{j-1} + 2^{j_0-1}, N_{\text{shot}}^{\text{2nd}}))/\sin\nu$.
19:        Set $\rho_j = \text{atan}\left(s_{2^{j-1}}, c_{2^{j-1}}\right)$.
20:        Set $n_j$ to $[\frac{1}{2\pi}\left((2^{j+1}+2)\theta_{\text{max}}^{j-1} - \rho_j + \pi/3\right)]$ where $[x]$ is the largest integer which does not exceed $x$.
21:        Set $\theta_{\text{min}}^j = (2\pi n_j + \rho_j - \pi/3)/(2^{j+1}+2)$ and $\theta_{\text{max}}^j = (2\pi n_j + \rho_j + \pi/3)/(2^{j+1}+2)$.
22:    end if
23: end for
return $(\theta_{\text{min}}^\ell + \theta_{\text{max}}^\ell)/2$, estimate of $\theta$ where the probability that $\theta \in [\theta_{\text{min}}^j, \theta_{\text{max}}^j]$ is larger than $1-(2\ell-j_0)\delta_c$.
```

There are two stages in the FAE algorithm; the estimation procedures are different in each stage. At the beginning of the iteration ($j = 1$), the algorithm is in the first stage. The algorithm may change into the second stage in later iterations if a condition is satisfied. The overview of the algorithm is shown in **Algorithm 1**[1]. We show the detail of each stage in the following. There are a typical number of measurements for each stage: N_{shot}^{1st} in the first stage and N_{shot}^{2nd} in the second stage.

Note that even though θ is not always inside the confidence interval: $[\theta_{\text{min}}^j, \theta_{\text{max}}^j]$, the probability exponentially decreases as N_{shot}^{1st} and N_{shot}^{2nd} increases. Thus, for simplicity, only the case where $\theta \in [\theta_{\text{min}}^j, \theta_{\text{max}}^j]$ holds for all js is discussed here. As we will see later, the probability that $\theta \in [\theta_{\text{min}}^j, \theta_{\text{max}}^j]$ holds for all j is larger than $1 - 2\ell\delta_c$.

[1]The source code of the algorithm is shown in https://github.com/quantum-algorithm/faster-amplitude-estimation.

First Stage

The algorithm is in the first stage either (a) when $j = 1$ or (b) when $j > 1$ and all $2^{k+1}\theta^k_{\max}(k = 1 \dots j-1)$ satisfy $2^{k+1}\theta^k_{\max} < \frac{3\pi}{8}$. In this stage, $\theta^j_{\min}, \theta^j_{\max}$ is gotten by inverting $c^{\min}_{2^{j-1}}$ and $c^{\max}_{2^{j-1}}$ as

$$\theta^j_{\max} = \frac{\arccos(c^{\min}_{2^{j-1}})}{2^{j+1}+2}, \qquad \theta^j_{\min} = \frac{\arccos(c^{\max}_{2^{j-1}})}{2^{j+1}+2}. \tag{4.14}$$

That is because $(2^{j+1}+2)\theta < \pi$ is guaranteed as we see in the following discussion. In case of $j = 1$, the bound (4.6) leads to $(2^{1+1}+2)\theta < 1.52 < \pi$. In another case where $j > 1$ and $2^{k+1}\theta^k_{\max} < \frac{3\pi}{8}$ for $(k = 1 \dots j-1)$, the following holds

$$(2^{j+1}+2)\theta < 2(2^j\theta^{j-1}_{\max}) + 2\theta < 3/4\pi + 0.504 < \pi. \tag{4.15}$$

The algorithm changes into the second stage if $2^{j+1}\theta^j_{\max} \geq 3\pi/8$. At the timing, the following two values are memorized for the second stage; one is j_0 defined as the last value of j in the first stage and another is ν defined as

$$\nu = 2^{j_0+1} \times \frac{\theta^{j_0}_{\max} + \theta^{j_0}_{\min}}{2}. \tag{4.16}$$

It should be noted that the above-defined ν is an estimate of $2^{j_0+1}\theta$ whose confidence interval is obtainable by using the Chernoff bound.

In case that $2^{j+1}\theta^j_{\max}$ is less than $3\pi/8$ for all $j(< \ell)$, the algorithm finishes without going to the second stage and the output of FAE is $(\theta^\ell_{\max} + \theta^\ell_{\min})/2$; j_0 is set to be $j_0 = \ell$. In the case, the error of the output is at most $\Delta\theta \equiv (\theta^\ell_{\max} - \theta^\ell_{\min})/2 = (\arccos(c^{\min}_{2^{\ell-1}}) - \arccos(c^{\max}_{2^{\ell-1}}))/(2^{\ell+2}+4)$. As a result, the error of the amplitude is bounded as

$$\epsilon = 4\left(\sin(\theta + \Delta\theta) - \sin\theta\right) < 4\Delta\theta < \frac{\arccos(c^{\min}_{2^{\ell-1}}) - \arccos(c^{\max}_{2^{\ell-1}})}{2^\ell} \tag{4.17}$$

as long as θ is inside the confidence interval for all j, whose probability is computed as $(1-\delta_c)^\ell > 1 - \ell\delta_c (= 1 - (2\ell - j_0)\delta_c)$.

Second Stage

In the second stage, $(2^{j+1}+2)\theta$ might be larger than π, and therefore, the value of $(2^{j+1}+2)\theta$ can not be estimated only by inverting $c_{2^{j-1}}$ due to the ambiguity of arc-cosine function. However, it is still possible to estimate the value of $(2^{j+1}+2)\theta$ by utilizing the results of measurements in the other angle: $(2^{j+1}+2^{j_0+1}+2)\theta$. Here, let us firstly show how to estimate $(2^{j+1}+2)\theta|_{\mathrm{mod}2\pi}$ and next let us show how to estimate $(2^{j+1}+2)\theta$ without mod(2π) ambiguity.

(i) The estimate of $(2^{j+1}+2)\theta|_{\mathrm{mod}2\pi}$

For estimating $(2^{j+1}+2)\theta|_{\mathrm{mod}2\pi}$, it is necessary to obtain not only the estimate of $\cos((2^{j+1}+2)\theta)$ (i.e., $c_{2^{j-1}}$) but also the estimate of $\sin((2^{j+1}+2)\theta)$. We can not obtain the estimate of $\sin((2^{j+1}+2)\theta)$ directly from measurement results, but it can be computed by the following process. From the trigonometric addition formula,

$$\cos((2^{j+1}+2^{j_0+1}+2)\theta) = \cos((2^{j+1}+2)\theta)\cos(2^{j_0+1}\theta) - \sin((2^{j+1}+2)\theta)\sin(2^{j_0+1}\theta). \tag{4.18}$$

As long as $\sin(2^{j_0+1}\theta)$ is non-zero,

$$\sin((2^{j+1}+2)\theta) = \frac{\cos((2^{j+1}+2)\theta)\cos(2^{j_0+1}\theta) - \cos((2^{j+1}+2^{j_0+1}+2)\theta)}{\sin(2^{j_0+1}\theta)}. \tag{4.19}$$

By replacing $\cos((2^{j+1}+2)\theta$ to $c_{2^{j-1}}$, $2^{j_0+1}\theta$ to ν and $\cos((2^{j+1}+2^{j_0+1}+2)\theta$ to $c_{2^{j-1}+2^{j_0+1}}$ in the right hand side of (4.19), we define $s_{2^{j-1}}$ by

$$s_{2^{j-1}} = \frac{c_{2^{j-1}}\cos\nu - c_{2^{j-1}+2^{j_0-1}}}{\sin\nu}, \tag{4.20}$$

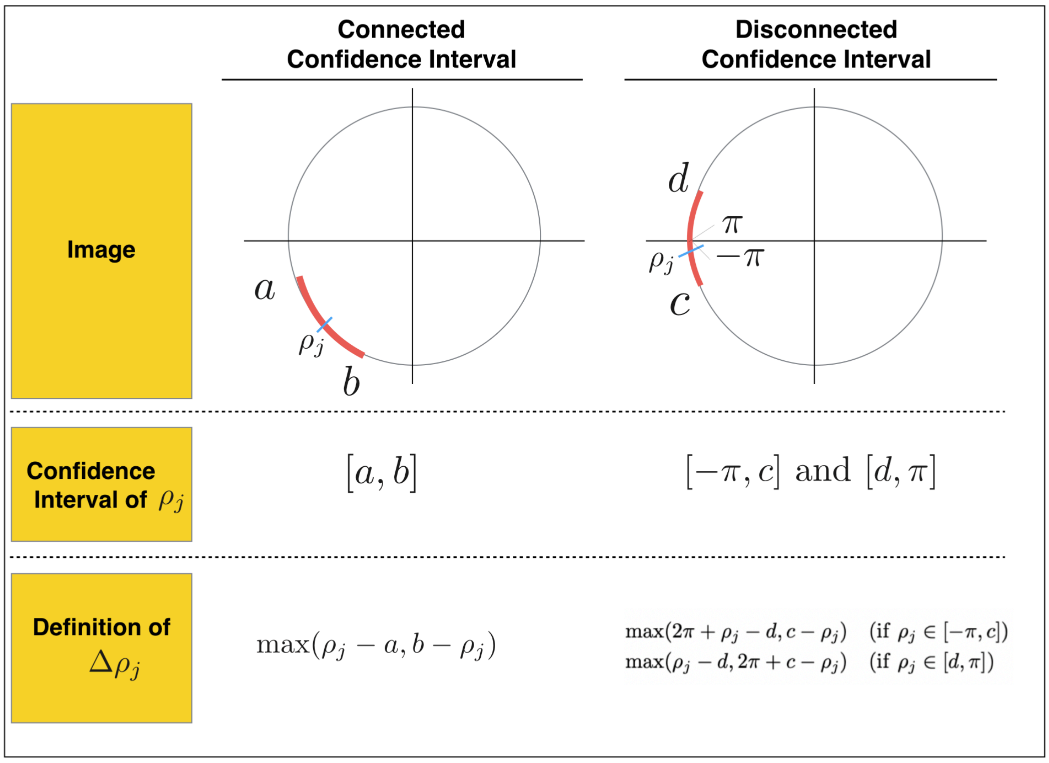

Connected Confidence Interval
Disconnected Confidence Interval
Image
a
ρ_j
b
d
π
ρ_j
$-\pi$
c
Confidence Interval of ρ_j
$[a,b]$
$[-\pi,c]$ and $[d,\pi]$
Definition of $\Delta\rho_j$
$\max(\rho_j - a, b-\rho_j)$
$\max(2\pi+\rho_j-d, c-\rho_j)$ (if $\rho_j \in [-\pi,c]$)
$\max(\rho_j-d, 2\pi+c-\rho_j)$ (if $\rho_j \in [d,\pi]$)

Figure 4.2: The overview of the definition of $\Delta\rho_j$. Reprinted figure from [DOI: 10.26421/QIC20.13-14-2]. Copyright 2020 by the Rinton press. The author is permitted to redistribute the figure.

which becomes an estimate of $\sin((2^{j+1}+2)\theta)$. The estimation error of s_{2^j-1} is determined by the estimation errors of c_{2^j-1}, $c_{2^{j-1}+2^{j_0-1}}$ and ν, which is discussed in Appendix A.1. Straightforwardly, we get the estimate of $(2^{j+1}+2)\theta|_{\text{mod}2\pi}$ from s_{2^j-1} and c_{2^j-1}; the $\rho_j \in [-\pi,\pi]$ defined by

$$\rho_j = \text{atan}\left(s_{2^j-1}, c_{2^j-1}\right) \tag{4.21}$$

is an estimate of $(2^{j+1}+2)\theta|_{\text{mod}2\pi}$.

The confidence interval of ρ_j can be derived from those of c_{2^j-1}, $c_{2^{j-1}+2^{j_0-1}}$ and ν as in s_{2^j-1}. There are two types of the confidence interval. One is the connected confidence interval; there is no discontinuities in the confidence interval, e.g., $[-\pi/3,\pi/4]$. The other is the disconnected confidence interval; the confidence interval is separated into an interval containing $-\pi$ and an interval containing π, e.g., $[-\pi,-2\pi/3]$ and $[3\pi/4,\pi]$. The discontinuity arises when the confidence interval of c_{2^j-1} contains -1 and that of s_{2^j-1} contains 0[2]. In the connected confidence interval case, given interval as $[a,b]$, let us define define $\Delta\rho_j = \max(\rho_j-a, b-\rho_j)$. In the disconnected confidence interval case, given intervals as $[-\pi,c]$ and $[d,\pi]$, let us define $\Delta\rho_j$ as

$$\Delta\rho_j = \begin{cases} \max(2\pi+\rho_j-d, c-\rho_j) & (\text{if } \rho_j \in [-\pi,c]) \\ \max(\rho_j-d, 2\pi+c-\rho_j) & (\text{if } \rho_j \in [d,\pi]) \end{cases}. \tag{4.22}$$

We show the conceptual image of the connected/disconnected intervals and that $\Delta\rho_j$ in Fig. 4.1. We can interpret above defined $\Delta\rho_j$ as the estimation error of ρ_j in a sense that

$$2\pi n_j + \rho_j - \Delta\rho_j \le (2^{j+1}+2)\theta \le 2\pi n_j + \rho_j + \Delta\rho_j \tag{4.23}$$

[2]If both the confidence intervals of c_{2^j-1} and s_{2^j-1} contain 0, there is a discontinuity in the confidence interval of ρ_j at $\rho_j = \pm\pi/2$. However, as long as we set N^{1st}_{shot} and N^{2nd}_{shot} enoughly large as the upper bound value derived in Appendix A.1, the estimation errors of c_{2^j-1} and s_{2^j-1} are suppressed so that either the confidence interval of c_{2^j-1} or that of s_{2^j-1} does not contain 0 (recall that $(c_{2^j-1})^2 + (s_{2^j-1})^2 \simeq 1$ holds when errors are suppressed). Thus, we do not discuss this type of discontinuity in the following argument.

holds with an unknown integer n_j if the true value of ρ_j (i.e. $(2^{j+1}+2)\theta|_{\mod 2\pi}$) is inside the confidence interval.

(ii)The estimate of $(2^{j+1}+2)\theta$

Next we show how to estimate $(2^{j+1}+2)\theta$ from ρ_j. By using (4.23) and the following inequality,

$$(2^{j+1}+2)\theta_{\text{min}}^{j-1} \le (2^{j+1}+2)\theta \le (2^{j+1}+2)\theta_{\text{max}}^{j-1}, \tag{4.24}$$

it holds that

$$(2^{j+1}+2)\theta_{\text{min}}^{j-1} - \rho_j - \Delta\rho_j \le 2\pi n_j \le (2^{j+1}+2)\theta_{\text{max}}^{j-1} - \rho_j + \Delta\rho_j. \tag{4.25}$$

Thus, as long as

$$(2^{j+1}+2)(\theta_{\text{max}}^{j-1} - \theta_{\text{min}}^{j-1}) + 2\Delta\rho_j < 2\pi \tag{4.26}$$

is satisfied, we can uniquely determine the integer n_j as

$$n_j = \frac{1}{2\pi}[(2^{j+1}+2)\theta_{\text{max}}^{j-1} - \rho_j + \Delta\rho_j], \tag{4.27}$$

where the symbol $[x]$ denotes the largest integer that does not exceed x. By using (4.24) and (4.27), we can inductively show that if all $\rho_k (k = j_0+1 \ldots j-1)$ are determined with the precision of $\Delta\rho_k \le \pi/3$ then the condition (4.26) is satisfied.

As we see above, (4.23) with n_j in (4.27) gives the upper/lower bounds of $(2^{j+1}+2)\theta$, but we do not need to evaluate $\Delta\rho_j$ inside the algorithm. Instead, in FAE, we set the upper/lower bounds of θ at the j-th iteration as

$$\theta_{\text{min}}^{j} = \frac{2\pi n_j + \rho_j - \pi/3}{2^{j+1}+2}, \qquad \theta_{\text{max}}^{j} = \frac{2\pi n_j + \rho_j + \pi/3}{2^{j+1}+2}, \tag{4.28}$$

and

$$n_j = \frac{1}{2\pi}[(2^{j+1}+2)\theta_{\text{max}}^{j-1} - \rho_j + \pi/3], \tag{4.29}$$

which are correct as far as $\Delta\rho_j \le \pi/3$. In Appendix A.1, we show that for all $j(> j_0)$, the conditions $\Delta\rho_j \le \pi/3$ and (4.23) are satisfied with the probability larger than $1-(2\ell - j_0)\delta_c$ when at least we choose the

$$N_{\text{shot}}^{1st} = 1944 \ln\left(\frac{2}{\delta_c}\right), \qquad N_{\text{shot}}^{2nd} = 972 \ln\left(\frac{2}{\delta_c}\right). \tag{4.30}$$

In the ℓ-th iteration, the output of FAE is given by $(\theta_{\text{max}}^{\ell} + \theta_{\text{min}}^{\ell})/2$. Then, the estimation error of the output $\Delta\theta$ is less than $\Delta\theta = (\theta_{\text{max}}^{\ell} - \theta_{\text{min}}^{\ell})/2 \le \pi/(3\cdot 2^{\ell+1})$. Thus, we estimate the error of the amplitude as

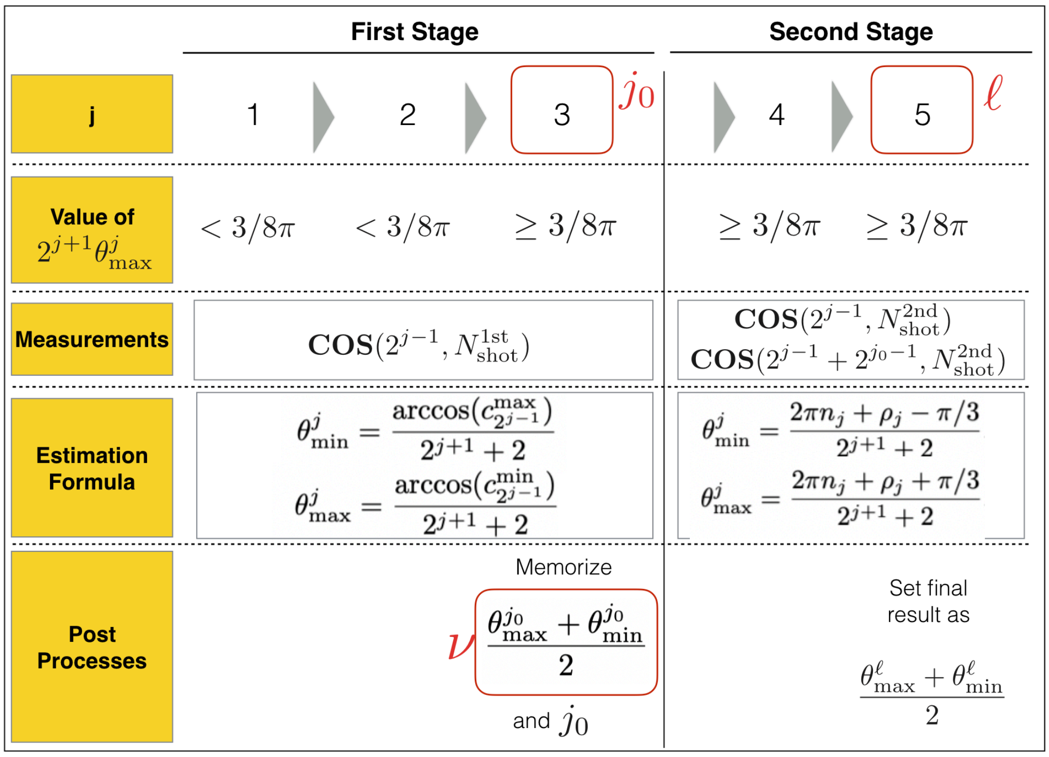
$$\epsilon = 4\left(\sin(\theta+\Delta\theta) - \sin\theta\right) < 4\Delta\theta < \frac{\pi}{3\cdot 2^{\ell-1}}. \tag{4.31}$$

The overview of FAE when $\ell = 5$ and $j_0 = 3$ is shown in Fig. 4.3.

Complexity upper bound

In Appendix A.1, we prove that the query complexity N_{orac} with which the estimation error of ξ is less than ϵ with the probability less than δ is bounded as

$$N_{\text{orac}} < \frac{4.1\cdot 10^3}{\epsilon} \ln\left(\frac{4\log_2(2\pi/3\epsilon)}{\delta}\right). \tag{4.32}$$



	First Stage			Second Stage	
j	1	2	$\boxed{3}$ j_0	4	$\boxed{5}$ ℓ
Value of $2^{j+1}\theta^j_{\max}$	$< 3/8\pi$	$< 3/8\pi$	$\geq 3/8\pi$	$\geq 3/8\pi$	$\geq 3/8\pi$
Measurements	$\mathrm{COS}(2^{j-1}, N^{\mathrm{1st}}_{\mathrm{shot}})$			$\mathrm{COS}(2^{j-1}, N^{\mathrm{2nd}}_{\mathrm{shot}})$ $\mathrm{COS}(2^{j-1} + 2^{j_0-1}, N^{\mathrm{2nd}}_{\mathrm{shot}})$	
Estimation Formula	$\theta^j_{\min} = \frac{\arccos(c^{\max}_{2^j-1})}{2^{j+1}+2}$ $\theta^j_{\max} = \frac{\arccos(c^{\min}_{2^j-1})}{2^{j+1}+2}$			$\theta^j_{\min} = \frac{2\pi n_j + \rho_j - \pi/3}{2^{j+1}+2}$ $\theta^j_{\max} = \frac{2\pi n_j + \rho_j + \pi/3}{2^{j+1}+2}$	
Post Processes			Memorize ν $\boxed{\frac{\theta^{j_0}_{\max} + \theta^{j_0}_{\min}}{2}}$ and j_0	Set final result as $\frac{\theta^{\ell}_{\max} + \theta^{\ell}_{\min}}{2}$	

Figure 4.3: The overview of FAE when $\ell = 5$ and $j_0 = 3$. Reprinted figure from [DOI: 10.26421/QIC20.13-14-2]. Copyright 2020 by the Rinton press. The author is permitted to redistribute the figure.

The worst case is when the algorithm moves to the second stage at the first iteration (when $j = 1$). We see that the upper bound of N_{orac} achieves the scaling: $N_{\mathrm{orac}} \propto 1/\epsilon$ (recall that the dependency of the factor $\ln(\log_2(\pi/\epsilon))$ on ϵ is small, e.g., even if $\epsilon = 10^{-20}$, the factor is at most 6).

Here let us show a brief sketch of the proof regarding why the upper bound is proportional to $1/\epsilon$. For ϵ to be suppressed as (4.31), it is suffice that the errors of all c_{2^j-1}s used in FAE are less than $1/9\sqrt{2}$, which is realized if $N_{\mathrm{shot}} \sim O(1000 \log(1/\delta))$ for each j. The number of calling $\mathcal{Q}$ in each j is about 2^{j-1} for each measurement. Therefore, $N_{\mathrm{orac}} \sim N_{\mathrm{shot}} \sum_{j=1}^{j=\ell} 2^{j-1} = N_{\mathrm{shot}} 2^{\ell} \propto N_{\mathrm{shot}}/\epsilon$ as we expected.

Number of controlled gates

Given the estimation error as $O(1/2^{\ell})$, the phase estimation algorithm needs the control-$Q^{2^{\ell}}$ operations, which requires many multi-Toffoli gates. On the contrary, the circuit in FAE includes $Q^{2^{\ell}}$ operation in the ℓ-th step, and therefore the only controlled gate needed is the one included in Q; in FAE, the number of controlled gates is dramatically reduced.

Still, the number of controlled gates in Q is harmful; particularly, even if the operator for a quantum algorithm $\mathcal{A}$ is realizable with a shallow circuit, we need a multi-Toffoli gate for realizing U_0. Thus, reducing the number of controlled gates in $\mathbf{Q}$ is important future work.

4.4 Numerical Demonstration

Now let us demonstrate FAE by numerical experiments. The amplitudes estimated are chosen to be $\xi = 0.1,\ 0.2,\ 0.3,\ 0.4$, and δ_c is taken as 0.01. The query complexity N_{orac} and the estimation error ϵ are computed with changing ℓ; in each choice of (ξ, ℓ), we execute 1000 trials of the algorithm.

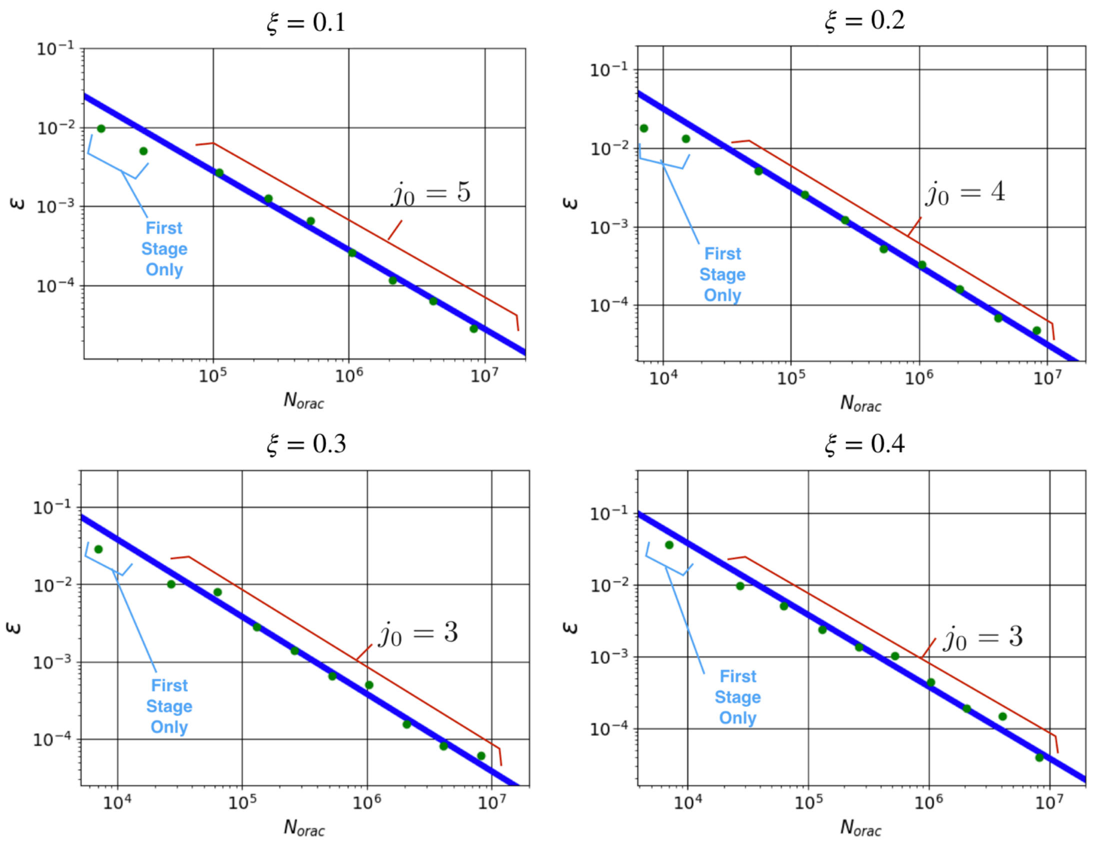


Figure 4.4: The estimation error (green dot) is plotted so that 95% of the estimation errors in 1000 trials are equals to or smaller than the plotted value. In the same figure, the value of j_0 is shown. We write "First Stage Only" instead of writing the value of j_0 for the data points where the algorithm does not go to the second stage. We fit the data points with $\log_{10}(N_{\mathrm{orac}}) = -\log_{10}(\epsilon) + b$ (blue lines) where we determine the fitting parameter b by the least-squares. Reprinted figure from [DOI: 10.26421/QIC20.13-14-2]. Copyright 2020 by the Rinton press. The author is permitted to redistribute the figure.

We show the computation results in Fig. 4.4. The horizontal axis is the value of N_{orac} and the vertical axis is the estimation error ϵ. For each N_{orac}, the estimation error (green dot) is plotted so that 95% of the estimation errors in 1000 trials are equals to or smaller than the plotted value. In the same figure, the value of j_0 is shown. We write "First Stage Only" instead of writing the value of j_0 for the data points where the algorithm does not go to the second stage. We fit the data points with $\log_{10}(N_{\mathrm{orac}}) = -\log_{10}(\epsilon) + b$ (blue lines) where we determine the fitting parameter b is by the least-squares.

We summarize the notable points in the following:

- The scaling $N_{\mathrm{orac}} \leq C \times 1/\epsilon$ is almost achieved since almost all green dots are on the blue lines.
- In case that the algorithm does not go to the second stage, the error tends to be below the blue line, i.e., the required number of N_{orac} is smaller for fixed $1/\epsilon$ than that in the case when the algorithm goes to the second stage.
- The larger ξ becomes, the smaller j_0 becomes. The reason is that as ξ increases, $2^{j+1}\theta_{\max} \geq 3/8\pi$ is satisfied with smaller j.

Chapter 5

Quantum semi-supervised generative adversarial network as an application of NISQ

In Section 3.3, we discuss the quantum generative adversarial network (QGAN) as a variant of the variational quantum algorithm. In classical machine learning, GAN is most successfully applied to the semi-supervised learning tasks [44–46]. Even in the quantum setting, the quantum generative adversarial network can be used for semi-supervised learning (SSL) tasks. The quantum semi-supervised generative adversarial network (qSGAN), proposed by the author of this thesis is the first algorithm that successfully applies QGAN to semi-supervised problems. In this chapter, we describe qSGAN, the algorithm for solving the SSL tasks in NISQ. In the following, we will first discuss the basics of SSL, and next, we will discuss the detail of qSGAN.

SSL is one of the most important machine learning tasks that use a dataset containing both labeled and unlabeled data as the training dataset. From those data, a model function $f_{\boldsymbol{\theta}}^{\text{model}}$ is optimized as it correctly predicts the label of a new data. The merit of using the unlabeled data is that the result of SSL is likely to become more robust than that of supervised learning. For example, unlabeled data clarifies the underlying data distribution in classification cases. As we demonstrate in Fig. 5.1, it contributes to getting a better decision boundary of the classifier, which is the boundary between different classes. The nature of SSL that leverages unlabeled data to obtain better performance is preferable, especially when getting labeled data is costly.

GAN is successfully applied to semi-supervised learning tasks. Originally the idea of using GAN in SSL was proposed in Ref. [44] and since then various studies [45–51] have been proposed. Same as the standard GAN discussed in Section 3.3.3, GAN in semi-supervised learning (SGAN) contains a generator and a discriminator, yet the role of the discriminator in SGAN is different from that in GAN. Namely, the discriminator not only classifies the input data as real or fake but also estimates the label of the data; the goal of SGAN is training the discriminator so that it becomes a good classifier, rather than training the generator. Notably, SGAN achieves competitive results in various SSL tasks [44–51]. The reason GAN contributes to the classification quality is that the samples generated by the generator complement the training data so that the discriminator finds a better decision boundary [50]; thus the generator should have a rich expressibility power for effectively complementing the training data.

5.1 Algorithm of qSGAN

Quantum circuits are expected to have more expressive power than the classical models; the quantum supremacy experiment [115] takes advantage of the rich expressibility of the quantum circuit. Thus, by changing the generator to the quantum generator in SGAN, we may obtain a better classifier as a result of the training. In the following, we describe the algorithm of qSGAN based on the original proposal [52], which is a contribution of the author of this thesis. Note

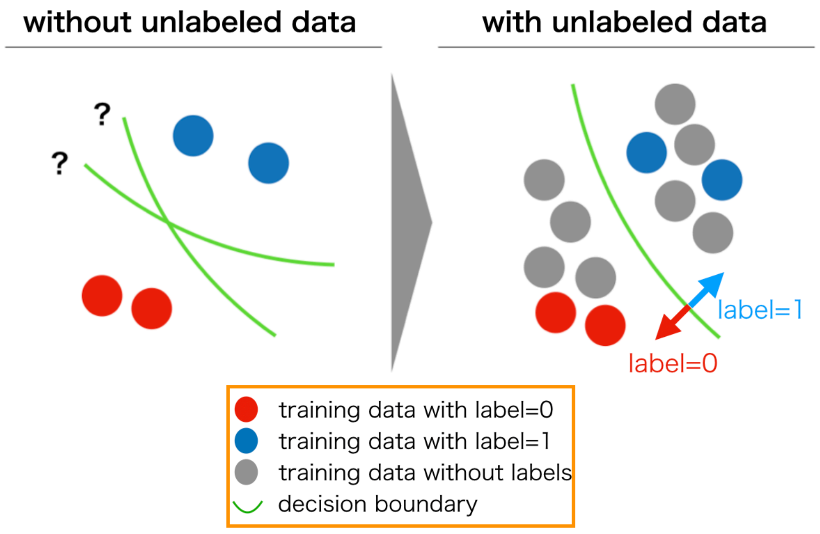

without unlabeled data
with unlabeled data
?
?
label=1
label=0
training data with label=0
training data with label=1
training data without labels
decision boundary

Figure 5.1: The benefit of using unlabeled data in SSL. We show the classification problem with two labels: red and blue. Depending on the label, labeled data is shown by red or blue circles. A gray circle shows unlabeled data. Green curves show possible decision boundaries. The left figure is the case when not using unlabeled data, and the right figure is when using unlabeled data. Without unlabeled data, the data distribution is unclear, and it is not easy to obtain a good decision boundary of the classifier. The data distribution becomes clearer with labeled data, and we get a better decision boundary.

that even though the idea of using the quantum generator is already proposed in QGAN, as we see in Section 3.3, qSGAN is the first algorithm that applies the quantum generator to the SSL problems.

Training data

Suppose that we can separate the labeled/unlabeled data into N_B groups (we call each group as a batch and N_B as the number of batches). Also, suppose each batch contains ℓ_D labeled and $m - \ell_D$ unlabeled data. We denote the set of labeled data in the a-th batch as

$$\{(\mathbf{x}_{\mathrm{L}}^{a,j}, y^{a,j})\}_{j=1}^{\ell_D}, \tag{5.1}$$

where $\mathbf{x}_{\mathrm{L}}^{a,j}$ and $y^{a,j}$ are the j-th labeled input vector and the j-th label vector in the a-th batch. Also we denote the set of unlabeled data as $\{\mathbf{x}_{\mathrm{UL}}^{a,j}\}_{j=1}^{m-\ell_D}$, Let us summarize $\mathbf{x}_{\mathrm{L}}$ and $\mathbf{x}_{\mathrm{UL}}$ into a single data vector as

$$\mathbf{x}_{\mathrm{data}}^{a,j} = \begin{cases} \mathbf{x}_{\mathrm{L}}^{a,j} & 1 \le i \le \ell_D \\ \mathbf{x}_{\mathrm{UL}}^{a,j-\ell} & \ell_D + 1 \le i \le m \end{cases}. \tag{5.2}$$

The label data $y^{a,j}$ takes one of the values of $\{1, \cdots, c\}$, where c is the number of classes.

Generator

The generator is built by PQC, which prepares the following quantum state $|\psi\rangle = U(\theta)|0\rangle^{\otimes n}$. The circuit outputs m fake data $\{\mathbf{x}_{\mathrm{fake}}^{j}\}_{j=1}^{m}$ as a result of the computational basis measurements of $|\psi\rangle$. The probability that $\mathbf{r}$ is measured is given by

$$p_G(\boldsymbol{\theta}, \mathbf{r}) = \mathrm{Tr}\left(\rho_0 U(\boldsymbol{\theta})^\dagger |\mathbf{r}\rangle\langle\mathbf{r}| U(\boldsymbol{\theta})\right), \tag{5.3}$$

which is same as QGANDL case (3.53).

Discriminator/Classifier(D/C)

In the framework of qSGAN, the discriminator not only has the function $D(\mathbf{x})$ that classifies the input data $\mathbf{x}$ as real or fake, but also have additional function: the classifier $C(\mathbf{x})$. The value of $D(\mathbf{x})$ represents the likelihood that the received $\mathbf{x}$ is included in the training dataset as in the case of the standard GAN. The classifier $C(\mathbf{x})$ is the vector whose dimension is $c+1$; the u-th ($1 \leq u \leq c$) element of $C(\mathbf{x})$ represents the likelihood that $\mathbf{x}$ belongs to the u-th class while the last element is the likelihood that $\mathbf{x}$ is a fake data. The $C(\mathbf{x})$ and $D(\mathbf{x})$ are built by a double-headed classical neural network, meaning that with the notation of (3.36),

$$f(\mathbf{x}^j) = \boldsymbol{\alpha}^{(L-1)}(\mathbf{x}^j), \quad C(\mathbf{x}) = \sigma^C\left(W\boldsymbol{\alpha}^{(L-1)}(\mathbf{x}^j) + b^C\right), \quad D(\mathbf{x}^j) = \sigma^D\left(W'\boldsymbol{\alpha}^{(L-1)}(\mathbf{x}^j) + b^D\right), \tag{5.4}$$

where $W \in \mathbb{R}^{(c+1)\times n_{L-1}}$ and $W' \in \mathbb{R}^{1\times n_{L-1}}$ are trainable matrices, σ^C and σ^D is activation functions; σ^C takes $c+1$ dimensional input and returns the same dimensional output while σ^D takes 1 dimensional input and returns the same dimensional output. In the following, we denote the discriminator having two functions $C(\mathbf{x})$ and $D(\mathbf{x})$ by D/C.

Training rule

In the training process of a-th batch, m real data are loaded from the dataset, and the generator generates m fake data. Those two types of data are the inputs of D/C. Those data are classified by D/C; $C(\mathbf{x})$ computes the likelihood that $\mathbf{x}$ belongs to each class, and $D(\mathbf{x})$ also computes the likelihood of real/fake. Based on those results, the parameters of the generator and the D/C are updated by the gradient descendant as follows.

For generator, we update the parameters so that fake data are classified as real by $D(\mathbf{x})$. More precisely, the parameters of the generator are updated to the direction that the following cost function L_G decreases,

$$L_G = -\sum_{j=1}^{m} \log(D(\mathbf{x}^j_{\text{fake}})), \tag{5.5}$$

as in case of QGAN discussed in Section 3.3.3. Also by the same argument to derive (3.58), the gradient vector is given by

$$\frac{\partial L_G(\boldsymbol{\theta})}{\partial \theta_k} \simeq \frac{1}{m}\sum_{j=1}^{m}\left(-\log(D(\mathbf{x}^{j(k+)}_{\text{fake}})) + \log(D(\mathbf{x}^{j(k-)}_{\text{fake}}))\right). \tag{5.6}$$

For D/C, we update the parameters so that it correctly judges if the input data is real/fake by $D(\mathbf{x})$ and additionally classifies them into the true class by $C(\mathbf{x})$. Thus, we update the parameters of the classical neural network to the direction that the following cost function $L_{D/C}$ decreases:

$$L_{D/C} = (L_D + L_C)/2, \tag{5.7}$$

where

$$L_D = -\mathbb{E}_{\mathbf{x}\sim p_G(\boldsymbol{\theta},\mathbf{x})}[\log(1 - D(\mathbf{x}))] - \mathbb{E}_{\mathbf{x}\sim q(\mathbf{x})}[\log(D(\mathbf{x}))] \tag{5.8}$$

$$\simeq -\frac{1}{m}\sum_{j=1}^{m}\left(\log\left(1 - D(\mathbf{x}^j_{\text{fake}})\right) + \log D(\mathbf{x}^{a,j}_{\text{data}})\right), \tag{5.9}$$

$$L_C = \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x})}[h(c+1, C(\mathbf{x}))] + \mathbb{E}_{\mathbf{x},y\sim q_l(\mathbf{x},y)}[h(y, C(\mathbf{x}))] \tag{5.10}$$

$$\simeq \frac{1}{m}\sum_{j=1}^{m} h(c+1, C(\mathbf{x}^j_{\text{fake}})) + \frac{1}{\ell}\sum_{j=1}^{\ell} h(y^j, C(\mathbf{x}^{a,j}_{\text{L}})), \tag{5.11}$$

with $q(\mathbf{x})$ as the underlying probability distribution of the input vectors in the training dataset and $q_l(\mathbf{x}, y)$ as the underlying distribution of the pairs of the input vector and the label in the

Algorithm 2 Quantum semi-supervised generative adversarial network [52]

```
1: for $i = 1$ to $N_{\text{iter}}$ do
2:   for $a = 1$ to $N_B$ do
3:     Load labeled data $\{(\mathbf{x}_{\text{L}}^{a,j}, y^{a,j})\}_{j=1}^{\ell_D}$ and unlabeled data as $\{\mathbf{x}_{\text{UL}}^{a,j}\}_{j=1}^{m-\ell_D}$.
4:     for $j = 1$ to $m$ do
5:       Set $\mathbf{x}_{\text{fake}}^{j}$ to the measurement result of $U(\boldsymbol{\theta})|0\rangle^{\otimes n}$.
6:     end for
7:     for $k = 1$ to $n$ do
8:       for $j = 1$ to $m$ do
9:         Set $\mathbf{x}_{\text{fake}}^{j(k+)}$ and $\mathbf{x}_{\text{fake}}^{j(k-)}$ to the measurement results of $U_{k+}(\boldsymbol{\theta})|0\rangle^{\otimes n}$ and $U_{k-}(\boldsymbol{\theta})|0\rangle^{\otimes n}$.
10:      end for
11:      Set $\partial L_G/\partial\theta_k$ to $\sum_{j=1}^{m}\left[-\log D(\mathbf{x}_{\text{fake}}^{(k+)j}) + \log D(\mathbf{x}_{\text{fake}}^{(k-)j})\right]/m$.
12:    end for
13:    Update $\boldsymbol{\theta}$ by the gradient descent algorithm using the gradient vector $\{\partial L_G/\partial\theta_k\}_{k=1}^{P}$ .
14:    Set $L_D$ to $-\sum_{j=1}^{m}\left[\log\left(1 - D(\mathbf{x}_{\text{fake}}^{j})\right) + \log D(\mathbf{x}_{\text{data}}^{a,j})\right]/m$.
15:    Set $L_C$ to $\sum_{j=1}^{m} h(c+1, C(\mathbf{x}_{\text{fake}}^{j}))/m + \sum_{i=1}^{\ell_D} h(y^{a,j}, C(\mathbf{x}_{\text{L}}^{a,j}))/\ell_D$.
16:    Set $L_{D/C}$ to $(L_D + L_C)/2$.
17:    Compute the gradients of $L_{D/C}$ using the back propagation and update the parameters by the
       gradient descent algorithm.
18:  end for
19: end for
20: The classifier $C(\mathbf{x})$ as the final result.
```

labeled dataset. Here $h(y, \mathbf{a})$ is the cross entropy loss defined by

$$h(y, \mathbf{a}) = -a_y + \log\left(\sum_{u=1}^{d} \exp(z_u)\right), \tag{5.12}$$

where z_u is the u-th element of the vector $\mathbf{z}$ and d is the dimension of $\mathbf{z}$. The gradient vector can be computed in a similar manner using the backpropagation as (3.51). Same as the training of the generator, the gradient descent is used for the update of the parameters in $L_{D/C}$.

In each iteration, we update the parameters of the generator and the discriminator by using all batches. After a sufficiently large number of iterations, we obtained $C(\mathbf{x})$ as a good classifier. The overall algorithm is summarized in Algorithm 2 with N_{iter} as the number of iterations.

5.2 Numerical Demonstration

In this section, we demonstrate the performance of the qSGAN method through numerical simulations. In particular, we find that a quantum generator with high expressive power leads to a better classifier after successful learning. Also, the resulting classification accuracy is comparable to that of using a standard classical neural network generator. The resulting classification accuracy is comparable to that obtained using a classical neural network generator.

5.2.1 Noiseless case

The source of the real data used in this simulation is the set of images of 1×8 pixels shown in Fig. 5.2. Each pixel takes a value of 0 (black) or 1 (white). Also, each image is assigned a label '0' or '1' (hence $c = 2$) for the following convention. If all white pixels in an image are connected, or there is only one white pixel in an image, the image is labeled '0'. If the white pixels in the image are split into two separate parts, then the image is labeled '1'. The number of images with

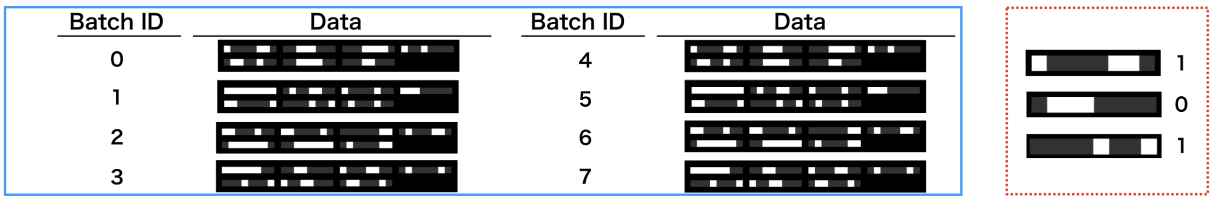
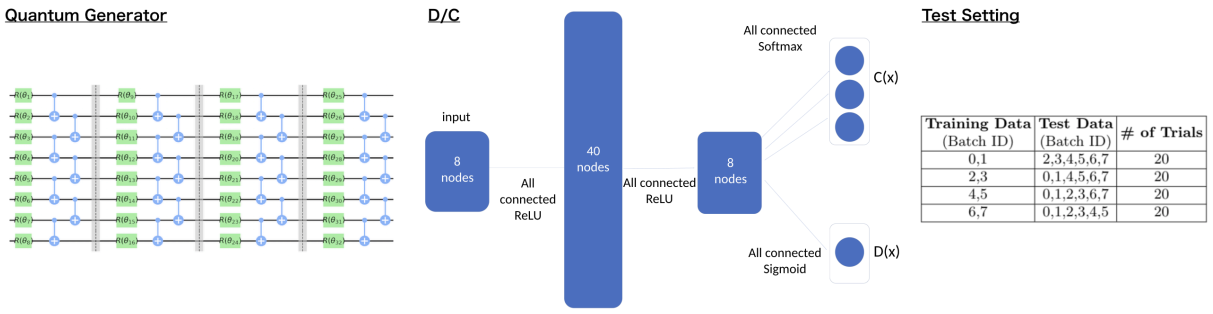


Figure 5.2: Left (enclosed by the blue rectangular): The real data used in the numerical simulation. The data is divided into eight batches. Right (enclosed by the red dotted rectangular): Examples of images and their labels. Reprinted figure from [DOI: 10.1038/s41598-021-98933-6]. Creative Commons Attribution 4.0 International license. Copyright 2021 by the Kouhei Nakaji and Naoki Yamamoto.



Training Data (Batch ID)	**Test Data** (Batch ID)	**# of Trials**
0,1	2,3,4,5,6,7	20
2,3	0,1,4,5,6,7	20
4,5	0,1,2,3,6,7	20
6,7	0,1,2,3,4,5	20

Figure 5.3: Left: The quantum circuit with four layers used in the simulation. Center: The D/C system, where the last layer functions as the classifier $C(x)$ or the discriminator $D(x)$. Right: The training/test dataset combination for the 4-fold cross-validation. Reprinted figure from [DOI: 10.1038/s41598-021-98933-6]. Creative Commons Attribution 4.0 International license. Copyright 2021 by the Kouhei Nakaji and Naoki Yamamoto.

the label '0' and the number of images with the label '1' are both 28; therefore, there are a total of 56 images. This dataset is divided into eight batches containing $m = 7$ images.

As the quantum generator, we use an eight-qubits parametrized quantum circuit with a single layer or four layers; the generator with four layers is shown in the left of Fig. 5.3. Each layer is composed of parametrized single-qubit rotational gates $\exp(-i\theta_i\sigma_{a_i}/2)$ and CNOT gates that connect adjacent qubits; here θ_i is the i-th parameter and σ_{a_i} is the Pauli operator ($a_i = x, y, z$). All θ_i and a_i are randomly initialized at the beginning of each training. The numerical simulation run on Qiskit QASM Simulator [74].

As the D/C, a neural network with four layers is used, which is shown in the center of Fig. 5.3. The first three layers of the network are shared by both the discriminator $D(\mathbf{x})$ and the classifier $C(\mathbf{x})$. The number of nodes in the first, second, and third layers is 8, 40, and 8, respectively; all nodes between the layers are fully connected, and we use ReLU as the activation function. The last layer for the classifier has three nodes, corresponding to the likelihood of label '0', label '1', and fake classes; these nodes are fully connected to those of the third layer, and the softmax function is used as the activation function. The last layer of the discriminator has one node, which gives the value of $D(\mathbf{x})$; this node is fully connected to the nodes of the third layer, and the sigmoid function ($\sigma(x) = 1/(1 + \exp(-x))$) is used as the activation. The neural network is implemented by PyTorch [116].

We choose two of the eight batches as the training dataset (hence $N_B = 2$) and the other six as the test dataset in each trial. We perform the 4-fold cross-validation by changing the training and test dataset, which is summarized in the right table of Fig. 5.3. For each training/test dataset, we execute 20 trials; therefore, we execute 80 trials totally. To demonstrate the semi-supervised learning, we mask some labels in each batch; recall that the number of labeled examples in each batch is denoted by ℓ, which takes $\ell = 2$ or 5 in this simulation. As the gradient descent algorithm, Adam [76] is used, whose learning coefficient is set to 0.001 for the case of generator and 0.005 for the case of D/C.

The left and center plots of Fig. 5.4 show the relationship between the average classification accuracy of the test data and the iterations. Each point is obtained as the average of 80 trials. The two subfigures correspond to different number of labeled data, as $\ell = 2$ (left) and $\ell = 5$ (center). In each subfigure, we show three different cases, depending on the generator; the quantum generator with one layer (blue) and that with four layers (orange), and the uniform-noise generator (green) that randomly generates 8-bit data with equal probability, which is not updated while training. The error bar represents the standard deviation of the average classification accuracy.

We see that when only a few labeled data is available ($\ell = 2$), the quantum generator with four layers results in the highest classification accuracy, which implies that the quantum generator with bigger expressibility contributes to the higher accuracy by effectively generating samples to train the classical D/C. On the other hand, in the case where five of eight image data in each batch are labeled ($\ell = 5$), the three generators achieve almost the same accuracy. This might be because, in this case, all the generators fail to generate a more valuable dataset than the set of labeled real data for effectively training the D/C. This observation is supported by the fact that the untrained uniform-noise generator, which, of course, is not related to the real dataset, achieves almost the same classification accuracy. Therefore, we expect that the quantum generator is useful when the number of labeled data is limited.

In this numerical simulation, we obtained the best classification accuracy when the constructed classical sample distribution corresponding to the output of the quantum generator does not match the distribution producing the real dataset, as predicted in [50]. In addition, we found that the cost for the quantum generator, L_G, is larger than that for the classical D/C, $L_{D/C}$, when the best accuracy is reached. These facts are favorable for the current noisy quantum devices that cannot be effectively trained due to the noise. Hence the next topic is to study how much the noise affects the quantum generator and accordingly the classification accuracy.

5.2.2 Noisy case

We examine the case where a noise channel is applied between every layer of the quantum generator. In particular, we assume the depolarizing channel:

$$\mathcal{E}(\rho) = (1-p)\rho + p\frac{I}{2^n}, \tag{5.13}$$

where ρ is a density matrix, I is the identity matrix, n is the number of qubits and p is a noise parameter. In this density matrix representation, the ideal unitary gate operation is expressed as $\mathcal{U}_i(\rho) = U_i \rho U_i^\dagger$, where U_i is the i-th layer unitary matrix. Then, the output density matrix of the four-layers quantum circuit under the above depolarizing noise is given by

$$\begin{aligned}
\rho_{\text{out}} &= \mathcal{E} \circ \mathcal{U}_4 \circ \mathcal{E} \circ \mathcal{U}_3 \circ \mathcal{E} \circ \mathcal{U}_2 \circ \mathcal{E} \circ \mathcal{U}_1(|0\rangle\langle 0|) \\
&= \mathcal{E} \circ \mathcal{U}_4 \circ \mathcal{E} \circ \mathcal{U}_3 \circ \mathcal{E} \circ \mathcal{U}_2 \left((1-p) U_1 |0\rangle\langle 0| U_1^\dagger + p\frac{I}{2^n} \right) \\
&= \cdots = (1-p)^4\, U_4 U_3 U_2 U_1 |0\rangle\langle 0| U_1^\dagger U_2^\dagger U_3^\dagger U_4^\dagger + \left(1 - (1-p)^4\right) \frac{I}{2^n},
\end{aligned} \tag{5.14}$$

where for instance, $\mathcal{E} \circ \mathcal{U}_i(\cdot)$ denotes the composite function $\mathcal{E}\left(\mathcal{U}_i(\cdot)\right)$ of $\mathcal{E}(\cdot)$ and $\mathcal{U}_i(\cdot)$. The samples are generated by measuring ρ_{out}. Other than the noise, the simulation setting is the same as the noiseless case. We examine the case when only two of eight image data in each batch are labeled ($\ell = 2$).

The resulting classification accuracy achieved via the quantum generator under the depolarization noise is shown in the right of Fig. 5.4, with several values of noise strength p. The horizontal axis represents the magnitude of total noise, i.e., the coefficient of the second term in Eq. (5.14), while the vertical axis represents the average classification accuracy at the final (= 100-th) iteration step. The error bar is the standard deviation of the average accuracy. The result is that, as discussed before, the classification accuracy does not become worse than the noiseless case as long as the depolarization noise for each layer of the quantum generator is suppressed to some level ($p = 0.05$). This demonstrates the second advantage of the proposed qSGAN; The quantum generator in our qSGAN framework does not need to generate a pure quantum state.

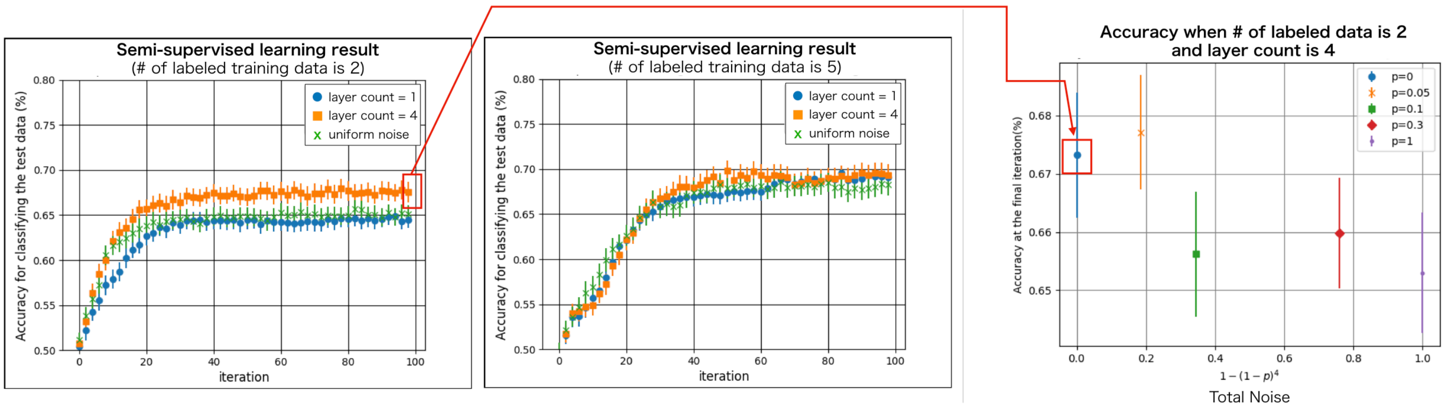


Figure 5.4: Left, Center: Classification accuracy of the classifier when using the quantum generator. The number of labeled data is $\ell = 2$ (left) and $\ell = 5$ (center). Right: Classification accuracy of the classifier for $\ell = 2$ when using the four-layers quantum generator under the depolarization noise (5.13). The data points surrounded by the red rectangles are drawn with the same data. Reprinted figure from [DOI: 10.1038/s41598-021-98933-6]. Creative Commons Attribution 4.0 International license. Copyright 2021 by the Kouhei Nakaji and Naoki Yamamoto.

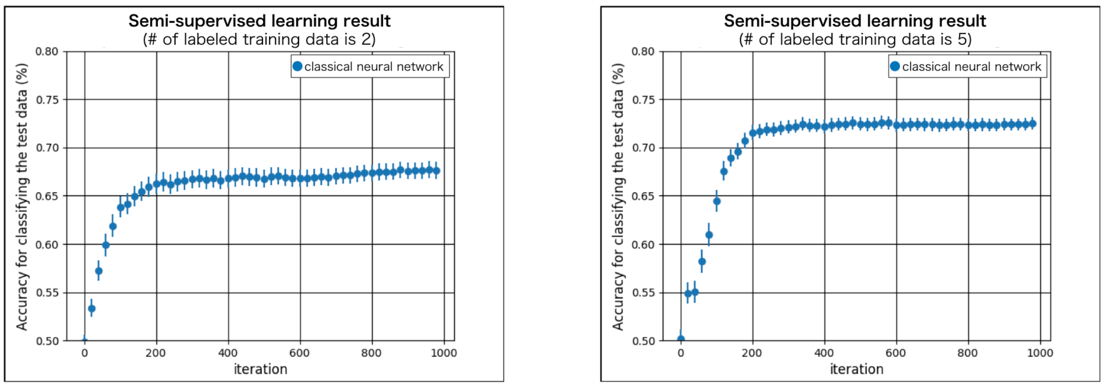


Figure 5.5: Classification accuracy of the classifier when using the five-layers classical neural network generator. The number of labeled data is $\ell = 2$ (left) and $\ell = 5$ (right). Reprinted figure from [DOI: 10.1038/s41598-021-98933-6]. Creative Commons Attribution 4.0 International license. Copyright 2021 by the Kouhei Nakaji and Naoki Yamamoto.

5.2.3 Comparison with classical neural network generator

Finally, we compare the performance of the proposed qSGAN to the fully classical case where the generator is given by a five-layer classical neural network. The input to the generator is the 1-dimensional normal Gaussian noise with zero mean and unit variance. The second, third, and fourth layers comprise 40 nodes and the fifth (= final) layer has eight nodes. The nodes between the layers are fully connected, and ReLU is used as the activation function. The output sample is obtained by transforming the values of the nodes at the final layer by the sigmoid function ($\sigma(x) = 1/(1 + e^{-x})$). We use the same D/C used in the quantum case. As the gradient descent algorithm, we use Adam, whose learning coefficients are set to 0.001 for both the generator and the D/C. Fig. 5.5 shows the classification accuracy for the test data over the number of iteration, which is obtained as the average of over 80 trials when using the classical neural network generator. The two subfigures are obtained with a different number of labeled data, $\ell = 2$ and $\ell = 5$.

The result is that for the case $\ell = 2$, the classifier aided by the classical neural network generator achieves the classification accuracy of about 67%, which is comparable to that of the four-layers quantum generator shown in the left subfigure of Fig. 5.4. The notable point is that the number of parameters of the classical and quantum generators are 3688 and 32, respectively. Hence, naively, the quantum generator has a rich expressibility power comparable to the classical one even with much fewer parameters. This means that the training of the quantum generator is easier than the classical one, which is actually shown in Figs. 5.4 and 5.5. More importantly, this result implies that a bigger quantum generator with a tractable number of parameters could have a potential to work even for some problems that are intractable via any classical one due to the explosion of the number of parameters.

Chapter 6

Solutions to the barren plateau issue assessed by the expressibility analysis

As we see in Section 3.4.1, the barren plateau issue is critical for all variational quantum algorithms. For mitigating this issue, several approaches have been proposed; e.g., circuit initialization [117], special structured ansatz [54], and parameter embedding [118]. In Section 6.1, we review those possible solutions.

Among those solutions, the literature [54] only gives the theoretical guarantee that the barren plateau issue is avoided, and therefore, is arguably the most promising among the solutions. However, for applying the method in [54], we must use the *alternating layered ansatz* whose entangling operations are limited compared to the hardware efficient ansatz. The limited entangling operations in general lead to the loss of expressive power for generating arbitrary quantum states; if the loss is huge, the solution in [54] is not usable in the variational quantum algorithm. In Section 6.2, which is the contribution of the author of this thesis, we examine the expressibility of the alternating layered ansatz and assess the method in [54].

6.1 Possible solutions to the barren plateau issue

6.1.1 Identity block strategy

The proof of the barren plateau utilizes the fact that the parameters in the hardware efficient ansatz are randomly initialized, which is why the distribution of the unitary matrix corresponding to the ansatz is close to unitary 2-design. The method proposed in [119] is to change the random initialization strategy to another one so that the barren plateau issue is avoided. In this method the hardware efficient ansatz is still used.

The initialization strategy introduced in [119] is so-called identity block strategy. In the strategy the deep PQC is separated into multiple blocks; the depth of each block, which is also a hardware efficient ansatz, is taken to be shallow enough so that even if we randomly initialize the parameters inside the block the distribution of the unitary operator corresponding to the block does not become close to unitary 2-design. Some of the parameters in each block are randomly initialized while the rest of the parameters are initialized so that the unitary operator corresponding to the block is identity. More precisely each block is separated into the former part and the latter part that has the same structure as the former part, and if the first part of the ℓ-th block is randomly initialized and the unitary operator corresponding to the former part is given as U_ℓ then the latter part of the block is initialized as $U_\ell^\dagger$ as we show in Fig. 6.1, which is realizable by inverting the signs of the parameters in the former block. Once the training starts the parameters in the latter part of each block is freely changed; given the unitary corresponding to the former part of ℓ-th block at the iteration t as $U(\boldsymbol{\theta}_1^{\ell,t})$, the unitary operator corresponding

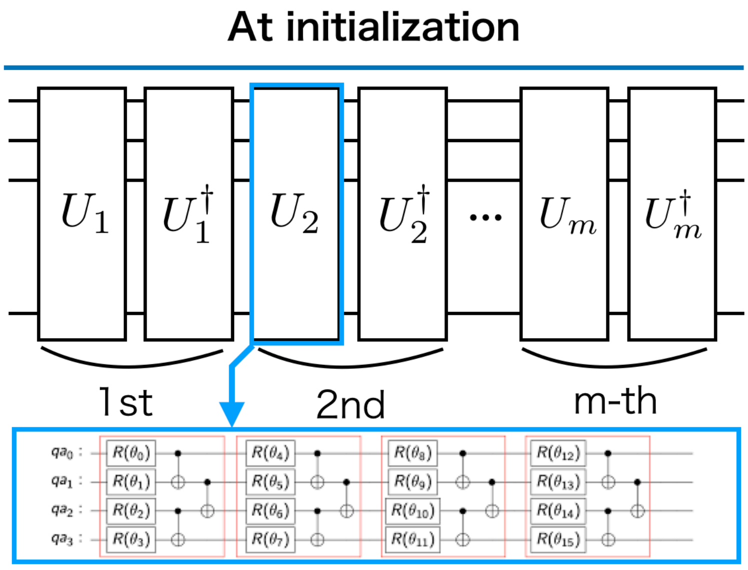


Figure 6.1: The overview of the initialization when using the identity block strategy.

to the latter part becomes $U(\boldsymbol{\theta}_2^{\ell,t}) \neq U(\boldsymbol{\theta}_1^{\ell,t})^\dagger$. Note that $U(\boldsymbol{\theta}_1^{\ell,0}) = U_\ell$ and $U(\boldsymbol{\theta}_2^{\ell,t}) = U_\ell^\dagger$.

As a consequence of introducing the above initialization strategy, the gradient of the cost function at the first iteration becomes large as we see in the following argument. Given the number of blocks as L and the cost function at iteration t as

$$C^t(\boldsymbol{\theta}) = \mathrm{Tr}\left(\rho U_{\mathrm{all}}^t(\boldsymbol{\theta})^\dagger \mathcal{O} U_{\mathrm{all}}^t(\boldsymbol{\theta})\right), \tag{6.1}$$

where $U(\boldsymbol{\theta})_{\mathrm{all}}^t \equiv \prod_{\ell=1}^L U(\boldsymbol{\theta}^{\ell,t})$ and $U(\boldsymbol{\theta}^{\ell,t}) \equiv U(\boldsymbol{\theta}_2^{\ell,t})U(\boldsymbol{\theta}_1^{\ell,t})$, the gradient of the cost function at $t=0$ is

$$\left.\frac{\partial C^t(\boldsymbol{\theta})}{\partial\theta_j}\right|_{t=0} = \mathrm{Tr}\left(\left.\frac{\partial U_{\mathrm{all}}^t(\boldsymbol{\theta})^\dagger}{\partial\theta_j}\right|_{t=0} \mathcal{O} U_{\mathrm{all}}^0(\boldsymbol{\theta})\right) + \mathrm{Tr}\left(U_{\mathrm{all}}^0(\boldsymbol{\theta})^\dagger \mathcal{O} \left.\frac{\partial U_{\mathrm{all}}^t(\boldsymbol{\theta})}{\partial\theta_j}\right|_{t=0}\right). \tag{6.2}$$

Suppose that the parameter θ_j is included in the ℓ'-th block, then

$$\left.\frac{\partial C^t(\boldsymbol{\theta})}{\partial\theta_j}\right|_{t=0} = \left.\frac{\partial \mathrm{Tr}\left(U(\boldsymbol{\theta}^{\ell',t})^\dagger \mathcal{O} U(\boldsymbol{\theta}^{\ell',t})\right)}{\partial\theta_j}\right|_{t=0}, \tag{6.3}$$

which is $O(1)$ since $U(\boldsymbol{\theta}^{\ell',t})$ corresponding to the ℓ'-th block is shallow.

The obvious flaw of the above argument is that we do not discuss how large the gradient becomes after the serveral iterations; it is only proved that large gradient vector is obtained at the first iteration. In [119] from a numerical experiment using VQE with 7-qubit hardware efficient ansatz, it is inferred that even after much iterations the gradient is still large and therefore the barren plateau issue is avoided.

6.1.2 Layerwise learning

Another method for devising the initialization strategy is the layerwise learning strategy proposed in Ref. [120]. As in the initial block strategy discussed in the previous subsection, the way of initializing the parameters is devised so that the barren plateau issue is avoided. In this strategy there are two stages: (i) layerwise learning stage and (ii) overall learning stage.

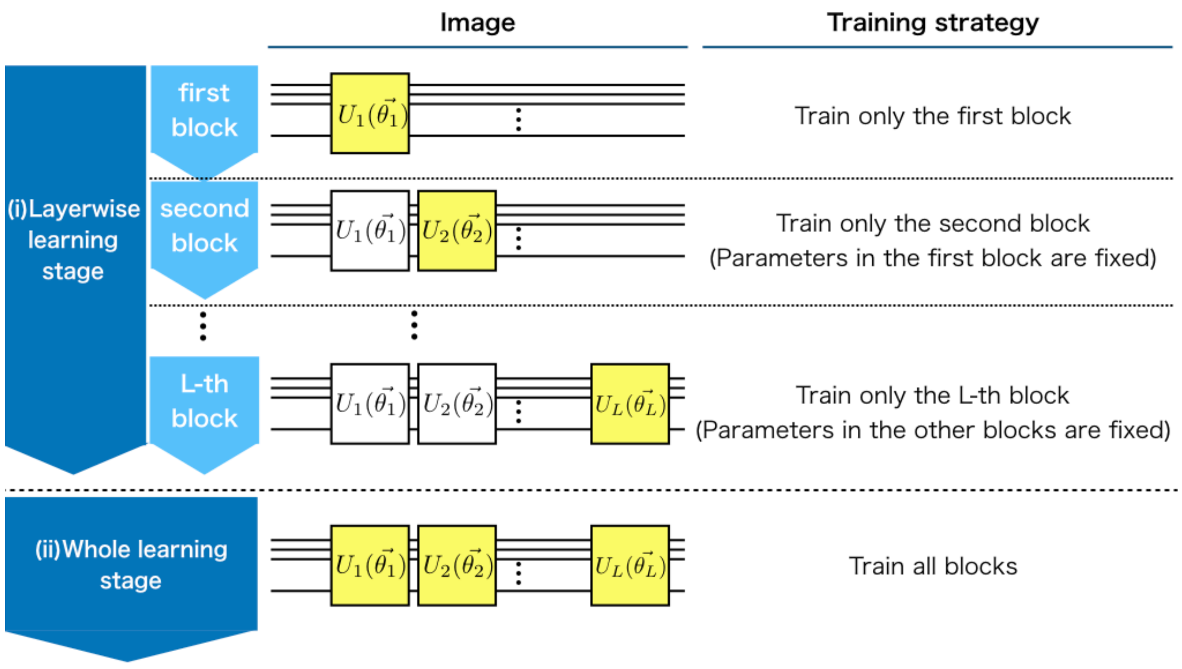


Image
Training strategy
(i)Layerwise learning stage
first block
second block
L-th block
(ii)Whole learning stage
$U_1(\vec{\theta_1})$
$U_2(\vec{\theta_2})$
$U_L(\vec{\theta_L})$
Train only the first block
Train only the second block (Parameters in the first block are fixed)
Train only the L-th block (Parameters in the other blocks are fixed)
Train all blocks

Figure 6.2: Overview of the layerwise learning.

(Phase 1) Layerwise learning stage

Suppose that we want to find $U(\boldsymbol{\theta})$ that minimizes the cost function

$$C(\boldsymbol{\theta}) = \mathrm{Tr}\left(\rho U(\boldsymbol{\theta})^\dagger \mathcal{O} U(\boldsymbol{\theta})\right). \tag{6.4}$$

In the layerwise learning stage, we start from training a shallow hardware efficient ansatz $U_1(\boldsymbol{\theta})$ by the gradient descendant so that the following cost is minimized

$$C_1(\boldsymbol{\theta}_1) = \mathrm{Tr}\left(\rho U_1(\boldsymbol{\theta}_1)^\dagger \mathcal{O} U_1(\boldsymbol{\theta}_1)\right). \tag{6.5}$$

Note that the training is not troubled by the barren plateau issue because $U_1(\boldsymbol{\theta}_1)$ is shallow.

Suppose that after enough training of $U_1(\boldsymbol{\theta}_1)$ we obtain an optimal parameter $\boldsymbol{\theta}_1^*$. Then in the second step we add a shallow hardware efficient ansatz $U_2(\boldsymbol{\theta}_2)$, which is initialized as $U_2(\boldsymbol{\theta}_2) = I$, after the circuit $U_1(\boldsymbol{\theta}_1^*)$ and we train $U_2(\boldsymbol{\theta}_2)$ so that the following $C_2(\boldsymbol{\theta}_2)$ is minimized:

$$C_2(\boldsymbol{\theta}_2) = \mathrm{Tr}\left(\rho U_2(\boldsymbol{\theta}_2)^\dagger U_1(\boldsymbol{\theta}_1^*)^\dagger \mathcal{O} U_2(\boldsymbol{\theta}_1^*) U_1(\boldsymbol{\theta}_1)\right). \tag{6.6}$$

It should be noted that the parameters in the firstly appended ansatz U_1 are fixed; the optimization that minimizes C_2 requires training of the shallow circuit $U_2(\boldsymbol{\theta})$. Therefore in this step again the training is not troubled by the barren plateau issue.

In the similar manner, new shallow hardware ansatz $U_\ell(\boldsymbol{\theta}_\ell)$, which is initialized as identity, is appended to the circuit in the ℓ-th step and only the parameters in this newly appended ansatz are trained. In every training steps, only the shallow ansatzes are trainied, and therefore, the barren plateau issue can be avoided in this stage. In the following, we assume that we stop this procedure at L-th step, i.e. we obtain the circuit

$$U^0(\boldsymbol{\theta}) = \prod_{\ell=1}^{L} U_\ell(\boldsymbol{\theta}_\ell^*), \tag{6.7}$$

which becomes the input of the next stage. The overview of the process of the layerwise learning stage is shown at the top of Fig. 6.2.

(Phase 2) Overall learning stage

In the overall learning stage, we train all parameters of the circuit built in the previous stage. More precisely the circuit is trained so that

$$C(\boldsymbol{\theta}) = \mathrm{Tr}\left(\rho U(\boldsymbol{\theta})^\dagger \mathcal{O} U(\boldsymbol{\theta})\right) \tag{6.8}$$

is minimzed, with

$$U(\boldsymbol{\theta}) = \prod_{\ell=1}^{L} U_\ell(\boldsymbol{\theta}_\ell). \tag{6.9}$$

As the initial parameters, $\boldsymbol{\theta}_\ell = \boldsymbol{\theta}_\ell^*$ are used for all ℓ that are obtain in the previous stage.

A merit of using the method

The training process in Phase 2 is same as that of the ordinary VQA using the hardware efficient ansatz. The only difference is the parameter initialization. In [120] it is claimed that the process in Phase 1 functions as the initialization process different from the random initialization, which is why the barren plateau is avoidable by this method since the barren plateau issue is proved under the condition that the parameters are randomly initialized. For supporting their craim, the authors in [120] also shows a numerical evidence that the barren plateau is avoided by using a quantum machine learning task.

6.1.3 Using alternating layered ansatz instead of hardware efficient ansatz

The above discussed two methods change the initial parameters but it does not change the cost function landscape. Ref. [54] propose a method to change the cost function landscape by using an ansatz named alternating layered ansatz.

The structure of the alternating layered ansatz is as follows. ALT is composed of multiple layers. Each layer of ALT has separated blocks, where each block has repetitions of parametrized single-qubit rotational gates and entanglers that entangle all qubits inside the block. In the odd-number-labeled layers, each block contains m qubits, so that m is an even number and n/m is an integer. Conversely, the odd-number-labeled layers contain n/m blocks which operate on $\{1, \dots, m\}$, $\{m+1 \dots, 2m\}$, ..., and $\{n-m+1, \dots, n\}$ qubits. The even-number-labeled layers contain $n/m+1$ blocks which operate on $\{1, \dots, m/2\}$, $\{m/2+1, \dots, 3m/2\}$, ..., and $\{n-m/2+1, \dots, n\}$ qubits; namely, the first and the last block operate on $m/2$ qubits, while the others operate on m qubits. The structure of the alternating layered ansatz when the number of layers is L, $m=4$, and the number of qubits is 12 is shown in Fig. 6.3. For later purpose, let us call m as *locality*.

The characteristics of the cost landscape when using the alternating layered ansatz depends on the form of the cost function. Before going into the detail of how the cost landscape is changed, we define two types of observables: the *local* observable and the *global* observable.

Local observable

An observable $\mathcal{O}$ is said to be *k-local* if $\mathcal{O}$ can be written in the form of

$$\mathcal{O} = \sum_{j=1}^{n/m} c_j O_j^m, \tag{6.10}$$

where O_j^m is an Hermitian operator that non-trivially operates on $m(j-1)+1 \sim mj$-th qubits and $\{c_j\}_{j=1}^n$ are real constants (some of c_j can be zero). For instance

$$\mathcal{O} = \sigma_z \otimes I_{n-1} + I_1 \otimes \sigma_z \otimes I_{n-2} \tag{6.11}$$

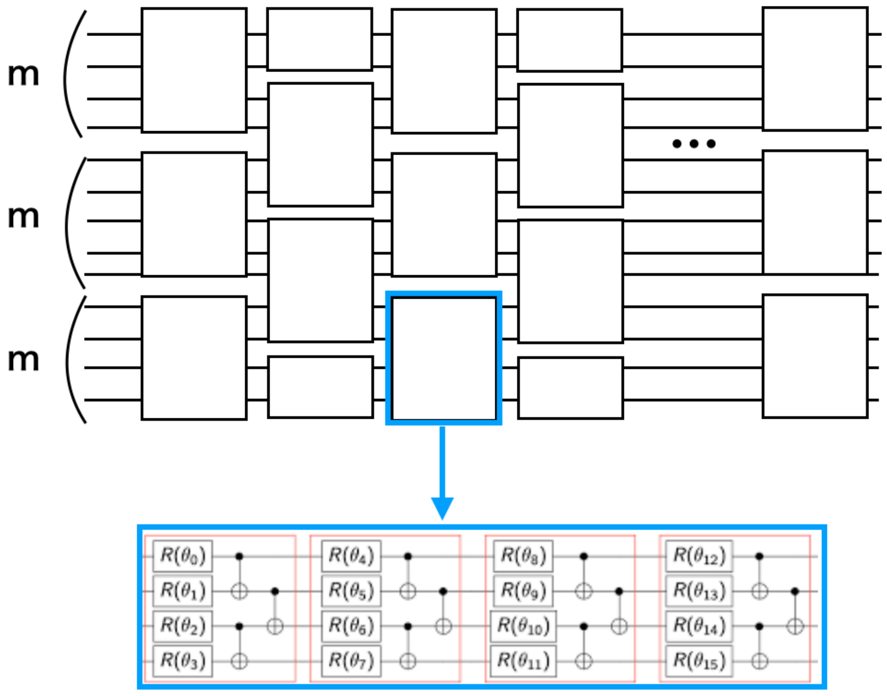


Figure 6.3: The structure of the alternating layered ansatz when the number of layers is L, $m = 4$, and the number of qubits is 12.

is a *1-local* observable. As another example

$$\mathcal{O} = \sigma_z \otimes \sigma_y \otimes I_{n-2} + I_2 \otimes \sigma_x \otimes \sigma_y \tag{6.12}$$

is *2-local* observable.

Global observable

An observable $\mathcal{O}$ is said to be global if $\mathcal{O}$ contains the term that non-trivially operates on all qubits. For instance

$$\mathcal{O} = \sigma_z \otimes \sigma_y \otimes \cdots \otimes \sigma_x \tag{6.13}$$

is the global observable.

The cost function landscape dramatically changes depending on whether the observable is local or global. Suppose that the distribution of the unitary operator corresponding to each block is unitary 2-design when randomly choosing the parameters. In [54] it is proved that if an observable $\mathcal{O}$ is m-local and we use the alternating layered ansatz with locality equals to m, then

$$\left\langle \frac{\partial C}{\partial \theta_j} \right\rangle = \int d\boldsymbol{\theta} \frac{\partial C}{\partial \theta_j} = 0, \quad \left\langle \left(\frac{\partial C}{\partial \theta_j} \right)^2 \right\rangle = \int d\boldsymbol{\theta} \left(\frac{\partial C}{\partial \theta_j} \right)^2 \geq O\left(\frac{1}{2^{mL}} \right) \tag{6.14}$$

where $C = \mathrm{Tr}(\rho U_{alt}(\boldsymbol{\theta})^\dagger \mathcal{O} U_{alt}(\boldsymbol{\theta}))$ with $U_{alt}(\boldsymbol{\theta})$ as the unitary operator corresponding to the alternating layered ansatz. As long as $mL = O(\log n)$ the gradient decreases only polynomially with the value of n.

To the contrary if an observable $\mathcal{O}$ is global, even if we use the alternating layered ansatz with locality equals to m, the gradient of the cost is suppressed as

$$\left\langle \frac{\partial C}{\partial \theta_j} \right\rangle = \int d\boldsymbol{\theta} \frac{\partial C}{\partial \theta_j} = 0, \quad \left\langle \left(\frac{\partial C}{\partial \theta_j} \right)^2 \right\rangle = \int d\boldsymbol{\theta} \left(\frac{\partial C}{\partial \theta_j} \right)^2 \leq O\left(\frac{1}{2^{n}} \right). \tag{6.15}$$

The proof is in [54].

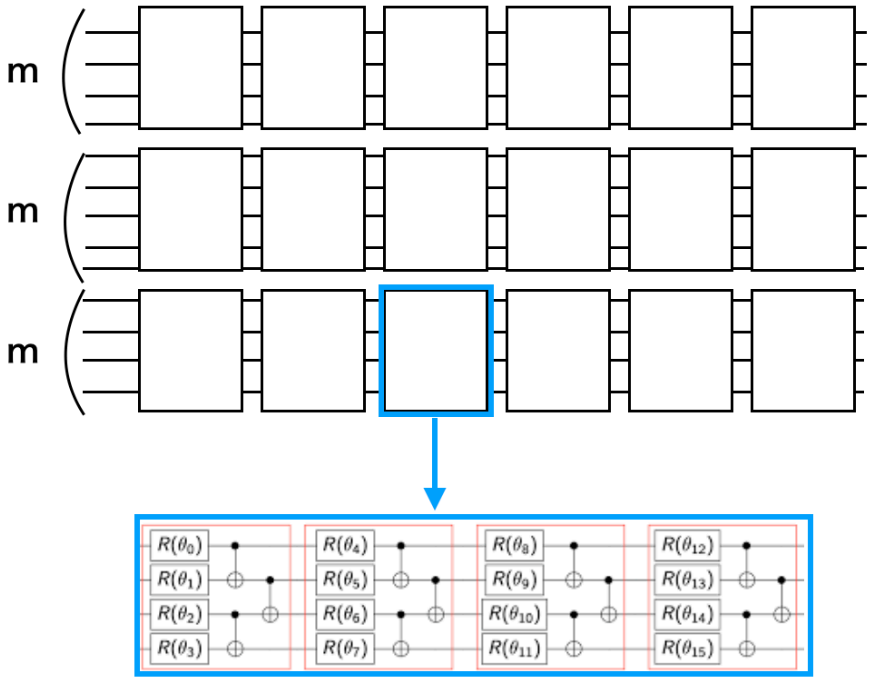


Figure 6.4: Overview of the tensor product ansatz.

6.1.4 Comparison of the three methods

We discussed three different methods to avoid the barren plateau issue. The first one and the second one are the methods to change the initialization strategy from the random initialization. The third one is the method to change the cost function landscape by using the alternating layered ansatz.

The flaw of the discussion in both the first and the second methods is that they do not give a good theoretical evidence. In particular, it is not assured whether or not the gradient is large after some iterations even if we start the optimization from the points where the gradient is large.

To the contrary for the third method using the alternating layered ansatz, it is theoretically shown that the barren plateau issue is avoided as long as the observable is local. Since there are a lot of problems that the observable can be local, e.g. VQE with the Bravyi-Kitaev transformation in the quantum chemistry or the quantum machine learning, the method using the alternating layered ansatz may solve the barren plateau issue in wide range of problems in VQA.

However, we can not immediately conclude that the third method is applicable to wide range of problems in VQA because the alternating layered ansatz may not have enough capability to express variety of unitary operations. We call the capability as *expressibility*, which we will formally define in the next chapter. The expressibility is especially important in case that there is lack of prior knowledge about the problem solution. For instance in VQE, where the goal is to find a quantum state that minimizes the Hamiltonian, it is required that the ansatz has enough capability to generate an arbitrary state as long as we do not have prior knowledge about the target state.

It is not clear whether or not the alternating layered ansatz has rich expressibility since the entangling operations are limited. As an extreme example let us consider the case where there are no entangling operations between the blocks, namely the case where the corresponding unitary operation U_{ten} can be written as

$$U_{ten}(\boldsymbol{\theta}) = U(\boldsymbol{\theta}_1) \otimes U(\boldsymbol{\theta}_2) \cdots U(\boldsymbol{\theta}_{n/m}), \tag{6.16}$$

where $U(\boldsymbol{\theta}_j)$ is the unitary operation for m-qubits. We call the ansatz tensor product ansatz. The structure of the tensor product ansatz is shown in Fig. 6.4. Also suppose that the cost function is given by $C(\boldsymbol{\theta}) = \mathrm{Tr}(\rho U_{ten}(\boldsymbol{\theta})^\dagger \mathcal{O} U_{ten}(\boldsymbol{\theta}))$. Then under the condition that the distribution of $U(\boldsymbol{\theta}_a)$ when $\boldsymbol{\theta}_a$ is randomly chosen is unitary 2-design, it can be proven that

$$\left\langle \frac{\partial C(\boldsymbol{\theta})}{\partial \theta_j} \right\rangle = \int d\boldsymbol{\theta} \frac{\partial C(\boldsymbol{\theta})}{\partial \theta_j} = 0, \quad \left\langle \left(\frac{\partial C(\boldsymbol{\theta})}{\partial \theta_j} \right)^2 \right\rangle = \int d\boldsymbol{\theta} \left(\frac{\partial C(\boldsymbol{\theta})}{\partial \theta_j} \right)^2 \simeq O\left(\frac{1}{2^m} \right), \tag{6.17}$$

with $\mathcal{O}$ as m-local. However it is obvious $U_{ten}(\boldsymbol{\theta})$ can not express an arbitrary unitary operator and also can only generate product states, which are expressed as

$$|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_{n/m}\rangle, \tag{6.18}$$

where $|\psi_j\rangle$ is an m-qubit quantum state. From this example we see that removing the entangling operations may do harm to the ansatz's expressibility; in the alternating layered ansatz, where some of the entangling operations is removed from the hardware efficient ansatz, may lose the expressibility.

Therefore we emphasize that studying the expressibility of the alternating layered ansatz is important for assessing the applicability of the third method, which is the most appealing solution for the barren plateau problem. The rest of the paper is devoted to study the expressibility of ansatzes.

6.2 Expressibility of alternating layered ansatz

In this section, we discuss the expressibility of the ansatzes for the variational quantum algorithm. Particularly, by checking the expressibility of the alternating layered ansatz (ALT), which is introduced for avoiding the barren plateau phenomenon, we see if the scalability issue of the VQA is solvable by using ALT. As a theoretical tool for analyzing the expressibility, we use the expressibility measure proposed in Ref. [56].

The structure of this chapter is as follows. In Section 6.2.1, we review the expressibility measures introduced in Ref. [56]. Next, we show the ansatzes we examine in this thesis. In Section 6.2.3, we describe the expressibility of ansatzes. Finally, we check the validity of the results with the expressibility analysis by the numerical experiment using the variational quantum eigensolver.

6.2.1 Expressibility measure

As in Ref. [56], we define the expressive power of a given ansatz by the randomness of the state generated from the ansatz. For the measure of the randomness, we discuss the frame potential and the Kullback-Leibler (KL) divergence.

Frame Potential

To define the expressive power of a given ansatz C, we consider the deviation of the state distribution generated by C from the Haar distribution. Suppose that $|\psi_\theta\rangle$ is the state generated by the ansatz C characterized by the parameters $\theta \in \Theta$, e.g., $|\psi_\theta\rangle = U_C(\theta)|0\rangle, \theta \in \Theta$, with $U_C(\theta)$ as the unitary operator corresponding to C and $|0\rangle$ as an initial state. Then the deviation can be defined by

$$\mathcal{D}^{(t)}(C) = \left\| \int_{\mathrm{Haar}} (|\psi\rangle\langle\psi|)^{\otimes t} d\psi - \int_\Theta (|\psi_\theta\rangle\langle\psi_\theta|)^{\otimes t} d\theta \right\|_{HS},$$

where $\int_{\mathrm{Haar}}$ is the integration over the state $|\psi\rangle$ that is distributed according to the Haar measure, $\|\cdot\|_{HS}$ is the Hilbert Schmidt norm. Then we call that the ansatz C with smaller $\mathcal{D}^{(t)}(C)$ has a higher expressibility. The reason why the definition of expressibility is reasonable is as follows. The state $|\psi\rangle$ generated from the Haar distribution can in principle represent an arbitrary state. Thus, the condition $\mathcal{D}^{(t)}(C) \approx 0$ means that the ansatz C generate almost all states, including the optimal solution (e.g., the ground state in VQE); also, in this case, the states generated from C are almost equally distributed, which is favorable when we do not have prior knowledge of the problem.

For the computation of $\mathcal{D}^{(t)}(C)$, we can use the following defined t-th generalized frame potential [121] of C:

$$\mathcal{F}^{(t)}(C) = \int_{\Phi}\int_{\Theta} |\langle\psi_\phi|\psi_\theta\rangle|^{2t} d\phi d\theta, \tag{6.19}$$

where both Θ and Φ represent the same set of parameters. In this thesis, we simply call it the t-th frame potential. In particular, we can compute the t-th frame potential of N-dimensional states distributed according to the Haar measure as $\mathcal{F}^{(t)}_{\mathrm{Haar}}(N) = \int_{\mathrm{Haar}}\int_{\mathrm{Haar}} |\langle\psi|\psi'\rangle|^{2t} d\psi d\psi'$. The t-th frame potential links to the value of $\mathcal{D}^{(t)}(C)$ in the following manner; that is, for an arbitrary positive integer t, it holds

$$\mathcal{F}^{(t)}(C) - \mathcal{F}^{(t)}_{\mathrm{Haar}}(N) = \mathcal{D}^{(t)}(C) \geq 0. \tag{6.20}$$

The equality in the last inequality holds if and only if the ensemble of $|\psi_\theta\rangle$ is a state t-design [122–124]. Therefore, the ansatz C with smaller $\mathcal{F}^{(t)}(C)$ has a higher expressibility. Additionally, since $\mathcal{F}^{(t)}(C)$ is lower bounded by $\mathcal{F}^{(t)}_{\mathrm{Haar}}(N)$, we can use the t-th frame potential as an indicator for the non-uniformity in the state distribution. In Sec. 6.2.3, we compute $\mathcal{F}^{(1)}(C)$ and $\mathcal{F}^{(2)}(C)$ for several ansatz C.

KL-Divergence

We define another measure of expressibility by using the fidelity distribution. The fidelity distribution is the distribution of the quantity $F = |\langle\psi_\theta|\psi_\phi\rangle|^2$, when the circuit parameters $\theta \in \Theta$ and $\phi \in \Phi$ are randomly sampled. Given the probability distribution as $P(C, F)$, the t-th frame potential is the t-th moment of $P(C, F)$. Thus, $P(C, F)$, contains more information for quantifying the randomness of C than $\mathcal{F}^{(t)}(C)$. Thus, as another indicator, we use the following measure to quantify the expressibility of C:

$$\mathcal{E}(\mathrm{C}) = D_{\mathrm{KL}}\left(P(C,F)\|P_{\mathrm{Haar}}(F)\right) = \int_0^1 P(C,F)\log\frac{P(C,F)}{P_{\mathrm{Haar}}(F)}dF, \tag{6.21}$$

where $D_{KL}(q\|p)$ is the KL divergence between the probability distributions q and p. Also $P_{\mathrm{Haar}}(F)$ is the probability distribution of the fidelity $F = |\langle\psi|\psi'\rangle|^2$, where $|\psi\rangle$ and $|\psi'\rangle$ are sampled from the Haar distribution. In Ref. [125], the equality $P_{\mathrm{Haar}}(F) = (N-1)(1-F)^{N-2}$ is derived, where N is the dimension of Hilbert space. In general, $D_{KL}(q\|p) = 0$ if and only if $q = p$, and small $D_{KL}(q\|p)$ means that the distributions q and p are close with each other. Thus, the smaller value of $\mathcal{E}(C)$ means that the ansatz C has a higher expressibility. Hence, we use $\mathcal{E}(C)$ as an indicator for quantifying the expressibility of an ansatz.

Finally, we emphasize that the t-th moment of $P(C, F)$ and $P_{\mathrm{Haar}}(F)$ are $\mathcal{F}^{(t)}(C)$ and $\mathcal{F}^{(t)}_{\mathrm{Haar}}(N)$, respectively. Thus, if the values of $\mathcal{F}^{(t)}(C)$ is close to $\mathcal{F}^{(t)}_{\mathrm{Haar}}(N)$, the value of $\mathcal{E}(C)$ is also close to zero.

6.2.2 Ansatzes

In this thesis, we will investigate the three types of ansatzes introduced in the previous sections: the hardware efficient ansatz (HEA), the alternating layered ansatz (ALT), and the tensor product ansatz (TEN). Let us summarize the three ansatzes in the following.

Hardware Efficient Ansatz

The HEA circuit consists of multiple layers of parametrized single qubit gates and entanglers which entangle all qubits. In the following, let $C^{\ell,n}_{\mathrm{HEA}}$ be the class of HEA with ℓ layers where each layer contains n-qubits.

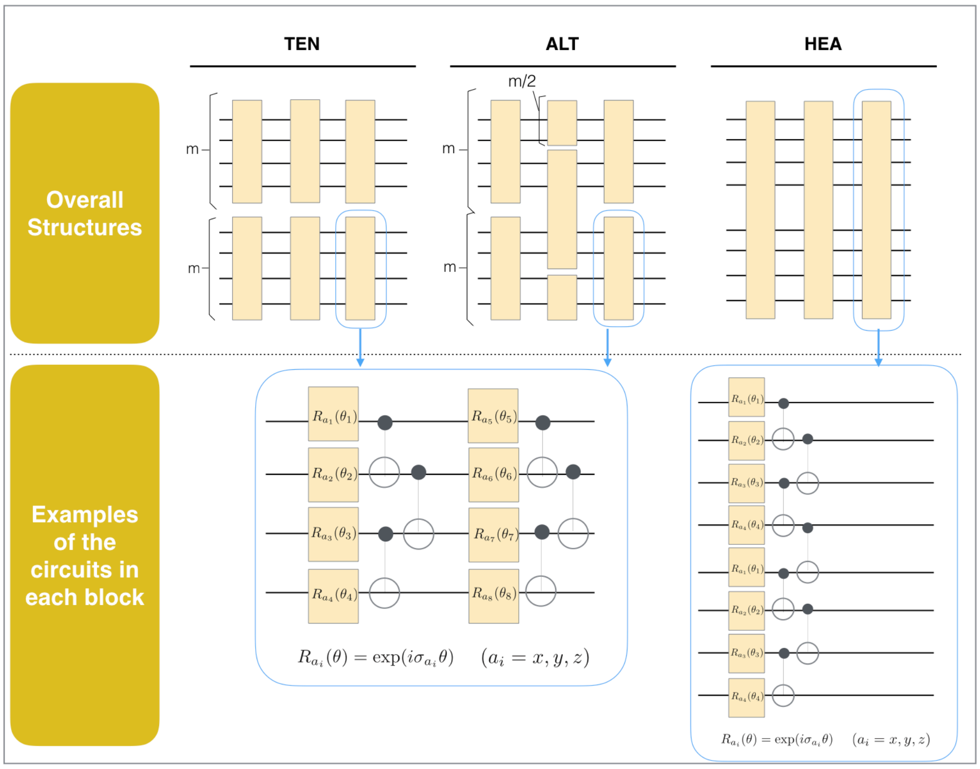

TEN
ALT
HEA
Overall Structures
m
m
m/2
m
m
Examples of the circuits in each block
$R_{a_i}(\theta) = \exp(i\sigma_{a_i}\theta) \quad (a_i = x, y, z)$

Figure 6.5: Examples of TEN, ALT, and HEA. The upper figures: overall structures of the ansatzes in the case $n = 8$ and $\ell = 3$ (and $m = 4$ for TEN and ALT). The lower figure: circuits inside the blocks, where σ_{a_i} ($a_i \in \{x, y, z\}$) is the Pauli matrix and $\theta_i \in [0, 2\pi]$ is a parameter. Reprinted figure from [DOI: 10.22331/Q-2021-04-19-434]. Creative Commons Attribution 4.0 International license. Copyright 2021 by the Kouhei Nakaji and Naoki Yamamoto.

Alternating Layered Ansatz

The ALT is composed of ℓ layers as in HEA. Each layer has separated blocks, where each block has parametrized single-qubit rotational gates and entanglers that entangle all qubits inside the block. In the odd-number-labeled layers, each block contains m qubits, so that m is an even number and n/m is an integer. Conversely, the odd-number-labeled layers contain n/m blocks which operate on $\{1, \ldots, m\}$, $\{m+1 \ldots, 2m\}$, ..., and $\{n-m+1, \ldots, n\}$ qubits. The even-number-labeled layers contain $n/m + 1$ blocks which operate on $\{1, \ldots, m/2\}$, $\{m/2 + 1, \ldots, 3m/2\}$, ..., and $\{n - m/2 + 1, \ldots, n\}$ qubits; namely, the first and the last block operate on $m/2$ qubits, while the others operate on m qubits. In the following, let us use $C^{\ell,m,n}_{\mathrm{ALT}}$ for denoting the class of ALT with the above-defined indices.

Tensor Product Ansatz

TEN also consists of ℓ layers, where each layer contains n/m blocks (we assume n/m is an integer), which contain repetitions of single-qubit rotation gates and entanglers combining all qubits in the block. Throughout all the layers, the blocks operate on $\{1, \ldots, m\}$, $\{m + 1 \ldots, 2m\}$, ..., and $\{n - m + 1, \ldots, n\}$ qubits. Therefore, TEN always generates a product state of the form $|\psi_1\rangle \otimes \cdots \otimes |\psi_{n/m}\rangle$ where each $|\phi_i\rangle$ is composed of m qubits. Let $C^{\ell,m,n}_{\mathrm{TEN}}$ be the class of TEN described above. In Fig. 6.5, the summary of the structures of these ansatzes is shown.

6.2.3 Expressibility of the ansatzes

In this section, we analyze the expressibility of the three ansatzes introduced in Section 6.2.2. Firstly, we analytically compute the first and the second frame potentials of those ansatzes and show that shallow ALT can have almost the same expressibility as that of HEA with suitable choice of ℓ and m. Next, we further confirm the result by a numerical simulation in terms of the KL-divergence.

Analytical expressions of the frame potential

First of all, we assume that the ensemble of the unitary matrices corresponding to $C^{\ell,n}_{\mathrm{HEA}}$ is 2-design. Similarly, we assume that the ensemble of the unitary matrices corresponding to each block of $C^{\ell,m,n}_{\mathrm{ALT}}$ and $C^{\ell,m,n}_{\mathrm{TEN}}$ are 2-design. Literatures [102] and [126] also adopt these assumptions. Note that such ensemble of unitary matrices can be generated by randomly choosing the parameters of the circuit having a specific structure.

Under the assumption, the following integration formulae hold. First, if the ensemble of $n \times n$ unitary matrices $\{U\}$ is 1-design, it holds:

$$\int_{1\mathrm{design}} dU U_{ij} U^*_{mk} = \frac{\delta_{im}\delta_{jk}}{n}, \tag{6.22}$$

where $\int_{1\mathrm{design}} dU$ is the integral over the 1-design ensemble of the unitary matrices. Note that if an ensemble of the unitary matrices is 2-design, the ensemble is also 1-design. Second, if the ensemble of $n \times n$ unitary matrices $\{U\}$ is 2-design, the following formula holds:

$$\int_{2\mathrm{design}} dU U_{i_1 j_1} U_{i_2 j_2} U^*_{i'_1 j'_1} U^*_{i'_2 j'_2} = \sum_{k=1}^{4} \lambda^{(n)}_k \Delta^k_{i_1 j_1 i_2 j_2 i'_1 j'_1 i'_2 j'_2}, \tag{6.23}$$

where

$$\begin{aligned}
\lambda^{(n)}_1 = \lambda^{(n)}_2 = \frac{1}{2^{2n}-1}&, \quad \lambda^{(n)}_3 = \lambda^{(n)}_4 = -\frac{1}{(2^{2n}-1)2^n}, \\
\Delta^1_{i_1 j_1 i_2 j_2 i'_1 j'_1 i'_2 j'_2} &= \delta_{i_1 i'_1}\delta_{j_1 j'_1}\delta_{i_2 i'_2}\delta_{j_2 j'_2}, \\
\Delta^2_{i_1 j_1 i_2 j_2 i'_1 j'_1 i'_2 j'_2} &= \delta_{i_1 i'_2}\delta_{j_1 j'_2}\delta_{i_2 i'_1}\delta_{j_2 j'_1}, \\
\Delta^3_{i_1 j_1 i_2 j_2 i'_1 j'_1 i'_2 j'_2} &= \delta_{i_1 i'_1}\delta_{j_1 j'_2}\delta_{i_2 i'_2}\delta_{j_2 j'_1}, \\
\Delta^4_{i_1 j_1 i_2 j_2 i'_1 j'_1 i'_2 j'_2} &= \delta_{i_1 i'_2}\delta_{j_1 j'_1}\delta_{i_2 i'_1}\delta_{j_2 j'_2},
\end{aligned} \tag{6.24}$$

and $\int_{2\mathrm{design}} dU$ is the integral over the 2-design ensemble of the unitary matrices. We use these formulae to derive the theorems shown below.

The First Frame Potential

For the first frame potentials, the following theorem holds.

Theorem 2. *If the ensemble of the unitary matrices corresponding to $C^{\ell,n}_{\mathrm{HEA}}$ and the ensemble of the unitary matrices corresponding to each block of $C^{\ell,m,n}_{\mathrm{ALT}}$ and $C^{\ell,m,n}_{\mathrm{TEN}}$ are 2-design, then the following equalities hold:*

$$\mathcal{F}^{(1)}(C^{\ell,n}_{\mathrm{HEA}}) = \mathcal{F}^{(1)}(C^{\ell,m,n}_{\mathrm{ALT}}) = \mathcal{F}^{(1)}(C^{\ell,m,n}_{\mathrm{TEN}}) = \mathcal{F}^{(1)}_{\mathrm{Haar}}(2^n). \tag{6.25}$$

We can readily prove he equality, $\mathcal{F}^{(1)}(C^{\ell,n}_{\mathrm{HEA}}) = \mathcal{F}^{(1)}_{\mathrm{Haar}}(2^n)$, from the assumption of the theorem, since if the ensemble of the unitary matrices corresponding to the circuit is 2-design, the ensemble of the states generated by the circuit is a state 2-design (and therefore a state 1-design). For the other equalities, the proof is given in Appendix B.1. It should be noted that, accordingly, the ensembles of the states generated by $C^{\ell,n}_{\mathrm{HEA}}$, $C^{\ell,m,n}_{\mathrm{ALT}}$, and $C^{\ell,m,n}_{\mathrm{TEN}}$ are all 1-design.

The Second Frame Potential

The second frame potential of the Haar random circuits, $\mathcal{F}^{(2)}_{\mathrm{Haar}}(2^n)$, can be computed as

$$\mathcal{F}^{(2)}_{\mathrm{Haar}}(2^n) = \int_0^1 dF F^2 (2^n - 1)(1-F)^{2^n-2} = \frac{1}{(2^n+1)2^{n-1}}. \tag{6.26}$$

Then, for $C^{\ell,n}_{\mathrm{HEA}}$ and $C^{\ell,m,n}_{\mathrm{TEN}}$, the following theorem holds.

Theorem 3. *If the ensemble of the unitary matrices corresponding to $C_{\mathrm{HEA}}^{\ell,n}$ and the ensemble of the unitary matrices corresponding to each block of $C_{\mathrm{TEN}}^{\ell,m,n}$ are 2-design, then the following equalities hold:*

$$\mathcal{F}^{(2)}(C_{\mathrm{HEA}}^{\ell,n}) = \mathcal{F}_{\mathrm{Haar}}^{(2)}(2^n), \tag{6.27}$$

$$\mathcal{F}^{(2)}(C_{\mathrm{TEN}}^{\ell,m,n}) = 2^{\frac{n}{m}-1} \cdot \frac{2^n+1}{(2^m+1)^{\frac{n}{m}}} \mathcal{F}_{\mathrm{Haar}}^{(2)}(2^n). \tag{6.28}$$

We can readily prove the equality (6.27) from the assumption of the theorem, because, as we mentioned above, if the ensemble of the unitary matrices corresponding to the circuit is 2-design, the ensemble of the states generated by the circuit is a state 2-design. For Eq.(6.28), we give the proof in Appendix B.2. From this theorem we find that $\mathcal{F}^{(2)}(C_{\mathrm{TEN}}^{\ell,m,n})$ is always larger than $\mathcal{F}_{\mathrm{Haar}}^{(2)}(2^n)$; in particular, $\mathcal{F}^{(2)}(C_{\mathrm{TEN}}^{\ell,m,n}) \simeq 2^{n/m-1}\mathcal{F}_{\mathrm{Haar}}^{(2)}(2^n)$ for large n, meaning that the expressibility of TEN is much smaller than that of HEA in the sense of the frame potential.

For ALT, it is complicated to obtain an explicit formula. Hence in Theorem 4 below, let us provide a formula for computing the values of $\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{2,m,n})$ and $\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{3,m,n})$; we left the computation methods for the other ℓs are for future work. As a preparation for stating a theorem, below we define a 16-dimensional vector $\mathbf{a}(2,m)$, a 16×16 matrix $B(2,m)$, a 64-dimensional vector $\mathbf{a}(3,m)$, and a 64×64 matrix $B(3,m)$. Given integers $k_a, k_b \in \{1,2,3,4\}$, the $(4(k_a-1)+k_b)$-th component of the vector $\mathbf{a}(2,m)$ is defined as

$$\mathbf{a}(2,m)_{4(k_a-1)+k_b} = \int_{\mathrm{2design}} dPdQ \sqrt{\lambda_{k_a}^{(m)}\lambda_{k_b}^{(m)}}\, \Delta^{(k_a,k_b)}(P,Q), \tag{6.29}$$

where $\Delta^{(k_a,k_b)}(P,Q)$ is the function of $m/2 \times m/2$ unitary matrices P and Q:

$$\Delta^{(k_a,k_b)}(P,Q) = \sum_{\substack{uu'ii'\\ jj'll'}} \sum_{pp'qq'} \Delta_{u0u'0i0i'0}^{k_a} \Delta_{p0p'0q0q'0}^{k_b} P_{ju}P_{j'u'}P_{li}^{*}P_{l'i'}^{*}Q_{lp}Q_{l'p'}Q_{jq}^{*}Q_{j'q'}^{*}. \tag{6.30}$$

Next, given integers $k_a, k_b, k_c \in \{1,2,3,4\}$, the $(16(k_a-1)+4(k_b-1)+k_c)$-th component of the vector $\mathbf{a}(3,m)$ is defined as

$$\mathbf{a}(3,m)_{16(k_a-1)+4(k_b-1)+k_c} = \int_{\mathrm{2design}} dPdQ \sqrt{\lambda_{k_a}^{(m)}\lambda_{k_b}^{(m)}\lambda_{k_c}^{(m)}}\, \Delta^{(k_a,k_b,k_c)}(P,Q), \tag{6.31}$$

where $\Delta^{(k_a,k_b,k_c)}(P,Q)$ is a function of $m/2 \times m/2$ unitary matrices P and Q:

$$\Delta^{(k_a,k_b,k_c)}(P,Q) = \sum_{\substack{uu'ii'\\ jj'll'}} \sum_{\substack{rr'tt'\\ pp'qq'}} \Delta_{u0u'0i0i'0}^{k_a} \Delta_{jlj'l'trt'r'}^{k_b} \Delta_{p0p'0q0q'0}^{k_c} P_{tu}P_{t'u'}P_{ji}^{*}P_{j'i'}^{*}Q_{lp}Q_{l'p'}Q_{rq}^{*}Q_{r'q'}^{*}. \tag{6.32}$$

Also, given integers $k_a, k_b, k_c, k_d \in \{1,2,3,4\}$, the $(4(k_a-1)+4k_b, 4(k_c-1)+k_d)$-th component of the matrix $B(m,2)$ is defined as

$$B(2,m)_{4(k_a-1)+k_b,4(k_c-1)+k_d} = \int_{\mathrm{2design}} dPdQ \sqrt{\lambda_{k_a}^{(m)}\lambda_{k_b}^{(m)}} \sqrt{\lambda_{k_c}^{(m)}\lambda_{k_d}^{(m)}}\, \Delta^{(k_a,k_b,k_c,k_d)}(P,Q), \tag{6.33}$$

where $\Delta^{(k_a,k_b,k_c,k_d)}(P,Q)$ is a function of $m \times m$ matrices P and Q:

$$\Delta^{(k_a,k_b,k_c,k_d)}(P,Q) = \sum_{\substack{u_2u_2'i_2i_2'\\ j_2j_2'l_2l_2'}} \sum_{p_2p_2'q_2q_2'} \sum_{\substack{u_3u_3'i_3i_3'\\ j_3j_3'l_3l_3'}} \sum_{p_3p_3'q_3q_3'} \Delta_{u_20u_2'0i_20i_2'0}^{k_a} \Delta_{p_20p_2'0q_20q_2'0}^{k_b} \Delta_{u_30u_3'0i_30i_3'0}^{k_c} \Delta_{p_30p_3'0q_30q_3'0}^{k_d}$$
$$\times P_{j_2u_2}^{j_3u_3} P_{j_2'u_2'}^{j_3'u_3'} P_{l_2i_2}^{*l_3i_3} P_{l_2'i_2'}^{*l_3'i_3'} Q_{l_2p_2}^{l_3p_3} Q_{l_2'p_2'}^{l_3'p_3'} Q_{j_2q_2}^{*j_3q_3} Q_{j_2'q_2'}^{*j_3'q_3'} \tag{6.34}$$

For the matrix component $M_{i,j}^{s,t}$, the upper indices correspond to the first $m/2$ qubits and the lower indices correspond to the last $m/2$. Given integers $k_a, k_b, k_c, k_d, k_e, k_f \in \{1,2,3,4\}$, the

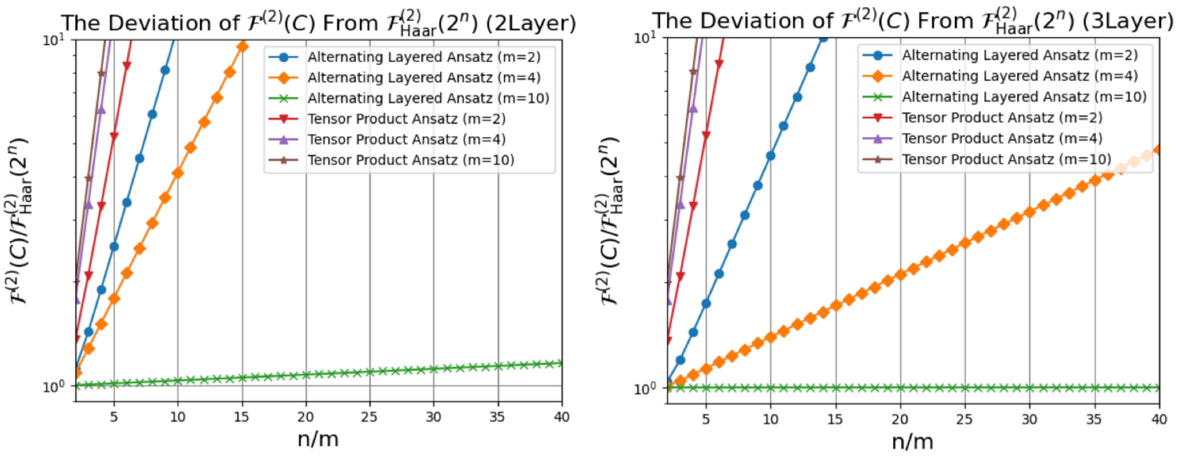


Figure 6.6: The values of $\mathcal{F}^{(2)}(C)/\mathcal{F}^{(2)}_{\mathrm{Haar}}(2^n)$ for ALT and TEN. The left and right figures show the case $\ell = 2$ and $\ell = 3$, respectively. The vertical axes are in the logarithmic scale. If $\mathcal{F}^{(2)}(C)/\mathcal{F}^{(2)}_{\mathrm{Haar}}(2^n)$ is close to 1, this means that the expressibility of the ansatz C is relatively high. Reprinted figure from [DOI: 10.22331/Q-2021-04-19-434]. Creative Commons Attribution 4.0 International license. Copyright 2021 by the Kouhei Nakaji and Naoki Yamamoto.

$(16(k_a-1)+4(k_b-1)+k_c, 16(k_d-1)+4(k_e-1)+k_f)$-th component of the matrix $B(m,3)$ is defined as

$$\begin{aligned} &B(3,m)_{16(k_a-1)+4(k_b-1)+k_c,16(k_d-1)+4(k_e-1)+k_f} \\ &\qquad = \int_{\mathrm{2design}} dPdQ \sqrt{\lambda^{(m)}_{k_a}\lambda^{(m)}_{k_b}\lambda^{(m)}_{k_c}} \sqrt{\lambda^{(m)}_{k_d}\lambda^{(m)}_{k_e}\lambda^{(m)}_{k_f}}\, \Delta^{(k_a,k_b,k_c,k_d,k_e,k_f)}(P,Q), \end{aligned} \tag{6.35}$$

where $\Delta^{(k_a,k_b,k_c,k_d,k_e,k_f)}(P,Q)$ is a function of $m\times m$ matrices P and Q:

$$\begin{aligned} \Delta^{(k_a,k_b,k_c,k_d,k_e,k_f)}(P,Q) = &\sum_{\substack{u_2u'_2i_2i'_2\\ j_2j'_2l_2l'_2}} \sum_{\substack{r_2r'_2t_2t'_2\\ p_2p'_2q_2q'_2}} \sum_{\substack{u_3u'_3i_3i'_3\\ j_3j'_3l_3l'_3}} \sum_{\substack{r_3r'_3t_3t'_3\\ p_3p'_3q_3q'_3}} \Delta^{k^a_{11}}_{u_20u'_20i_20i'_20} \Delta^{k^a_{31}}_{j_2l_2j'_2l'_2t_2r_2t'_2r'_2} \Delta^{k^b_{11}}_{p_20p'_2q_20q'_20} \\ &\times \Delta^{k^a_{12}}_{u_30u'_30i_30i'_30} \Delta^{k^a_{32}}_{j_3l_3j'_3l'_3t_3r_3t'_3r'_3} \Delta^{k^b_{12}}_{p_30p'_3q_30q'_30} P^{t_3u_3}_{t_2u_2} P^{t'_3u'_3}_{t'_2u'_2} P^{*j_3i_3}_{j_2i_2} P^{*j'_3i'_3}_{j'_2i'_2} \\ &\times Q^{l_3p_3}_{l_2p_2} Q^{l'_3p'_3}_{l'_2p'_2} Q^{*r_3q_3}_{r_2q_2} Q^{*r'_3q'_3}_{r'_2q'_2}. \end{aligned} \tag{6.36}$$

Now we can give the theorem as follows (the proof is given in Appendix B.3).

Theorem 4. *If the ensemble of the unitary matrices corresponding to each block of $C^{2,m,n}_{\mathrm{ALT}}$ and $C^{3,m,n}_{\mathrm{ALT}}$ are 2-design, then we have*

$$\mathcal{F}^{(2)}(C^{2,m,n}_{\mathrm{ALT}}) = \mathbf{a}(2,m)^{\mathrm{T}} B(2,m)^{\frac{n}{m}-1}\mathbf{a}(2,m), \tag{6.37}$$

$$\mathcal{F}^{(2)}(C^{3,m,n}_{\mathrm{ALT}}) = \mathbf{a}(3,m)^{\mathrm{T}} B(3,m)^{\frac{n}{m}-1}\mathbf{a}(3,m). \tag{6.38}$$

The vectors $\mathbf{a}(\ell,m)$ and the matrices $B(\ell,m)$ for $\ell=2,3$ are obtained by directly computing Eqs. (6.29), (6.31), (6.33), and (6.35), which then lead to $\mathcal{F}^{(2)}(C^{2,m,n}_{\mathrm{ALT}})$ and $\mathcal{F}^{(2)}(C^{3,m,n}_{\mathrm{ALT}})$. Now our interest is in the gap of these quantities from $\mathcal{F}^{(t)}_{\mathrm{Haar}}(2^n)$, to see the expressibility of ALT. For this purpose, in Fig. 6.6 we show $\mathcal{F}^{(2)}(C^{2,m,n}_{\mathrm{ALT}})/\mathcal{F}^{(2)}_{\mathrm{Haar}}(2^n)$ and $\mathcal{F}^{(2)}(C^{3,m,n}_{\mathrm{ALT}})/\mathcal{F}^{(2)}_{\mathrm{Haar}}(2^n)$ as a function of n/m, for several values of (m, n). For comparison, $\mathcal{F}^{(2)}(C_{\mathrm{TEN}}^{2,m,n})/\mathcal{F}_{\mathrm{Haar}}^{(2)}(2^n)$ and $\mathcal{F}^{(2)}(C_{\mathrm{TEN}}^{3,m,n})/\mathcal{F}_{\mathrm{Haar}}^{(2)}(2^n)$ are shown in the figure. Recall that, if this measure takes a smaller value, this means that the corresponding ansatz has a higher expressibility. Here is the list of notable points:

- For any pair of (n, m), it is clear that $\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{\ell,m,n})$ is much smaller than $\mathcal{F}^{(2)}(C_{\mathrm{TEN}}^{\ell,m,n})$ for both $\ell = 2, 3$ meaning that, as expected, ALT has a much higher expressibility than TEN.
- For any pair of (n, m), $\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{2,m,n}) > \mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{3,m,n})$ holds, i.e., as ℓ increases, the expressibility increases.
- For any fixed n/m, the ALT with bigger m always has a higher expressibility. For example, the ALT with $(n, m) = (50, 10)$ has a higher expressibility than the ALT with $(n, m) = (20, 4)$. This is because, if the structure of the circuit (the number of division in each layer for making the block) is the same, then an ALT with bigger block components has a higher expressibility.
- For a fixed n, we have ALT with the smaller second order frame potential by taking m bigger. For instance $n = 100$, we have $\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{2,2,100}) > \mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{2,4,100}) > \mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{2,10,100})$. That is, for a limited number of available qubits, the ALT with less blocks has a higher expressibility.
- $\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{\ell,m,n}) \simeq \mathcal{F}_{\mathrm{Haar}}^{(2)}(2^n)$ when $m = 10$ for all n/m within the figure and for both $\ell = 2, 3$. Hence the ALT composed from the blocks with $m = 10$ qubits in each layer has almost the same expressibility as the HEA without respect to the total qubits number, n. In other words, for a given HEA with fixed n, we can divide each layer into separated 10-qubits blocks to make an ALT, without decreasing the expressibility.

The last point is of particular important in our scenario. That is, we are concerned with the condition on the number m such that $\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{\ell,m,n}) \simeq \mathcal{F}_{\mathrm{Haar}}^{(2)}(2^n)$ holds. The following Theorem 4 and the subsequent Corollary 1, which can be readily derived from the theorem, provide a means for evaluating such m.

Theorem 5. *If the ensemble of the unitary matrices corresponding to each block of $C_{\mathrm{ALT}}^{2,m,n}$ and $C_{\mathrm{ALT}}^{3,m,n}$ are 2-design, then the following inequalities hold:*

$$\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{2,m,n}) < \left(1 + \frac{1}{2^n}\right)\left(1 + \frac{1.2}{2^m}\right)^2 \left(1 + 8\left(\left(1 + \frac{20.8}{2^{m/2}}\right)^{\frac{n}{m}-1} - 1\right)\right)\mathcal{F}_{\mathrm{Haar}}^{(2)}(2^n), \tag{6.39}$$

$$\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{3,m,n}) < \left(1 + \frac{1}{2^n}\right)\left(1 + \frac{1.2}{2^m}\right)^2 \left(1 + 32\left(\left(1 + \frac{83.2}{2^{m/2}}\right)^{\frac{n}{m}-1} - 1\right)\right)\mathcal{F}_{\mathrm{Haar}}^{(2)}(2^n). \tag{6.40}$$

Corollary 1. *If $m = 2a \log_2 n$ and $143/(an^{a-1}\log_2 n) < 1$,*

$$\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{2,m,n}) < \left(1 + \frac{1}{2^n}\right)\left(1 + \frac{1.2}{n^{2a}}\right)^2 \left(1 + \frac{143}{an^{a-1}\log_2 n}\right)\mathcal{F}_{\mathrm{Haar}}^{(2)}(2^n). \tag{6.41}$$

If $m = 2a \log_2 n$ and $2288/(an^{a-1}\log_2 n) < 1$,

$$\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{3,m,n}) < \left(1 + \frac{1}{2^n}\right)\left(1 + \frac{1.2}{n^{2a}}\right)^2 \left(1 + \frac{2288}{an^{a-1}\log_2 n}\right)\mathcal{F}_{\mathrm{Haar}}^{(2)}(2^n). \tag{6.42}$$

The proof is in Appendix B.4 and Appendix B.5. Recall from Eq. (6.20) that $\mathcal{F}^{(t)}(C) \geq \mathcal{F}_{\mathrm{Haar}}^{(t)}(N)$ holds for any ansatz C. Therefore, if $m \geq 4\log n$ and n is enough large, Corollary 1 implies that $\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{2,m,n}) \sim \mathcal{F}_{\mathrm{Haar}}^{(2)}(2^n)$ and $\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{3,m,n}) \sim \mathcal{F}_{\mathrm{Haar}}^{(2)}(2^n)$. This means that the ensembles of the states generated by $C_{\mathrm{ALT}}^{2,m,n}$ and $C_{\mathrm{ALT}}^{3,m,n}$ are almost 2-design. Hence in this case, from Theorem 2, the expressibility of ALT is as high as that of HAE. It is worth mentioning that, when $m = O(\log_2 n)$, the vanishing gradient problem does not happen in ALT as long as the cost function is local and ℓ is small [126]. More precisely, it was shown there that the variance of the gradient of such a cost function is larger than the value proportional to $O(1/2^{m\ell})$; thus, by taking $m = O(\log_2 n)$, the variance decreases with only $O(1/\mathrm{poly}(n))$ as a function of n, whereas in the HEA case the same variance decreases exponentially fast as n becomes large. Therefore, the expressibility and the trainability coexists in the shallow ALT with $m = O(\log_2 n)$.

Expressibility measured by KL divergence

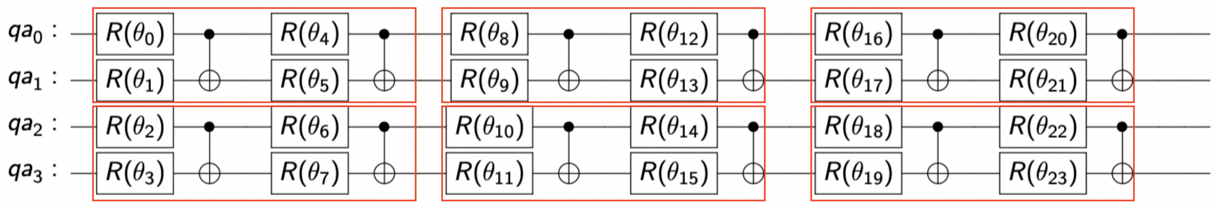
n	**Ansatz**	ℓ	m	Depth of each block	# of gate parameters	Example of the circuit
4	**TEN**	2	2	2	16	-
		3	2	2	24	Fig. 6.7a
	ALT	2	2	2	16	-
		3	2	2	24	Fig. 6.7b
	HEA	4	-	-	16	Fig. 6.7c
6	**TEN**	2	2	2	24	-
		3	2	2	36	-
	ALT	2	2	2	24	-
		3	2	2	36	-
	HEA	6	-	-	36	-
8	**TEN**	2	2	2	32	-
			4	4	64	-
		3	2	2	48	-
			4	4	96	-
	ALT	2	2	2	32	-
			4	4	64	-
		3	2	2	48	-
			4	4	96	-
	HEA	8	-	-	64	-

Table 6.1: Parameters chosen for computing the KL-divergence. The number of gate parameters are computed by $n \times \ell \times$ (Depth of each block) for TEN and ALT, and $n \times \ell$ for HEA.

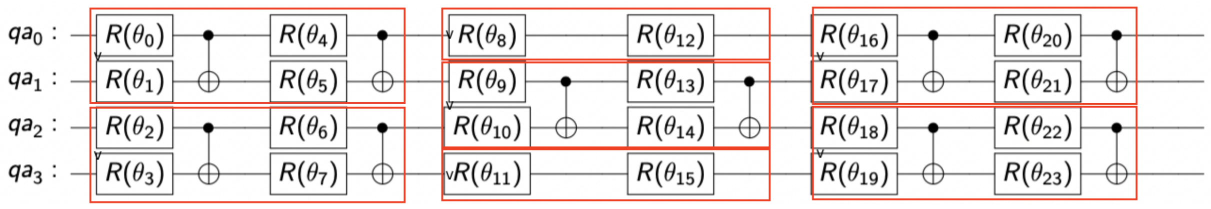
In Subsection 6.2.3, we showed that the first two moments of $P(C_{\rm ALT}^{\ell,m,n}, F)$ and $P(C_{\rm HEA}^{\ell,n}, F)$ are close to those of $P_{\rm Haar}(F)$ as long as we choose $m = O(\log_2 n)$, and the unitary corresponding to the block is sufficiently random. The result implies that both the distributions $P(C_{\rm ALT}^{\ell,m,n}, F)$ and $P(C_{\rm HEA}^{\ell,n}, F)$ themselves are close to $P_{\rm Haar}(F)$; namely $P(C_{\rm ALT}^{\ell,m,n}, F) \simeq P(C_{\rm HEA}^{\ell,n}, F) \simeq P_{\rm Haar}(F)$. In this subsection, we compare the distributions and support the conjecture. In this subsection, to support the conjecture, we evaluate the values of KL-divergence $\mathcal{E}(C) = D_{\rm KL}\left(P(C,F) \| P_{\rm Haar}(F)\right)$ for the case $C = C_{\rm ALT}^{\ell,m,n}$ and $C = C_{\rm HEA}^{\ell,n}$, in addition to $C = C_{\rm TEN}^{\ell,m,n}$ for comparison with various sets of (ℓ, m, n). In particular, we focus on the relationship between the values of $\mathcal{F}^{(2)}(C)$ and $\mathcal{E}(C)$, and check if $\mathcal{F}^{(2)}(C) \simeq \mathcal{F}^{(2)}(C')$ would lead to $\mathcal{E}(C) \simeq \mathcal{E}(C')$ for a fixed n.

We summarize the parameters chosen for calculating the KL-divergence in Table 6.1. The structure of the circuits are chosen to be similar to those used in Section 6.2.3; namely, for TEN and ALT, we set the depth of the circuits inside each block is to m so that the ensemble of the unitary matrices corresponding to each block becomes close to 2-design [127, 128]; for HEA, ℓ is set to n so that the ensemble of the unitary matrices corresponding to the *whole circuits* becomes close to 2-design. We expect that $\mathcal{F}^{(2)}(C_{\rm ALT}^{3,2,4}) \approx \mathcal{F}^{(2)}(C_{\rm HEA}^{4,4})$ and $\mathcal{F}^{(2)}(C_{\rm ALT}^{3,4,8}) \approx \mathcal{F}^{(2)}(C_{\rm HEA}^{8,8})$, because, in Fig 6.6, we see that $\mathcal{F}^{(2)}(C_{\rm ALT}^{3,2,4})$ and $\mathcal{F}^{(2)}(C_{\rm ALT}^{3,4,8})$ are almost equals to the Haar counterparts when the ensembles of unitary matrices corresponding to each block are 2-design. Thus, we here check if $\mathcal{F}^{(2)}(C) \simeq \mathcal{F}^{(2)}(C')$ would mean $\mathcal{E}(C) \simeq \mathcal{E}(C')$ in these parameter sets. As an example of the circuit, the whole structure of $C_{\rm TEN}^{3,2,4}$, $C_{\rm ALT}^{3,2,4}$, and $C_{\rm HEA}^{4,4}$ in our settings are shown in Figs. 6.7a, 6.7b, and 6.7c, respectively. As illustrated in the figures, each layer is composed of parametrized single qubit gates and fixed 2-qubit CNOT gates.

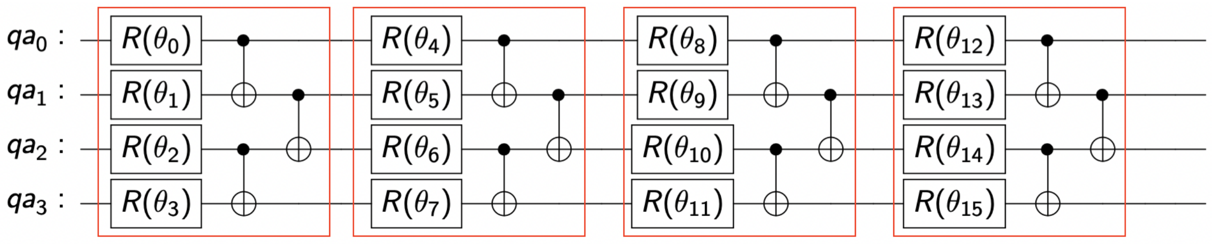
In each trial, we generate 200 states. When generating a state in each trial, we randomly choose the parameters and the type fo single-qubit gate of the circuit. That is, for the i-th single qubit gate $R_i(\theta_i) = \exp(\sigma_{a_i}\theta_i)$ with $a_i = \{x, y, z\}$ and $\theta_i \in [0, 2\pi]$, in each trial all a_i and θ_i are randomly chosen. Then 200 fidelity values are computed, which are then used to construct the histogram with 1000 bins to approximate the probability distribution $P(C, F)$. Increasing the number of generated states and the number of bins do not affect the following conclusions.



(a) The structure of $C^{3,2,4}_{\text{TEN}}$. The red boxes correspond to blocks.



(b) The structure of $C^{3,2,4}_{\text{ALT}}$. The red boxes correspond to blocks.



(c) The structure of $C^{4,4}_{\text{HEA}}$.

Figure 6.7: The structures of ansatzes in our setting. Reprinted figure from [DOI: 10.22331/Q-2021-04-19-434]. Creative Commons Attribution 4.0 International license. Copyright 2021 by the Kouhei Nakaji and Naoki Yamamoto.

In this setting, we show the KL divergences $\mathcal{E}(C^{\ell,m,n}_{\text{ALT}})$, $\mathcal{E}(C^{\ell,n}_{\text{HEA}})$, and $\mathcal{E}(C^{\ell,m,n}_{\text{TEN}})$ in Fig. 6.8. As a reference, we also show the values of $\mathcal{F}^{(2)}(C)/\mathcal{F}^{(2)}_{\text{Haar}}(2^n)$ computed from the second moment of the fidelity distributions. Each data point and associated error bar is the average and the standard deviation of 10 trials of computation, respectively. Here is the list of points:

- For a fixed n, $\mathcal{E}(C^{\ell,m,n}_{\text{TEN}})$ is always bigger than $\mathcal{E}(C^{\ell,m,n}_{\text{ALT}})$ and $\mathcal{E}(C^{\ell,n}_{\text{HEA}})$.
- As the number of layers increases, the KL-divergence decreases for fixed (m, n).
- For a fixed n, the tendency of the values of $\mathcal{F}^{(2)}(C)/\mathcal{F}^{(2)}_{\text{Haar}}(2^n)$ is strongly correlated with that of KL-divergence.
- As expected, $\mathcal{F}^{(2)}(C^{3,m,n}_{\text{ALT}}) \approx \mathcal{F}^{(2)}(C^{n,n}_{\text{HEA}})$ is realized when $(m, n) = (2, 4)$ and $(4, 8)$.
- $\mathcal{E}(C^{3,m,n}_{\text{ALT}})$ is as small as $\mathcal{E}(C^{n,n}_{\text{HEA}})$ in the parameter sets where $\mathcal{F}^{(2)}(C^{3,m,n}_{\text{ALT}}) \approx \mathcal{F}^{(2)}(C^{n,n}_{\text{HEA}})$ is realized, i.e., $(m, n) = (2, 4)$ and $(4, 8)$. This result implies that the state distribution in ALT is also close to that in HEA, in the setting where the second frame potential is close to $\mathcal{F}^{(2)}_{\text{Haar}}(2^n)$.

From some of the above observations, we find the strong correlation between $\mathcal{F}^{(2)}(C)/\mathcal{F}^{(2)}_{\text{Haar}}(2^n)$ and $\mathcal{E}(C)$; that is, as $\mathcal{F}^{(2)}(C)$ becomes close to 1, then $\mathcal{E}(C)$ becomes close to 0. Thus, combining the result obtained in Section 6.2.3, we get a clear evidence that, as far as $m = O(\log_2 n)$, $\mathcal{E}(C^{2,m,n}_{\text{ALT}}) \approx 0$ and $\mathcal{E}(C^{3,m,n}_{\text{ALT}}) \approx 0$ hold. That is, the high expressibility and trainability in ALT proven in Section 6.2.3 are assured also in terms of KL-divergence.

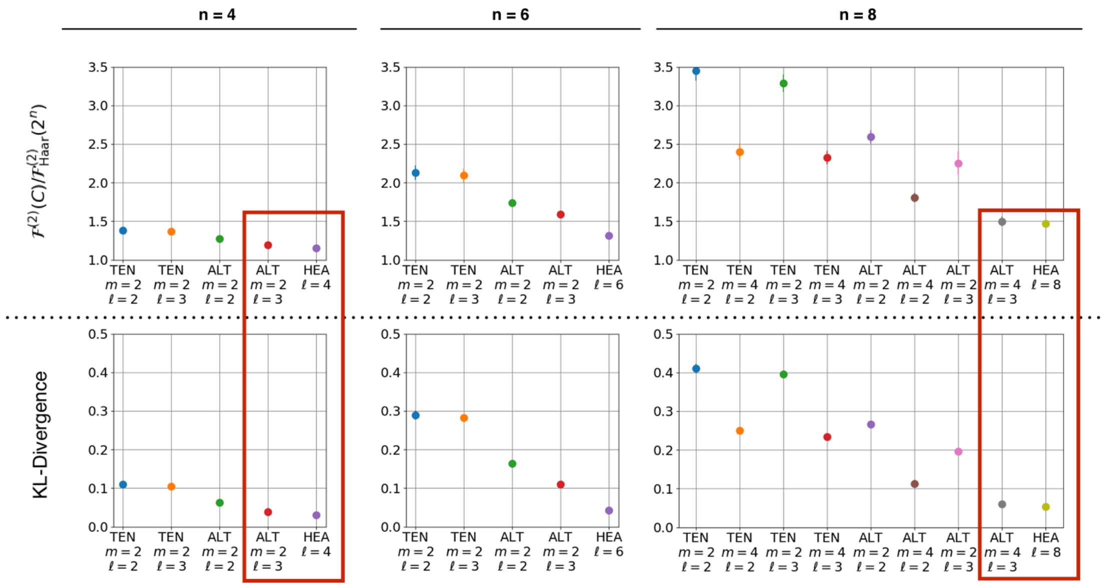


Figure 6.8: $\mathcal{F}^{(2)}(C)/\mathcal{F}^{(2)}_{\text{Haar}}(2^n)$ (top) and KL-divergence (bottom) for each ansatz. The sets of points with which $\mathcal{F}^{(2)}(C^{3,m,n}_{\text{ALT}}) \approx \mathcal{F}^{(2)}(C^{3,n}_{\text{HEA}})$ hold are enclosed by the red rectangles. Reprinted figure from [DOI: 10.22331/Q-2021-04-19-434]. Creative Commons Attribution 4.0 International license. Copyright 2021 by the Kouhei Nakaji and Naoki Yamamoto.

6.2.4 Numerical Experiment

Recall that ALT was originally introduced with the motivation to resolve the vanishing gradient problem in VQE, which has been often observed when using HEA; then we were concerned with the expressibility of ALT in VQE, meaning that ALT would not offer a chance to reach the optimal solution due to the possible loss of expressibility. But we now know that this concern has been resolved under some conditions, as concluded in the previous section; that is, the expressibility and the trainability coexists in the shallow ALT with $m = O(\log_2 n)$. This section provides a case study of VQE that implies this desirable fact.

We choose the Hamiltonian of 4-qubits Heisenberg model on a 1-dimensional lattice with periodic boundary conditions:

$$\mathcal{H} = \sum_{i=1}^{4} (\sigma_x^i \sigma_x^{i+1} + \sigma_y^i \sigma_y^{i+1} + \sigma_z^i \sigma_z^{i+1}), \tag{6.43}$$

where σ_a^i ($a \in \{x, y, z\}$) is the Pauli matrix that operates on the i-th qubit and $\sigma_a^5 = \sigma_a^1$. The goal of VQE problem is to find the minimum eigenvalue of $\mathcal{H}$, by calculating the mean energy $\langle \mathcal{H} \rangle = \langle \psi_\theta | \mathcal{H} | \psi_\theta \rangle = \langle 0 | U_C(\theta)^\dagger \mathcal{H} U_C(\theta) | 0 \rangle$ via a quantum computer and then updating the parameter $\theta \in \Theta$ to decrease $\langle \mathcal{H} \rangle$ via a classical computer, in each iteration. As ansatzes, $C^{3,2,4}_{\text{TEN}}$, $C^{3,2,4}_{\text{ALT}}$, and $C^{4,4}_{\text{HEA}}$ are chosen. As indicated in Fig. 6.8, the values of KL divergence corresponding to these ansatzes show that $\mathcal{E}(C^{3,2,4}_{\text{TEN}}) > \mathcal{E}(C^{3,2,4}_{\text{ALT}}) \simeq \mathcal{E}(C^{4,4}_{\text{HEA}})$. That is, this ALT has the expressibility as high as that of HEA, and further, it is expected to enjoy the trainability unlike the HEA.

The simulation results are shown in Fig. 6.9. In the top three subfigures of Fig. 6.9, the blue lines and the associated error bars represent the average and the standard deviation of $\langle \mathcal{H} \rangle$ in total 100 trials, respectively. In each trial, the initial parameters of the ansatz are randomly chosen, and the optimization to decrease $\langle \mathcal{H} \rangle$ in each iteration is performed by using the Adam Optimizer with learning rate 0.001 [76]. The green line shows the theoretical minimum energy (i.e., the ground energy) of $\mathcal{H}$. Also the bottom subfigures of Fig. 6.9 show three trajectories of $\langle \mathcal{H} \rangle$ (red lines) whose energies at the final iteration step are the smallest three.

The ansatz $C^{3,2,4}_{\text{TEN}}$, which has the least expressibility in the sense of frame potentials and the

KL divergence analysis, clearly gives the worst result; its least mean-energy is far above from the ground energy. This is simply because the state generated via $C_{\rm TEN}^{3,2,4}$ cannot represent the ground state for any parameter choice. The result on $C_{\rm HEA}^{4,4}$ is the second worst, which also does not reach the ground energy as in the case of TEN. Note that increasing the number of parameters does not change this result; that is, we also executed the simulation with $C_{\rm HEA}^{4,6}$ that has the same number of parameters as $C_{\rm ALT}^{3,2,4}$ but did not find a better result than that of $C_{\rm HEA}^{4,4}$. On the other hand, $C_{\rm ALT}^{3,2,4}$ succeeds in finding the ground state; in fact, 5 of the total 100 trajectories generated via this ALT reach the ground energy. Hence this result implies that, in this example, the expressibility and the trainability coexist in ALT, while the latter is lacking in HEA.

This better trainability of ALT than that of HEA could be explained in terms of the "magnitude" of the gradient vector $\nabla_\theta\langle\mathcal{H}\rangle = [\partial\langle\mathcal{H}\rangle/\partial\theta_1,\ldots,\partial\langle\mathcal{H}\rangle/\partial\theta_P]$, where P is the number of parameters. Now care should be taken to define an appropriate magnitude, because the focused ansatzes have different number of parameters. In this work, we regard $\partial\langle\mathcal{H}\rangle/\partial\theta_p, (p=1,\ldots,P)$ as random variables and, based on this view, define the magnitude of $\nabla_\theta\langle\mathcal{H}\rangle$ as the mean of the absolute value of those random variables:

$$\|g(\theta)\| = \frac{1}{P}\sum_{p=1}^{P}\left|\frac{\partial\langle\mathcal{H}\rangle}{\partial\theta_p}\right|. \tag{6.44}$$

We evaluate the average of the magnitude (6.44) over sample trajectories, at several values of energy reached through the update of θ. For this purpose, let $\theta_C^{(i)}(t)$ denote the vector of parameters of a given ansatz C at the t-th step (number of iteration) of the i-th trajectory, and $E_C^{(i)}(t) = \langle 0|U_C(\theta_C^{(i)}(t))^\dagger\mathcal{H}U_C(\theta_C^{(i)}(t))|0\rangle$ be the energy at $\theta_C^{(i)}(t)$. Next, to define the average of $\|g(\theta_C)\|$ over the sample trajectories at the given energy value E, let $t_E^{(i)}$ be the smallest integer such that the energy of the i-th trajectory satisfies $E_C^{(i)}(t_E^{(i)}) \leq E$; in other words, $t_E^{(i)}$ represents the number of iteration such that the i-th trajectory first reaches the value E. Note that some trajectories may not reach a given value E for all the repetition of θ_C. (For example, as seen above, all trajectories of TEN never reached the value $E=-7$.) Hence, let $\mathcal{I}_E$ be the set of index i such that the i-th trajectory reaches the value E at some point of $t_E^{(i)}$. We can now define the average of $\|g(\theta_C)\|$ as

$$\langle\|g(\theta_C)\|\rangle_E = \frac{1}{|\mathcal{I}_E|}\sum_{i\in\mathcal{I}_E}\left\|g(\theta_C^{(i)}(t_E^{(i)}))\right\|, \tag{6.45}$$

where $|\mathcal{I}_E|$ denotes the size of $\mathcal{I}_E$. Fig. 6.10 shows Eq. (6.45) for the specific values of E (integers from -7 to 0) for the three ansatzes $C_{\rm TEN}^{3,2,4}$, $C_{\rm ALT}^{3,2,4}$, and $C_{\rm HEA}^{4,4}$. The standard deviation of the average is indicated by the error bar. For instance, $\langle\|g(\theta_C)\|\rangle_E \simeq 5.6$ for the case of orange point (i.e., the case of ALT) at $E=-1$ was calculated with $|\mathcal{I}_E| = 100$; actually, all 100 trajectories become lower than $E=-1$. The figure shows that $\langle\|g(\theta_C)\|\rangle_E$ in ALT is always larger than that in HEA for all E, and this result is consistent to the theorems given in [126]. Such a larger gradient vector might enable ALT to circumvent possible flat energy landscapes and eventually realize the better trainability than HEA, but further studies are necessary to confirm this observation. Note that TEN has the largest values of $\langle\|g(\theta_C)\|\rangle_E$ when $E \geq -5$, which yet do not lead to the convergence to the global minimum due to the lack of expressibility.

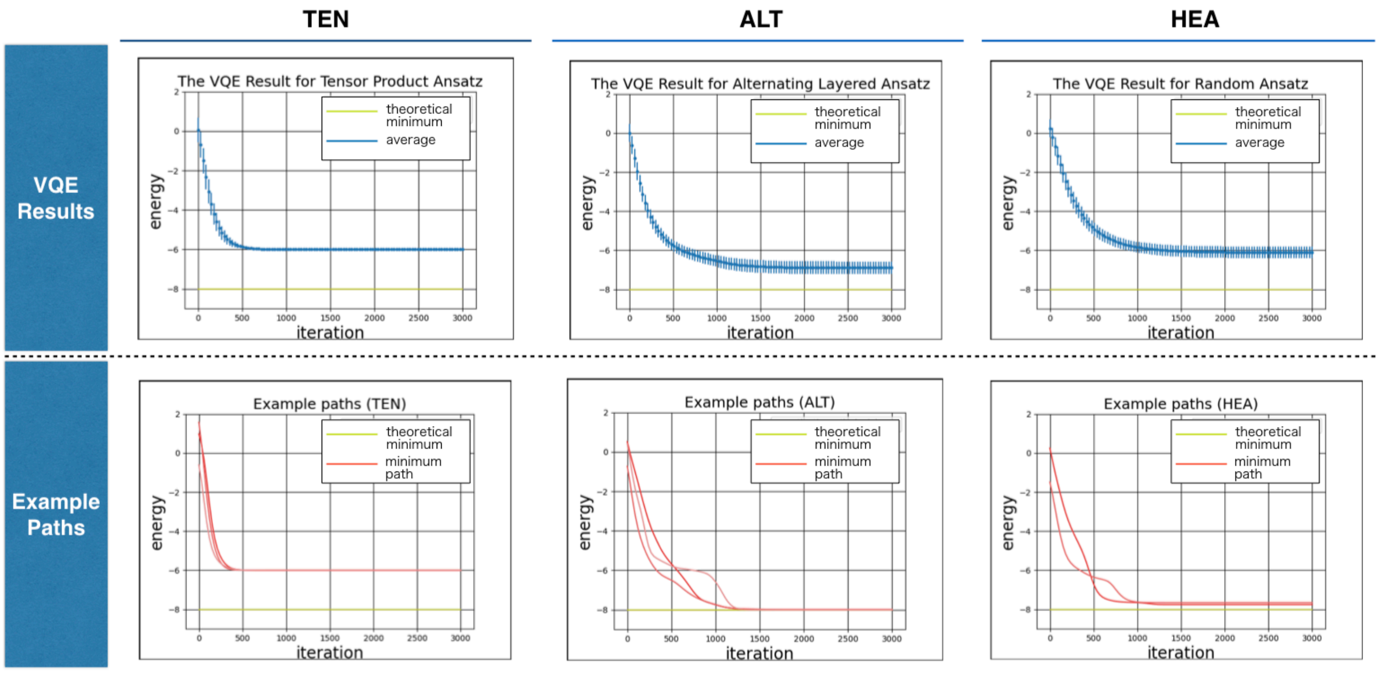


Figure 6.9: Top: Energy versus the iteration step in the VQE problem for the Hamiltonian (6.43), with the ansatz TEN (left), ALT (center), and HEA (right). The blue lines and the associated error bars represent the average and the standard deviation of the mean energies in total 100 trials, respectively; in each trial, the initial parameters of the ansatz are randomly chosen. Optimization to decrease $\langle \mathcal{H} \rangle$ in each iteration is performed by using Adam Optimizer with learning rate 0.001. Bottom: Three of 100 trajectories for each ansatz TEN (left), ALT (center), and HEA (right), indicated by red lines. The trajectories are chosen such that the energies at the final iteration step are the three smallest values. Reprinted figure from [DOI: 10.22331/Q-2021-04-19-434]. Creative Commons Attribution 4.0 International license. Copyright 2021 by the Kouhei Nakaji and Naoki Yamamoto.

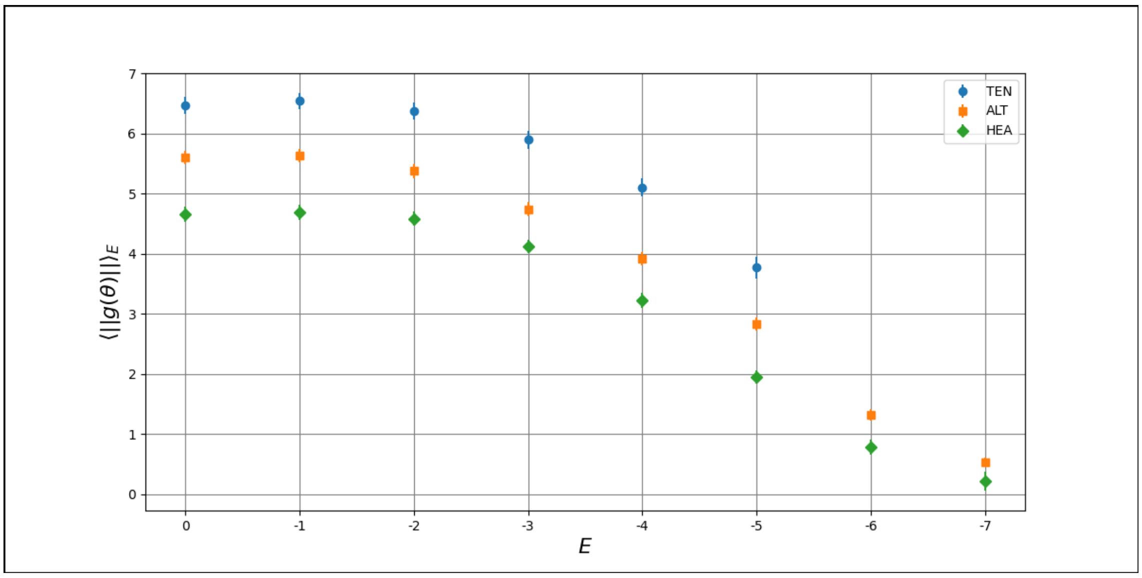


Figure 6.10: The average and the standard deviation of $\|g(\theta)\|$ versus E. The norm of the gradient is defined as $\|g(\theta)\| = \frac{1}{P}\sum_{p=1}^{P} \left| \frac{\partial \langle \mathcal{H} \rangle}{\partial \theta_p} \right|$. Reprinted figure from [DOI: 10.22331/Q-2021-04-19-434]. Creative Commons Attribution 4.0 International license. Copyright 2021 by the Kouhei Nakaji and Naoki Yamamoto.

Chapter 7

Conclusion and outlook

7.1 Conclusion

The quantum computer is expected to have many real-world applications, but the capability of the quantum devices developed in the next few decades is limited; a small number of available qubits and much noise are especially severe. In this thesis, we proposed algorithms maximizing the functionality of NISQ and assessed their feasibility. Chapter 2 and Chapter 3 were devoted to the basics of the quantum computation and NISQ. In Chapter 4 and later, we proposed and assessed the algorithms tailored for NISQ. Let us summarize the content of each chapter in the following.

In Chapter 2, we discussed the basics of quantum computation. After introducing the building blocks of the quantum computation, we demonstrated the power of the quantum computation by the phase estimation algorithm.

In Chapter 3, we discussed the noisy intermediate quantum device (NISQ). In the chapter, we first reviewed the characteristics of NISQ. Particularly, we discussed how the noise affects the computation results in NISQ. Next, we checked some methods to mitigate the error by the error mitigation technique. The main content of this section was the introduction of the variational quantum algorithm (VQA), which effectively utilizes NISQ to solve real-world problems. Finally, we discussed critical issues in VQA. We showed that the barren plateau issue, where the gradient of the cost function becomes exponentially small when the number of qubits increases, is especially severe.

In Chapter 4, we proposed the faster amplitude estimation (FAE), which is the method for efficiently extracting the computational results from the amplitude of the quantum state. We showed that the proposed algorithm without using the phase estimation algorithm successfully reduces the number of gates, and the computational complexity is as small as the phase estimation algorithm.

In Chapter 5, we proposed the quantum semi-supervised generative adversarial network (qS-GAN), which provides a method to utilize NISQ as an adversary for training a classifier. We demonstrated that the high expressibility of the quantum circuit contributes to the classifier's performance. Also, even with small noise, the classification performance did not worsen; it implies that qSGAN is suitable for the execution in NISQ.

Chapter 6 was devoted to the solution for the barren plateau issue of VQA reviewed in Chapter 3. In the former part of the chapter, we discussed the methods to avoid the barren plateau issue. Three methods were introduced there: the method devising the initializing strategy, the method using the layer-wise learning, and the method using the alternating layered ansatz (ALT). Additionally, we discussed that only the method utilizing ALT is theoretically guaranteed. However, we also showed that the applicability of ALT is unclear since ALT may not have enough expressibility to generate the solution of VQA. Based on the above discussion, in the latter part of Chapter 6, we examined the expressibility of ALT. Our conclusion of the analysis was in addition to that the shallow ALT can solve the barren plateau issue, the ALT has enough expressibility. Namely, in the expressibility measure of the frame potential and the

KL-divergence, those shallow ALTs had almost the same expressibility as HEAs, which are often used in hybrid algorithms but suffer from the barren plateau issue. In particular, we proved that such expressibility holds if the number of entangled qubits in each block are of the order of the logarithm of the number of all resource qubits, which is consistent with the previous result discussing the trainability of ALT. We also provided a case study of the VQE problem, implying that the ALT enjoys both the expressibility and trainability.

7.2 Outlook

There are open problems regarding the contents we discussed in Chapter 4, Chapter 5, and Chapter 6. Let us show the problems in the following.

As for FAE, the main issue is how to reduce the effect of noise for realizing practical applications in the current device. We list the open problems as follows:

- Are there any methods to reduce the number of two-qubit gates in the amplitude amplification operator? As we see in Fig. 4.1, the amplitude amplification operator includes a multi-controlled operation, which still requires a lot of two-qubit gates. Since we repeatedly use the operator, the impact of reducing the two-qubit gate from the operator is enormous.
- How does the qubit connectivity affect the algorithm? Can we build the 'hardware efficient' amplitude amplification operator free from the limitation of the qubit connectivity?
- Can we utilize the error mitigation technique for FAE? Some of the subsequent works [129–132] discuss the QAE algorithm in the presence of noise, but they consider a limited class of the noise models. By utilizing the error mitigation, we may build the noise model-independent QAE methods in the presence of unknown noise.
- What are the device requirements for achieving the quantum advantage with FAE in practical problems?

Regarding qSGAN, applying QGAN to SSL problems is new and immature. Therefore, there is still large room for improvement. The open problems of qSGAN are as follows:

- How does the algorithm changes if we use the other SGAN architecture such as the one proposed in [45]?
- Can we apply qSGAN for the case where the dataset is given by the quantum states? Note that some of the QGAN algorithms, such as [27] propose algorithms in the case of using the quantum dataset.
- Can we use the quantum-classical hybrid generator instead of the quantum generator? The quantum-classical hybrid generator is the generator that the outputs of the quantum circuit are loaded into the classical neural network, and the outputs of the neural network become the fake samples.

Finally, VQA as a whole has its issues to be solved. Let us summarize those open problems as follows:

- Can we give the theoretical guarantee in the convergence performance of VQA? In classical deep neural networks, the convergence property is theoretically proven at a certain limit [133]. Can we show a similar theorem in VQA? Note that the author of this thesis has been working on this topic and proved the convergence property in the case of the hybrid quantum-classical neural network [134]. The subsequent work [135] also tackles the problem in the case of the quantum neural network, but there is still room for improvement.
- Are there any methods that avoid the barren plateau issue, and the ansatz has enough expressibility even when the cost function is global?
- If we measure the expressibility of the ansatz by the capability of expressing arbitrary function rather than that of generating arbitrary quantum states, how does the result of Chapter 6 change? We used the expressibility measure for measuring the capability of generating an arbitrary quantum state. The definition is reasonable as long as the problem we solve is finding the optimal quantum state. However, in some problems, such as quantum

machine learning, we need to create a quantum circuit that approximates the target function; in this case, the above definition of expressibility may be unsatisfactory. Thus, it is a good direction to define an expressibility measure to capture the capability of approximating the target function and analyze the expressibility of ALT by using the measure. The literature [136–138] examines the capability of approximating the function, but the useful measure has not been proposed yet.

The practical use of quantum computers still requires significant progress. However, solving the open problems mentioned above will surely bring us closer to the realization. I hope that this thesis will contribute to the solution of these problems.

Bibliography

[1] R. P. Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467–488, 1982.

[2] P. Shor. Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[3] J. R. McClean, R. Babbush, P. J. Love, and A. Aspuru-Guzik. Exploiting locality in quantum computation for quantum chemistry. *The Journal of Physical Chemistry Letters*, 5(24):4368–4380, 2014.

[4] D. Wecker, B. Bauer, B. K. Clark, M. B. Hastings, and M. Troyer. Gate-count estimates for performing quantum chemistry on small quantum computers. *Physical Review A*, 90(2):022305, 2014.

[5] R. Babbush, D. W. Berry, I. D. Kivlichan, A. Y. Wei, P. J. Love, and A. Aspuru-Guzik. Exponentially more precise quantum simulation of fermions in second quantization. *New Journal of Physics*, 18(3):033032, 2016.

[6] R. Babbush, D. W. Berry, J. R. McClean, and H. Neven. Quantum simulation of chemistry with sublinear scaling in basis size. *npj Quantum Information*, 5(1):1–7, 2019.

[7] P. Rebentrost, B. Gupt, and T. R. Bromley. Quantum computational finance: Monte carlo pricing of financial derivatives. *Physical Review A*, 98(2):022321, 2018.

[8] D. J. Egger, R. G. Gutiérrez, J. C. Mestre, and S. Woerner. Credit risk analysis using quantum computers. *IEEE Transactions on Computers*, 2020.

[9] N. Stamatopoulos, D. J. Egger, Y. Sun, C. Zoufal, R. Iten, N. Shen, and S. Woerner. Option pricing using quantum computers. *Quantum*, 4:291, 2020.

[10] J. Biamonte, P. Wittek, N. Pancotti, P. Rebentrost, N. Wiebe, and S. Lloyd. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

[11] S. Srinivasan, C. Downey, and B. Boots. Learning and inference in hilbert space with quantum graphical models. In *Advances in Neural Information Processing Systems*, pages 10338–10347, 2018.

[12] M. Schuld, I. Sinayskiy, and F. Petruccione. Prediction by linear regression on a quantum computer. *Physical Review A*, 94(2):022342, 2016.

[13] V. Havlíček, A. D. Córcoles, K. Temme, A. W. Harrow, A. Kandala, J. M. Chow, and J. M. Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.

[14] C. Ciliberto, M. Herbster, A. D. Ialongo, M. Pontil, A. Rocchetto, S. Severini, and L. Wossnig. Quantum machine learning: a classical perspective. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 474(2209):20170551, 2018.

[15] M. Schuld, A. Bocharov, K. M. Svore, and N. Wiebe. Circuit-centric quantum classifiers. *Physical Review A*, 101(3):032308, 2020.

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

[17] A. Montanaro. Quantum speedup of monte carlo methods. *Proceedings. Mathematical, Physical, and Engineering Sciences / The Royal Society*, 471, 2015.

[18] A. Y. Kitaev. Quantum measurements and the abelian stabilizer problem. *arXiv preprint arXiv:quant-ph/9511026*, 1995.

[19] M. A. Nielsen and I. L. Chuang. Quantum computation and quantum information (10th anniversary edition). 2010.

[20] Y. Suzuki, S. Uno, R. Raymond, T. Tanaka, T. Onodera, and N. Yamamoto. Amplitude estimation without phase estimation. *Quantum Information Processing*, 19(2):1–17, 2020.

[21] D. Grinko, J. Gacon, C. Zoufal, and S. Woerner. Iterative quantum amplitude estimation. *npj Quantum Information*, 7:1–6, 2019.

[22] K. Nakaji. Faster amplitude estimation. *Quantum Inf. Comput.*, 20:1109–1122, 2020.

[23] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, et al. Variational quantum algorithms. *Nature Reviews Physics*, pages 1–20, 2021.

[24] A. Peruzzo, J. McClean, P. Shadbolt, M.-H. Yung, X.-Q. Zhou, P. J. Love, A. Aspuru-Guzik, and J. L. O'brien. A variational eigenvalue solver on a photonic quantum processor. *Nature Communications*, 5(1):1–7, 2014.

[25] A. Kandala, A. Mezzacapo, K. Temme, M. Takita, M. Brink, J. M. Chow, and J. M. Gambetta. Hardware-efficient variational quantum eigensolver for small molecules and quantum magnets. *Nature*, 549(7671):242–246, 2017.

[26] P.-L. Dallaire-Demers and N. Killoran. Quantum generative adversarial networks. *Physical Review A*, 98(1):012324, 2018.

[27] S. Lloyd and C. Weedbrook. Quantum generative adversarial learning. *Physical Review Letters*, 121(4):040502, 2018.

[28] H. Situ, Z. He, Y. Wang, L. Li, and S. Zheng. Quantum generative adversarial network for generating discrete distribution. *Information Sciences*, 538:193–208, 2020.

[29] C. Zoufal, A. Lucchi, and S. Woerner. Quantum generative adversarial networks for learning and loading random distributions. *npj Quantum Information*, 5(1):1–9, 2019.

[30] M. Benedetti, E. Grant, L. Wossnig, and S. Severini. Adversarial quantum circuit learning for pure state approximation. *New Journal of Physics*, 21(4):043023, 2019.

[31] L. Hu, S.-H. Wu, W. Cai, Y. Ma, X. Mu, Y. Xu, H. Wang, Y. Song, D.-L. Deng, C.-L. Zou, and L. Sun. Quantum generative adversarial learning in a superconducting quantum circuit. *Science advances*, 5(1):eaav2761, 2019.

[32] J. Zeng, Y. Wu, J.-G. Liu, L. Wang, and J. Hu. Learning and inference on generative adversarial quantum circuits. *Physical Review A*, 99(5):052306, 2019.

[33] J. Romero and A. Aspuru-Guzik. Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions. *arXiv preprint arXiv:1901.00848*, 2019.

[34] S. Chakrabarti, T. Yiming, Huang an d Li, S. Feizi, and X. Wu. Quantum wasserstein generative adversarial networks. In *Advances in Neural Information Processing Systems*, pages 6781–6792, 2019.

[35] K. Huang, Z.-A. Wang, C. Song, K. Xu, H. Li, Z. Wang, Q. Guo, Z. Song, Z.-B. Liu, D. Zheng, et al. Realizing a quantum generative adversarial network using a programmable superconducting processor. *arXiv preprint arXiv:2009.12827*, 2020.

[36] H.-L. Huang, Y. Du, M. Gong, Y. Zhao, Y. Wu, C. Wang, S. Li, F. Liang, J. Lin, Y. Xu, et al. Experimental quantum generative adversarial networks for image generation. *arXiv preprint arXiv:2010.06201*, 2020.

[37] A. Anand, J. Romero, M. Degroote, and A. Aspuru-Guzik. Experimental demonstration of a quantum generative adversarial network for continuous distributions. *arXiv preprint arXiv:2006.01976*, 2020.

[38] S. Ahmed, C. S. Muñoz, F. Nori, and A. F. Kockum. Quantum state tomography with conditional generative adversarial networks. *arXiv preprint arXiv:2008.03240*, 2020.

[39] S. A. Stein, B. Baheri, R. M. Tischio, Y. Mao, Q. Guan, A. Li, B. Fang, and S. Xu. Qugan: A generative adversarial network through quantum states. *arXiv preprint arXiv:2010.09036*, 2020.

[40] D. Herr, B. Obert, and M. Rosenkranz. Anomaly detection with variational quantum generative adversarial networks. *arXiv preprint arXiv:2010.10492*, 2020.

[41] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2672–2680. Curran Associates, Inc., 2014.

[42] J. Gui, Z. Sun, Y. Wen, D. Tao, and J. Ye. A review on generative adversarial networks: Algorithms, theory, and applications. *arXiv preprint arXiv:2001.06937*, 2020.

[43] P. Shamsolmoali, M. Zareapoor, E. Granger, H. Zhou, R. Wang, M. E. Celebi, and J. Yang. Image synthesis with adversarial networks: A comprehensive survey and case studies. *Information Fusion*, 2021.

[44] Z. Liu, J. Wang, and Z. Liang. Catgan: Category-aware generative adversarial networks with hierarchical evolutionary learning for category text generation. In *AAAI*, pages 8425–8432, 2020.

[45] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in neural information processing systems*, pages 2234–2242, 2016.

[46] A. Odena. Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*, 2016.

[47] A. Madani, M. Moradi, A. Karargyris, and T. Syeda-Mahmood. Semi-supervised learning with generative adversarial networks for chest x-ray classification with ability of data domain adaptation. In *2018 IEEE 15th International symposium on biomedical imaging (ISBI 2018)*, pages 1038–1042. IEEE, 2018.

[48] C. Li, K. Xu, J. Zhu, and B. Zhang. Triple generative adversarial nets. *arXiv preprint arXiv:1703.02291*, 2017.

[49] Z. Gan, L. Chen, W. Wang, Y. Pu, Y. Zhang, H. Liu, C. Li, and L. Carin. Triangle generative adversarial networks. *arXiv preprint arXiv:1709.06548*, 2017.

[50] Z. Dai, Z. Yang, F. Yang, W. W. Cohen, and R. Salakhutdinov. Good semi-supervised learning that requires a bad gan. *arXiv preprint arXiv:1705.09783*, 2017.

[51] T. Miyato, S.-i. Maeda, M. Koyama, and S. Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

[52] K. Nakaji and N. Yamamoto. Quantum semi-supervised generative adversarial network for enhanced data classification. *Scientific Reports*, 11, 2021.

[53] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1):1–6, 2018.

[54] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature Communications*, 12(1):1–12, 2021.

[55] K. Nakaji and N. Yamamoto. Expressibility of the alternating layered ansatz for quantum computation. *Quantum*, 5:434, 2021.

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

[57] H. Häffner, C. F. Roos, and R. Blatt. Quantum computing with trapped ions. *Physics reports*, 469(4):155–203, 2008.

[58] P. Krantz, M. Kjaergaard, F. Yan, T. P. Orlando, S. Gustavsson, and W. D. Oliver. A quantum engineer's guide to superconducting qubits. *Applied Physics Reviews*, 6(2):021318, 2019.

[59] S. J. Devitt, W. J. Munro, and K. Nemoto. Quantum error correction for beginners. *Reports on Progress in Physics*, 76(7):076001, 2013.

[60] A. Aspuru-Guzik, A. D. Dutoi, P. J. Love, and M. Head-Gordon. Simulated quantum computation of molecular energies. *Science*, 309:1704 – 1707, 2005.

[61] H. F. Trotter. On the product of semi-groups of operators. 1959.

[62] S. McArdle, S. Endo, A. Aspuru-Guzik, S. C. Benjamin, and X. Yuan. Quantum computational chemistry. *Reviews of Modern Physics*, 92:015003, 2020.

[63] A. W. Harrow, A. Hassidim, and S. Lloyd. Quantum algorithm for linear systems of equations. *Physical Review Letters*, 103(15):150502, 2009.

[64] J. M. Gambetta, J. M. Chow, and M. Steffen. Building logical qubits in a superconducting quantum computing system. *npj Quantum Information*, 3(1):1–7, 2017.

[65] Y. Xu, J. Chu, J. Yuan, J. Qiu, Y. Zhou, L. Zhang, X. Tan, Y. Yu, S. Liu, J. Li, et al. High-fidelity, high-scalability two-qubit gate scheme for superconducting qubits. *Physical Review Letters*, 125(24):240503, 2020.

[66] B. Foxen, C. Neill, A. Dunsworth, P. Roushan, B. Chiaro, A. Megrant, J. Kelly, Z. Chen, K. Satzinger, R. Barends, et al. Demonstrating a continuous set of two-qubit gates for near-term quantum algorithms. *Physical Review Letters*, 125(12):120504, 2020.

[67] D. C. McKay, C. J. Wood, S. Sheldon, J. M. Chow, and J. M. Gambetta. Efficient z gates for quantum computing. *Physical Review A*, 96(2):022330, 2017.

[68] F. B. Maciejewski, Z. Zimborás, and M. Oszmaniec. Mitigation of readout noise in near-term quantum devices by classical post-processing based on detector tomography. *Quantum*, 4:257, April 2020.

[69] Y. Li and S. C. Benjamin. Efficient variational quantum simulator incorporating active error minimization. *Physical Review X*, 7:021050, 2017.

[70] K. Temme, S. Bravyi, and J. M. Gambetta. Error mitigation for short-depth quantum circuits. *Physical Review Letters*, 119 18:180509, 2017.

[71] S. Endo, S. C. Benjamin, and Y. Li. Practical quantum error mitigation for near-future applications. *Physical Review X*, 8(3):031027, 2018.

[72] T. Giurgica-Tiron, Y. Hindy, R. LaRose, A. Mari, and W. J. Zeng. Digital zero noise extrapolation for quantum error mitigation. In *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, pages 306–316. IEEE, 2020.

[73] A. G. Taube and R. J. Bartlett. New perspectives on unitary coupled-cluster theory. *International Journal of Quantum Chemistry*, 106:3393–3401, 2006.

[74] H. Abraham, AduOffei, R. Agarwal, I. Y. Akhalwaya, G. Aleksandrowicz, T. Alexander, et al. Qiskit: An open-source framework for quantum computing, 2019.

[75] G. E. Crooks. Gradients of parameterized quantum gates using the parameter-shift rule and gate decomposition. *arXiv preprint arXiv:1905.13311*, 2019.

[76] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[77] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii. Quantum circuit learning. *Physical Review A*, 98(3):032309, 2018.

[78] H.-Y. Huang, M. Broughton, M. Mohseni, R. Babbush, S. Boixo, H. Neven, and J. R. McClean. Power of data in quantum machine learning. *Nature Communications*, 12, 2021.

[79] L. Tran, X. Yin, and X. Liu. Representation learning by rotating your faces. *IEEE transactions on pattern analysis and machine intelligence*, 41(12):3007–3021, 2018.

[80] R. Huang, S. Zhang, T. Li, and R. He. Beyond face rotation: Global and local perception gan for photorealistic and identity preserving frontal view synthesis. In *Proceedings of the IEEE international conference on computer vision*, pages 2439–2448, 2017.

[81] L. Ma, X. Jia, Q. Sun, B. Schiele, T. Tuytelaars, and L. Van Gool. Pose guided person image generation. *arXiv preprint arXiv:1705.09368*, 2017.

[82] W. Jiang, S. Liu, C. Gao, J. Cao, R. He, J. Feng, and S. Yan. Psgan: Pose and expression robust spatial-aware gan for customizable makeup transfer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5194–5202, 2020.

[83] R. Yi, Y.-J. Liu, Y.-K. Lai, and P. L. Rosin. Apdrawinggan: Generating artistic portrait drawings from face photos with hierarchical gans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10743–10752, 2019.

[84] J.-Y. Zhu, P. Kr¨ahenb¨uhl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *European conference on computer vision*, pages 597–613. Springer, 2016.

[85] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Neural photo editing with introspective adversarial networks. *arXiv preprint arXiv:1609.07093*, 2016.

[86] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2337–2346, 2019.

[87] K. Lin, D. Li, X. He, Z. Zhang, and M.-T. Sun. Adversarial ranking for language generation. *arXiv preprint arXiv:1705.11001*, 2017.

[88] J. Wang, L. Yu, W. Zhang, Y. Gong, Y. Xu, B. Wang, P. Zhang, and D. Zhang. Irgan: A minimax game for unifying generative and discriminative information retrieval models. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 515–524, 2017.

[89] S. Lu, Z. Dou, X. Jun, J.-Y. Nie, and J.-R. Wen. Psgan: A minimax game for personalized search with limited and noisy click data. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 555–564, 2019.

[90] T. Qiao, J. Zhang, D. Xu, and D. Tao. Mirrorgan: Learning text-to-image generation by redescription. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1505–1514, 2019.

[91] K. Ehsani, R. Mottaghi, and A. Farhadi. Segan: Segmenting and generating the invisible. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6144–6153, 2018.

[92] J. Li, X. Liang, Y. Wei, T. Xu, J. Feng, and S. Yan. Perceptual generative adversarial networks for small object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1222–1230, 2017.

[93] Y. Bai, Y. Zhang, M. Ding, and B. Ghanem. Sod-mtgan: Small object detection via multi-task generative adversarial network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 206–221, 2018.

[94] L. K. Grover. Synthesis of quantum superpositions by quantum computation. *Physical Review Letters*, 85(6):1334, 2000.

[95] Y. R. Sanders, G. H. Low, A. Scherer, and D. W. Berry. Black-box quantum state preparation without arithmetic. *Physical Review Letters*, 122(2):020502, 2019.

[96] M. Plesch and Č. Brukner. Quantum-state preparation with universal gate decompositions. *Physical Review A*, 83(3):032302, 2011.

[97] V. V. Shende, S. S. Bullock, and I. L. Markov. Synthesis of quantum-logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(6):1000–1010, 2006.

[98] L. Grover and T. Rudolph. Creating superpositions that correspond to efficiently integrable probability distributions. *arXiv preprint arXiv:quant-ph/0208112*, 2002.

[99] M. Möttönen, J. Vartiainen, V. Bergholm, and M. Salomaa. Transformation of quantum states using uniformly controlled rotations. *Quantum Information and Computation.*, 5:467–473, 2005.

[100] V. Shende and I. Markov. Quantum circuits for incompletely specified two-qubit operators. *Quantum Information and Computation.*, 5:49–57, 2005.

[101] K. Nakaji, S. Uno, Y. Suzuki, R. Raymond, T. Onodera, T. Tanaka, H. Tezuka, N. Mitsuda, and N. Yamamoto. Approximate amplitude encoding in shallow parameterized quantum circuits and its application to financial market indicator. 2021.

[102] J. R. McClean, S. Boixo, V. N. Smelyanskiy, R. Babbush, and H. Neven. Barren plateaus in quantum neural network training landscapes. *Nature Communications*, 9(1):1–6, 2018.

[103] S. Wang, E. Fontana, M. Cerezo, K. Sharma, A. Sone, L. Cincio, and P. J. Coles. Noise-induced barren plateaus in variational quantum algorithms. *Nature Communications*, 12(1):1–11, 2021.

[104] A. Arrasmith, M. Cerezo, P. Czarnik, L. Cincio, and P. J. Coles. Effect of barren plateaus on gradient-free optimization. *Quantum*, 5:558, 2021.

[105] C. Dankert, R. Cleve, J. Emerson, and E. R. Livine. Exact and approximate unitary 2-designs and their application to fidelity estimation. *Physical Review A*, 80:012304, 2009.

[106] D. Wierichs, C. Gogolin, and M. Kastoryano. Avoiding local minima in variational quantum eigensolvers with the natural gradient optimizer. *Physical Review Research*, 2(4):043246, 2020.

[107] J. Stokes, J. A. Izaac, N. Killoran, and G. Carleo. Quantum natural gradient. *Quantum*, 4:269, 2020.

[108] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, et al. Variational quantum algorithms. *arXiv preprint arXiv:2012.09265*, 2020.

[109] H. R. Grimsley, S. E. Economou, E. Barnes, and N. J. Mayhall. An adaptive variational algorithm for exact molecular simulations on a quantum computer. *Nature Communications*, 10(1):1–9, 2019.

[110] H. L. Tang, V. Shkolnikov, G. S. Barron, H. R. Grimsley, N. J. Mayhall, E. Barnes, and S. E. Economou. qubit-adapt-vqe: An adaptive algorithm for constructing hardware-efficient ansätze on a quantum processor. *PRX Quantum*, 2(2):020310, 2021.

[111] I. G. Ryabinkin, T.-C. Yen, S. N. Genin, and A. F. Izmaylov. Qubit coupled cluster method: a systematic approach to quantum chemistry on a quantum computer. *Journal of chemical theory and computation*, 14(12):6317–6326, 2018.

[112] N. V. Tkachenko, J. Sud, Y. Zhang, S. Tretiak, P. M. Anisimov, A. T. Arrasmith, P. J. Coles, L. Cincio, and P. A. Dub. Correlation-informed permutation of qubits for reducing ansatz depth in the variational quantum eigensolver. *PRX Quantum*, 2(2):020337, 2021.

[113] W. Tang, T. Tomesh, M. Suchara, J. Larson, and M. Martonosi. Cutqc: using small quantum computers for large quantum circuit evaluations. In *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 473–486, 2021.

[114] T. Peng, A. W. Harrow, M. Ozols, and X. Wu. Simulating large quantum circuits on a small quantum computer. *Physical Review Letters*, 125(15):150504, 2020.

[115] F. Arute, K. Arya, R. Babbush, D. Bacon, J. C. Bardin, R. Barends, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574:505–510, 2019.

[116] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. pages 8024–8035, 2019.

[117] E. Grant, L. Wossnig, M. Ostaszewski, and M. Benedetti. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3:214, 2019.

[118] T. Volkoff and P. J. Coles. Large gradients via correlation in random parameterized quantum circuits. *Quantum Science and Technology*, 6(2):025008, 2021.

[119] E. Grant, L. Wossnig, M. Ostaszewski, and M. Benedetti. An initialization strategy for addressing barren plateaus in parametrized quantum circuits. *Quantum*, 3:214, 2019.

[120] A. Skolik, J. R. McClean, M. Mohseni, P. van der Smagt, and M. Leib. Layerwise learning for quantum neural networks. *Quantum Machine Intelligence*, 3:1–11, 2021.

[121] T. Ali, A. Bhattacharyya, S. S. Haque, E. H. Kim, N. Moynihan, and J. Murugan. Chaos and complexity in quantum mechanics. *Physical Review D*, 101(2):026021, 2020.

[122] J. M. Renes, R. Blume-Kohout, A. J. Scott, and C. M. Caves. Symmetric informationally complete quantum measurements. *Journal of Mathematical Physics*, 45(6):2171–2180, 2004.

[123] A. Klappenecker and M. Rotteler. Mutually unbiased bases are complex projective 2-designs. In *Proceedings. International Symposium on Information Theory, 2005. ISIT 2005.*, pages 1740–1744. IEEE, 2005.

[124] I. Bengtsson and K. Życzkowski. *Geometry of quantum states: an introduction to quantum entanglement*. Cambridge university press, 2017.

[125] K. Życzkowski and H.-J. Sommers. Average fidelity between random quantum states. *Physical Review A*, 71(3):032313, 2005.

[126] M. Cerezo, A. Sone, T. Volkoff, L. Cincio, and P. J. Coles. Cost function dependent barren plateaus in shallow parametrized quantum circuits. *Nature Communications*, 12 1:1791, 2021.

[127] F. G. Brandao, A. W. Harrow, and M. Horodecki. Local random quantum circuits are approximate polynomial-designs. *Communications in Mathematical Physics*, 346(2):397–434, 2016.

[128] A. Harrow and S. Mehraban. Approximate unitary t-designs by short random quantum circuits using nearest-neighbor and long-range gates. *arXiv preprint arXiv:1809.06957*, 2018.

[129] E. G. Brown, O. Goktas, and W. Tham. Quantum amplitude estimation in the presence of noise. *arXiv preprint arXiv:2006.14145*, 2020.

[130] S. Uno, Y. Suzuki, K. Hisanaga, R. Raymond, T. Tanaka, T. Onodera, and N. Yamamoto. Modified grover operator for quantum amplitude estimation. *New Journal of Physics*, 23(8):083031, 2021.

[131] T. Tanaka, Y. Suzuki, S. Uno, R. H. Putra, T. Onodera, and N. Yamamoto. Amplitude estimation via maximum likelihood on noisy quantum computer. *Quantum Information Processing*, 20:293, 2021.

[132] S. Herbert, R. Guichard, and D. Ng. Noise-aware quantum amplitude estimation. *arXiv preprint arXiv:2109.04840*, 2021.

[133] A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *arXiv preprint arXiv:1806.07572*, 2018.

[134] K. Nakaji, H. Tezuka, and N. Yamamoto. Quantum-enhanced neural networks in the neural tangent kernel framework. *arXiv preprint arXiv:2109.03786*, 2021.

[135] J. Liu, F. Tacchino, J. R. Glick, L. Jiang, and A. Mezzacapo. Representation learning via quantum neural tangent kernels. *arXiv preprint arXiv:2111.04225*, 2021.

[136] F. J. Gil Vidal and D. O. Theis. Input redundancy for parameterized quantum circuits. *Frontiers in Physics*, 8:297, 2020.

[137] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre. Data re-uploading for a universal quantum classifier. *Quantum*, 4:226, 2020.

[138] M. Schuld, R. Sweke, and J. J. Meyer. Effect of data encoding on the expressive power of variational quantum-machine-learning models. *Physical Review A*, 103(3):032430, 2021.

Appendix A

Theorems for Faster amplitude estimation

A.1 Proof of Complexity Upper Bound

In this appendix, we provide proof of the complexity upper bound.
Theorem *The following upper bound holds for N_{orac}:*

$$N_{\mathrm{orac}} < \frac{4.1 \cdot 10^3}{\epsilon} \ln\left(\frac{4\log_2(2\pi/3\epsilon)}{\delta}\right). \tag{A.1}$$

Proof. Our strategy to obtain the upper bound is calculating the required number of N^{1st}_{shot} and N^{2nd}_{shot} for the algorithm to work correctly with the probability $1-\delta$. Both upper bounds of N^{1st}_{shot} and N^{2nd}_{shot} can be derived from the condition that our algorithm works correctly in the second stage because even though the condition from the first stage also bounds N^{1st}_{shot} loosely, the most strict upper bound of N^{1st}_{shot} can be gotten from the condition that the estimation error of ν is small enough. Thus, we only discuss the condition from the second stage in the following.

In the second stage, as we mention in Section 4.3, the algorithm works correctly as long as $\Delta\rho_j \leq \pi/3$. Even though the conditions for $\mathrm{atan}\left(s_{2^{j-1}}, c_{2^{j-1}}\right)$ derived from $\Delta\rho_j \leq \pi/3$ are different depending on whether the confidence interval of ρ_j is the connected confidence interval or the disconnected confidence interval, the required precisions for $s_{2^{j-1}}$ and $c_{2^{j-1}}$ do not change depending on the interval type. Therefore, in the following, we discuss only the case of the connected confidence interval. Then, the condition $\Delta\rho_j \leq \pi/3$ can be converted to

$$\left|\mathrm{atan}\left(s_{2^{j-1}}, c_{2^{j-1}}\right) - \mathrm{atan}\left(s^*_{2^{j-1}}, c^*_{2^{j-1}}\right)\right| \leq \frac{\pi}{3} \tag{A.2}$$

where $s^*_{2^{j-1}}, c^*_{2^{j-1}}$ are the true values of $s_{2^{j-1}}$ and $c_{2^{j-1}}$ respectively. Given $\Delta c_{2^{j-1}} = |c_{2^{j-1}} - c^*_{2^{j-1}}|$, $\Delta s_{2^{j-1}} = |s_{2^{j-1}} - s^*_{2^{j-1}}|$, by using the result (A.18), which we show in Appendix A.2, the following inequality holds for the left hand side of (A.2):

$$\left|\mathrm{atan}\left(s_{2^{j-1}}, c_{2^{j-1}}\right) - \mathrm{atan}\left(s^*_{2^{j-1}}, c^*_{2^{j-1}}\right)\right| < \max(2\Delta c_{2^{j-1}} + 2\Delta s_{2^{j-1}}, 3\Delta c_{2^{j-1}}) \tag{A.3}$$

as long as $\Delta c_{2^{j-1}} < 1/4$ and $\Delta s_{2^{j-1}} < 1/3$. On the other hand, from (4.20), it holds that

$$\Delta s_{2^{j-1}} = \left|\frac{-s^*_{2^{j-1}}(\sin\nu - \sin(\nu-\Delta\nu)) + c^*_{2^{j-1}}(\cos\nu - \cos(\nu-\Delta\nu)) + \Delta c_{2^{j-1}}\cos\nu + \Delta c_{2^{j-1}+2^{j_0-1}}}{\sin\nu}\right| \tag{A.4}$$

$$\leq \frac{\sqrt{2-2\cos(\Delta\nu)} + |\Delta c_{2^{j-1}}\cos\nu| + |\Delta c_{2^{j-1}+2^{j_0-1}}|}{\sin\nu}, \tag{A.5}$$

where $\Delta\nu = \nu - 2^{j_0+1}\theta$ and $3\pi/8 - |\Delta\nu| \leq \nu \leq 3\pi/4 - |\Delta\nu|$. Thus, if at least the estimation errors are bounded as

$$\Delta c_{2^{j-1}} \leq \frac{1}{9}, \tag{A.6}$$

$$\Delta c_{2^{j-1}+2^{j_0-1}} \leq \frac{1}{9}, \tag{A.7}$$

$$|\Delta\nu| < \frac{\pi}{60}, \tag{A.8}$$

then it holds

$$\Delta s_{2^{j-1}} < \frac{\sqrt{2-2\cos(\frac{\pi}{60})} + \frac{1}{9}|\cos(\frac{3\pi}{4} - \frac{\pi}{60})| + \frac{1}{9}}{\sin(\frac{3\pi}{4} - \frac{\pi}{60})} < \frac{1}{3}. \tag{A.9}$$

As a result,

$$|\mathrm{atan}\,(s_{2^{j-1}}, c_{2^{j-1}}) - \mathrm{atan}\,(s^*_{2^{j-1}}, c^*_{2^{j-1}})| < \max(2 \cdot \frac{1}{9} + 2 \cdot \frac{1}{3}, 3 \cdot \frac{1}{9}) < \frac{\pi}{3} \tag{A.10}$$

is satisfied. By using (4.12), the direct calculation shows that if we choose

$$N^{2nd}_{\mathrm{shot}} = 972 \ln\left(\frac{2}{\delta_c}\right), \tag{A.11}$$

then both the conditions (A.6) and (A.7) are satisfied with the probability $1 - 2\delta_c$. On the other hand, (A.8) is achieved if at least

$$\Delta c_{2^{j-1}} < \frac{1}{9\sqrt{2}} \tag{A.12}$$

holds in the first stage because

$$\begin{aligned}\Delta\nu &= \frac{1}{2}\left(\arccos\left(c^{\min}_{2^{j_0-1}}\right) - \arccos\left(c^{\max}_{2^{j_0}-1}\right)\right) \\ &< \frac{1}{2}\left(\arccos\left(\cos\left(\frac{3\pi}{4}\right)\right) - \arccos\left(\cos\left(\frac{3\pi}{4}\right) + \frac{1}{9\sqrt{2}}\right)\right) \\ &< \frac{\pi}{60}.\end{aligned} \tag{A.13}$$

Thus, by using (4.12) again, we can leadily show that by setting

$$N^{1st}_{\mathrm{shot}} = 1944 \ln\left(\frac{2}{\delta_c}\right), \tag{A.14}$$

then (A.12) and hence (A.8) is satisfied with the probability $1 - \delta_c$. In summary, as far as (A.11) and (A.14) are satisfied, for all $j(> j_0)$, $\Delta\rho_j \leq \pi/3$ holds and (4.23) is satisfied under the assumption that all the estimates of cosines are inside the confidence interval. Note that the probability that all the estimates are inside the interverl is $(1-\delta_c)^{j_0+2(\ell-j_0)} > 1 - (2\ell - j_0)\delta_c$.

Finally, we evaluate the query complexity in the worst case. The worst case is that the algorithm moves to the second stage at the first iteration($j = 1$). In this case, the number of oracle call is

$$N_{\mathrm{orac}} < N^{1st}_{\mathrm{shot}} + \sum_{j=2}^{\ell}(2N^{2nd}_{\mathrm{shot}} \times 2^{j-1}) = 1944 \ln\left(\frac{2}{\delta_c}\right) + 1944(2^\ell - 2)\ln\left(\frac{2}{\delta_c}\right), \tag{A.15}$$

and the success probability of the algorithm is $1-(2\ell-1)\delta_c$. Thus, if we demand that the success probability is more than $1 - \delta$ then $\delta_c < \delta/2\ell$ and

$$N_{\mathrm{orac}} < 1944 \cdot 2^\ell \ln\left(\frac{4\ell}{\delta}\right). \tag{A.16}$$

By combining with (4.31)

$$N_{\mathrm{orac}} < \frac{4.1 \cdot 10^3}{\epsilon} \ln\left(\frac{4\log_2(2\pi/3\epsilon)}{\delta}\right). \tag{A.17}$$

□

A.2 Theorem for atan function

Theorem *When $c, c^*, s \in [-1, 1]$, s^* takes one of the value of $\pm\sqrt{1-c^{*2}}$, $\Delta c = |c - c^*|$ and $\Delta s = |s - s^*|$, the following inequality holds:*

$$|\text{atan}(s,c) - \text{atan}(s^*,c^*)| < \max(2\Delta c + 2\Delta s, 3\Delta c) \tag{A.18}$$

if $\Delta s < 1/2$ and $\Delta c < 1/4$ and if there is no discontinuity of $\text{atan}(s,c)$ *in the intervals: $s^* - \Delta s \leq s \leq s^* + \Delta s$ and $c^* - \Delta c \leq c \leq c^* + \Delta c$.*

Proof. It is suffice to prove in following three cases: (i) $cc^* > 0$ (ii) $cc^* < 0$ and (iii) $cc^* = 0$. In case (i) $cc^* > 0$, using trigonometric addition formulas for arctan, it holds that

$$\begin{aligned} |\text{atan}(s,c) - \text{atan}(s^*,c^*)| &= |\text{arctan}(s,c) - \text{arctan}(s^*,c^*)| \\ &= \left|\text{arctan}\left(\frac{s^*\Delta c - c^*\Delta s}{1 + c^*\Delta c + s^*\Delta s}\right)\right| \\ &\leq \left|\frac{|s^*|\Delta c + |c^*|\Delta s}{1 - |c^*|\Delta c - |s^*|\Delta s}\right| \\ &< 2\Delta c + 2\Delta s. \end{aligned} \tag{A.19}$$

To show the last inequality, we use $1 - |c^*|\Delta c - |s^*|\Delta s > 1 - \sqrt{(1/3)^2 + (1/4)^2} > 1/2$.

In case (ii) $cc^* < 0$,

$$\begin{aligned} |\text{atan}(s,c) - \text{atan}(s^*,c^*)| = \lim_{\eta\to 0} &(|\text{arctan}(s,c) - \text{arctan}(s,\eta)| \\ &+ |\text{arctan}(s,\eta) - \text{arctan}(s^*,-\eta)|) \\ &+ |\text{arctan}(s^*,-\eta) - \text{arctan}(s^*,c^*)|), \end{aligned} \tag{A.20}$$

where the sign of η is same as that of c. The first term in (A.20) can be bounded as

$$\begin{aligned} \lim_{\eta\to 0} |\text{arctan}(s,c) - \text{arctan}(s,\eta)| &= \lim_{\eta\to 0}\left|\frac{\partial}{\partial c}\text{arctan}\left(\frac{s}{c}\right)|_{c=c_0}(c-\eta)\right| \\ &= \lim_{\eta\to 0}\left|\frac{-s}{c_0^2 + s^2}(c-\eta)\right| \\ &\leq \left|\frac{s}{c_0^2+s^2}c\right| \\ &\leq \left|\frac{1}{(c_* - (c_* - c_0))^2 + (s_* - (s_* - s))^2}c\right| \\ &\leq \left|\frac{1}{\left(\frac{3}{5}-\frac{1}{4}\right)^2 + \left(\frac{4}{5}-\frac{1}{3}\right)^2}c\right| \\ &< 3|c|, \end{aligned} \tag{A.21}$$

where c_0 take the value between η and c, and we use the mean value theorem for showing the first equality. Similary,

$$\lim_{\eta\to 0} |\text{arctan}(s_*, -\eta) - \text{arctan}(s_*, c_*)| < 3|c_*|. \tag{A.22}$$

By substituting (A.21), (A.22) and $\lim_{\eta\to 0} |\text{arctan}(s,\eta) - \text{arctan}(s^*,-\eta)| = 0$ (that follows from no-discontinuity condition) to the right-hand side of (A.20), it follows

$$|\text{atan}(s,c) - \text{atan}(s^*,c^*)| < 3(|c| + |c_*|) = 3\Delta c. \tag{A.23}$$

The last equality holds because the signs of c and c_* are different.

In case (iii) $cc_* = 0$, when $c_* = 0$,

$$\begin{aligned} |\text{atan}(s,c) - \text{atan}(s^*,c^*)| = \lim_{\eta\to 0}&\left(\left|\pm\frac{\pi}{2} - \text{arctan}(s_*,\eta)\right| + |\text{arctan}(s_*,\eta) - \text{arctan}(s,\eta)|\right. \\ &+ |\text{arctan}(s,\eta) - \text{arctan}(s,c)|), \end{aligned} \tag{A.24}$$

where the sign $\pm$ is the same as the sign of s and the sign of η is same as that of c. The values of the first line go to 0 and the value of the second line can be evaluated by the same arguments as (A.21). Thus, it follows

$$|\mathrm{atan}(s, c) - \mathrm{atan}(s^*, c^*)| < 3(|c|) = 3\Delta c. \tag{A.25}$$

By the same discussion, when $c = 0$

$$|\mathrm{atan}(s, c) - \mathrm{atan}(s^*, c^*)| < 3(|c_*|) = 3\Delta c. \tag{A.26}$$

In all of the three cases, (A.18) is proved.

Appendix B

Theorem for the expressibility computation

Here we prove several theorems introduced in Section 6.2. In the following, for the unitary matrix U_a corresponding to the entire circuit, we denote the unitary matrix corresponding to the i-th layer of the circuit to be $U_a(i)$ and the unitary matrix corresponding to the j-th block in the i-th layer as $U_a(i,j)$.

B.1 Proof of Theorem 2

First, because the probability distribution of $F = |\langle\psi|\psi'\rangle|^2$ with n-qubits states $|\psi\rangle$ and $|\psi'\rangle$ taken from the Haar measure is given by $P_{\mathrm{Haar}}(F) = (2^n-1)(1-F)^{2^n-2}$, the value of $\mathcal{F}^{(1)}_{\mathrm{Haar}}(2^n)$ is straightforwardly computed as

$$\mathcal{F}^{(1)}_{\mathrm{Haar}}(2^n) = \int_{\mathrm{Haar}}\int_{\mathrm{Haar}} |\langle\psi|\psi'\rangle|^2 d\psi d\psi' = \int_0^1 dF F(2^n-1)(1-F)^{2^n-2} = \frac{1}{2^n}. \tag{B.1}$$

Next, we provide the proof of $\mathcal{F}^{(1)}(C^{\ell,m,n}_{\mathrm{ALT}}) = \mathcal{F}^{(1)}_{\mathrm{Haar}}(2^n)$. Given two final states $|\phi\rangle = U_a|0\rangle$ and $|\psi\rangle = U_b|0\rangle$ generated by $C^{\ell,m,n}_{\mathrm{ALT}}$, we have

$$\begin{aligned}
\mathcal{F}^{(1)}(C^{\ell,m,n}_{\mathrm{ALT}}) &= \int_{\mathrm{1design}} dU_a dU_b \langle 0|U_a^\dagger U_b|0\rangle\langle 0|U_b^\dagger U_a|0\rangle \\
&= \int_{\mathrm{1design}} \left(\prod_{i=1}^{\ell} dU_a(i)\right)\left(\prod_{i=1}^{\ell} dU_b(i)\right) \\
&\quad \times \langle 0|U_a(1)^\dagger U_a(2)^\dagger\cdots U_a(\ell)^\dagger U_b(\ell)\cdots U_b(2)U_b(1)|0\rangle \\
&\quad \times \langle 0|U_b(1)^\dagger U_b(2)^\dagger\cdots U_b(\ell)^\dagger U_a(\ell)\cdots U_a(2)U_a(1)|0\rangle \\
&= \int_{\mathrm{1design}} \left(\prod_{i=1}^{\ell}\left(\prod_{j=1}^{k(i)} dU_a(i,j)\right)\right)\left(\prod_{i'=1}^{\ell}\left(\prod_{j'=1}^{k(i')} dU_b(i',j')\right)\right) \\
&\quad \times \langle 0|U_a(1)^\dagger U_a(2)^\dagger\cdots U_a(\ell)^\dagger U_b(\ell)\cdots U_b(2)U_b(1)|0\rangle \\
&\quad \times \langle 0|U_b(1)^\dagger U_b(2)^\dagger\cdots U_b(\ell)^\dagger U_a(\ell)\cdots U_a(2)U_a(1)|0\rangle,
\end{aligned} \tag{B.2}$$

where $k(i)$ is the number of blocks in the i-th layer, and each $\int dU$ is the average over the ensemble of the unitary matrix U. Because the distribution of each $U_a(i,j)$ is 2-design (and is therefore 1-design), we can apply the formula (6.22) to the integrals with respect to $U_a(i,j)$.

Actually, by integrating $\prod_{j=1}^{k(\ell)} dU_a(\ell,j)$ for all a in the last line of (B.2), we have

$$\begin{aligned}
\mathcal{F}^{(1)}(C_{\mathrm{ALT}}^{\ell,m,n}) &= \int_{\mathrm{1design}} \left(\prod_{i=1}^{\ell}\left(\prod_{j=1}^{k(i)} dU_a(i,j)\right)\right)\left(\prod_{i'=1}^{\ell}\left(\prod_{j'=1}^{k(i')} dU_b(i',j')\right)\right) \\
&\qquad \times \langle 0|U_a(1)^\dagger U_a(2)^\dagger \cdots U_a(\ell-1)^\dagger U_a(\ell-1)\cdots U_a(2)U_a(1)|0\rangle \\
&\qquad \times \langle 0|U_b(1)^\dagger U_b(2)^\dagger \cdots U_b(\ell)^\dagger U_b(\ell)\cdots U_b(2)U_b(1)|0\rangle \\
&= \left(\frac{1}{2^m}\right)^{\ell} \int_{\mathrm{1design}} \left(\prod_{i=1}^{L-1}\left(\prod_{\alpha=1}^{k(i)} dU_a^{i\alpha}\right)\right)\left(\prod_{j=1}^{L}\left(\prod_{\beta=1}^{k(i')} dU_b^{j\beta}\right)\right)\times 1 \\
&= \frac{1}{2^n} = \mathcal{F}_{\mathrm{Haar}}^{(1)}(2^n). \qquad \text{(B.3)}
\end{aligned}$$

The other equality in Eq. (6.25) can be proved in the same manner.

B.2 Proof of Theorem 3

Similar to the first frame potential, the value of $\mathcal{F}_{\mathrm{Haar}}^{(1)}(2^n)$ is straightforwardly computed as

$$\mathcal{F}_{\mathrm{Haar}}^{(2)}(2^n) = \int_0^1 dF F^2(2^n-1)(1-F)^{2^n-2} = \frac{1}{2^{n-1}(2^n+1)}. \qquad \text{(B.4)}$$

Next, we compute $\mathcal{F}^{(2)}(C_{\mathrm{TEN}}^{\ell,m,n})$. Given two final states $|\phi\rangle = U_a|0\rangle$ and $|\psi\rangle = U_b|0\rangle$, it is computed as

$$\begin{aligned}
\mathcal{F}^{(2)}(C_{\mathrm{TEN}}^{\ell,m,n}) &= \int_{\mathrm{2design}} \left(\prod_{i=1}^{\ell} dU_a(i,1)dU_b(i,1)\right) |\langle 0|U_a(1,1)^\dagger U_a(2,1)^\dagger\cdots U_a(\ell,1)^\dagger U_b(\ell,1)\cdots U_b(2,1)U_b(1,1)|0\rangle|^4 \\
&\quad\times \int_{\mathrm{2design}} \left(\prod_{i=1}^{\ell} dU_a(i,2)dU_b(i,2)\right) |\langle 0|U_a(1,2)^\dagger U_a(2,2)^\dagger\cdots U_a(\ell,2)^\dagger U_b(\ell,2)\cdots U_b(2,2)U_b(1,2)|0\rangle|^4 \\
&\quad\times\cdots\times \int_{\mathrm{2design}} \left(\prod_{i=1}^{\ell} dU_a\Big(i,\frac{n}{m}\Big)dU_b\Big(i,\frac{n}{m}\Big)\right) \\
&\qquad\qquad \times |\langle 0|U_a(\ell,1)^\dagger U_a(\ell,2)^\dagger\cdots U_a(\ell,\ell)^\dagger U_b(\ell,\ell)\cdots U_b(\ell,2)U_b(\ell,1)|0\rangle|^4 \\
&= \left(\frac{1}{(2^m+1)2^{m-1}}\right)^{\frac{n}{m}} = 2^{\frac{n}{m}-1}\cdot\frac{2^n+1}{(2^m+1)^{\frac{n}{m}}}\mathcal{F}_{\mathrm{Haar}}^{(2)}(2^n). \qquad \text{(B.5)}
\end{aligned}$$

B.3 Proof of Theorem 4

Here we only show the computation of $\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{3,m,n})$. The computation of $\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{2,m,n})$ can be done similarly. The second frame potential can be expressed as follows:

$$\begin{aligned}
\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{3,m,n}) &= \int dU_a(1)dU_a(2)dU_a(3)dU_b(1)dU_b(2)dU_b(3)\ |\langle 0|U_a(1)^\dagger U_a(2)^\dagger U_a(3)^\dagger U_b(3)U_b(2)U_b(1)|0\rangle|^4 \\
&= \int_{\mathrm{2design}} \left(\prod_{i=1}^{3}\prod_{j=1}^{k(i)} dU_a(i,j)\right)\left(\prod_{i'=1}^{3}\prod_{j'=1}^{k(i)} dU_b(i',j')\right) \\
&\qquad \times |\langle 0|U_a(1,1)^\dagger U_a(1,2)^\dagger\cdots U_a(1,n/m)^\dagger U_a(2,1)^\dagger U_a(2,2)^\dagger\cdots U_a(2,n/m+1)^\dagger \\
&\qquad\quad \times U_a(3,1)^\dagger U_a(3,2)^\dagger\cdots U_a(3,n/m)^\dagger U_b(3,n/m)\cdots U_b(3,2)U_b(3,1) \\
&\qquad\qquad \times U_b(2,n/m+1)\cdots U_b(2,2)U_b(2,1)U_b(1,n/m)\cdots U_b(1,2)U_b(1,1)|0\rangle|^4.
\end{aligned}$$

Recall that $k(i)$ denotes the number of blocks in i-th layer; $k(i) = n/m$ for $i = 1,3$ and $k(i) = n/m+1$ for $i = 2$. We can get the final formula in the theorem by integrating only the unitary matrices in the first layer and the third layer. Executing integrals $\int_{\text{2design}} \prod_{j=1}^{n/m} dU_a(\ell, j)$ and $\int_{\text{2design}} \prod_{j'=1}^{n/m} dU_b(\ell, j')$ for $\ell = 1, 3$, we have

$$
\begin{aligned}
&\mathcal{F}^{(2)}(C_{\text{ALT}}^{3,m,n}) \\
&= \int_{\text{2design}} \left(\prod_{j=1}^{n/m+1} dU_a(2, j) \right) \left(\prod_{j'=1}^{n/m+1} dU_b(2, j') \right) \\
&\sum_{k_{11}^a=1}^{4} \sum_{k_{12}^a=1}^{4} \cdots \sum_{k_{1\frac{n}{m}}^a=1}^{4} \sum_{k_{31}^a=1}^{4} \sum_{k_{32}^a=1}^{4} \cdots \sum_{k_{3\frac{n}{m}}^a=1}^{4} \sum_{k_{11}^b=1}^{4} \sum_{k_{12}^b=1}^{4} \cdots \sum_{k_{1\frac{n}{m}}^b=1}^{4} \\
&\lambda_{k_{11}^a}^{(m)} \lambda_{k_{12}^a}^{(m)} \cdots \lambda_{k_{1\frac{n}{m}-1}^a}^{(m)} \lambda_{k_{1\frac{n}{m}}^a}^{(m)} \lambda_{k_{31}^a}^{(m)} \lambda_{k_{32}^a}^{(m)} \cdots \lambda_{k_{3\frac{n}{m}-1}^a}^{(m)} \lambda_{k_{3\frac{n}{m}}^a}^{(m)} \lambda_{k_{11}^b}^{(m)} \lambda_{k_{12}^b}^{(m)} \cdots \lambda_{k_{1\frac{n}{m}-1}^b}^{(m)} \lambda_{k_{1\frac{n}{m}}^b}^{(m)} \\
&\quad \times \Delta^{(k_{11}^a, k_{31}^a, k_{11}^b)} \left(U_a(2,1), U_b(2,1)\right) \times \Delta^{\left(k_{11}^a, k_{31}^a, k_{11}^b, k_{12}^a, k_{32}^a, k_{12}^b\right)} (U_a(2,2), U_b(2,2)) \\
&\quad \times \Delta^{\left(k_{12}^a, k_{32}^a, k_{12}^b, k_{13}^a, k_{33}^a, k_{13}^b\right)} (U_a(2,3), U_b(2,3)) \times \cdots \\
&\quad \times \Delta^{\left(k_{1\frac{n}{m}-1}^a, k_{3\frac{n}{m}-1}^a, k_{1\frac{n}{m}-1}^b, k_{1\frac{n}{m}}^a, k_{3\frac{n}{m}}^a, k_{1\frac{n}{m}}^b\right)} (U_a\Big(2, \frac{n}{m}\Big), U_b\Big(2, \frac{n}{m}\Big)) \\
&\quad \times \Delta^{(k_{1\frac{n}{m}}^a, k_{3\frac{n}{m}}^a, k_{1\frac{n}{m}}^b)} \left(U_a\Big(2, \frac{n}{m}+1\Big), U_b\Big(2, \frac{n}{m}+1\Big)\right) \\
&= \sum_{k_{11}^a=1}^{4} \sum_{k_{12}^a=1}^{4} \cdots \sum_{k_{1\frac{n}{m}}^a=1}^{4} \sum_{k_{31}^a=1}^{4} \sum_{k_{32}^a=1}^{4} \cdots \sum_{k_{3,\frac{n}{m}}^a=1}^{4} \sum_{k_{11}^b=1}^{4} \sum_{k_{12}^b=1}^{4} \cdots \sum_{k_{1\frac{n}{m}}^b=1}^{4} \\
&\int_{\text{2design}} dU_a(2,1) dU_b(2,1) \sqrt{\lambda_{k_{11}^a}^{(m)} \lambda_{k_{31}^a}^{(m)} \lambda_{k_{11}^b}^{(m)}} \Delta^{(k_{11}^a, k_{31}^a, k_{11}^b)} \left(U_a(2,1), U_b(2,1)\right) \\
&\times \int_{\text{2design}} dU_a(2,2) dU_b(2,b) \sqrt{\lambda_{k_{11}^a}^{(m)} \lambda_{k_{31}^a}^{(m)} \lambda_{k_{11}^b}^{(m)}} \sqrt{\lambda_{k_{12}^a}^{(m)} \lambda_{k_{32}^a}^{(m)} \lambda_{k_{12}^b}^{(m)}} \Delta^{\left(k_{11}^a, k_{31}^a, k_{11}^b, k_{12}^a, k_{32}^a, k_{12}^b\right)} (U_a(2,2), U_b(2,2)) \\
&\times \int_{\text{2design}} dU_a(2,3) dU_b(2,3) \sqrt{\lambda_{k_{12}^a}^{(m)} \lambda_{k_{32}^a}^{(m)} \lambda_{k_{12}^b}^{(m)}} \sqrt{\lambda_{k_{13}^a}^{(m)} \lambda_{k_{33}^a}^{(m)} \lambda_{k_{13}^b}^{(m)}} \Delta^{\left(k_{12}^a, k_{32}^a, k_{12}^b, k_{13}^a, k_{33}^a, k_{13}^b\right)} (U_a(2,3), U_b(2,3)) \\
&\times \cdots \times \int_{\text{2design}} dU_a\Big(2, \frac{n}{m}\Big) dU_b\Big(2, \frac{n}{m}\Big) \sqrt{\lambda_{k_{1\frac{n}{m}-1}^a}^{(m)} \lambda_{k_{3\frac{n}{m}-1}^a}^{(m)} \lambda_{k_{1\frac{n}{m}-1}^b}^{(m)}} \sqrt{\lambda_{k_{1\frac{n}{m}}^a}^{(m)} \lambda_{k_{3\frac{n}{m}}^a}^{(m)} \lambda_{k_{1\frac{n}{m}}^b}^{(m)}} \\
&\qquad\qquad \times \Delta^{\left(k_{1\frac{n}{m}-1}^a, k_{3\frac{n}{m}-1}^a, k_{1\frac{n}{m}-1}^b, k_{1\frac{n}{m}}^a, k_{3\frac{n}{m}}^a, k_{1\frac{n}{m}}^b\right)} (U_a\Big(2, \frac{n}{m}\Big), U_b\Big(2, \frac{n}{m}\Big)) \\
&\times \int_{\text{2design}} dU_a\Big(2, \frac{n}{m}+1\Big) dU_b\Big(2, \frac{n}{m}+1\Big) \sqrt{\lambda_{k_{1\frac{n}{m}}^a}^{(m)} \lambda_{k_{3\frac{n}{m}}^a}^{(m)} \lambda_{k_{1\frac{n}{m}}^b}^{(m)}} \\
&\qquad\qquad \times \Delta^{(k_{1\frac{n}{m}}^a, k_{3\frac{n}{m}}^a, k_{1\frac{n}{m}}^b)} \left(U_a\Big(2, \frac{n}{m}+1\Big), U_b\Big(2, \frac{n}{m}+1\Big)\right) \\
&= \mathbf{a}(3,m)^{\mathrm{T}} B(3,m)^{\frac{n}{m}-1} \mathbf{a}(3,m),
\end{aligned}
\tag{B.6}
$$

where we use definitions: (6.31), (6.32), (6.35), and (6.36). For the purpose of exemplifying the computation in the first equality of (B.6), we show the computation when $n/m = 2$ in the following. When $n/m = 2$, the second frame potential can be computed as follows:

$$
\begin{aligned}
&\mathcal{F}^{(2)}(C_{\text{ALT}}^{3,n,m}) \\
&= \int_{\text{2design}} dU_a(1,1) dU_a(1,2) dU_a(2,1) dU_a(2,2) dU_a(2,3) dU_a(3,1) dU_a(3,2) \\
&\qquad\qquad dU_b(1,1) dU_b(1,2) dU_b(2,1) dU_b(2,2) dU_b(2,3) dU_b(3,1) dU_b(3,2) \\
&\qquad\qquad |\langle 0| U_a^\dagger(1,1) U_a^\dagger(1,2) U_a^\dagger(2,1) U_a^\dagger(2,2) U_a^\dagger(2,3) U_a^\dagger(3,1) U_a^\dagger(3,2) \\
&\qquad\qquad\quad U_b(3,1) U_b(3,2) U_b(2,1) U_b(2,2) U_b(2,3) U_b(1,1) U_b(1,2) |0\rangle|^4
\end{aligned}
$$

$$
\begin{aligned}
= \int_{\text{2design}} & dU_a(1,1)dU_a(1,2)dU_a(2,1)dU_a(2,2)dU_a(2,3)dU_a(3,1)dU_a(3,2) \\
& dU_b(1,1)dU_b(1,2)dU_b(2,1)dU_b(2,2)dU_b(2,3)dU_b(3,1)dU_b(3,2) \\
& \sum_{\mathbf{i},\mathbf{j},\mathbf{k},\mathbf{l},\mathbf{p}} \Big(U_a^*(1,1)_{i_1 0}^{i_2 0} U_a^*(1,2)_{i_3 0}^{i_4 0} U_a^*(2,1)_{j_1 i_1} U_a^*(2,2)_{j_2 i_2}^{j_3 i_3} U_a^*(2,3)_{j_4 i_4} U_a^*(3,1)_{k_1 j_1}^{k_2 j_2} U_a^*(3,2)_{k_3 j_3}^{k_4 j_4} \\
& \quad U_b(3,1)_{k_1 l_1}^{k_2 l_2} U_b(3,2)_{k_3 l_3}^{k_4 l_4} U_b(2,1)_{l_1 p_1} U_b(2,2)_{l_2 p_2}^{l_3 p_3} U_b(2,3)_{l_4 p_4} U_b(1,1)_{p_1 0}^{p_2 0} U_b(1,2)_{p_3 0}^{p_4 0} \Big) \times \\
& \sum_{\mathbf{q},\mathbf{r},\mathbf{s},\mathbf{t},\mathbf{u}} \Big(U_b^*(1,1)_{q_1 0}^{q_2 0} U_b^*(1,2)_{q_3 0}^{q_4 0} U_b^*(2,1)_{r_1 q_1} U_b^*(2,2)_{r_2 q_2}^{r_3 q_3} U_b^*(2,3)_{r_4 q_4} U_b^*(3,1)_{s_1 r_1}^{s_2 r_2} U_b^*(3,2)_{s_3 r_3}^{s_4 r_4} \\
& \quad U_a(3,1)_{s_1 t_1}^{s_2 t_2} U_a(3,2)_{s_3 t_3}^{s_4 t_4} U_a(2,1)_{t_1 u_1} U_a(2,2)_{t_2 u_2}^{t_3 u_3} U_a(2,3)_{t_4 u_4} U_a(1,1)_{u_1 0}^{u_2 0} U_a(1,2)_{u_3 0}^{u_4 0} \Big) \times \\
& \sum_{\mathbf{i}',\mathbf{j}',\mathbf{k}',\mathbf{l}',\mathbf{p}'} \Big(U_a^*(1,1)_{i'_1 0}^{i'_2 0} U_a^*(1,2)_{i'_3 0}^{i'_4 0} U_a^*(2,1)_{j'_1 i'_1} U_a^*(2,2)_{j'_2 i'_2}^{j'_3 i'_3} U_a^*(2,3)_{j'_4 i'_4} U_a^*(3,1)_{k'_1 j'_1}^{k'_2 j'_2} U_a^*(3,2)_{k'_3 j'_3}^{k'_4 j'_4} \\
& \quad U_b(3,1)_{k'_1 l'_1}^{k'_2 l'_2} U_b(3,2)_{k'_3 l'_3}^{k'_4 l'_4} U_b(2,1)_{l'_1 p'_1} U_b(2,2)_{l'_2 p'_2}^{l'_3 p'_3} U_b(2,3)_{l'_4 p'_4} U_b(1,1)_{p'_1 0}^{p'_2 0} U_b(1,2)_{p'_3 0}^{p'_4 0} \Big) \times \\
& \sum_{\mathbf{q}',\mathbf{r}',\mathbf{s}',\mathbf{t}',\mathbf{u}'} \Big(U_b^*(1,1)_{q'_1 0}^{q'_2 0} U_b^*(1,2)_{q'_3 0}^{q'_4 0} U_b^*(2,1)_{r'_1 q'_1} U_b^*(2,2)_{r'_2 q'_2}^{r'_3 q'_3} U_b^*(2,3)_{r'_4 q'_4} U_b^*(3,1)_{s'_1 r'_1}^{s'_2 r'_2} U_b^*(3,2)_{s'_3 r'_3}^{s'_4 r'_4} \\
& \quad U_a(3,1)_{s'_1 t'_1}^{s'_2 t'_2} U_a(3,2)_{s'_3 t'_3}^{s'_4 t'_4} U_a(2,1)_{t'_1 u'_1} U_a(2,2)_{t'_2 u'_2}^{t'_3 u'_3} U_a(2,3)_{t'_4 u'_4} U_a(1,1)_{u'_1 0}^{u'_2 0} U_a(1,2)_{u'_3 0}^{u'_4 0} \Big)
\end{aligned}
\tag{B.7}
$$

where the bold symbols in the bottom the summation denote the multiple indices, e.g., $\mathbf{i} = i_1, i_2, i_3, i_4$. For the integrals $U_a(1,1), U_b(1,1), U_a(1,2), U_b(1,2)$,

$$
\int_{\text{2design}} dU_a(1,1) U_a(1,1)_{u_1 0}^{u_2 0} U_a(1,1)_{u'_1 0}^{u'_2 0} U_a^*(1,1)_{i_1 0}^{i_2 0} U_a^*(1,1)_{i'_1 0}^{i'_2 0} = \sum_{k_{11}^a=1}^{4} \lambda_{k_{11}^a}^{(m)} \Delta_{u_1 0 u'_1 0 i_1 0 i'_1 0}^{k_{11}^a} \Delta_{u_2 0 u'_2 0 i_2 0 i'_2 0}^{k_{11}^a}, \tag{B.8}
$$

$$
\int_{\text{2design}} dU_b(1,1) U_b(1,1)_{p_1 0}^{p_2 0} U_b(1,1)_{p'_1 0}^{p'_2 0} U_b^*(1,1)_{q_1 0}^{q_2 0} U_b^*(1,1)_{q'_1 0}^{q'_2 0} = \sum_{k_{11}^b=1}^{4} \lambda_{k_{11}^b}^{(m)} \Delta_{p_1 0 p'_1 0 q_1 0 q'_1 0}^{k_{11}^b} \Delta_{p_2 0 p'_2 0 q_2 0 q'_2 0}^{k_{11}^b}, \tag{B.9}
$$

$$
\int_{\text{2design}} dU_a(1,2) U_a(1,2)_{u_3 0}^{u_4 0} U_a(1,2)_{u'_3 0}^{u'_4 0} U_a^*(1,2)_{i_3 0}^{i_4 0} U_a^*(1,2)_{i'_3 0}^{i'_4 0} = \sum_{k_{12}^a=1}^{4} \lambda_{k_{12}^a}^{(m)} \Delta_{u_3 0 u'_3 0 i_3 0 i'_3 0}^{k_{12}^a} \Delta_{u_4 0 u'_4 0 i_4 0 i'_4 0}^{k_{12}^a}, \tag{B.10}
$$

$$
\int_{\text{2design}} dU_b(1,2) U_b(1,2)_{p_3 0}^{p_4 0} U_b(1,2)_{p'_3 0}^{p'_4 0} U_b^*(1,2)_{q_3 0}^{q_4 0} U_b^*(1,2)_{q'_3 0}^{q'_4 0} = \sum_{k_{12}^b=1}^{4} \lambda_{k_{12}^b}^{(m)} \Delta_{p_3 0 p'_3 0 q_3 0 q'_3 0}^{k_{12}^b} \Delta_{p_4 0 p'_4 0 q_4 0 q'_4 0}^{k_{12}^b} \tag{B.11}
$$

hold. For the integrals $U_a(3,1), U_b(3,1), U_a(3,2), U_b(3,2)$,

$$\int_{\text{2design}} dU_a(3,1)dU_b(3,1)$$
$$\sum_{\substack{k_1,k_2\\ s_1,s_2}}\sum_{\substack{k'_1,k'_2\\ s'_1,s'_2}} U_a^*(3,1)^{k_2 j_2}_{k_1 j_1} U_b(3,1)^{k_2 l_2}_{k_1 l_1} U_b^*(3,1)^{s_2 r_2}_{s_1 r_1} U_a(3,1)^{s_2 t_2}_{s_1 t_1} U_a^*(3,1)^{k'_2 j'_2}_{k'_1 j'_1} U_b(3,1)^{k'_2 l'_2}_{k'_1 l'_1} U_b^*(3,1)^{s'_2 r'_2}_{s'_1 r'_1} U_a(3,1)^{s'_2 t'_2}_{s'_1 t'_1}$$
$$= \sum_{k^a_{31}=1}^{4} \lambda^{(m)}_{k^a_{31}} \Delta^{k^a_{31}}_{j_1 l_1 j'_1 l'_1 t_1 r_1 t'_1 r'_1} \Delta^{k^a_{31}}_{j_2 l_2 j'_2 l'_2 t_2 r_2 t'_2 r'_2} \tag{B.12}$$

$$\int_{\text{2design}} dU_a(3,2)dU_b(3,2)$$
$$\sum_{\substack{k_3,k_4\\ s_3,s_4}}\sum_{\substack{k'_3,k'_4\\ s'_3,s'_4}} U_a^*(3,2)^{k_4 j_4}_{k_3 j_3} U_b(3,2)^{k_4 l_4}_{k_3 l_3} U_b^*(3,1)^{s_4 r_4}_{s_3 r_3} U_a(3,2)^{s_4 t_4}_{s_3 t_3} U_a^*(3,2)^{k'_4 j'_4}_{k'_3 j'_3} U_b(3,2)^{k'_4 l'_4}_{k'_3 l'_3} U_b^*(3,2)^{s'_4 r'_4}_{s'_3 r'_3} U_a(3,2)^{s'_4 t'_4}_{s'_3 t'_3}$$
$$= \sum_{k^a_{31}=1}^{4} \lambda^{(m)}_{k^a_{31}} \Delta^{k^a_{32}}_{j_3 l_3 j'_3 l'_3 t_3 r_3 t'_3 r'_3} \Delta^{k^a_{32}}_{j_4 l_4 j'_4 l'_4 t_4 r_4 t'_4 r'_4} \tag{B.13}$$

hold. Substituting (B.8), (B.9), (B.10), (B.11), (B.12), and (B.13) to (B.7), we get

$$\mathcal{F}^{(2)}(C^{3,n,m}_{\text{ALT}})$$
$$= \sum_{k^a_{11}=1}^{4}\sum_{k^a_{31}=1}^{4}\sum_{k^b_{11}=1}^{4}\sum_{k^a_{12}=1}^{4}\sum_{k^a_{32}=1}^{4}\sum_{k^b_{12}=1}^{4} \lambda^{(m)}_{k^a_{11}}\lambda^{(m)}_{k^a_{31}}\lambda^{(m)}_{k^b_{11}}\lambda^{(m)}_{k^a_{12}}\lambda^{(m)}_{k^a_{32}}\lambda^{(m)}_{k^b_{12}}$$

$$\int_{\text{2design}} dU_a(2,1)dU_b(2,1)\Bigg[\sum_{\substack{u_1u'_1i_1i'_1\\ j_1j'_1l_1l'_1}}\sum_{\substack{r_1r'_1t_1t'_1\\ p_1p'_1q_1q'_1}} \Delta^{k^a_{11}}_{u_1 0 u'_1 0 i_1 0 i'_1 0}\Delta^{k^a_{31}}_{j_1 l_1 j'_1 l'_1 t_1 r_1 t'_1 r'_1}\Delta^{k^b_{11}}_{p_1 0 p'_1 0 q_1 0 q'_1 0} \tag{B.14}$$
$$U_a(2,1)_{t_1 u_1} U_a(2,1)_{t'_1 u'_1} U_a^*(2,1)_{j_1 i_1} U_a^*(2,1)_{j'_1 i'_1} U_b(2,1)_{l_1 p_1} U_b(2,1)_{l'_1 p'_1} U_b^*(2,1)_{r_1 q_1} U_b^*(2,1)_{r'_1 q'_1}\Bigg] \times$$

$$\int_{\text{2design}} dU_a(2,2)dU_b(2,2)\Bigg[\sum_{\substack{u_2u'_2i_2i'_2\\ j_2j'_2l_2l'_2}}\sum_{\substack{r_2r'_2t_2t'_2\\ p_2p'_2q_2q'_2}}\sum_{\substack{u_3u'_3i_3i'_3\\ j_3j'_3l_3l'_3}}\sum_{\substack{r_3r'_3t_3t'_3\\ p_3p'_3q_3q'_3}} \Delta^{k^a_{11}}_{u_2 0 u'_2 0 i_2 0 i'_2 0}\Delta^{k^a_{31}}_{j_2 l_2 j'_2 l'_2 t_2 r_2 t'_2 r'_2}\Delta^{k^b_{11}}_{p_2 0 p'_2 q_2 0 q'_2 0}$$
$$\Delta^{k^a_{12}}_{u_3 0 u'_3 0 i_3 0 i'_3 0}\Delta^{k^a_{32}}_{j_3 l_3 j'_3 l'_3 t_3 r_3 t'_3 r'_3}\Delta^{k^b_{12}}_{p_3 0 p'_3 q_3 0 q'_3 0} U_a(2,2)^{t_3 u_3}_{t_2 u_2} U_a(2,2)^{t'_3 u'_3}_{t'_2 u'_2} U_a^*(2,2)^{j_3 i_3}_{j_2 i_2} U_a^*(2,2)^{j'_3 i'_3}_{j'_2 i'_2}$$
$$U_b(2,2)^{l_3 p_3}_{l_2 p_2} U_b(2,2)^{l'_3 p'_3}_{l'_2 p'_2} U_b^*(2,2)^{r_3 q_3}_{r_2 q_2} U_b^*(2,2)^{r'_3 q'_3}_{r'_2 q'_2}\ \Bigg] \times$$

$$\int_{\text{2design}} dU_a(2,3)dU_b(2,3)\Bigg[\sum_{\substack{u_4u'_4i_4i'_4\\ j_4j'_4l_4l'_4}}\sum_{\substack{r_4r'_4t_4t'_4\\ p_4p'_4q_4q'_4}} \Delta^{k^a_{12}}_{u_4 0 u'_4 0 i_4 0 i'_4 0}\Delta^{k^a_{32}}_{j_4 l_4 j'_4 l'_4 t_4 r_4 t'_4 r'_4}\Delta^{k^b_{12}}_{p_4 0 p'_4 0 q_4 0 q'_4 0} \tag{B.15}$$
$$U_a(2,3)_{t_4 u_4} U_a(2,3)_{t'_4 u'_4} U_a^*(2,3)_{j_4 i_4} U_a^*(2,3)_{j'_4 i'_4} U_b(2,3)_{l_4 p_4} U_b(2,3)_{l'_4 p'_4} U_b^*(2,3)_{r_4 q_4} U_b^*(2,3)_{r'_4 q'_4}\Bigg]$$
$$= \sum_{k^a_{11}=1}^{4}\sum_{k^a_{31}=1}^{4}\sum_{k^b_{11}=1}^{4}\sum_{k^a_{12}=1}^{4}\sum_{k^a_{32}=1}^{4}\sum_{k^b_{12}=1}^{4} \lambda^{(m)}_{k^a_{11}}\lambda^{(m)}_{k^a_{31}}\lambda^{(m)}_{k^b_{11}}\lambda^{(m)}_{k^a_{12}}\lambda^{(m)}_{k^a_{32}}\lambda^{(m)}_{k^b_{12}}$$
$$\int_{\text{2design}} dU_a(2,1)dU_b(2,1)\Delta^{(k^a_{11},k^a_{31},k^b_{11})}\left(U_a(2,1),U_b(2,1)\right)\times$$
$$\int_{\text{2design}} dU_a(2,2)dU_b(2,2)\Delta^{\left(k^a_{11},k^a_{31},k^b_{11},k^a_{12},k^a_{32},k^b_{12}\right)}(U_a(2,2),U_b(2,2))\times$$
$$\int_{\text{2design}} dU_a(2,3)dU_b(2,3)\Delta^{(k^a_{12},k^a_{32},k^b_{12})}\left(U_a(2,3),U_b(2,3)\right), \tag{B.16}$$

which is the right hand side of the first equality in (B.6) when $n/m = 2$.

B.4 Proof of Theorem 5

Similar to Theorem 4, we only show the inequality for $\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{3,m,n})$ here. The inequality for $\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{2,m,n})$ can be shown in the same manner. In the process of showing the final inequality of the theorem, we expand $\mathbf{a}(3,m)$ and $B(3,m)$ as the sum of a vector/matrix whose components are $O(1)$ and a vector/matrix whose components are $O(1/2^{m/2})$.

To evaluate Eq. (6.31)[1], we can expand $\mathbf{a}(3,m)$ as

$$\mathbf{a}(3,m) = \frac{1}{2^m}\left(\mathbf{v}_0 + \frac{1.2}{2^{m/2}}\mathbf{v}_1\right), \tag{B.17}$$

where

$$\mathbf{v}_{0i} = \begin{cases} 1 & i=1 \quad (k_a,k_b,k_c=1), \\ 1 & i=22 \quad (k_a,k_b,k_c=2), \\ 0 & \text{otherwise}, \end{cases} \qquad |\mathbf{v}_{1i}| < 1. \tag{B.18}$$

Also, evaluating Eq. (6.35), we can expand $B(3,m)$ as

$$B(3,m) = \frac{1}{2^{2m}}\left(D + \frac{1.3}{2^{m/2-6}}X\right), \tag{B.19}$$

where

$$D_{ij} = \begin{cases} 1 & i=1, j=1 \quad (k_a,k_b,k_c,k_d,k_e,k_f=1), \\ 1 & i=22, j=22 \quad (k_a,k_b,k_c,k_d,k_e,k_f=2), \\ 0 & \text{otherwise}, \end{cases} \qquad |X_{ij}| < \frac{1}{64}. \tag{B.20}$$

With $\alpha = n/m$, let $g_\alpha^k(X,D)$ be as the set of matrices expressed by $\prod_{i=1}^{\alpha} R_i$ where $R_i = D$ or X and the number of Xs in $\{R_i\}$ is k. For example, $XDXX \in g_4^3(X,D)$ and $XDDD \in g_4^1(X,D)$. Then, $\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{3,m,n})$ is expanded as

$$\begin{aligned}\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{3,m,n}) &= \frac{1}{2^{2m}}\left(\mathbf{v}_0^T + \frac{1.2}{2^{m/2}}\mathbf{v}_1^T\right)\frac{1}{2^{2n-2m}}\left(D + \frac{1.3}{2^{m/2-6}}X\right)^{\alpha-1}\left(\mathbf{v}_0 + \frac{1.2}{2^{m/2}}\mathbf{v}_1\right) \\ &= \frac{1}{2^{2n}}\left(\mathbf{v}_0^T D^{\alpha-1}\mathbf{v}_0 + \left(\frac{2.4}{2^{m/2}}\right)\mathbf{v}_1^T D^{\alpha-1}\mathbf{v}_0 + \left(\frac{1.2^2}{2^m}\right)\mathbf{v}_1^T D^{\alpha-1}\mathbf{v}_1\right) \\ &\quad + \frac{1}{2^{2n}}\sum_{k=1}^{\alpha-1}\left(\frac{1.3}{2^{m/2-6}}\right)^k \sum_{i=1}^{{}_{\alpha-1}\mathrm{C}_k}\left(\mathbf{v}_0^T g_{\alpha i}^k \mathbf{v}_0 + \left(\frac{2.4}{2^{m/2}}\right)\mathbf{v}_1^T g_{\alpha i}^k\mathbf{v}_0 + \left(\frac{1.2^2}{2^m}\right)\mathbf{v}_1^T g_{\alpha i}^k \mathbf{v}_1\right),\end{aligned} \tag{B.21}$$

where $g_{\alpha i}^k$ $(i = 1, 2 \ldots {}_\alpha\mathrm{C}_k)$ is an element of $g_\alpha^k(X,D)$. For an arbitrary $g \in g_\alpha^k(X,D)$ with $k \geq 1$,

$$\mathbf{v}_r^{\mathrm{T}} g \mathbf{v}_s < (1,1,\cdots,1)\begin{pmatrix} 1 & 0\cdots & 0 \\ 0 & 1\cdots & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0\cdots & 1 \end{pmatrix}^{\alpha-k}\begin{pmatrix} \frac{1}{64} & \frac{1}{64}\cdots & \frac{1}{64} \\ \frac{1}{64} & \frac{1}{64}\cdots & \frac{1}{64} \\ \vdots & \vdots & \vdots \\ \frac{1}{64} & \frac{1}{64}\cdots & \frac{1}{64} \end{pmatrix}^{k}\begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{pmatrix} = 64 \tag{B.22}$$

holds. For D and $\mathbf{v}_0, \mathbf{v}_1$

$$\mathbf{v}_1^{\mathrm{T}} D^{\frac{n}{m}-1}\mathbf{v}_s < \sum_{i=1}^{64}(1\cdot(\delta_{i1}+\delta_{i22})\cdot 1) = 2 \tag{B.23}$$

$$\mathbf{v}_0^{\mathrm{T}} D^{\frac{n}{m}-1}\mathbf{v}_0 = 2 \tag{B.24}$$

[1]The evaluation procedure is straightforward, but a lot of computation is required. Thus, instead of performing a hand-calculation, we built an algorithm to evaluate (6.31) for arbitrary m and derived the expansion formula by computational calculation. We also built an algorithm for evaluating (6.35) and derived the expansion formula by computational calculation.

holds where $r, s = 0, 1$. By using the inequalities (B.22), (B.23), and (B.24), the upper bound for $\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{3,m,n})$ is derived as follows:

$$\begin{aligned}\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{3,m,n}) &< \frac{1}{2^{2n}}\left(2+2\left(\frac{2.4}{2^{m/2}}\right)+2\left(\frac{1.2^2}{2^m}\right)+64\sum_{k=1}^{\alpha-1}{}_{\alpha-1}\mathrm{C}_k\left(\frac{1.3}{2^{m/2-6}}\right)^k 1^k\left(1+\frac{1.2}{2^{m/2}}\right)^2\right)\\ &= \frac{1}{2^{2n-1}}\left(1+\frac{1.2}{2^m}\right)^2\left(1+32\left(\left(1+\frac{1.3}{2^{m/2-6}}\right)^{\alpha-1}-1\right)\right)\\ &= \left(1+\frac{1}{2^n}\right)\left(1+\frac{1.2}{2^m}\right)^2\left(1+32\left(\left(1+\frac{83.2}{2^{m/2}}\right)^{\alpha-1}-1\right)\right)\mathcal{F}_{\mathrm{Haar}}^{(2)}(2^n). \qquad \text{(B.25)}\end{aligned}$$

B.5 Proof of Corollary 1

As in the above theorems, we only show the inequality for $\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{3,m,n})$ here. When $m = 2a\log_2 n$,

$$\left(1+\frac{83.2}{2^{m/2}}\right)^{\alpha-1} < \left[\left(1+\frac{83.2}{2^{m/2}}\right)^{\frac{2^{m/2}}{83.2}}\right]^{\frac{83.2n}{2^{m/2}m}} < e^{\frac{83.2n}{2^{m/2}m}} = e^{\frac{41.6}{an^{a-1}\log_2 n}}. \qquad \text{(B.26)}$$

If $41.6/(an^{a-1}\log_2 n) < 1$,

$$e^{\frac{41.6}{an^{a-1}\log_2 n}} < 1+(e-1)\frac{41.6}{an^{a-1}\log_2 n}. \qquad \text{(B.27)}$$

Substituting Eqs. (B.26), (B.27), and $m = 2a\log_2 n$ into (B.25), we get

$$\mathcal{F}^{(2)}(C_{\mathrm{ALT}}^{3,m,n}) < \left(1+\frac{1}{2^n}\right)\left(1+\frac{1.2}{n^{2a}}\right)^2\left(1+\frac{2288}{an^{a-1}\log_2 n}\right)\mathcal{F}_{\mathrm{Haar}}^{(2)}(2^n). \qquad \text{(B.28)}$$