國立臺灣大學理學院物理學研究所

碩士論文

Department of Physics

College of Science

National Taiwan University

Master's Thesis

以深度強化學習實現抗噪量子閘

Robust quantum gates by deep reinforcement learning

林晉揚

Chin-Yang Lin

指導教授：管希聖 博士

Advisor: Hsi-Sheng Goan, Ph.D.

中華民國 113 月 1 月

January, 2024

doi:10.6342/NTU202400695

國立臺灣大學碩士學位論文
口試委員會審定書
MASTER'S THESIS ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

（論文中文題目）(Chinese title of Master's thesis)

以深度強化學習實現抗噪量子閘

（論文英文題目）(English title of Master's thesis)

Robust quantum gates by deep reinforcement learning

本論文係 林晉揚 (姓名) R10222046 (學號)在國立臺灣大學 物理所 (系/所/學位學程)完成之碩士學位論文，於民國 113 年 01 月 23 日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Department / Institute of Physics on 23 (date) 01 (month) 2024 (year) have examined a Master's thesis entitled above presented by Chin-Yang Lin (name) R10222046 (student ID) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

管希聖

（指導教授 Advisor）

林俊達　伍祥華

doi:10.6342/NTU202400695

摘要

量子計算在加密、金融、科學模擬等領域革新中深具潛力，然而現實世界中，量子硬體的雜訊會嚴重地妨礙實行量子演算法，因此實現抗噪量子閘是使量子計算發揮成效的重要前提。本文以創新的方法將量子控制問題整合進強化學習框架中，並使用一種稱為近似策略最佳化的強化學習演算法配合深度神經網路，建立出容錯量子計算所需的高保真、抗雜訊的量子閘。

關鍵字：強化學習、機器學習、神經網路、近似策略最佳化、量子控制、抗噪量子閘

doi:10.6342/NTU202400695

Abstract

Quantum computing holds immense promise to revolutionize several industries such as cryptography, finance, scientific simulations and so on. However, the real-world application of quantum algorithms is severely hindered by the presence of noise in quantum hardware. Achieving noise-robust quantum gates is an important prerequisite to harness the power of quantum computing. This thesis presents an innovative way to address the challenge by mapping the quantum gate control problem into the reinforcement learning (RL) framework. Utilizing a RL algorithm called proximal policy optimization equipped with deep neural networks, we achieve constructing high-fidelity and robust single-qubit and two-qubit quantum gates in the presence of quasi-static noise, paving the way for fault-tolerant quantum computation.

Keywords: Reinforcement learning, Machine learning, Neural networks, Proximal policy optimization, Quantum control, Robust quantum gates

doi:10.6342/NTU202400695

Contents

doi:10.6342/NTU202400695

doi:10.6342/NTU202400695

doi:10.6342/NTU202400695

List of figures

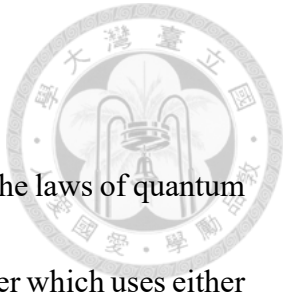
doi:10.6342/NTU202400695

doi:10.6342/NTU202400695

doi:10.6342/NTU202400695

List of tables

doi:10.6342/NTU202400695



Chapter 1 Introduction

Quantum computing is a cutting-edge technology that leverages the laws of quantum mechanics to process and store information. Unlike a classical computer which uses either one of Boolean states to represent information, a quantum computer utilizes superposition of states along with quantum entanglement. Several quantum algorithms have been theoretically proved to outperform their classical counterparts for some problems. For example, Shor's algorithm factorizes integers within only polynomial time [1]. Grover's algorithm searches an unsorted database with a quadratic speed up [2]. Quantum computing with fast and efficient quantum algorithms provides a paradigm shift that promises to potentially impact numerous domains like cryptography, finance and drug industry. To realize the power of quantum computing, it is important to build a set of high-fidelity and noise-robust quantum gates in real-world quantum computers.

Recently, machine learning (ML) has been rapidly developed and widely applied in many versatile domains. ML algorithms seek to identify patterns and make predictions based on a large amount of data they have learned. There are three types of ML: supervised learning, unsupervised learning and reinforcement learning (RL). We here choose the RL method to address the challenge of constructing robust high-fidelity quantum gates. Equipped with deep neural networks (DNN), deep reinforcement learning (DRL) agents are able to plan policies or make estimations when interacting with the

doi:10.6342/NTU202400695

environment. Mapping the quantum gate control problem into the RL framework, we use a RL algorithm called proximal policy optimization (PPO) [3], to optimize the piecewise constant (PWC) control pulses, generating a set of robust quantum gates with fidelities beyond the fault-tolerant threshold of quantum error correction for generic qubit models in the presence of quasi-static noise (QSN).

Some research groups have also applied the DRL method to the optimal control of quantum gates in an ideal environment with infidelities around 10^{-3} to 10^{-4} [4][5][6]. Lin (2022) has constructed the ideal X, H and CNOT gates with the lowest ideal infidelities as 10^{-16}, 10^{-16} and 10^{-12}, respectively by the same PPO DRL algorithm [7]. Niu et al. (2019) have used the similar policy optimization DRL algorithm to construct a two-qubit gates in a noisy environment, but they introduce the time-varying noise into their deep neural network (DNN) model such that the output control pulses will alter with the unknown noise, not giving the unique robust pulses [8]. In this thesis, we give a preliminary attempt of applying the PPO DRL agent to perform robust high-fidelity quantum gates in a noisy environment, and we have shown that it works for the QSN. Moreover, most of these DRL approaches [4][8] use the quantum states of the qubit systems as environment observations, requiring knowing the underlying quantum dynamics. In contrast, the environment observations for our DRL agent are the control pulse strengths, which are more feasible and practical for the realistic control

doi:10.6342/NTU202400695

experiments.

We organize the thesis as follows. First, we introduce the DRL in Chapter 2. Next, we describe the quantum gate control problem in Chapter 3. Then, we integrate the control problem into the DRL framework in Chapter 4, and construct three iconic quantum gates, namely X, H and CNOT gates, in the presence of noise in Chapter 5. Finally, we discuss these results in Chapter 6, and finally we conclude our work in Chapter 7.

doi:10.6342/NTU202400695

Chapter 2 Deep reinforcement learning

2.1 Reinforcement learning

Inspired by behavioral psychology, reinforcement learning focuses on training an intelligent agent to act responses to its observations in order to maximize a cumulative reward. We first introduce some terminologies and their relationships in RL:

State (s): It is a representation of an environment at a certain time t. To be more specific, information of state that is fully or partially observed by an agent is called an observation o. However, state s and observation o are usually interchangeable in the RL notation.

Action (a): It is a decision made by an agent to interact with an environment.

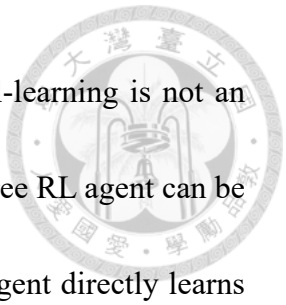
Policy (π): It is a deterministic or stochastic strategy that defines an agent's behavior, mapping states to actions, $a_t \sim \pi(\cdot|s_t)$.

State transition (P): Given a current state s_t and an agent's action a_t, it tells how an environment maps them to a next state $s_{t+1} \sim P(\cdot\,|s_t, a_t)$.

Reward (r): It is a numerical value that an environment provides after each agent's action, serving as feedback.

RL is classified into two main taxonomies, model-based and model-free, by telling whether a RL agent learns a model of environment or not. With a learned model, a model-based RL owns sample efficiency and allows an agent to plan ahead. A famous example

doi:10.6342/NTU202400695



of this kind is Google DeepMind's AlphaZero [9]. However, model-learning is not an easy task, a model-free RL agent is instead easier to train. A model-free RL agent can be further classified into policy-based or value-based. A policy-based agent directly learns policies to take responses, while a value-based one make decision according to its estimation of options. We here choose the model-free approach, called proximal policy optimization (PPO) with actor-critic style implementation which leverages both merits of policy-based and value-based algorithms.

2.2 Markov decision process

Markov decision process (MDP) refers to a state transition P of environment obeying Markov property. It says that a future state of an environment depends only on a current state and action, not on an entire history of states and actions. Though not being a strict requirement, MDP is widely used in RL because it provides a well-defined structure of environment.

2.3 Proximal policy optimization

PPO is known for its stability, robustness, and ease of implementation. It is an on-policy agent who updates its policy typically by taking several epochs of objective

doi:10.6342/NTU202400695

maximization with a batch of data collected by its current policy,

$$\theta_{k+1} = \arg\max_{\theta} \mathbb{E}_{s,a\sim\pi_{\theta_k}} [L(s, a, \theta_k, \theta)] \ , \qquad (2.3\text{-}1)$$

where $\mathbb{E}$ is the expectation function, π is PPO's stochastic policy whose implicit parameters are θ with the subscript index k denoting generations of policies. The objective function $L(\theta)$ is,

$$\min \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A(s,a), \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon \right) A(s,a) \right] , \qquad (2.3\text{-}2)$$

where clip is a function that clip the first argument between the second and the third ones, ϵ is a small-value hyperparameter that limits updating policy parameters, $A(s,a)$ is an advantage function that estimates how good an action a is. To understand what L is doing, we first define a ratio,

$$r(s, a, \theta_k, \theta) = \frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} \ . \qquad (2.3\text{-}3)$$

Taking a positive $A(s,a)$ as an example, we certainly hope that probability of taking the action a is as high as possible, that is, r is far greater than one. However, a large update from θ_k to θ can bring training instability, we clip r by $1+\epsilon$ such that a new policy does not benefit by going far away from the current one. The overall PPO algorithm is shown in Algorithm 1.

doi:10.6342/NTU202400695

2.4 Value function

To determine the design of advantage function $A(s,a)$, we recall the goal of RL is to maximize the cumulative reward, or called return,

$$G_t = \sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k} \ , \tag{2.4-1}$$

where $\gamma \in [0,1]$ introduced here is a future discount factor that reflects preference of an immediate reward or is mathematically designed to avoid divergence of an infinite sum. Nevertheless, PPO's policy as well as the state transition function P of an environment is stochastic. We are not going to focus on only one trajectory of rewards but an expected one,

$$\mathbb{E}_{s,a\sim\pi,P}\left[\sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k}\right] . \tag{2.4-2}$$

Practically, it is still hardly possible to exactly calculate this expected value which requires traversing all of attainable trajectories. Given an agent taking action by its policy π and starting at the state s_t, we use an estimator called value function to approximate the expected value,

$$V(s_t) = \mathbb{E}_{s,a\sim\pi}\left[\sum_{k=0}^{\infty} \gamma^k \cdot r_{t+k}\right] . \tag{2.4-3}$$

With the value function $V(s_t)$, we can estimate consequences of choosing different actions. For example, starting at the state s_0, there are two actions a_1 and a_2 that lead

doi:10.6342/NTU202400695

us to the states s_1 and s_2 with the values $v_1 = V(s_1)$ and $v_2 = V(s_2)$ respectively. The PPO agent then updates the probabilities of these two actions according to their consequent values.

2.5 Advantage function

2.5.1 Temporal difference error

Sometimes, we are not concerned about the goodness of actions in absolute sense which directly consider the consequent value $V(s_{t+1})$. Instead, we want to update a policy such that it increases probability of better-than-average actions and decreases opposite ones. We use temporal difference (TD) error,

$$\delta_t^V = r_t + \gamma \cdot V(s_{t+1}) - V(s_t) \ , \qquad (2.5\text{-}1)$$

to be an advantage function estimation $A_t^{(1)}$ of a one-step action a_t. The first two terms $r_t + \gamma \cdot V(s_{t+1})$ is the estimated value at the state s_t when we intentionally choose the action a_t, while the last term $V(s_t)$ is the overall estimated value at the state s_t.

2.5.2 General advantage estimation

We can similarly calculate following advantage function estimations if we further take actions a_{t+1}, a_{t+2}, and all the way up to the final state,

doi:10.6342/NTU202400695

$$A_t^{(1)} = \delta_t^V = -V(s_t) + r_t + V(s_{t+1}) \ ,$$

$$A_t^{(2)} = \delta_t^V + \gamma \cdot \delta_{t+1}^V = -V(s_t) + r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot V(s_{t+2}) \ ,$$

$$A_t^{(3)} = \delta_t^V + \gamma \cdot \delta_{t+1}^V + \gamma^2 \cdot \delta_{t+2}^V = -V(s_t) + r_t + \gamma \cdot r_{t+1} + \gamma^2 \cdot r_{t+1} + \gamma^3 \cdot V(s_{t+2}) \ ,$$

$$\vdots$$

$$\begin{aligned} A_t^{(k)} &= \sum_{k=0}^{k-1} \gamma^k \cdot \delta_{t+k}^V \\ &= -V(s_t) + r_t + \gamma \cdot r_{t+1} + \cdots + \gamma^{k-1} \cdot r_{t+k-1} + V(s_{t+k}) \ . \end{aligned} \tag{2.5-2}$$

We use generalized advantage estimator $A_t^{GAE(\gamma,\lambda)}$ to be our final advantage function estimator $A^{\pi_{\theta_k}}(s,a)$, which is defined as exponentially-weighted average of these above k-step advantage function estimators,

$$A_t^{GAE(\gamma,\lambda)} = (1-\lambda)\left(\sum_{k=1}^{\infty} \lambda^{k-1} \cdot A_t^{(k)}\right) = \sum_{l=0}^{\infty} (\gamma\lambda)^l \cdot \delta_{t+l}^V \ , \tag{2.5-3}$$

where the parameter $\lambda \in [0,1]$ controls a bias-variance tradeoff by discounting future TD errors δ_{t+l}^V [10].

From Eq. (2.5-2), we know that a return G_t^k with definite rewards up to the k-step is $A_t^{(k)} + V(s_t)$. Similarly, we define a more general return,

$$G_t^\lambda = A_t^{GAE(\gamma,\lambda)} + V(s_t) \ , \tag{2.5-4}$$

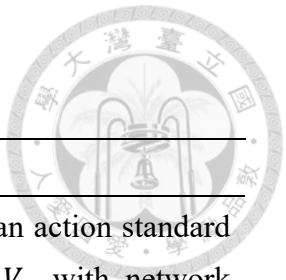
called λ-return.

doi:10.6342/NTU202400695

2.6 Deep neural network

The policy function $\pi_\theta(a|s)$ and the value function $V(s_t)$ of PPO require sophisticated mechanisms to map a state (observation) into an action probability distribution and an estimated value of expected return respectively. It is barely possible to directly record every relation of state, action and value into a large lookup table. Instead, we utilize two deep neural networks (DNN), called policy (actor) network and value (critic) network, to approximate the policy function $\pi_\theta(a|s)$ and the value functions $V(s_t)$. In a policy network, there is an embedded network called action projection network which projects quasi output of policy network to final output action that satisfies an action spec of environment.

doi:10.6342/NTU202400695



Algorithm 1 PPO with GAE

1. Initialize a replay buffer, an optimizer with learning rate lr, an action standard deviation σ_a, a policy network π_θ and a value network V_ϕ with network parameters θ and ϕ.
2. For iteration = 1, 2, …:
3. Clear the replay buffer.
4. For number of data collections = 1, 2, …:
5. Run π_{θ_k} in the environment to get a trajectory of $(s_t,\ a_t,\ \pi_{\theta_k}(a_t|s_t),\ r_t)$.
6. Get values $v_t = V_\phi(s_t)$ to compute $A_t^{GAE(\gamma,\lambda)}$ in Eq. (2.5-3) and G_t^λ in Eq. (2.5-4).
7. Add data $(s_t,\ a_t,\ \pi_{\theta_k}(a_t|s_t),\ A_t^{GAE(\gamma,\lambda)},\ G_t^\lambda)$ to the replay buffer.
8. For number of epochs = 1, 2, …:
9. Randomly sample a batch of data from the replay buffer.
10. Get $\pi_\theta(a_t|s_t)$ and $V_\phi(s_t)$.
11. Get $L(\theta)$ in Eq. (2.3-2).
12. Get mean squared error:

$$L(\phi) = \sum \left(G_t^\lambda - V_\phi(s_t)\right)^2 .$$

13. Update π_θ and V_ϕ with the optimizer:

$$\theta \leftarrow \theta + lr \cdot \nabla_\theta L(\theta) , \qquad \phi \leftarrow \phi - lr \cdot \nabla_\phi L(\phi) .$$

14. If using adaptive learning:
15. Update lr and/or σ_a.

doi:10.6342/NTU202400695

Chapter 3 Quantum computing

3.1 Qubit

A qubit, short for “quantum bit”, is a fundamental unit of quantum information in quantum computing. Several kinds of quantum system can be implemented as a qubit such as superconducting qubit, ion trap and semiconductor quantum dot. Unlike a classical bit which represent information in either 0 or 1, a qubit can exist in a superposition of both states,

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle \ , \tag{3.1-1}$$

where $\alpha, \beta \in \mathbb{C}$ are probability amplitudes which require that $|\alpha|^2 + |\beta|^2 = 1$, $|x\rangle$ is a ket vector in Dirac bra-ket notation to represent a quantum state. They can also be written in vector forms,

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \ , \qquad |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \ . \tag{3.1-2}$$

For a more general n-qubit representation, we construct them by the tensor product of each single qubit,

$$\begin{aligned} |\psi\rangle &= \sum_{b_i=0,1}^{2^n} c_{b_1 b_2 \dots b_n} |b_1\rangle \otimes |b_2\rangle \otimes \dots \otimes |b_n\rangle \\ &= \sum_{b_i=0,1}^{2^n} c_{b_1 b_2 \dots b_n} |b_1 b_2 \dots b_n\rangle \ , \end{aligned} \tag{3.1-3}$$

where the sum of these squared probability amplitudes is also required to be one.

doi:10.6342/NTU202400695

3.2 Quantum gate

Quantum gates are mathematical operators that manipulate the qubits to perform specific operations. Some common quantum gates and their matrix forms are shown below,

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \; Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \; Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \; H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \qquad (3.2\text{-}1)$$

$$\mathrm{CZ} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}, \qquad \mathrm{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \qquad (3.2\text{-}2)$$

X, *Y* and *Z* are Pauli gates which is also usually noted as σ_x, σ_y and σ_z. *H* is Hadamard gate which plays an important role in making a superposition state. CZ and CNOT are controlled gates that act on two qubits simultaneously to generate critical entangled states. These gates are vital part of universal quantum gate sets to which any operation possible on a quantum computer can be reduced.

3.3 Quantum control

3.3.1 Hamiltonian and propagator

Quantum control involves using external fields such as electromagnetic pulses or laser beams to steer qubits. It is often described in terms of Hamiltonian $\mathcal{H}(t)$, which represents system dynamic over time and how it responds to the external control fields.

doi:10.6342/NTU202400695

It can be further decomposed into ideal and noisy parts,

$$\mathcal{H}(t) = \mathcal{H}_I(t) + \mathcal{H}_N(t) \ . \tag{3.3-1}$$

If a system is ideal, then an ideal gate propagator over time t is given as,

$$U_I(t) = \mathcal{T} \exp\left(-i \int_0^t \mathcal{H}_I(t')dt'\right) , \tag{3.3-2}$$

where $\mathcal{T}$ is the time-ordering operator, and we set $\hbar = 1$ throughout this thesis. However, we get company from noise in reality, a total propagator should be,

$$U(t) = U_I(t)\, \mathcal{T} \exp\left(-i \int_0^t \widetilde{\mathcal{H}}_N(t')dt'\right) , \tag{3.3-3}$$

where $\widetilde{\mathcal{H}}_N(t') = U_I^\dagger(t)\mathcal{H}_N(t)U_I(t)$ is noisy Hamiltonian in the interaction picture transformed by $U_I(t)$. With a careful choice of control parameters by optimization or machine learning, we can make a propagator to be any desire quantum gate we want.

3.3.2 Rotating wave approximation

The rotating wave approximation (RWA) is a common approximation used in quantum optics. When applied electromagnetic field is near resonance with a qubit transition frequency and its intensity is low, it neglects terms in Hamiltonian that oscillate rapidly.

For simplicity, considering a two-level qubit with intrinsic transition frequency ω_0, its Hamiltonian is,

doi:10.6342/NTU202400695

$$\mathcal{H}_0 = \frac{\omega_0}{2} Z \ . \tag{3.3-4}$$

We drive the qubit with an external classical electric field $\vec{E} = \vec{E}_0 e^{-i\omega_d t} + \vec{E}_0^* e^{i\omega_d t}$ with a driving frequency ω_d and $*$ denoting the complex conjugate. Under the dipole approximation, the driving Hamiltonian is,

$$\mathcal{H}_d = -\vec{d} \cdot \vec{E} \ , \tag{3.3-5}$$

where $\vec{d}$ is the dipole moment of the qubit. We assume that the qubit does not have a dipole moment when it is in an eigenstate such that,

$$\vec{d} = \vec{d}_0 \sigma_+ + \vec{d}_0^* \sigma_- \ , \tag{3.3-6}$$

where the $\sigma_\pm = X \pm iY$. The driving Hamiltonian is then,

$$\mathcal{H}_d = -\left(\Omega e^{-i\omega_d t} + \tilde{\Omega} e^{i\omega_d t}\right)\sigma_+ - \left(\tilde{\Omega}^* e^{-i\omega_d t} + \Omega^* e^{i\omega_d t}\right)\sigma_- \ , \tag{3.3-7}$$

where $\Omega = \vec{d}_0 \cdot \vec{E}_0$ is the Rabi frequency and $\tilde{\Omega} = \vec{d}_0 \cdot \vec{E}_0^*$ is counter-rotating frequency. We consider the unitary transformation to the interaction picture,

$$\mathcal{H}_{d.I} = U_0 \mathcal{H}_d U_0^\dagger \ , \tag{3.3-8}$$

where $U_0 = e^{-i\mathcal{H}_0 t}$ and $\dagger$ is the conjugate transpose of matrix. With $U_0 \sigma_\pm U_0^\dagger = e^{\pm i\omega_0 t}\sigma_\pm$, Eq. (3.3-8) can be further deduced to,

$$\mathcal{H}_{d.I} = -\left(\Omega e^{-i\Delta t} + \tilde{\Omega} e^{i(\omega_0+\omega_d)t}\right)\sigma_+ - \left(\tilde{\Omega}^* e^{-i(\omega_0+\omega_d)t} + \Omega^* e^{i\Delta t}\right)\sigma_- \ , \tag{3.3-9}$$

where $\Delta = \omega_d - \omega_0$. This is the point where we apply the RWA. If the driving frequency of electric field is near resonance with the qubit frequency, i.e., $\Delta \ll \omega_0 + \omega_d \approx 2\omega_0$, those complex exponentials multiplying counter-rotating frequency is considered to be

doi:10.6342/NTU202400695

rapidly oscillating. Given an appreciate time scale, these oscillations average to 0 such that Eq. (3.3-9) reduces to,

$$\mathcal{H}_{d.I}^{RWA} = -\Omega e^{-i\Delta t}\sigma_+ - \Omega^* e^{i\Delta t}\sigma_- \ . \tag{3.3-10}$$

The approximation error is about $\Omega^2 T/\omega_0$, where T is the gate time [11]. We transforming Eq. (3.3-10) back to the Schrödinger picture, while $\mathcal{H}_0$ remains unaffected by the RWA. The total Hamiltonian under the RWA is,

$$\mathcal{H}^{RWA} = \frac{\omega_0}{2}Z - \Omega e^{-i\omega_d t}\sigma_+ - \Omega^* e^{i\omega_d t}\sigma_- \ . \tag{3.3-11}$$

3.3.3 Effective control Hamiltonian

In the numerical simulation, we usually favor each coefficient in Hamiltonian to be comparable. However, ω_0 is usually larger than Ω and Δ. We transform Eq. (3.3-11) into the driving frame with $U_d = e^{-i\omega_d tZ/2}$ so that it becomes,

$$\mathcal{H}_{1q} = U_d \mathcal{H}^{RWA} U_d^\dagger + i\dot{U}_d U_d^\dagger = \frac{\Delta}{2}Z - \Omega\sigma_+ - \Omega^*\sigma_- \ . \tag{3.3-12}$$

If we further assume a real driving such that $\Omega = \Omega^*$, Eq. (3.3-12) is simplified to,

$$\mathcal{H}_{1q} = \frac{\Delta}{2}Z - \Omega X \ , \tag{3.3-13}$$

which is the commonly used effective single-qubit control Hamiltonian. If the driving is on-resonant ($\Delta = 0$), it can generate X gate whose gate time is determined by the Ω, therefore the strength of the external field. In contrast, if the driving is off-resonant ($\Delta \neq$

doi:10.6342/NTU202400695

0), it can generate H gate by carefully designing the Δ and Ω.

For a two-qubit control Hamiltonian, we consider the Ising model that two qubits are coupled by the ZZ interaction,

$$\mathcal{H}_0 = \frac{\omega_{0,1}}{2} Z_1 + \frac{\omega_{0,2}}{2} Z_2 + \frac{J}{2} Z_1 Z_2 \ , \tag{3.3-14}$$

where $\omega_{0,i}$ is an intrinsic qubit frequency and J is coupling strength between qubits, subscript 1 and 2 denote the indices of qubits. The ZZ interaction is the natural generator of $CNOT$ gate up to one-qubit operations [12], which can be realized in many quantum systems such as Transmons [13], flux qubits [14], trapped ions [15][16] and neutral atoms [17]. Similarly, we can drive these two qubits independently with Eq. (3.3-7) such that the total Hamiltonian is,

$$\mathcal{H} = \frac{\omega_{0,1}}{2} Z_1 + \mathcal{H}_{d,1}\left(\Omega_1, \omega_{d,1}\right) + \frac{\omega_{0,2}}{2} Z_2 + \mathcal{H}_{d,1}(\Omega_2, \omega_{d,2}) + \frac{J}{2} Z_1 Z_2 \ . \tag{3.3-15}$$

where Ω_i and $\omega_{d,i}$ are Rabi and driving frequencies of the i-th qubit. Since Pauli operators of different qubits are commute, we can apply the same RWA procedure in Sec. 3.3.2 and driving-frame transformation in the previous single-qubit case. Besides, we assume both qubits have identical qubit and driving frequencies ($\omega_{0,1} = \omega_{0,2}$ and $\omega_{d,1} = \omega_{d,2}$) such that they have the same detuning Δ, obtaining our effective two-qubit control Hamiltonian with comparable coefficients,

$$\mathcal{H}_{2q} = \frac{\Delta}{2} Z_1 - \Omega_1 X_1 + \frac{\Delta}{2} Z_2 + -\Omega_2 X_2 + \frac{J}{2} Z_1 Z_2 \ . \tag{3.3-16}$$

doi:10.6342/NTU202400695

3.3.4 Piecewise constant control

Since the RL requires an environment to possess an episodic characteristic, we choose PWC control pulse to build quantum gates. In the PWC control protocol, the maximum gate time T is predefined. Entire pulses are divided into N step pulses with equal step time $\delta t = T/N$. During i-th time step in time interval $(i-1)\delta t < t < i\delta t$ with $i \in \mathbb{N}$ and $i \leq N$, the control parameter vector p_i is constant such that $\mathcal{H}(p_i)$ is time-independent, then the i-th unitary propagator is,

$$U_i(p_i) = \exp(-i\mathcal{H}(p_i)\delta t) \ . \tag{3.3-17}$$

The total propagator up to time $i\delta t$ can be determined iteratively,

$$U(i\delta t) = U_i(p_i)U\big((i-1)\delta t\big) \ , \tag{3.3-18}$$

with the initial propagator is identity, $U(0) = I$.

3.3.5 Exponential of Pauli vector

During a step pulse of PWC protocol in Sec. 3.3.4, if this constant control Hamiltonian can be represented by combination of Pauli matrices,

$$\mathcal{H} = |\vec{p}|\hat{p} \cdot \vec{\sigma} \ , \tag{3.3-19}$$

where $\vec{p} = \big(p_x, p_y, p_z\big)$ is the control vector, $|\vec{p}|$ is the norm of $\vec{p}$ so that $\hat{p} = \vec{p}/|\vec{p}|$, and $\vec{\sigma} = (X, Y, Z)$ is a Pauli vector, we can rewrite the exponential form of unitary

doi:10.6342/NTU202400695

propagator in Eq. (3.3-17) with sine and cosine functions,

$$U(\vec{p}) = I\cos(|\vec{p}|\delta t) - i(\hat{p}\cdot\vec{\sigma})\sin(|\vec{p}|\delta t)\ , \tag{3.3-20}$$

which avoids the computational burden of matrix exponential in numerical simulation.

3.3.6 Dynamic decoupling

The dynamic decoupling is a quantum control technique based on the concept of Hahn spin echo [18], employed in quantum control to suppress decoherence with sequences of carefully timed and tuned control pulses. The pulses often implement periodic flip-flop operations at intervals shorter than characteristic timescale of environmental noise to continuously refresh and protect quantum information [19].

3.4 Gate infidelity

3.4.1 Definition of infidelity

To have an idea how good our quantum gates are, we need a metric to determine a gate infidelity. Given a target gate U_T, final gate time t_f and number of qubits n, the gate infidelity is defined as,

$$\mathbb{I} = 1 - \frac{1}{4^n}\left|\mathrm{Tr}\left[U_T^{\dagger}U(t_f)\right]\right|^2\ , \tag{3.4-1}$$

doi:10.6342/NTU202400695

where Tr is the trace operator.

3.4.2 Dyson expansion

Referring to the analysis of noise contribution in infidelity in [19], if the noise strength is not too strong, we can expand the propagator $U(t_f)$ in terms of $\widetilde{\mathcal{H}}_N(t)$ in Eq. (3.3-3) by Dyson series [21],

$$U(t_f) = U_I(t_f)\left[I + \sum_{j=1}^{\infty} \Psi_j\right], \tag{3.4-2}$$

where Ψ_j is,

$$\Psi_j = (-i)^j \int_0^{t_f} dt_1 \int_0^{t_1} dt_2 \dots \int_0^{t_{j-1}} \widetilde{\mathcal{H}}_N(t_1)\widetilde{\mathcal{H}}_N(t_2) \dots \widetilde{\mathcal{H}}_N(t_j) dt_j \ . \tag{3.4-3}$$

We substitute the expanded propagator back into the infidelity in Eq. (3.4-1),

$$\begin{aligned} \mathbb{I} = J_0 - \frac{2}{4^n} \mathrm{Re}\left\{\mathrm{Tr}\left[U_T^\dagger U_I(t_f)\right]\right\}^\star \mathrm{Tr}\left[U_T^\dagger U_I(t_f) \cdot \sum_{j=1}^{\infty} \Psi_j\right] \\ - \frac{1}{4^n}\left|\mathrm{Tr}\left[U_T^\dagger U_I(t_f) \cdot \sum_{j=1}^{\infty} \Psi_j\right]\right|^2 , \end{aligned} \tag{3.4-4}$$

where Re denotes taking the real part of the quantity. The first term J_0 is exactly the gate infidelity for the ideal system,

$$J_0 = 1 - \frac{1}{4^n}\left|\mathrm{Tr}\left[U_T^\dagger U_I(t_f)\right]\right|^2 \ . \tag{3.4-5}$$

We then introduce an error shift matrix U_ϵ of the ideal gate propagator $U_I(t_f)$ relative from the target gate propagator U_T up to a global phase ϕ,

$$U_I(t_f) = e^{i\phi} U_T (I + U_\epsilon) \ . \tag{3.4-6}$$

doi:10.6342/NTU202400695

We substitute this back into the infidelity in Eq. (3.3-17),

$$\mathbb{I} = J_0 + \sum_{k=1}^{\infty} J_k + \epsilon \ , \tag{3.4-7}$$

where those J_k are noisy terms without containing U_ϵ with k defined as noise orders, and ϵ instead contains U_ϵ,

$$\begin{aligned}\epsilon = &-\frac{1}{2^{n-1}}\mathrm{Re}\left\{\mathrm{Tr}\left[U_\epsilon \sum_{j=1}^{\infty}\Psi_j\right]\right\} - \frac{2}{4^n}\mathrm{Re}\left\{\mathrm{Tr}[U_\epsilon]^\star \mathrm{Tr}\left[\sum_{j=1}^{\infty}\Psi_j\right]\right\} \\ &-\frac{2}{4^n}\mathrm{Re}\left\{\mathrm{Tr}\left[\sum_{j=1}^{\infty}\Psi_j\right]^\star \mathrm{Tr}\left[U_\epsilon \sum_{j=1}^{\infty}\Psi_j\right]\right\} \\ &-\frac{2}{4^n}\mathrm{Re}\left\{\mathrm{Tr}[U_\epsilon]^\star \mathrm{Tr}\left[U_\epsilon \sum_{j=1}^{\infty}\Psi_j\right]\right\} - \frac{1}{4^n}\left|\mathrm{Tr}\left[U_\epsilon \sum_{j=1}^{\infty}\Psi_j\right]\right|^2 .\end{aligned} \tag{3.4-8}$$

When J_0 is small, the matrix elements of U_ϵ are also small. Meanwhile, if noise strength is not too strong such that $|\Psi_{k+1}| \ll |\Psi_k|$, we can neglect the effect of ϵ. We look into the first four noise order terms of J_k,

$$J_1 = -\frac{1}{2^{n-1}}\mathrm{Re}[\mathrm{Tr}(\Psi_1)] \ , \tag{3.4-9}$$

$$J_2 = -\frac{1}{2^{n-1}}\mathrm{Re}[\mathrm{Tr}(\Psi_2)] - \frac{1}{4^n}|\mathrm{Tr}(\Psi_1)|^2 \ , \tag{3.4-10}$$

$$J_3 = -\frac{1}{2^{n-1}}\mathrm{Re}[\mathrm{Tr}(\Psi_3)] - \frac{2}{4^n}\mathrm{Re}\{\mathrm{Tr}(\Psi_1)\mathrm{Tr}(\Psi_2)^\star\} \ , \tag{3.4-11}$$

$$J_4 = -\frac{1}{2^{n-1}}\mathrm{Re}[\mathrm{Tr}(\Psi_4)] - \frac{1}{4^n}|\mathrm{Tr}(\Psi_2)|^2 - \frac{2}{4^n}\mathrm{Re}\{\mathrm{Tr}(\Psi_1)\mathrm{Tr}(\Psi_3)^\star\}. \tag{3.4-12}$$

According to Eq. (3.4-3), the noisy Hamiltonian $\widetilde{\mathcal{H}}_N(t)$ is a Hermitian operator, so the real part of traced Ψ_j with odd j are zero such that these odd noise order terms of J_k such as J_1 and J_3 are all vanish. Since noise is stochastic, we use ensemble average of

doi:10.6342/NTU202400695

infidelities,

$$\langle \mathbb{I} \rangle = J_0 + \left\langle \sum_{k=1}^{\infty} J_{2k} \right\rangle + \langle \epsilon \rangle \ . \tag{3.4-13}$$

Though we are not going to explicitly use the expanded infidelity as our loss function of machine learning, the expansion gives us insights of which noise order terms dominate (see the analysis in Sec. 3.6).

3.5 Quasistatic noise model

For an ideal quantum system, the ensemble infidelity $\langle \mathbb{I} \rangle$ is simply the ideal infidelity J_0. But for the noisy one, practically, we need to test our pulses several times in the noisy environment and then average these result infidelities. To test our RL approach, we here choose the quasistatic Z-noise (QSN) model which the noise is constant during the gate time of each running, exerted on the Z component in Hamiltonian. Its noise strength is sampled from the Gaussian distribution with a zero mean and a noise standard deviation σ_N.

doi:10.6342/NTU202400695

3.6 Gate infidelity estimation

3.6.1 Noise contribution

With the gate infidelity expansion introduced in Sec. 3.4.2, we can estimate noise contribution of each order in a gate infidelity. Taking a single qubit gate in quasistatic Z-noise model (Sec. 3.5) as an example, the noisy Hamiltonian in the interaction picture $\widetilde{\mathcal{H}}_N(t')$ is $U_I^\dagger(t)[\beta \cdot \omega_0 \cdot Z/2]U_I(t)$, where β is the sampled noise amplitude from the Gaussian distribution. We substitute it into the Ψ_k in Eq. (3.4-3),

$$\Psi_j = (-i)^j \int_0^{t_f} \omega_0 dt_1 \int_0^{t_1} \omega_0 dt_2 \dots \int_0^{t_{j-1}} \omega_0 dt_j \times \beta^j R_Z(t_1)R_Z(t_2)\dots R_Z(t_N) \tag{3.6-1}$$

where $R_Z(t) = U_I^\dagger(t)[Z/2]U_I(t)$. Since matrix element magnitudes of Z and $U_I(t)$ are all smaller than one, matrix element magnitudes of $R_Z(t_1)R_Z(t_2)\dots R_Z(t_N)$ are also smaller than one. The time integral $\int_0^{t_f} \omega_0 dt_1 \int_0^{t_1} \omega_0 dt_2 \dots \int_0^{t_{j-1}} \omega_0 dt_j$ can be estimated as $(\omega_0 t_f)^j/j!$ and the β^j can be estimated as the noise standard deviation σ_N^j. The overall estimation of Ψ_j is $(\omega_0 t_f \sigma_N)^j/j!$. Substituting the estimated Ψ_j into each noise order J_k with even k such as Eq. (3.4-10) and Eq. (3.4-12), we get,

$$J_2 \sim (\omega_0 t_f \sigma_N)^2 \cdot \frac{1}{2}\ , \qquad J_4 \sim (\omega_0 t_f \sigma_N)^4 \cdot \frac{5}{48}\ . \tag{3.6-2}$$

Once we construct a control pulse, we can test them in a certain noise model with a series of σ_N. From slopes between noise standard deviations and corresponding ensemble gate

doi:10.6342/NTU202400695

infidelities, we can tell which noise order dominate an ensemble gate infidelity $\langle \mathbb{I} \rangle$. Given $\sigma_N < 1$, the higher the noise order dominates, the more robust a quantum gate is, because the exponent on the noise standard deviation is greater.

3.6.2 Control deviation

The noise contribution estimation (Sec. 3.6.1) can also reflect a response of control deviation. The control deviation can be thought as intentionally sampling a certain noise strength β into a perfect control pulse. Not considering a noise-robust quantum gate, we know that J_2 usually dominates such that a gate infidelity $\langle \mathbb{I} \rangle$ will be around the order of β^2 according to Eq. (3.6-2).

doi:10.6342/NTU202400695

Chapter 4 Integration

4.1 Framework mapping

We are going to map the quantum control problem into the RL framework. There are several options which can be designed as agent's observations o such as control pulse vector, Hamiltonian, or unitary propagator. Though these options have same amount of information, we choose pulse vector as observations because the last two terms need to access information of quantum states which is unavailable in real quantum devices. Instead of only one previous step pulse vector, we include an entire sequence of N-step PWC pulse vectors as observations,

$$q_i = (p_1, p_2, \dots, p_i, 0, \dots, 0), \tag{4.1-1}$$

where each pulse vector p_i has M control dimensions. It makes this RL task to hold the merit of Markov decision process property, avoiding the situation that an agent may confuse whether an observation of a previous step comes from the same pulse history or not. The agent's action a is then intuitively the next step pulse vector p_{i+1}. To specify this observation and action formulation, we take $N = 2$ for example: The agent's initial observation is a sequence of zeros $q_0 = (0, 0)$, then it gives the first step pulse vector p_1 as the action. The agent's next observation is the first step pulse vector p_1 followed with

doi:10.6342/NTU202400695

zeros, $q_1 = (p_1, 0)$, then it gives the final action p_2, leading to the final state $q_2 = (p_1, p_2)$ that is no need to be observed.

4.2 Reward design

4.2.1 Sampling-based method

To get the ensemble infidelity $\langle \mathbb{I} \rangle$ in the QSN model (Sec. 3.5), we indeed do the random sampling to calculate $\langle \mathbb{I} \rangle$ in a noise test. However, it requires large amount of pulse data collection to train the agent. We instead use the sampling-based method [22] which samples a small fixed set of noise strength to relieve the computational burden. We attempt to use a set that includes $\{0, +\sigma_N, -\sigma_N\}$. This noise strength set has a zero mean and a standard deviation $\sqrt{2/3}\sigma_N$, which is not equal to the original Gaussian noise standard deviation σ_N. We scale back this noise strength set to $\{0, +k\sigma_N, -k\sigma_N\}$ with $k = \sqrt{3/2}$ such that the training and testing noise strength set has the same mean and standard deviation.

doi:10.6342/NTU202400695

4.2.2 Weighted infidelity

The RL agent's performance really depends on the design of the reward function. Though we assume the Gaussian quasistatic noise model in Sec. 3.5, we do not know the noise standard deviation σ_N not to mention to adjust it. With an ideal infidelity redefined as $\mathbb{I}_I = J_0$, the smaller the true σ_N is, the smaller the noisy infidelity $\mathbb{I}_N = \langle \mathbb{I} \rangle - \mathbb{I}_I$ is as a proportion of the reward function such that the agent may not aware it to learn a robust quantum gate. We design an adjustable weighted infidelity,

$$\mathbb{I}_W = w_I \cdot \mathbb{I}_I + w_N \cdot \mathbb{I}_N \ . \tag{4.2-1}$$

By assigning a different ratio between the ideal weight w_I and noisy weight w_N, we can adjust the importance between ideal and noisy infidelities.

In practice of training our agent in noisy environment, an $\mathbb{I}_I$ is given by testing its control pulse in ideal environment, while a $\mathbb{I}_N$ is given by subtracting the $\mathbb{I}_I$ from the $\langle \mathbb{I} \rangle$ of sampling-based noises (Sec. 4.2.1). Since we require the agent achieve the target gate U_T right at the gate time T, the i-th step reward is design as,

$$r_i = \begin{cases} 0, & i \in \{0, 1, \dots, N-1\} \\ -\log_{10} \mathbb{I}_W, & i = N \end{cases}, \tag{4.2-2}$$

where we take the logarithm to increase the sensitivity of infidelity improvement.

doi:10.6342/NTU202400695

4.2.3 Hyperparameters γ and λ

Since our quantum gate control environment has a small number of time steps $N\sim10^1$ and the high important final reward design in Eq. (4.2-2), we assign both future discount factor γ and exponentially-weighted parameter λ introduced in Secs. 2.4 and 2.5 to be 1.

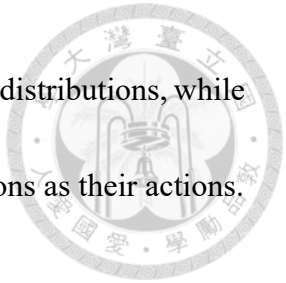
4.3 Neural network design

4.3.1 Network size

Our PPO agent possesses a policy network, a normal projection network and a value network. For the value network, its input size is the size of flatten pulse sequence $N \times M$, its number of hidden layers is 2 with their numbers of hidden neurons usually designed to be at the order of its input size, and its output size is only 1. For the policy network, its input size and hidden structure are the same as the value network, while its output size is two times the size of pulse vector $2 \times M$. These outputs are then fed into an action projection network, called normal projection network, to generate action means μ_a, action standard deviations σ_a, and action normal distributions which they are truncated by given boundaries of pulse amplitude and renormalized. In the RL training phase, the

doi:10.6342/NTU202400695

agent's actions are finally sampled from these truncated action normal distributions, while in our RL testing phase, we force agent to use the means of distributions as their actions.



4.3.2 Network initialization

According to the concept of dynamic decoupling in Sec. 3.3.6, a flip-flop pulse can mitigate noise. To better incent an agent to generate the flip-flop pulse that across the zero line, we intentionally set the initial network parameters of normal projection network to be zeros such that it gives a flat pulse right at the zero line at the beginning. To show why this design works, we take the Pauli X gate to be the example: To achieve a low ideal infidelity $\mathbb{I}_i$, it requires the area of control pulse on the X direction to be the odd multiples of π. However, due to the recursive nature of pulse generation in RL, we usually get a nearly-flat pulse after training. If we do not set the initial pulse near the zero line, the agent may generate a 3π or higher pulse area pulse that is away from the zero line. Then, if we further want to get a robust X gate which requires a flip-flop pulse shape, the pulse that is away from the zero line is harder to achieve the goal.

doi:10.6342/NTU202400695

4.4 Adaptive learning

To build a low infidelity quantum gate, the adaptive learning rate mechanism in the commonly-used optimizer, Adam [23], is not enough. We additionally adjust a learning rate lr and an action standard deviation σ_a according to a gate infidelity. In Sec. 3.6.2, we know that an infidelity is roughly square of control deviation when an ideal infidelity $\mathbb{I}_I$ is small. Hence, we automatically or manually assign a learning rate lr and an action standard deviation σ_a to roughly be a square-root of infidelity $\langle\mathbb{I}\rangle^{1/2}$ every certain number of training iterations. At the automatic mode, we will set initial lr and σ_a to also play as the upper bounds of them. That is, taking a lr for example, a result lr will be the smaller one between the initial lr and $\langle\mathbb{I}\rangle^{1/2}$.

doi:10.6342/NTU202400695

Chapter 5 Result

We here apply our method to construct three iconic quantum gates, X, H and CNOT gates. For X and H gates. we begin with a trivial one-step pulse in ideal environment to preliminarily verify the feasibility of our RL approach (Sec. 2.3), the intentional flat pulse initialization (Sec. 4.3.2) and the efficiency of automatic adaptive learning (Sec. 4.4). Then, we construct their multi-step versions in ideal and noisy environments, where we set the same max gate time T and step numbers N within each kind of gate to better compare the effect of noise learning.

We organize each test as follows. We first declare the target gate and our effective control Hamiltonian. Next, we list a setting of control configuration, network sizes and learning hyperparameters in a table. Finally, we show the results of training including agents' learning curves, control pulses as well as their noise test to recognize dominant noise orders.

5.1 X gate

We first deal with the simplest X gate to test our approach. Its matrix form is,

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} . \tag{5.1-1}$$

According to Eq. (3.3-13) with an on-resonant driving field, our effective control

doi:10.6342/NTU202400695

Hamiltonian is,

$$\mathcal{H}(t) = \frac{\pi}{2}\Omega(t)X \ , \qquad (5.1\text{-}2)$$

where $\Omega(t) \in \{p_1, p_2, \dots, p_N\}$ is in the PWC control protocol.

5.1.1 Trivial case

In the trivial case of X gate, we will test the efficiency of automatic adaptive learning which assigns a learning rate lr and an action standard deviation σ_a to be a square root of infidelity after every training iteration. First, we train a PPO agent without the automatic adaptive learning. An effective learning rate is fully determined by Adam optimizer with the initial $lr = 3 \times 10^{-4}$ and an action standard deviation is given by the agent's normal projection network. Later, we adopt the automatic adaptive learning with the initial $lr = 1$ to speed up training. The common settings of these two are shown in Table 5.1-1, where the batch size is set to "All" means that we use the whole collected data instead of mini-batch of them in every gradient descent epoch. The comparison results are shown from Figure 5.1-1 to Figure 5.1-3, where blue and green colors indicate the learnings with and without the automatic adaptive learning respectively.

Comparing the learning curves in Figure 5.1-1, the blue one which adopts the automatic adaptive learning easily reaches the lower bound double precision (10^{-15}) at

doi:10.6342/NTU202400695

the early 22-th iteration, while the other green one wobbles around 10^{-6} and touches the 10^{-12} just by chance. Moreover in Figure 5.1-2, the control deviation of the green pulse is at the order of 10^{-6} which is the square root of its infidelity, indeed consistent with our control deviation analysis in Sec. 3.6.2. Besides, from the result that their pulse areas are both equal to 1.0 but not 3.0 or higher odd values (note that our Hamiltonian contains $\pi/2$), we realize that our network initialization in Sec. 4.3.2 that intentionally sets initial pulses to be flat around the zero truly works. Finally, when testing with the QSN model in Figure 5.1-3, these X gates are both mainly dominated by the Second order noise J_2 since the agents trained in the ideal environment are unable to aware the characteristic of noise.

Parameter	**Value**
T	1.0
N	1
Pulse boundary	[-4, 4]
Policy network size	(4, 4)
Value network size	(4, 4)
Number of collections	30
Number of epochs	10
Batch size	All
Initial lr	1 and 3×10^{-4}
Initial σ_a	3×10^{-1}

Table 5.1-1. The setting of trivial X gate.

doi:10.6342/NTU202400695

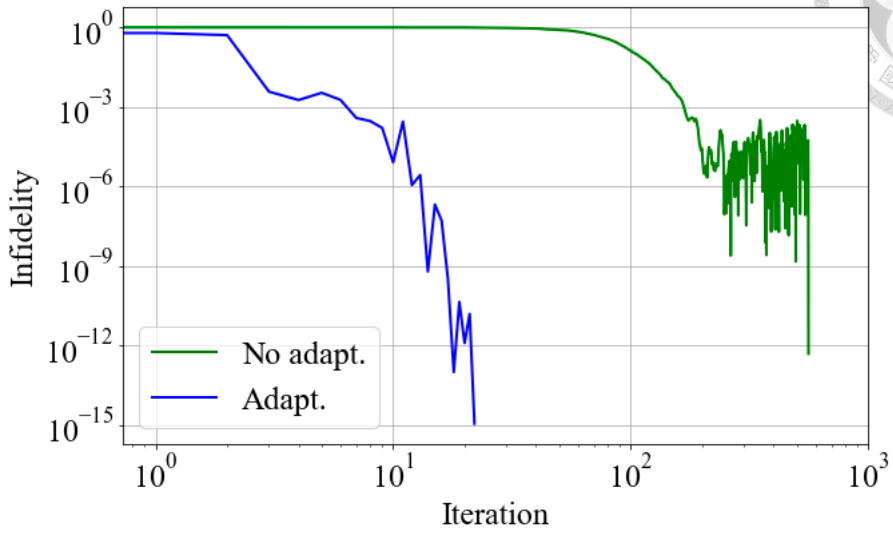


Figure 5.1-1. The learning curves of trivial X gates with and without the automatic adaptive learning.

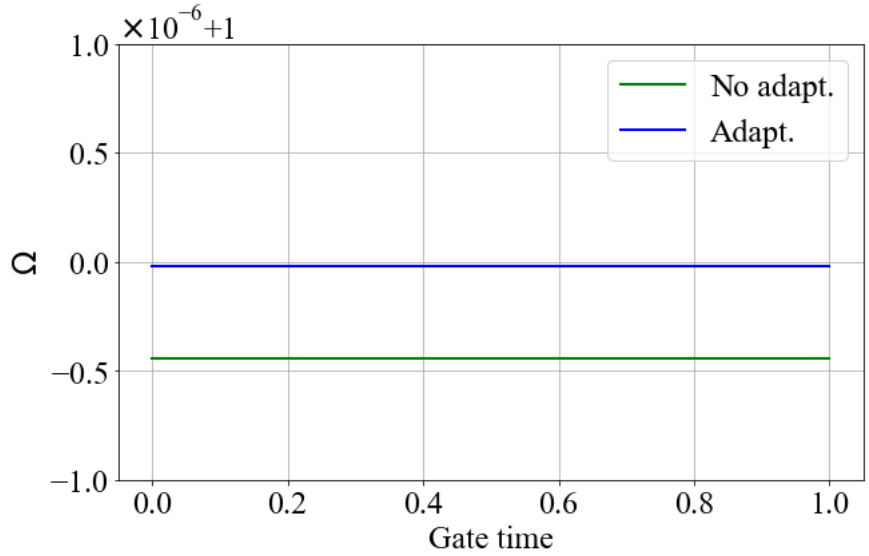


Figure 5.1-2. The one-step control pulses of trivial X gates with and without the automatic adaptive learning.

doi:10.6342/NTU202400695

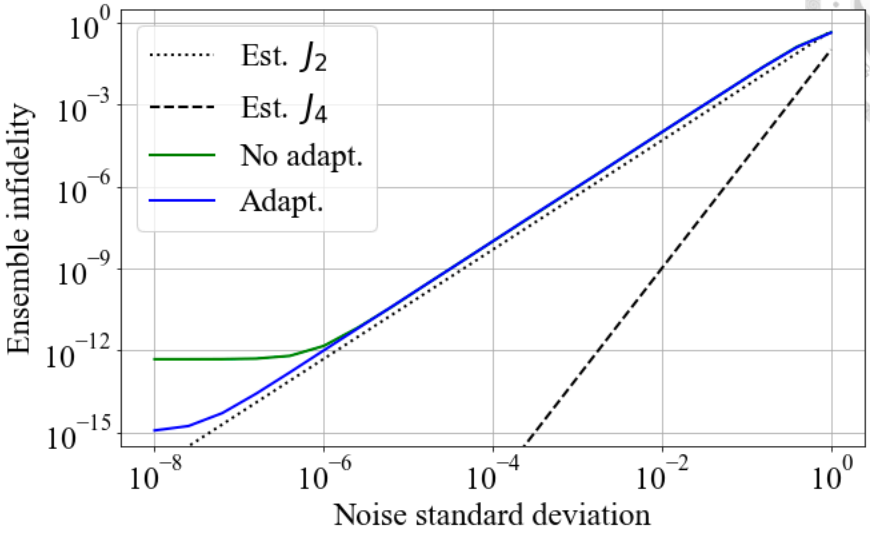
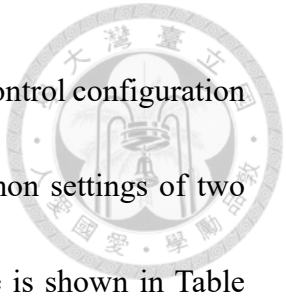


Figure 5.1-3. The ensemble infidelity versus noise standard deviation of trivial X gates with and without the automatic adaptive learning.

5.1.2 Ideal and noisy cases

To construct a robust X gate in the QSN environment with $\sigma_N = 10^{-1}$, we need a multi-step pulse to realize a flip-flop feature, so we increase step numbers N to 8. In the noisy case, we use the sampling-based method (Sec. 4.2.1) to get an effective ensemble infidelity for an agent's training. Also, we adopt the manual adaptive learning, starting with larger lr and σ_a at the first stage. At this stage, it is crucial that the weighted ideal and noisy infidelities should be comparable in magnitudes to let the agent aware of them both. We then manually decrease lr, σ_a and noisy weight w_N after every training stage once the agent learns a robust pulse with a low enough ideal infidelity, while an ideal

doi:10.6342/NTU202400695



weight w_I is always set to 1.0. We also do an ideal case in the same control configuration except for still adopting the automatic adaptive learning. The common settings of two cases are shown in Table 5.1-2, the adaptive schedule of noisy case is shown in Table 5.1-3, and the results are shown from Figure 5.1-4 to Figure 6-4.

Comparing the learning curves in Figure 5.1-4 and Figure 5.1-5, the ideal X gate adopting the automatic adaptive learning converges to 10^{-15} in short iterations, while the noisy X gate takes a lot of time, especially trying to get a robust pulse at the stage-1 learning. We terminate this learning stage only when the agent learns a pulse that possesses the 4th noise order behavior shown as the solid cyan curve in Figure 6-4. The following stages are mainly served as decreasing the ideal infidelity while maintaining the 4th noise order behavior. Finally, in Figure 5.1-6, in contrast with the flat pulse of ideal X gate, the robust pulse of noisy X gate really behaves a flip-flop shape across the zero line.

doi:10.6342/NTU202400695

Parameter	Value
T	1.0
N	8
Pulse boundary	[-4, 4]
Policy network size	(8, 8)
Value network size	(8, 8)
Number of collections	30
Number of epochs	10
Batch size	All
Initial lr	3×10^{-4}
Initial σ_a	3×10^{-1}

Table 5.1-2. The common settings of ideal and noisy X gates.

Stage	Number of iterations	w_N	Initial lr	Initial σ_a
1	8427	1	3×10^{-4}	3×10^{-1}
2	99	10^{-1}	10^{-4}	3×10^{-2}
3	16	10^{-2}	3×10^{-5}	10^{-2}
4	938	10^{-3}	10^{-7}	10^{-3}
5	29	10^{-4}	10^{-8}	10^{-3}
6	145	0	10^{-10}	10^{-6}

Table 5.1-3. The adaptive schedule of noisy X gate.

doi:10.6342/NTU202400695

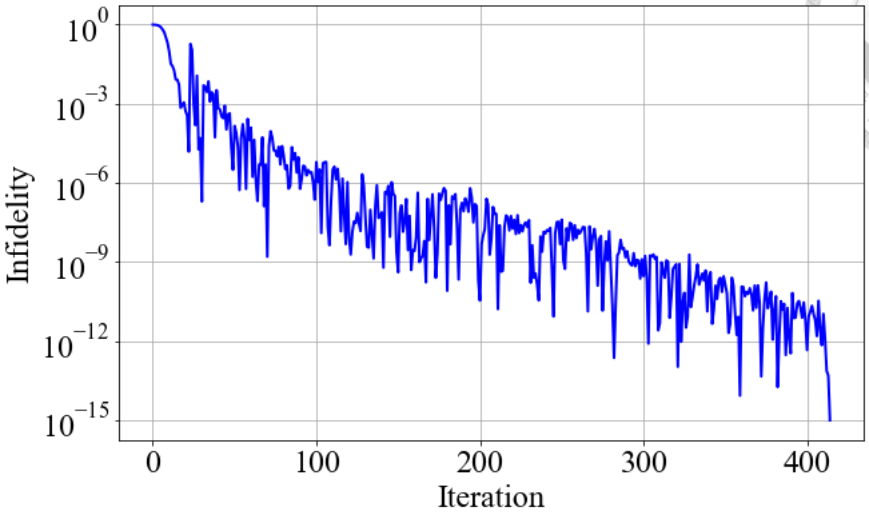


Figure 5.1-4. The learning curve of ideal X gate.

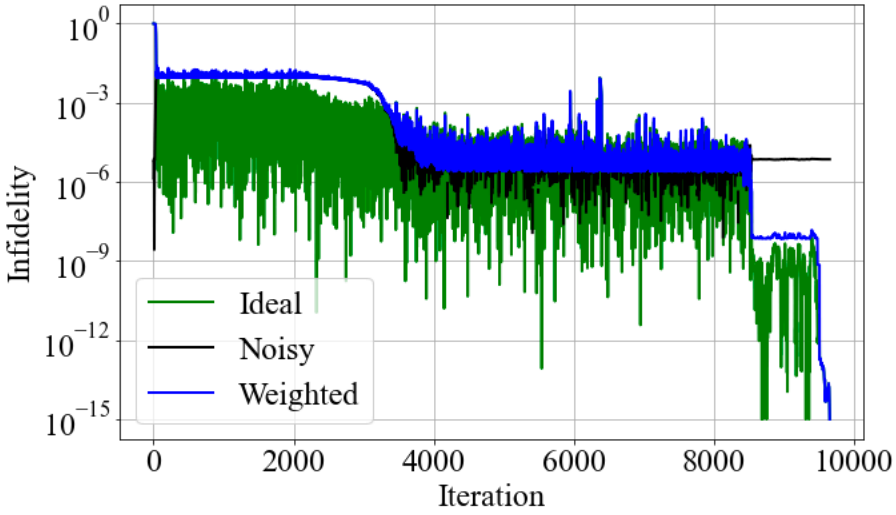


Figure 5.1-5. The learning curves of noisy X gate. These infidelities are defined in Sec. 4.2.2.

doi:10.6342/NTU202400695

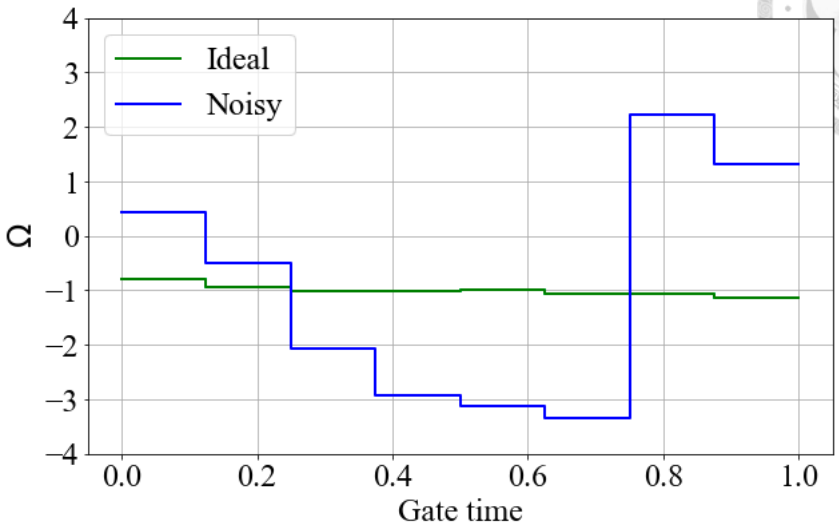


Figure 5.1-6. The control pulse of ideal and noisy X gates.

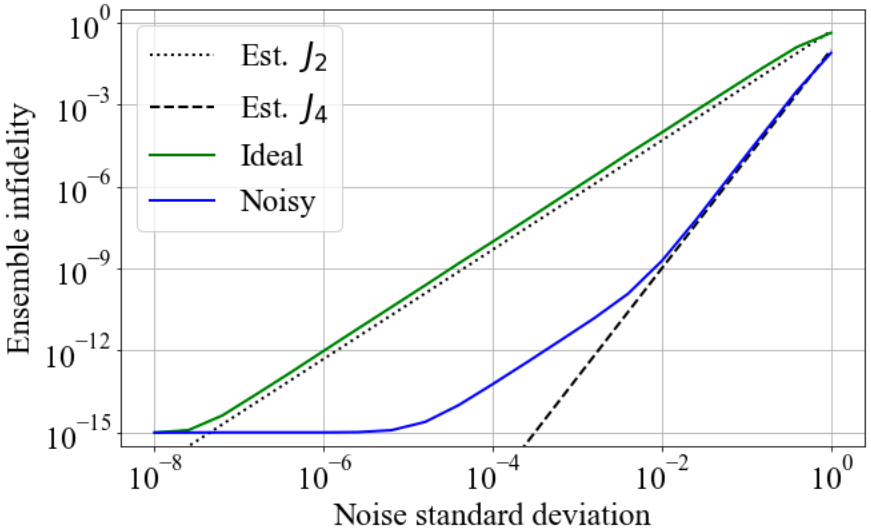


Figure 5.1-7. The ensemble infidelity versus noise standard deviation of ideal and noisy X gates.

doi:10.6342/NTU202400695

5.2 H gate

The matrix form of target H gate is,

$$H = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} . \tag{5.2-1}$$

According to an off-resonant driving field with a detuning Δ in Eq. (3.3-13), our effective control Hamiltonian is,

$$\mathcal{H}(t) = \frac{\pi}{2}(\Delta Z + \Omega(t)X) , \tag{5.2-2}$$

where $\Omega(t) \in \{p_1, p_2, \dots, p_N\}$ is in the PWC control protocol. The all settings are similar to those in the X gate learning. However, the extra Z operator restricts the lower bound of gate time and makes a control more difficult by requiring a coefficient match between the Z and X operators.

5.2.1 Trivial case

In the trivial case of H gate, we as well adopt the automatic adaptive learning and set T to be the minimal required gate time $1/\sqrt{2}$. The other settings shown in Table 5.2-1 are the same as the trivial X gate in Sec. 5.1.1. The results shown from Figure 5.2-1 to Figure 5.2-3 are also similar to trivial X gate.

doi:10.6342/NTU202400695

Parameter	Value
T	$1/\sqrt{2}$
N	1
Pulse boundary	[-4, 4]
Policy network size	(4, 4)
Value network size	(4, 4)
Number of collections	30
Number of epochs	10
Batch size	All
Initial lr	10^0
Initial σ_a	3×10^{-1}

Table 5.2-1. The setting of trivial one-step ideal H gate.

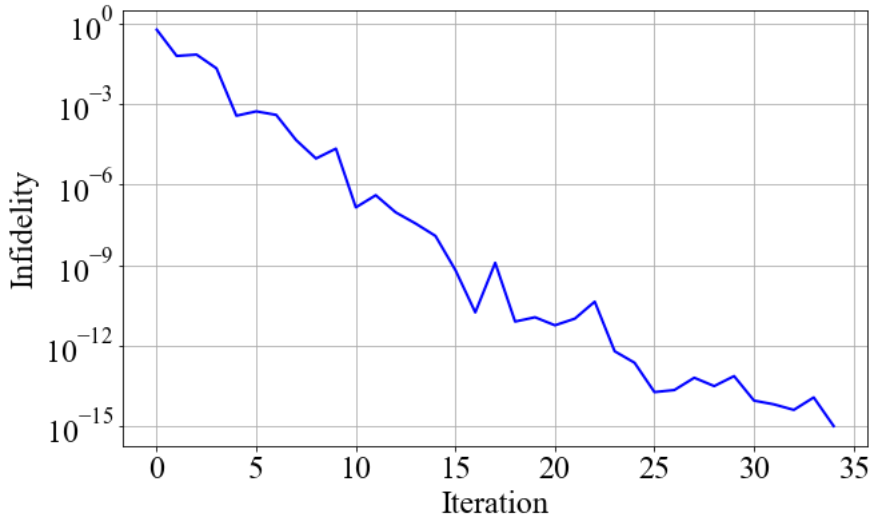


Figure 5.2-1. The learning curve of trivial H gate.

doi:10.6342/NTU202400695

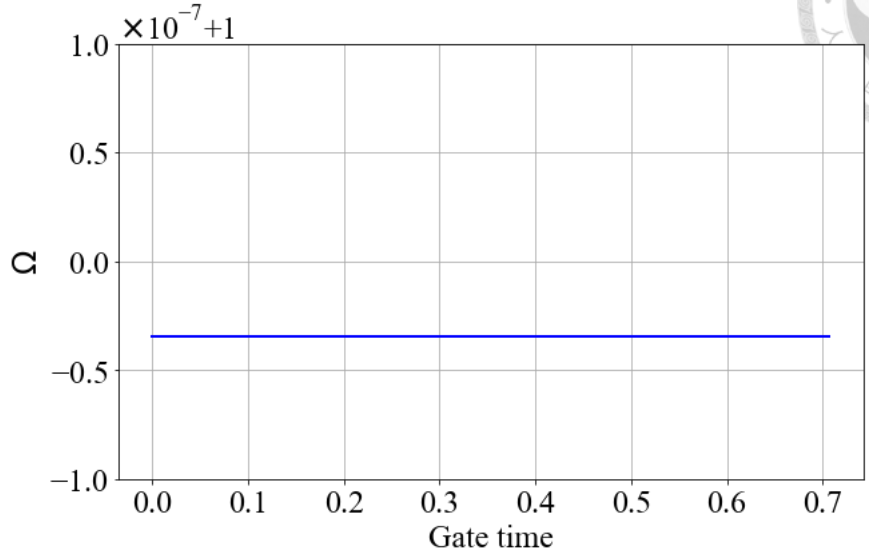


Figure 5.2-2. The control pulse of trivial H gate.

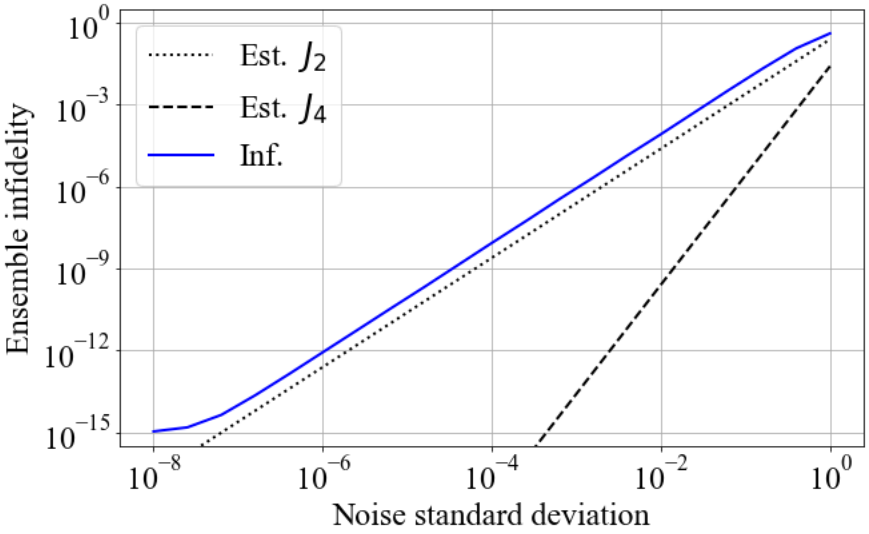
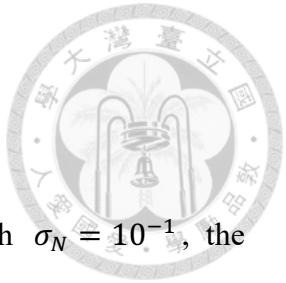


Figure 5.2-3. The ensemble infidelity versus noise standard deviation of trivial H gate.

doi:10.6342/NTU202400695



5.2.2 Ideal and noisy cases

To construct a robust H gate in the QSN environment with $\sigma_N = 10^{-1}$, the minimal gate time $1/\sqrt{2}$ in the previous trivial case is no longer enough, so we set $T = 4$. Furthermore, though a large N may make the control more complicated, we set $N = 16$ such that it provides more control degrees of freedom and effective higher pulse frequencies to conquer the noise. We also enlarge network sizes to (32, 32) to tackle the sophisticated control of the large N. Similar to the ideal and noisy X gates in Sec. 5.1.2, we adopt the automatic adaptive learning in the ideal H gate, while manually adjust lr, σ_a and w_N in the noisy one. The common settings of two cases are shown in Table 5.2-2, the adaptive schedule of noisy case is shown in Table 5.2-3, and the results are shown from Figure 5.2-4 to Figure 6-5.

Comparing the learning curves in Figure 5.2-4 and Figure 5.2-5, the ideal case adopting the automatic adaptive learning reaches 10^{-15} in short iterations. The noisy case takes most of the time at the stage-1 learning, trying to optimize the ideal and noisy infidelities simultaneously. The two pulses in Figure 5.2-6 look very different with one flat and the other wiggled. Finally, in Figure 5.2-7, though the noisy case sacrifices its ideal infidelity as 5×10^{-10}, it owns the robust 4th noise order behavior in the large σ_N region.

doi:10.6342/NTU202400695

Parameter	Value
T	1.0
N	8
Pulse boundary	[-4, 4]
Policy network size	(32, 32)
Value network size	(32, 32)
Number of collections	30
Number of epochs	10
Batch size	All
Initial lr	3×10^{-4}
Initial σ_a	3×10^{-1}

Table 5.2-2. The common settings of ideal and noisy H gates.

Stage	Number of iterations	w_N	Initial lr	Initial σ_a
1	6845	1	3×10^{-4}	3×10^{-1}
2	1309	10^{-1}	10^{-6}	5×10^{-1}
3	201	10^{-2}	10^{-6}	10^{-2}
4	45	10^{-3}	10^{-7}	10^{-2}
5	403	10^{-4}	10^{-8}	10^{-2}
6	725	0	10^{-9}	10^{-4}

Table 5.2-3. The adaptive schedule of noisy H gate.

doi:10.6342/NTU202400695

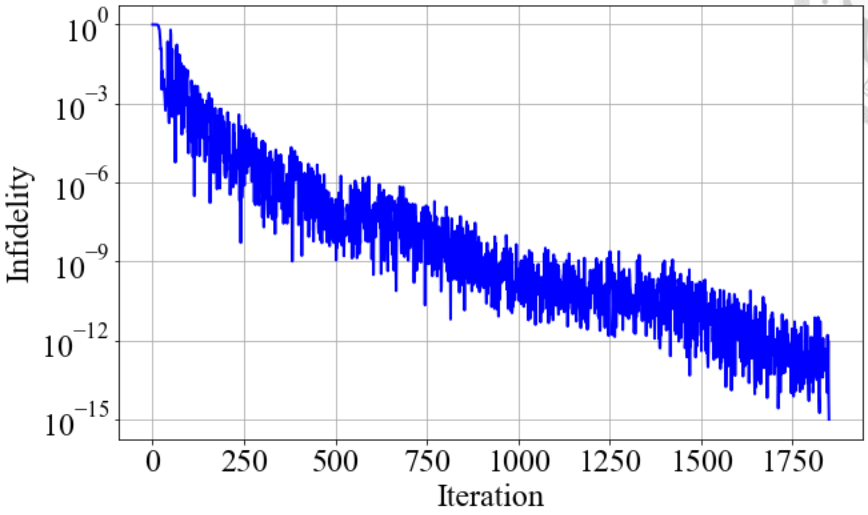


Figure 5.2-4. The learning curve of ideal H gate.

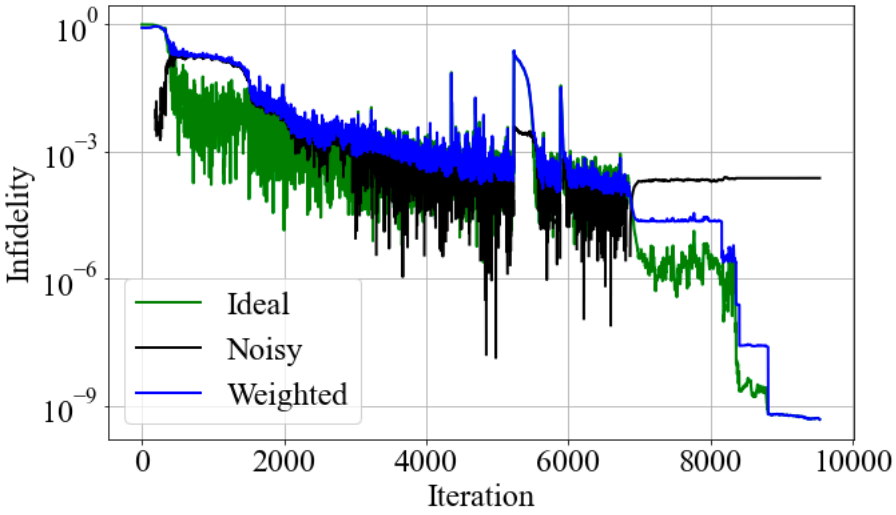


Figure 5.2-5. The learning curves of noisy H gate. These infidelities are defined in Sec. 4.2.2.

doi:10.6342/NTU202400695

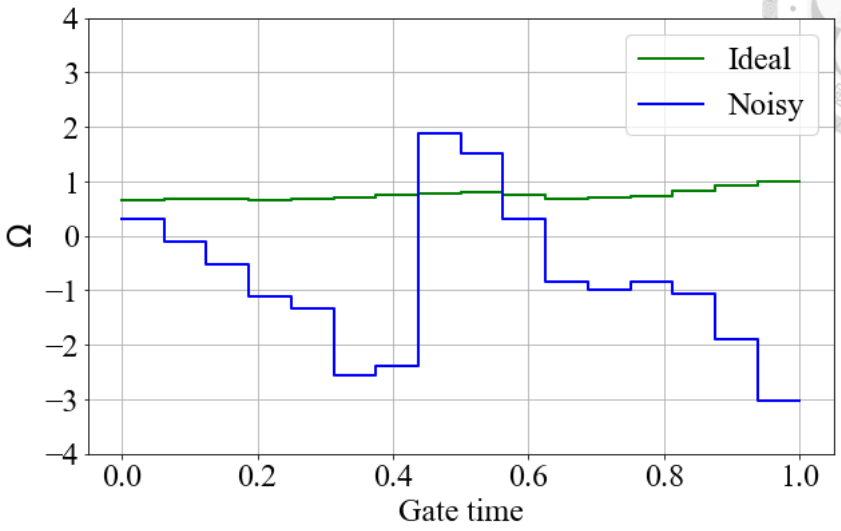


Figure 5.2-6. The control pulse of ideal and noisy H gates.

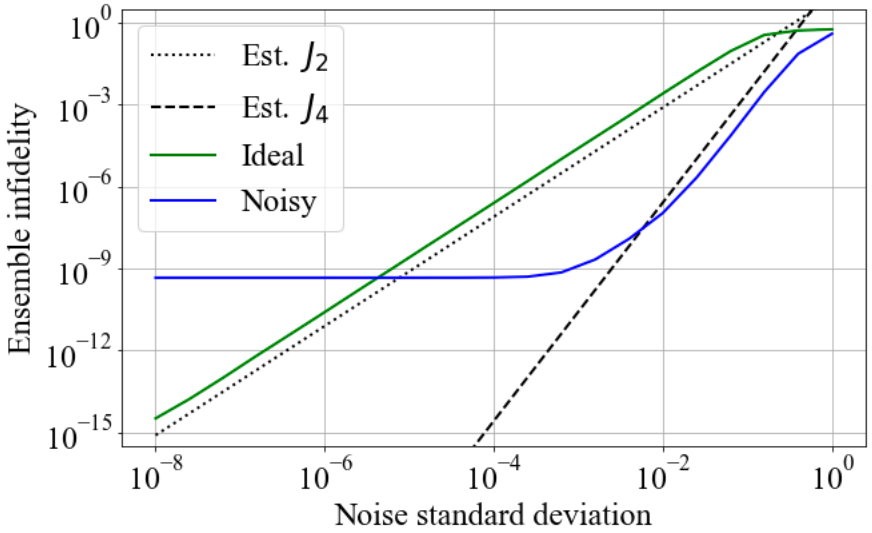


Figure 5.2-7. The ensemble infidelity versus noise standard deviation of ideal and noisy H gates.

doi:10.6342/NTU202400695

5.3 CNOT gate

The matrix form of target CNOT gate is,

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{5.3-1}$$

According to Eq. (3.3-16) with the same detuning Δ on qubits, our effective two-qubit control Hamiltonian is,

$$\mathcal{H}(t) = \frac{\pi}{2}(\Omega_1(t)X_1 + \Delta Z_1 + \Omega_2(t)X_2 + \Delta Z_2 + J(t)Z_1Z_2)\ , \tag{5.3-2}$$

where we define a $M = 3$ pulse vector $q(t) = \big(\Omega_1(t), \Omega_2(t), J(t)\big) \in \{p_1, p_2, \dots, p_N\}$ with $p_i = \big(\Omega_{1,i}, \Omega_{2,i}, J_i\big)$ in the PWC protocol. For CNOT gate, we conduct only the ideal and noisy cases with $\sigma_N = 10^{-2}$. Our Hamiltonian has two Z operators Z_1 and Z_2 where we exert the noises on both of them in the QSN model. We set $T = 4$ and $N = 16$. The network sizes are enlarged to (128, 128) since an agent's pulse sequence observation is now as large as 48 with $N = 16$ and $M = 3$. Besides, the control problems get so difficult that we give up the automatic adaptive learning in the ideal case and turn to the manual one in both cases. The common settings are shown in Table 5.3-1. The common settings of ideal and noisy CNOT gates. The adaptive schedules of ideal and noisy cases are respectively shown in Table 5.3-2 and Table 5.3-3. The comparisons of noise learning are shown from Figure 5.3-1 to Figure 5.3-4.

Comparing the learning curves in Figure 5.3-1 and Figure 5.3-2, without the merit

doi:10.6342/NTU202400695

of automatic adaptive learning, two cases both reach the ideal infidelities around 10^{-10} in long iterations, but the noisy one takes more longer to also optimize the noisy infidelity. The pulses of noisy case wiggle strongly opposite to the ideal ones in Figure 5.3-3. These robust pulses, though unable to possess the 4th noise order behavior like those in X and H gates, still reduce the two orders of magnitude in 2nd order noise from the ideal case in Figure 5.3-4.

Parameter	**Value**
T	4.0
N	16
Pulse boundary	[-4, 4]
Policy network size	(128, 128)
Value network size	(128, 128)
Number of collections	30
Number of epochs	10
Batch size	All

Table 5.3-1. The common settings of ideal and noisy CNOT gates.

doi:10.6342/NTU202400695

Stage	Number of iterations	Initial lr	Initial σ_a
1	3901	3×10^{-4}	10^{-1}
2	1941	10^{-5}	3×10^{-3}
3	10065	10^{-6}	3×10^{-4}
4	8130	10^{-9}	3×10^{-5}
5	18747	3×10^{-11}	3×10^{-6}

Table 5.3-2. The adaptive schedule of ideal CNOT gate.

Stage	Number of iterations	w_N	Initial lr	Initial σ_a
1	8597	10^2	10^{-4}	10^{-1}
2	19905	10^2	10^{-5}	10^{-2}
3	14444	10^1	10^{-6}	10^{-3}
4	407	1	3×10^{-7}	3×10^{-4}
5	17422	10^{-1}	10^{-7}	10^{-4}
6	19522	10^{-2}	10^{-7}	10^{-4}
7	1795	10^{-3}	3×10^{-10}	10^{-5}
8	3078	10^{-4}	3×10^{-10}	10^{-5}
9	9606	0	3×10^{-10}	10^{-5}
10	13995	0	3×10^{-11}	3×10^{-6}

Table 5.3-3. The adaptive schedule of noisy CNOT gate.

doi:10.6342/NTU202400695

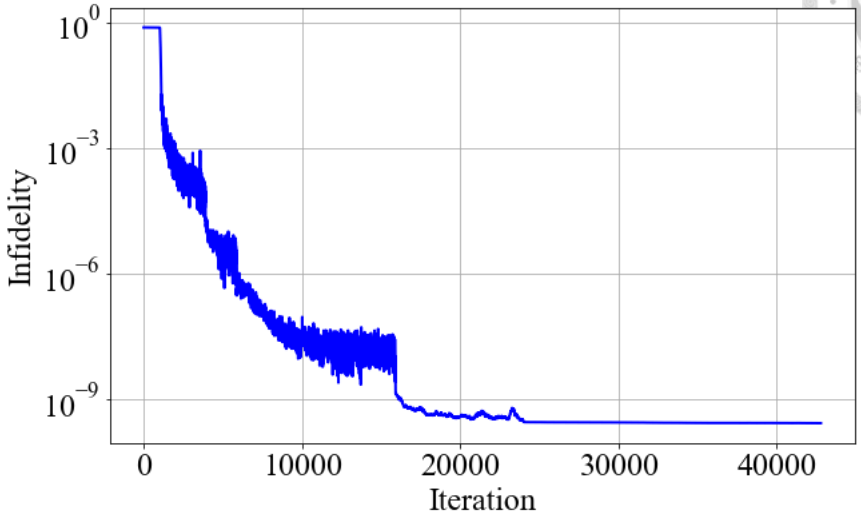


Figure 5.3-1. The learning curve of ideal CNOT gate.

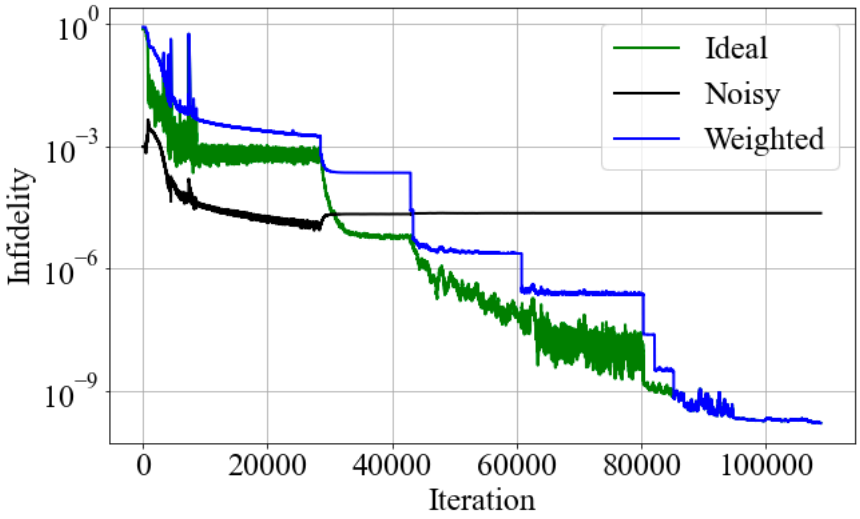


Figure 5.3-2. The learning curve of noisy CNOT gate. These infidelities are defined in Sec. 4.2.2.

doi:10.6342/NTU202400695

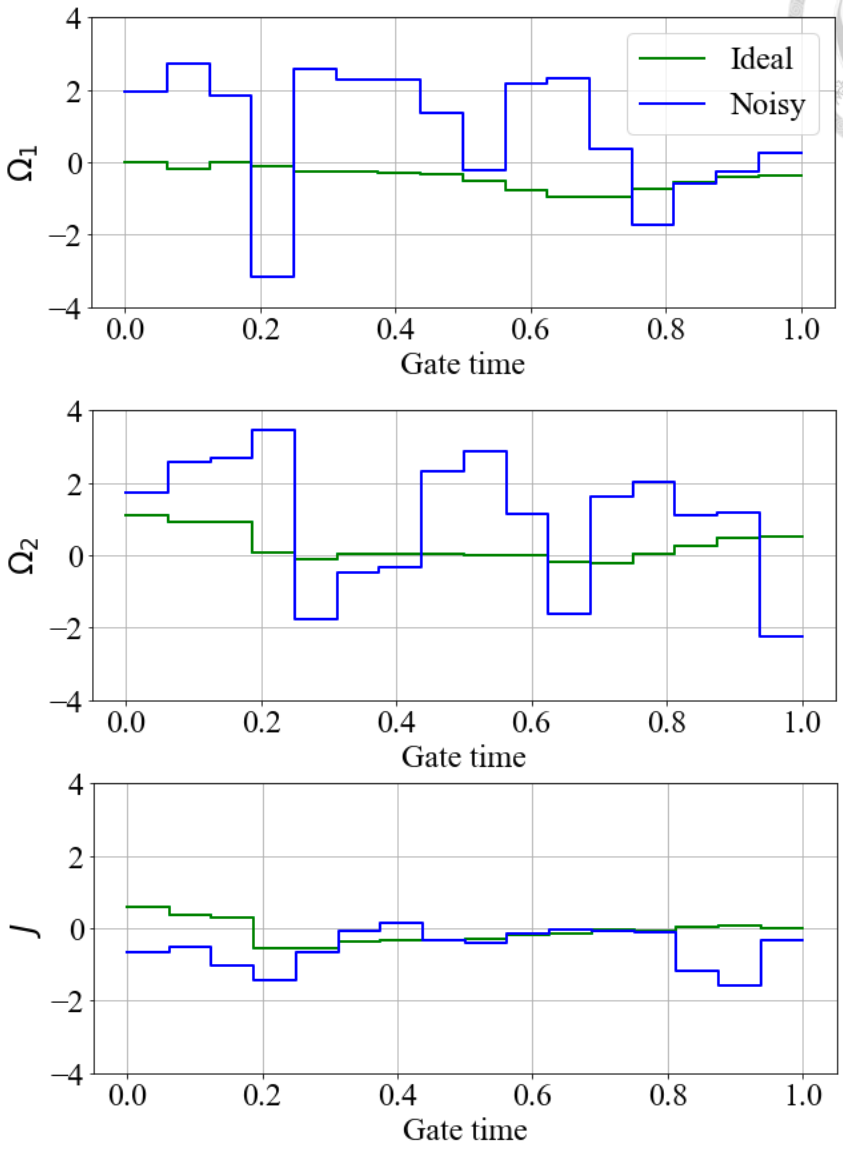


Figure 5.3-3. The control pulses of ideal and noisy CNOT gates.

doi:10.6342/NTU202400695

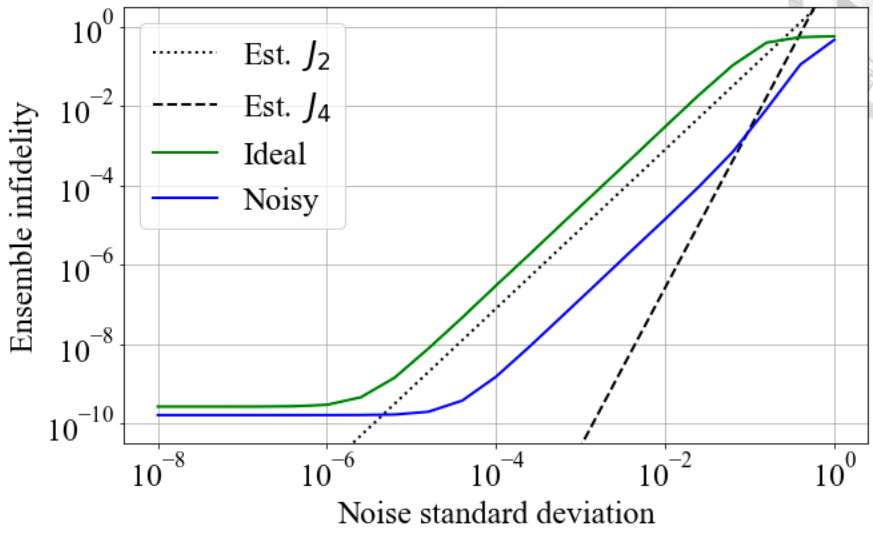


Figure 5.3-4. The ensemble infidelity versus noise standard deviation of ideal and noisy CNOT gates.

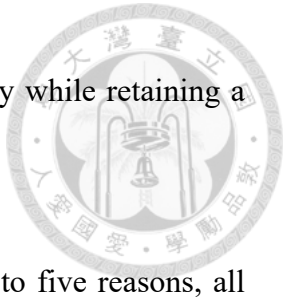
doi:10.6342/NTU202400695

Chapter 6 Discussion

In this Chapter, we discuss factors that could affect the control complexity for the PPO agents. The first factor is the number of PWC steps N. For the trivial one-step X and H gates in Secs. 5.1.1 and 5.2.1, we set a really large initial learning rate $lr = 1$ such that these learnings take only within 40 iterations. As N goes up to 8 or 16, we must decrease the initial $lr = 3 \times 10^{-4}$ to let the agents take their time to learn the relation between each step. The second factor is the Hamiltonian. For the ideal cases of the X and H gates in Secs. 5.1.2 and 5.2.2, their Hamiltonians are simple, containing only one control parameter. We are able to adopt the automatic adaptive learning which swiftly converges the learning. For the Hamiltonian of two-qubit CNOT gate in Sec. 5.3, even though the environment is ideal, the automatic adaptive learning fails. The final factor is whether the environment is ideal or noisy. For these three gates in a noisy environment, we all adopt the manual adaptive learning with a large number of learning iterations.

In the below comparisons of pulses and noise tests before and after lowering the noisy weights w_N in their adaptive schedules of X, H and CNOT gates from Figure 6-1 to Figure 6-6, we find that a pulse shape is determined at the very beginning of learning for which the weighted ideal and noisy infidelities must be comparable. Otherwise, an agent will incline to focus on learning the one with a larger weight in a reward and be stuck in a sub-optimal control minimum. The learning stages later on where

doi:10.6342/NTU202400695



we decrease noise weight w_N are meant to lower the ideal infidelity while retaining a robust behavior.

We attribute the success of constructing robust quantum gates to five reasons, all introduced in Chapter 4. First, the environment observation of pulse sequence rather than quantum states in Sec. 4.1 enables an agent to control a quantum gate without the need to know the underlying quantum dynamic. Especially in a noisy case, if a sampled QSN comes into an agent's policy network, its output changes consequently after each episode, and one could not get a unique robust pulse. Second, the sampling-based ensemble infidelity in Sec. 4.2.1 reliefs the computational burden of introducing noise information into a reward when training an agent. Third, the weighted infidelity in Sec. 4.2.2 makes the ideal and noisy infidelities play comparable roles in a reward, though not knowing an actual σ_N of the environment. Also, we can further adjust them when we want to, for example, focus on lowering the ideal infidelity. Fourth, the zero-pulse initialization in Sec. 4.3.2 makes an agent get a flip-flop pulse more possible. Finally, the adaptive learning in Sec. 4.4 serves as a strategy to converge a learning to a really low infidelity. All these techniques work together to make constructing robust quantum gates by RL possible.

doi:10.6342/NTU202400695

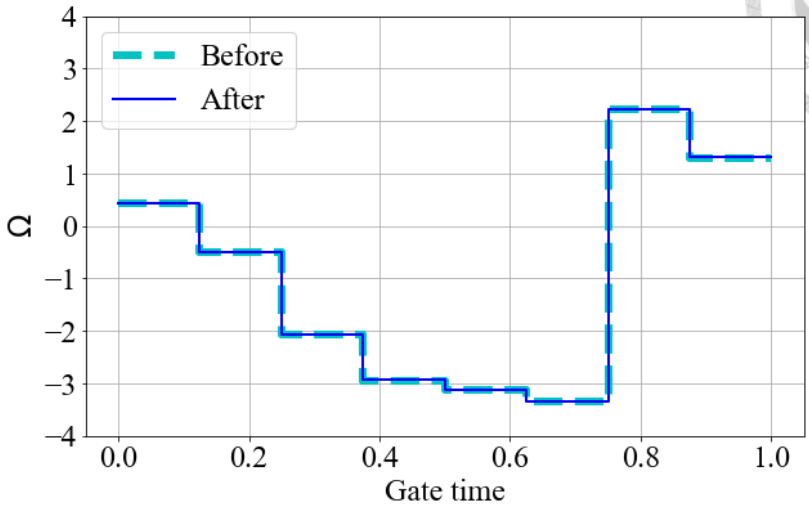


Figure 6-1. The control pulses of noisy X gate before and after lowering the noisy weight.

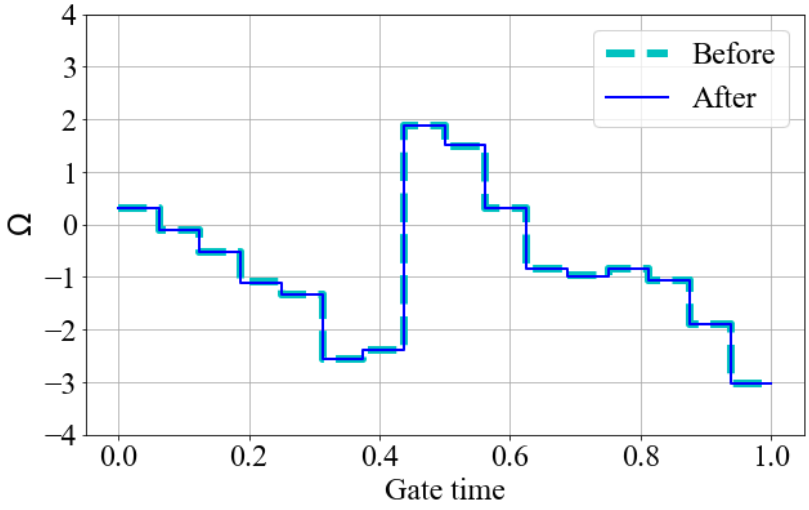


Figure 6-2. The control pulses of noisy H gate before and after lowering the noisy weight.

doi:10.6342/NTU202400695

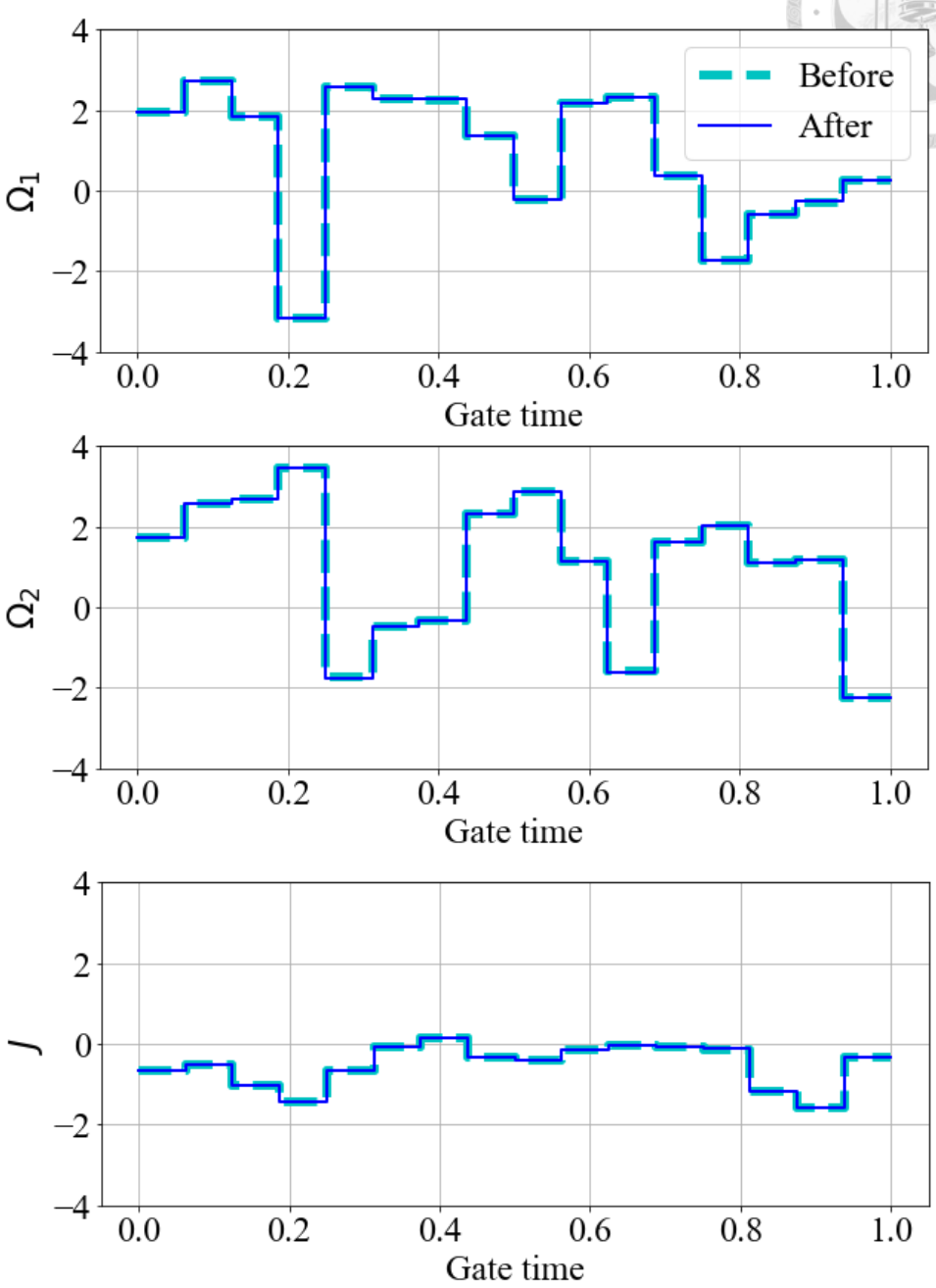


Figure 6-3. The control pulses of noisy CNOT gate before and after lowering the noisy weight.

doi:10.6342/NTU202400695

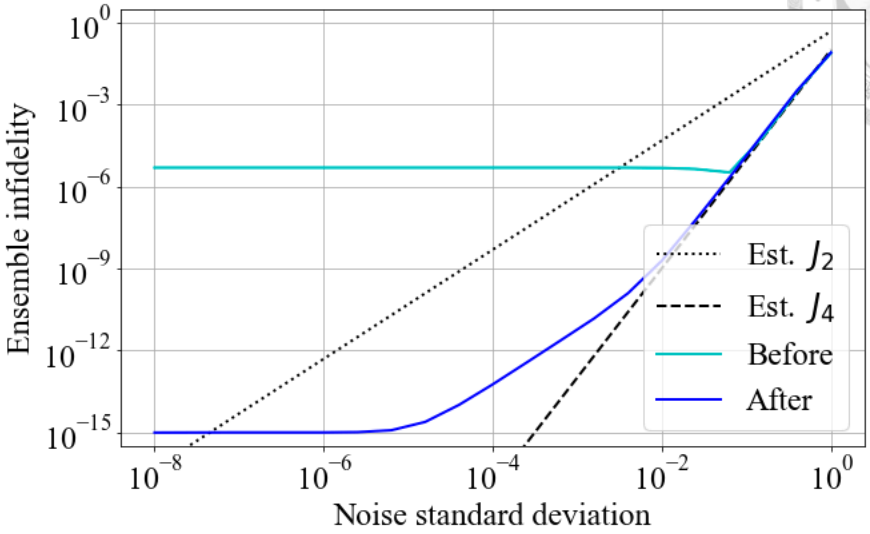


Figure 6-4. The ensemble infidelity versus noise standard deviation of noisy X gates before and after lowering the noisy weight.

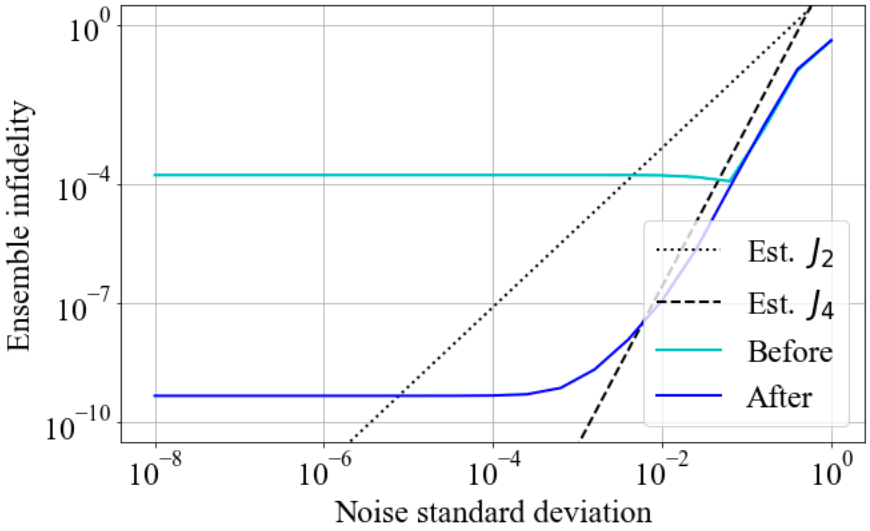


Figure 6-5. The ensemble infidelity versus noise standard deviation of noisy H gates before and after lowering the noisy weight.

doi:10.6342/NTU202400695

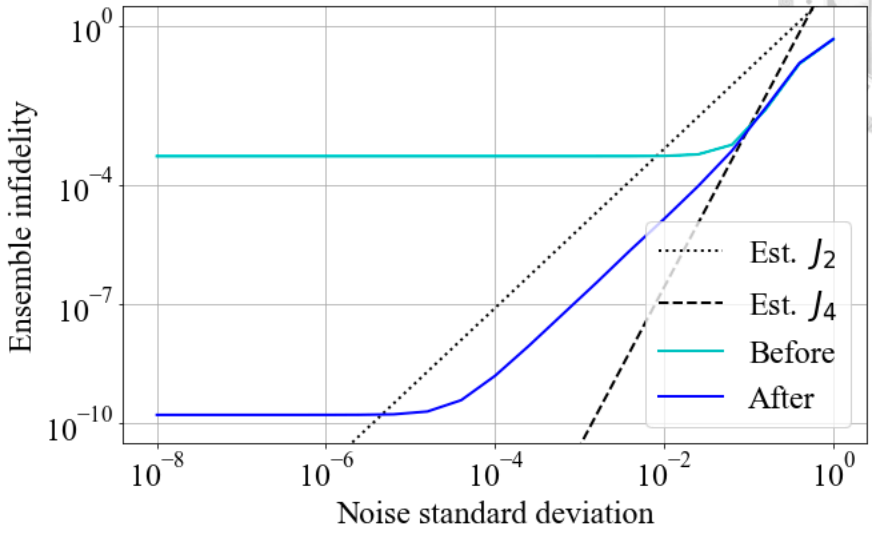


Figure 6-6. The ensemble infidelity versus noise standard deviation of noisy CNOT gates before and after lowering the noisy weight.

doi:10.6342/NTU202400695

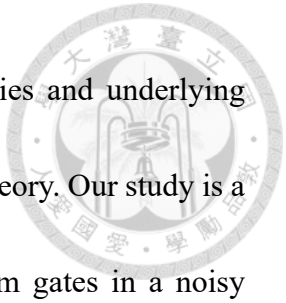
Chapter 7 Conclusion

We give a brief summary of the thesis here. We have established an DRL control of robust quantum gates in the presence of QSN. We have used the PPO algorithm with the actor-critic model for the DRL agent to learn an optimal control policy under the policy gradient framework. Our PPO DRL agent does not need to have prior knowledge of any quantum system dynamics or noise information, but only a control pulse sequence as an observation (Sec. 4.1). With the adaptive learning (Sec. 4.4), we have constructed the ideal X, H and CNOT gates with lowest ideal infidelities of 10^{-15}, 10^{-15} and 10^{-10} respectively, in contrast to those in [4][5][6], meeting the performance in [7].

As for the noise-learning, with our design of weighted infidelity (Sec. 4.2.2) and zero-pulse initialization (Sec. 4.3.2), we have constructed robust quantum gates capable of suppressing the 2nd to 4th order effects of the noise. Our RL method performs better than other machine learning methods that deal with even the ideal noiseless case [8][24]. Even at a large standard deviation of the noise strength of $\sigma_N = 10^{-1}$, ensemble gate infidelities $< 10^{-3}$, lower than the error threshold of surface code of quantum error correction for fault-tolerant quantum computation [24], can still be achieved.

Though the performance of our RL method is not superior to the quantum optimal control method employing classical optimization algorithms, such as the Nelder-Mead optimization algorithm [14], in the noisy cases, our RL method generates a control policy

doi:10.6342/NTU202400695



that does not rely on the detailed information of the noise properties and underlying quantum system dynamics required in the quantum optimal control theory. Our study is a preliminary attempt of applying the DRL agent to perform quantum gates in a noisy environment, and we have shown that it works for the QSN. Future work is to extend the applicability of our method from the QSN investigated here to other more general noises, making it particularly useful for the system and noise models that are not exactly known. In addition, our study so far has focused on constructing high-fidelity robust quantum gates at the pulse level with a PPO DRL agent, and have not exploited the flexibility of employing other possible ML algorithms. For example, the pulse sequence observation in our study scales with the size of observation N, and we may use recurrent neural networks as agent's networks to overcome this problem. We also believe that it is promising to apply the ML methods at the gate level for more general and broader quantum control and quantum computing tasks.

doi:10.6342/NTU202400695

Reference

[1] Shor, P. W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5), 1484-1509.

[2] Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 212-219.

[3] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017) Proximal policy optimization algorithms. arXiv:1707.06347.

[4] An, Z. & Zhou, D. (2019). Deep reinforcement learning for quantum gate control. *Europhysics Letters*, 126, 60002.

[5] Bukov, M., Day, A. G. R., Sels, D., Weinberg, P., Polkovnikov, A. & Mehta, P. (2018). Reinforcement Learning in Different Phases of Quantum Control. Physical Review X, 8, 031086.

[6] Daraeizadeh, S., Premaratne, S. P. & Matsuura, A. Y. (2020). Designing high-fidelity multi-qubit gates for semiconductor quantum dots through deep reinforcement learning. *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*, 2020, 30-36.

[7] Lin, J. H. (2022). Simulation of Quantum Gate Control via Proximal Policy Optimization Algorithm [master's thesis, National Taiwan University]. Airiti

doi:10.6342/NTU202400695

Library. https://doi.org/10.6342/NTU202104530

[8] Niu, M. Y., Boixo, S., Smelyanskiy, V. N. & Neven, H. (2019). Universal quantum control through deep reinforcement learning. *NPJ Quantum Information*, 5, 33.

[9] Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K. & Hassabis, D. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. arXiv:1712.01815.

[10] Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2018). High-dimensional continuous control using generalized advantage estimation. arXiv:1506.02438.

[11] Sarkar, S., Paruchuri, P. & Khaneja, N. (2021). Error Analysis of Rotating Wave Approximation in Control of Spins in Nuclear Magnetic Resonance Spectroscopy. *60th IEEE Conference on Decision and Control (CDC)*, 2021, 605-610.

[12] Schuch, N. & Siewert, J. (2003). Natural two-qubit gate for quantum computation using the XY interaction. *Physical Review A*, 67, 032301.

[13] Long, J., Zhao, T., Bal, M., Zhao, R., Barron, G. S., Ku, H. S., … Pappas, D. P. (2021). A universal quantum gate set for transmon qubits with strong ZZ interactions. arXiv:2103.12305.

[14] Orlando, T. P., Mooij, J. E., Tian, L., van der Wal, C. H., Levitov, L. S., Lloyd, S.,

doi:10.6342/NTU202400695

& Mazo, J. J. (1999). Superconducting persistent-current qubit. *Physics Review B*, 60, 15398.

[15] Porras, D. & Cirac, J. I. (2004). Effective Quantum Spin Systems with Trapped Ions. *Physics Review Letters*, 92, 207901.

[16] Britton, J. W., Sawyer, B. C., Keith, A., Wang, C.-C. J., Freericks, J. K., Uys, H., … Bollinger, J. J. (2012). Engineered two-dimensional Ising interactions in a trapped-ion quantum simulator with hundreds of spins. *Nature*, 484, 489–492.

[17] Simon, J., Bakr, W. S., Ma, R., Tai, M. E., Preiss, P. M. & Greiner, M. (2011). Quantum simulation of antiferromagnetic spin chains in an optical lattice. *Nature*, 472, 307–312.

[18] Hahn, E. L. (1950). Spin echoes. *Physical Review*, 80, 580.

[19] Viola, L., Knill, E. & Lloyd, S. (1999). Dynamical Decoupling of Open Quantum Systems. *Physical Review Letters*, 82(12), 2417-2421.

[20] Huang, C. H. & Goan, H. S. (2017). Robust quantum gates for stochastic time-varying noise. *Physical Review A*, 95, 062325.

[21] Dyson, F. J. (1949). The radiation theories of Tomonaga, Schwinger, and Feynman. *Physical Review*, 75, 486.

[22] Chen, C. L., Dong, D. Y., Long, R. X., Ian R. Petersen, I. R. & Rabitz, H. A. (2014). Sampling-based learning control of inhomogeneous quantum ensembles. *Physical*

doi:10.6342/NTU202400695

Review A, 89, 023402.

[23] Kingma, D. P. & Ba, J. (2014). Adam: A Method for Stochastic Optimization. arXiv:1412.6980.

[24] Baum, Y., Amico, M., Howell, S., Hush, M., Maggie Liuzzi, M., Mundada, P., Merkh, T., Carvalho, A. R. R. & Biercuk, M. J. (2021). Experimental Deep Reinforcement Learning for Error-Robust Gate-Set Design on a Superconducting Quantum Computer. *Physical Review X Quantum*, 2, 040324.

[25] Gottesman, D. (2009). An introduction to quantum error correction and fault-tolerant quantum computation. arXiv:0904.2557.

doi:10.6342/NTU202400695