Fast quantum gate design with deep reinforcement learning using real-time feedback on readout signals

by

Emily Wright
B.A.Sc., Queen's University, 2021

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Physics and Astronomy

We acknowledge and respect the lək̓ʷəŋən peoples on whose territory the university stands and the Songhees, Esquimalt and W̱SÁNEĆ peoples whose historical relationships with the land continue to this day.

Fast quantum gate design with deep reinforcement learning using real-time feedback on readout signals

by

Emily Wright
B.A.Sc., Queen's University, 2021

Supervisory Committee

Dr. Rogério de Sousa, Supervisor
(Department of Physics and Astronomy)

Dr. Kristan Jensen, Departmental Member
(Department of Physics and Astronomy)

ABSTRACT

The design of high-fidelity quantum gates is difficult because it requires the optimization of two competing effects, namely maximizing gate speed and minimizing leakage out of the qubit subspace. We propose a deep reinforcement learning algorithm that uses two agents to address the speed and leakage challenges simultaneously on superconducting transmon qubits. The first agent constructs the qubit in-phase control pulse using a policy learned from rewards that compensate short gate times. The rewards are obtained at intermediate time steps throughout the construction of a full-length pulse, allowing the agent to explore the landscape of shorter pulses. The second agent determines an out-of-phase pulse to target leakage. Both agents are trained on real-time data from noisy hardware, thus providing model-free gate design that adapts to unpredictable hardware noise. To reduce the effect of measurement classification errors, the agents are trained directly on the readout signal from probing the qubit. We present proof-of-concept experiments by designing X and square root of X gates of various durations on IBM hardware. After just 200 training iterations, our algorithm is able to construct novel control pulses up to two times faster than the default IBM gates, while matching their performance in terms of state fidelity and leakage rate. As the length of our custom control pulses increases, they begin to outperform the default gates. Improvements to the speed and fidelity of gate operations open the way for higher circuit depth in quantum simulation, quantum chemistry and other algorithms on near-term and future quantum devices.

Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank:

Dr. Rogério de Sousa for providing innumerable insights and never allowing me to undervalue my work,

my friends for always being there (with special thanks to **Camryn, Scott, Max, and Breanna** for letting me crash their office), and

my family for their encouragement and support, and making sure I was away on an adventure more often than not.

Dedication

To **Ranger**, who has no idea what a thesis is.

Chapter 1

Introduction

1.1 Motivation

Quantum computers promise to spur advancements in industries ranging from medicine to finance through simulation of chemical reactions, new cryptographic protocols and many other novel algorithms. Performing reliable quantum computations at a large-scale requires controlling quantum systems subject to hardware noise and fabrication variability [1]. Error-correcting protocols [2] can protect quantum information against some errors but their implementation relies on high-quality quantum gate operations. The fidelity of gates suffers due to long control pulse times leading to decoherence. Decoherence can be reduced by creating faster gates but these cause excitations out of the computational subspace. Leakage out of the computational subspace is of particular concern as it requires substantial additional resources to correct and can significantly impact the threshold of certain error correction protocols [3–6]. Thus, engineering faster and higher-fidelity gates is of timely importance.

Optimal and robust control strategies for gate design have been shown to increase the fidelity of quantum operations [7, 8]. However, these techniques are mostly analytic [8–12] or based on numerically simulated environments [7, 13–18] thus they re-

quire precise physical and noise models of the hardware. Even model-free approaches such as gradient-based optimizations [7, 17, 18] and genetic algorithms [19] require complex calculations that make them better suited to training on simulations rather than real quantum hardware. Simple parameter optimizations like Optimized Randomized Benchmarking for Immediate Tune-up (ORBIT) [20] may also be used, but do not lead to novel control pulse shapes. The quality of all these control pulses also suffers due to time-dependent changes in the processor parameters. Frequent calibration to combat these fluctuations is costly and even when properly calibrated, the control pulse shapes are sub-optimal and allow for errors. In large-scale quantum processors, the difficulty of completely characterizing the system along with other engineering constraints inhibit the use of model-based control techniques.

Reinforcement learning [21–23] is an alternative approach for gate design which operates without prior knowledge of the hardware model. Reinforcement learning and its variants have been applied to myriad quantum control problems using numerical simulated environments [24–41]. Such set-ups demonstrate the potential of reinforcement learning, but suffer from the same modelling constraints as other methods. Recently, a few experiments have been carried out using reinforcement learning directly on noisy hardware [42, 43]. As well, advancements in control hardware using custom built field-programmable gate arrays (FPGAs) have opened the possibility for real-time feedback control to speed up the learning process [42].

This thesis is motivated by the short-comings of existing gates on quantum computers, the demonstrated success of reinforcement learning to solve quantum control problems, and the new opportunities for real-time feedback on noisy quantum hardware.

1.2 Overview

In this thesis, we present a deep reinforcement learning algorithm to design fast quantum gates with built in error-mitigation specifically targeting leakage errors in superconducting transmon qubits. We use two agents to address the competing effects of maximizing gate speed and minimizing leakage out of the qubit subspace. The first agent constructs the qubit in-phase control pulse using a policy learned from rewards that compensate short gate times. The second agent determines an out-of-phase pulse to target leakage. Our algorithm has several advantages over existing proposals for gate design including:

1. enabling design of faster gates by rewarding intermediate steps in the pulse,
2. reducing the impact of measurement errors by training directly on the readout signal rather than classifying the state,
3. mitigating leakage and creating fast gates simultaneously with a dual agent architecture,
4. speeding up training using low measurement overhead and real-time feedback, and
5. accounting for realistic noise in the quantum processor by training directly on hardware.

We also pre-train our agents with the default gates calibrated on the quantum hardware so our initial control policy captures information about the system.

We conducted proof-of-concept experiments by running our algorithm on an IBM superconducting quantum computer. We optimized X and $\sqrt{X}$ gates of various

durations for state preparation from the ground state. After just 200 training iterations, our agent created gates up to two times faster than the IBM default gates while matching its performance in other metrics. Our results show that our algorithm can be used to design fast quantum gates on superconducting hardware. The proof-of-concept demonstrated single-qubit operations, but our algorithm can easily be extended to two-qubit gates by modifying the reward function and state space to complete a universal gateset. Our algorithm can also be applied to other quantum computing hardware such as trapped ions, quantum dots, or neutral atoms with obvious modifications (e.g. photon counts and laser pulses for trapped ion quantum computing [44]).

This thesis is organized as follows:

Chapter 1 motivates the problem of fast quantum gate design and gives an overview of the thesis.

Chapter 2 provides an introduction to quantum information processing and superconducting quantum computers.

Chapter 3 gives an overview of analytic and numerical methods for quantum gate design, introduces the reinforcement learning algorithm, and provides a review of related works.

Chapter 4 describes our deep reinforcement learning algorithm for fast quantum gate design in detail.

Chapter 5 describes our results from proof-of-concept experiments with X and $\sqrt{X}$ gates.

Chapter 6 explains how our algorithm can be extended to two-qubit gates and other hardware platforms.

Chapter 7 gives conclusions and provides avenues for future work.

Chapter 2

Background

We begin this chapter by introducing the concept of quantum information processing. We then describe the implementation of quantum information processing operations on superconducting quantum hardware. In the final section of this chapter, we provide a short introduction to the Qiskit Pulse library for programming quantum control hardware.

2.1 Quantum information processing

Quantum information processing refers to the manipulation, transmission, and storage of data contained in quantum systems. The fundamental unit of quantum information processing is a two-level quantum system called a qubit. A qubit can be represented as a unit vector $|\psi\rangle$ in a Hilbert space $H \simeq \mathbb{C}^2 = \text{span}\{|0\rangle, |1\rangle\}$. A register of n qubits lies in the tensor product Hilbert space $H^{\otimes n}$. Information stored in qubits can be manipulated to perform computations inaccessible to classical computers [1]. The general steps of gate-model quantum computation are:

1. initialization, usually into the ground state $|0\rangle$,

2. computation represented by unitary operations, and

3. readout via projective measurement in the computational basis.

In the language of quantum computing, the unitary operations are called gates. In this thesis, we focus on engineering gate operations to perform high-quality quantum computations. The Pauli operators

$$X = \sigma^x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y = \sigma^y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z = \sigma^z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{2.1}$$

form a basis for single-qubit operations. An arbitrary single-qubit state (ignoring global phase) can be generated from the ground state using the identity gate I and the Pauli rotation gates

$$R_X(\alpha) = \exp\left(-i\frac{\alpha}{2}\sigma^x\right),\ R_Y(\beta) = \exp\left(-i\frac{\beta}{2}\sigma^y\right),\ R_Z(\phi) = \exp\left(-i\frac{\phi}{2}\sigma^z\right)\ . \tag{2.2}$$

We also consider an entangling operation

$$CX = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{2.3}$$

called a controlled-NOT or CX gate. The action of the CX gate on a two qubit system is

$$\begin{aligned} CX\left|0\right\rangle \otimes \left|0\right\rangle &= \left|0\right\rangle \otimes \left|0\right\rangle, \quad CX\left|0\right\rangle \otimes \left|1\right\rangle = \left|0\right\rangle \otimes \left|1\right\rangle, \\ CX\left|1\right\rangle \otimes \left|0\right\rangle &= \left|1\right\rangle \otimes \left|1\right\rangle, \quad CX\left|1\right\rangle \otimes \left|1\right\rangle = \left|1\right\rangle \otimes \left|0\right\rangle\ . \end{aligned} \tag{2.4}$$

The set $\{R_X(\alpha), R_Y(\beta)\}$ for all angles α, β can be used to construct all possible

one-qubit unitary operators (apart from a global phase). Together with CX gates connecting all nearest-neighbour qubits, this set is said to be *universal* in that it allows the construction of all possible n-qubit unitaries (For a proof see Section 4.5 of [45]).

Because it is challenging to implement arbitrary qubit operations on quantum hardware, a fixed set of gates is used. Current IBM hardware uses the following gateset $\{R_Z(\phi), X, \sqrt{X}, CX\}$. To see that this gateset is universal, consider the relations

$$R_X(\alpha) = R_Z\left(\frac{\pi}{2}\right)\sqrt{X}R_Z(\alpha+\pi)\sqrt{X}R_Z\left(\frac{\pi}{2}\right) \tag{2.5}$$

and

$$R_Y(\beta) = R_Z\left(\frac{\pi}{2}\right)R_X(\beta)R_Z\left(-\frac{\pi}{2}\right). \tag{2.6}$$

Thus, we have arbitrary single-qubit rotations and can generate entanglement with CX gates. In this thesis, we aim to improve the implementation of the gates in this universal gateset on superconducting quantum computers.

2.2 Superconducting quantum computers

Superconducting quantum computers are one realization of a quantum information processing unit. They are built with transmon qubits, formed of the lowest two energy levels in the quantized spectrum of a superconducting circuit [46]. In the following section, we describe the physics of transmon qubits and we explain their control and measurement.

2.2.1 The transmon qubit

To understand the physics of the transmon qubit, it is useful to first understand the dynamics of an LC circuit. An LC circuit (Figure 2.1a) is composed of an inductor L in parallel with a capacitor C. The energy in the circuit oscillates between magnetic energy in the inductor and electric energy in the capacitor. The instantaneous, time-dependent energy in each element is

$$E(t) = \int_{-\infty}^{t} d\tau \, V(\tau) I(\tau) \tag{2.7}$$

where V and I denote the voltage and current of the capacitor or inductor. The flux

$$\Phi(t) = \int_{-\infty}^{t} d\tau \, V(\tau) \tag{2.8}$$

is a generalized coordinate of the circuit and can be associated with the canonical position to derive the system Hamiltonian. Using the circuit relations

$$V = L\frac{dI}{dt} \quad I = C\frac{dV}{dt} \tag{2.9}$$

we obtain the energy in the capacitor and inductor

$$\mathcal{T}_C = \frac{C\dot{\Phi}^2}{2} \quad \mathcal{U}_L = \frac{\Phi^2}{2L} \tag{2.10}$$

in terms of flux. The Lagrangian for the system is thus

$$\mathcal{L} = \mathcal{T}_C - \mathcal{U}_L = \frac{C\dot{\Phi}^2}{2} - \frac{\Phi^2}{2L} \, . \tag{2.11}$$

As well, the momentum conjugate to flux is the charge on the capacitor

$$Q = \frac{d\mathcal{L}}{d\dot{\Phi}} = C\dot{\Phi}. \tag{2.12}$$

We can now find the system Hamiltonian via a Legendre transformation to give

$$\mathcal{H} = Q\dot{\Phi} - \mathcal{L} = \frac{Q^2}{2C} + \frac{\Phi^2}{2L}\,. \tag{2.13}$$

The Hamiltonian in eq. (2.13) is analogous to a mechanical harmonic oscillator with mass $m = C$ and frequency $\omega = 1/\sqrt{LC}$. We promote the flux and charge to quantum operators (indicated by a hat) which satisfy the commutation relation

$$\left[\hat{\Phi}, \hat{Q}\right] = i\hbar\,. \tag{2.14}$$

Re-scaling to reduced flux $\phi = \hat{\Phi}/\Phi_0$ using the magnetic flux quantum $\Phi_0 = h/(2e)$ and reduced charge $n = \hat{Q}/(2e)$ (and removing hats for ease of notation) gives the harmonic oscillator Hamiltonian

$$\mathcal{H} = 4E_c n^2 + \frac{1}{2}E_L\phi^2 \tag{2.15}$$

where $E_c = e^2/2C$ and $E_L = (\Phi_0/2\pi)^2/L$. The eigenstates of the Hamiltonian labelled $|k\rangle$ for $k = 0, 1, 2\ldots$ have equally spaced energies with gap $E_k - E_{k-1} = \hbar\omega_r$ for the resonant frequency $\omega_r = 1/\sqrt{LC}$ as plotted in Figure 2.1c.

The equally-spaced energy spectrum poses a challenge for quantum information processing because driving a transition at the $|0\rangle \leftrightarrow |1\rangle$ resonant frequency also induces transitions outside the computational subspace (i.e., to higher eigenstates $|2\rangle, |3\rangle, \ldots$). Replacing the inductor with a nonlinear superconducting circuit element called a Josephson junction introduces the desired isolation of the qubit resonance

frequency $\omega_q = (E_1 - E_0)/\hbar$. A Josephson junction is formed of two superconducting metals separated by a thin insulating barrier [47, 48]. Cooper-pairs (paired electrons that form the superconducting state) can tunnel through this barrier in a quantum-coherent way. This gives rise to the Josephson relations describing current and voltage across the barrier:

$$I = I_c \sin(\phi) \quad V = \frac{\hbar}{2e}\dot{\phi}\,, \tag{2.16}$$

Here I_c is the junction critical current, the maximum current that can flow without resistance across the barrier. These lead to the modified Hamiltonian

$$\mathcal{H} = 4E_c n^2 - E_J \cos\phi \tag{2.17}$$

where $E_C = e^2/(2C)$ ($C = C_s + C_J$ is the sum of the shunt capacitance and the self-capacitance of the junction) and $E_J = I_c\Phi_0/(2\pi)$. The circuit and its quantized energy spectrum are depicted in Figures 2.1b and 2.1d. The dynamics of the qubit are governed by the ratio E_J/E_C. Transmon qubits are engineered to have $E_J \gg E_C$ by using a large shunt capacitance $C_s \gg C_J$ [46]. For small fluctuations in phase, we expand the potential energy term in a power series to achieve an anharmonic oscillator with quartic perturbation (Duffing oscillator)

$$\mathcal{H} = 4E_c n^2 - \frac{1}{2}E_J\phi^2 + \frac{1}{24}E_J\phi^4 + \mathcal{O}(\phi^6)\,. \tag{2.18}$$

Using creation and destruction operators $n = -in_{zpf}(a - a^\dagger)$ and $\phi = \phi_{zpf}(a + a^\dagger)$ we can re-write the Hamiltonian in a simpler form

$$\mathcal{H} = \omega_q a^\dagger a + \frac{\alpha}{2}a^\dagger a^\dagger a a\,, \tag{2.19}$$

by approximating to the fourth order with $[a, a^\dagger] = 1$ and dropping terms due to

the rotating wave approximation (RWA) (for details, see Section 2.2.2). The qubit frequency is

$$\omega_q = \frac{\sqrt{8E_J E_C} - E_C}{\hbar} \tag{2.20}$$

and the anharmonicity is $\alpha = -E_C$, with $E_C \ll E_J$ and $|\alpha| \ll \omega_q$. The quantities

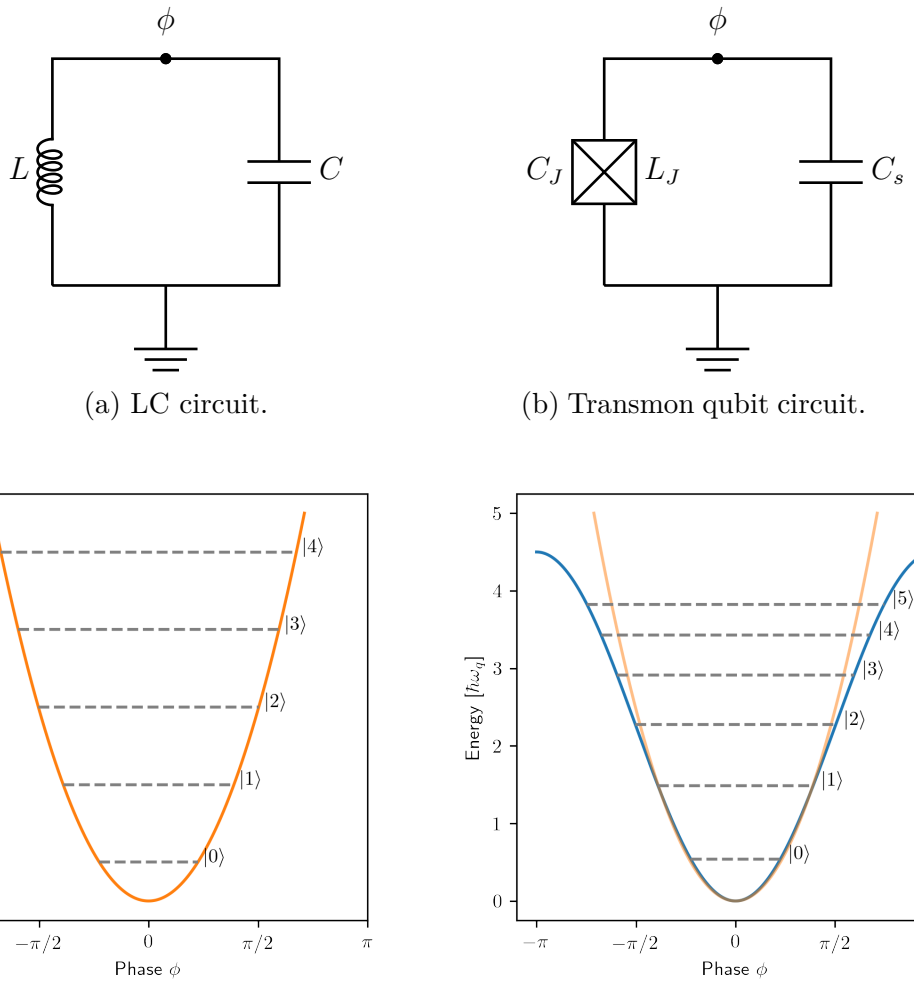
$$n_{zpf} = \left(\frac{E_J}{32E_C}\right)^{1/4} \quad \phi_{zpf} = \left(\frac{2E_C}{E_J}\right)^{1/4} \tag{2.21}$$

refer to the zero-point fluctuations of the charge and phase variables. Although the Hamiltonian in eq. (2.19) contains higher energy terms, the ideal transmon qubit is operated entirely in the ground state $|0\rangle$ and first excited state $|1\rangle$. Leakage refers to undesirable transitions out of the $\{|0\rangle, |1\rangle\}$ computational subspace into higher energy states such as $|2\rangle, |3\rangle$, etc during computations. Even if the qubit returns to the computational subspace, the temporary $|2\rangle$ population during the gate operation leads to an undesirable phase rotation [49]. It is hard to detect and correct leakage errors [3–6]. In this thesis, we aim to control transmon qubits with minimal leakage.

2.2.2 Control of transmon qubits

In this section, we describe the control of transmon qubits. To carry out computations, transmon qubits are manipulated by applied AC voltages. A voltage drive line is coupled to the qubit via a capacitor C_d as shown in Figure 2.2. Similar circuit analysis to the previous section yields a drive Hamiltonian [46]

$$\mathcal{H}_d = -i2e\frac{C_d}{C} n_{zpf} V_d(t)(a - a^\dagger) \tag{2.22}$$



(a) LC circuit.

(b) Transmon qubit circuit.

(c) Quantum harmonic oscillator spectrum.

(d) Transmon qubit spectrum.

Figure 2.1: Circuit diagrams and plots of the quantized energy spectrum for an LC circuit and a transmon qubit. The superconducting phase is denoted ϕ. Fig. (a) is a circuit diagram for an inductor L in parallel with a capacitor C. Fig. (b) is a circuit diagram for a transmon qubit formed of a Josephson junction (with inductance L_J and capacitance C_J) and a shunt capacitor C_S. Fig. (c) shows the quantized energy spectrum of a quantum harmonic oscillator where the energy levels are spaced $\hbar\omega_r$ apart. Fig. (d) shows the quantized energy spectrum of a transmon where the Josephson junction introduces non-linearity in the energy levels. The ground state and first excited state are separated by a distance $\hbar\omega_q$. The orange underlay shows the quadratic shape of the LC circuit spectrum before adding the Josephson junction. Modified from Krantz et al. [46].

where $V_d(t)$ is an arbitrary voltage envelope. We define a control envelope

$$c(t) := 2e\frac{C_d}{C}n_{zpf}V_d(t) \tag{2.23}$$

which absorbs the pre-factor. A useful form for $c(t)$ is

$$c(t) = \begin{cases} c^x(t)\cos(\omega_d t) + c^y(t)\sin(\omega_d t) & 0 < t < t_g \\ 0 & \text{otherwise} \end{cases}, \tag{2.24}$$

composed of two independent quadratures $c^x(t)$ and $c^y(t)$ on a single drive frequency ω_d for the duration of the gate operation t_g. Transforming to the frame of reference that rotates with frequency ω_d via the operator

$$\mathcal{R} = \exp\left(-i\omega_d t a^\dagger a\right) \tag{2.25}$$

yields

$$\mathcal{H}_d^{\mathcal{R}} = \mathcal{R}^\dagger \mathcal{H}_d \mathcal{R} - i\hbar\mathcal{R}^\dagger\dot{\mathcal{R}} \tag{2.26}$$

$$= c(t)\left(\exp\left(-i\omega_d t\right)a + \exp\left(i\omega_d t\right)a^\dagger\right) . \tag{2.27}$$

Finally, making the RWA by ignoring terms at high frequency ($2\omega_d$) gives

$$\mathcal{H}_d^{\text{RWA}} = \frac{c^x(t)}{2}(a + a^\dagger) - i\frac{c^y(t)}{2}(a - a^\dagger) . \tag{2.28}$$

To clarify the role of the voltage drive, let us briefly consider the ideal two-level regime. The creation and annihilation operators correspond to Pauli operators in the

following manner:

$$\sigma^x = a + a^\dagger, \quad \sigma^y = -i(a - a^\dagger), \quad \sigma^z = 1 - 2a^\dagger a\,. \tag{2.29}$$

Adding in the free Hamiltonian from eq. (2.19) and dropping constants, we can write the controlled transmon Hamiltonian as

$$\mathcal{H}^{\text{RWA}} = \frac{\omega_d - \omega_q}{2}\sigma^z + \frac{c^x(t)}{2}\sigma^x + \frac{c^y(t)}{2}\sigma^y\,. \tag{2.30}$$

Matching the drive frequency to the qubit frequency allows for arbitrary x- and y-rotations. For example, by choosing $\omega_d = \omega_q$ and $c(t)$ such that

$$\int_0^{t_g} dt\, c^x(t) = \pi \tag{2.31}$$

and $c^y(t) = 0$, we can create an X gate apart from a global phase

$$\exp\left(-i\frac{\pi}{2}\sigma^x\right) = -iX. \tag{2.32}$$

Similarly, by integrating the $c^x(t)$ control to $\pi/2$ we implement a $\sqrt{X}$ gate. For fixed-frequency transmon qubits (i.e., ω_q is an intrinsic property set during fabrication), arbitrary z-rotations are performed in software rather than hardware [50].

To take full advantage of quantum resources, some qubits are capacitively coupled to generate entanglement. Transmon qubits are operated in the weak coupling regime where the capacitance C_g is much less than the qubit capacitances C_1 and C_2. Figure 2.3 depicts the circuit for a pair of coupled qubits. Circuit analysis leads to a

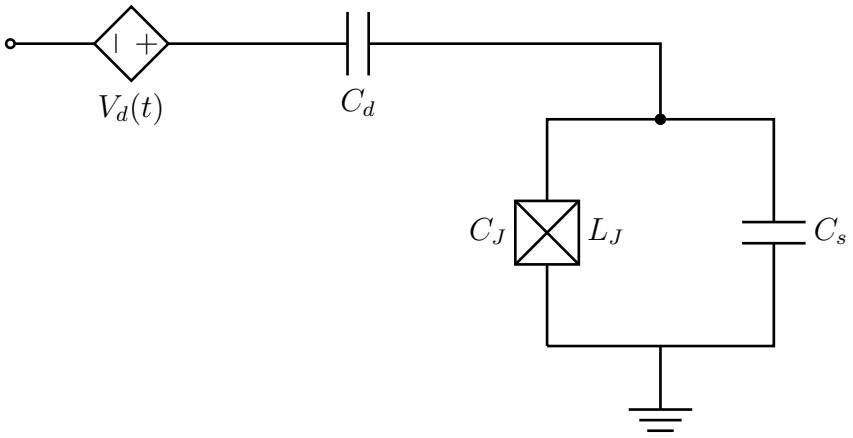


Figure 2.2: A circuit diagram for transmon qubit capacitively coupled to a microwave drive line characterized by a time-dependent voltage $V_d(t)$.

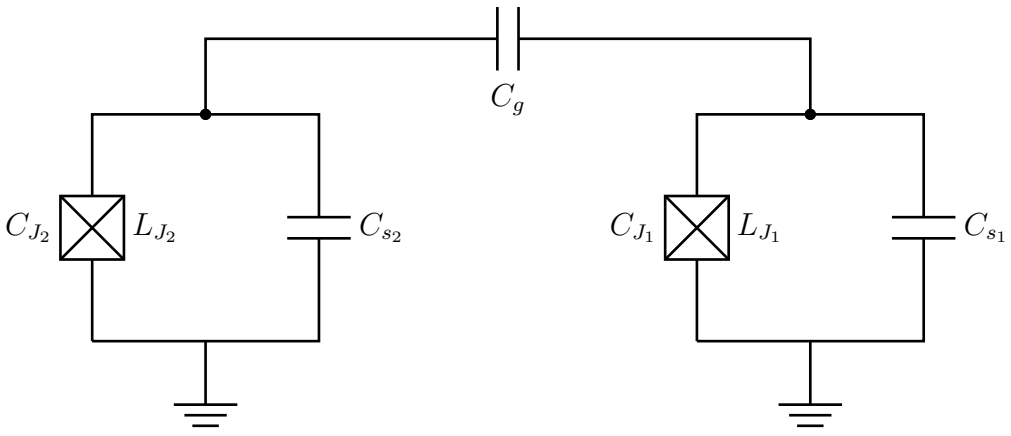


Figure 2.3: A circuit diagram for two capacitively coupled transmon qubits

Hamiltonian [46]

$$\mathcal{H} = \sum_{j=1,2} \left[\omega_{q_j} a_j^\dagger a_j + \frac{\alpha_j}{2} a_j^\dagger a_j^\dagger a_j a_j \right] - g(a_1 - a_1^\dagger)(a_2 - a_2^\dagger) \tag{2.33}$$

for two interacting qubits, where the expression in brackets represents the Hamiltonian from eq. (2.19) for each qubit and g is the coupling strength. In the two level simplification, this amounts to

$$\mathcal{H} = \sum_{j=1,2} -\frac{1}{2} \omega_{q_j} \sigma_j^z + g \sigma_1^y \sigma_2^y. \tag{2.34}$$

Driving the first qubit at the resonant frequency of the second qubit induces a rotation on the second qubit conditional on the state of the first qubit. To see this, we diagonalize the Hamiltonian using a Schrieffer-Wolff transformation. Let $\Delta_{12} = \omega_{q_1} - \omega_{q_2}$ be the frequency difference between the qubits with $g \ll \Delta_{12}$. The Schrieffer-Wolff transformation applied to the external drives gives [51]

$$\mathcal{H}_{d,1} = c_1(t) \left(\sigma_1^x - \nu_1 \sigma_2^x - \mu_1 \sigma_1^z \sigma_2^x \right) \tag{2.35}$$

$$\mathcal{H}_{d,2} = c_2(t) \left(\sigma_2^x - \nu_2 \sigma_1^x - \mu_2 \sigma_1^z \sigma_2^x \right) \tag{2.36}$$

where

$$\mu_1 = \frac{g}{\Delta_{12}} \frac{\alpha_1}{\alpha_1 + \Delta_{12}} \tag{2.37}$$

$$\mu_2 = \frac{g}{\Delta_{12}} \frac{\alpha_2}{\alpha_2 - \Delta_{12}} \tag{2.38}$$

$$\nu_1 = \frac{g}{\alpha_1 + \Delta_{12}} \tag{2.39}$$

$$\nu_2 = \frac{-g}{\alpha_2 - \Delta_{12}} \tag{2.40}$$

and $c_j(t)$ is the control envelope for qubit j and α_j is the anharmonicity of qubit j.

A drive $c_1(t)$ at frequency $\omega_{d,1} = \omega_{q_2}$ on the first qubit induces a conditional rotation $\sigma_1^z\sigma_2^x$ as well as an unconditional rotation σ_2^x and a phase rotation σ_1^z from driving off-resonance. This type of drive is called a cross-resonance ($CR(\theta)$) pulse and was first developed in [52]. The angle of rotation

$$\theta = -\mu_1 \int_0^{t_g} dt\, c_1(t) \tag{2.41}$$

depends on the voltage envelope. The unconditional single-qubit rotations can be eliminated by an "echo" scheme consisting of four pulses – $CR(\theta/2)$ gate, X gate on q_1, $CR(-\theta/2)$ gate, X gate on the q_1 – as proposed in [53]. The unitary matrix realized by a $CR(\theta)$ pulse is

$$U_{CR}(\theta) = e^{-i\theta\sigma_1^z\sigma_2^x} \begin{pmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} & 0 & 0 \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} & 0 & 0 \\ 0 & 0 & \cos\frac{\theta}{2} & i\sin\frac{\theta}{2} \\ 0 & 0 & i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{pmatrix} \tag{2.42}$$

By combining the $CR(\theta)$ pulse with appropriate single-qubit gates, we can create a controlled X gate

$$CX = CR(-\pi/2)I\sqrt{X}R_Z(\pi/2)I \tag{2.43}$$

which is ubiquitous in quantum algorithms.

These simple models for qubit control and coupling provide a framework to discuss the control of transmon qubits; however, they are insufficient to capture the stochastic dynamics in an experimental setting. In particular, we cannot ignore higher energy levels and we cannot assume the qubits are isolated from their environment. Over time qubits return to the ground state and lose superposition due to interactions with their environment in a process called decoherence. Longer gates suffer from

decoherence, but imperfect operations lead to leakage errors, especially in shorter pulses which excite the qubit abruptly. These competing effects are difficult to model even without additional considerations such as errors in classical drive lines. In this thesis, we propose to overcome these challenges with reinforcement learning which does not require a model.

2.2.3 Measurement of transmon qubits

We now explain how to read out the state of transmon qubits after performing quantum computations. The transmon qubit is coupled to a resonator for measurement in the computational basis $\{|0\rangle, |1\rangle\}$. Each transmon qubit circuit is coupled to a superconducting coplanar waveguide resonator. The resonator can be modelled as a harmonic oscillator which is capacitively coupled to the qubit (just as the two transmon qubits were coupled to each other). The qubit-resonator Hamiltonian is given by [54]

$$\mathcal{H} = \omega_r \left(b^\dagger b + \frac{1}{2}\right) + \frac{\omega_q}{2}\sigma^z + g\left((\sigma^x + i\sigma^y)b + (\sigma^x - i\sigma^y)b^\dagger\right) \tag{2.44}$$

where ω_r is the resonator frequency, g defines the coupling strength, $b^\dagger, b$ are the creation and annihilation operators for the resonator, and the Pauli operators $\sigma^x, \sigma^y, \sigma^z$ act on the qubit in the two-level approximation. In the dispersive regime $\Delta_{qr} = |\omega_q - \omega_r| \gg g$, the qubit frequency is far detuned from the resonator frequency compared to the coupling strength. The interaction can be simplified with a change of basis which eliminates the interaction to first order in g/Δ_{qr}. Using again the Schrieffer-Wolff transformation, we diagonalize the Hamiltonian to get [51]

$$\mathcal{H} = \omega_r \left(b^\dagger b + \frac{1}{2}\right) + \frac{\omega_q}{2}\sigma^z + \frac{g^2}{\Delta_{qr}}\sigma^z b^\dagger b \tag{2.45}$$

$$= \left(\omega_r + \frac{g^2}{\Delta_{qr}}\sigma^z\right)\left(b^\dagger b + \frac{1}{2}\right) + \left(\frac{\omega_q}{2} - \frac{g^2}{2\Delta_{qr}}\right)\sigma^z\,. \tag{2.46}$$

The first term encapsulates a qubit-state dependent shift in the resonator frequency. The second term shows that the vacuum fluctuations in the resonator produce a shift in the qubit resonance frequency (the analog of the atomic "Lamb" shift in quantum electrodynamics).

The state of the transmon qubit is measured by driving the resonator frequency with a microwave, and measuring the reflected or transmitted wave. It is convenient to think of the driving microwave as a time-dependent force applied to a classical harmonic oscillator (the resonator). Driving the resonator produces a response that can be measured as a reflected (or transmitted) microwave signal given by

$$v(t) = A\cos(\omega_m t + \gamma) = \mathrm{Re}\left[Ae^{i\gamma}e^{-i\omega_m t}\right] \tag{2.47}$$

where ω_m is the frequency of the microwave signal and γ is a phase shift. When $\omega_m = \omega_r$, the driving microwave will be *below* resonance if the qubit state is in state $|0\rangle$, and *above* resonance if the qubit is in state $|1\rangle$. Following the analogy with the classical harmonic oscillator, we get $\gamma \approx 0$ in the former case and $\gamma \approx \pi$ in the latter. The signal is measured by means of homodyne detection [46] leading to an observation vector that is comprised of time traces of the in-phase I and quadrature Q components of the digitized signal

$$Ae^{i\gamma} = A\cos\gamma + iA\sin\gamma = I + iQ\ . \tag{2.48}$$

The state of the qubit can be inferred from the location of the signal in the (I, Q)-plane. The separation between states is maximized when the resonator is probed midway between the qubit-state dependent resonator frequencies [46]. However, on noisy hardware, the locations often overlap in the (I, Q)-plane and a decision maker such as a linear discriminant analyzer [55] is necessary to predict the state. The

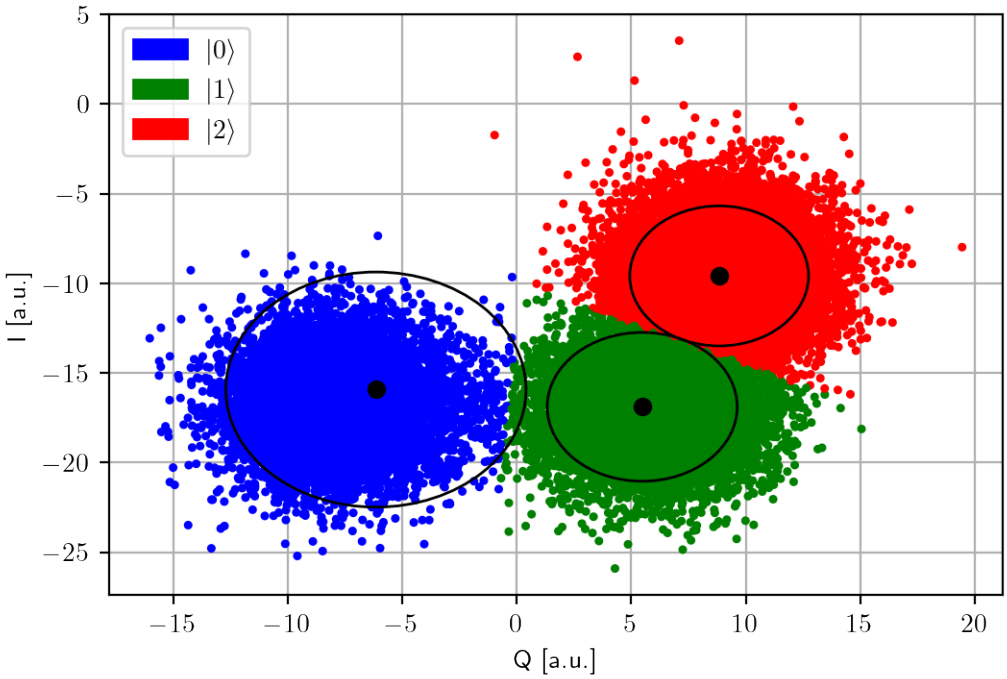


Figure 2.4: A plot of the (I, Q) readout signal for different qubit states (10000 shots each) taken on IBM Lima April 12, 2023. Black circles indicate the mean and standard deviation of each cluster. The overlap between clusters shows the difficulty of distinguishing between states.

outcomes of experimental quantum measurements generally have a Gaussian distribution. The spread is due in part to quantum noise associated with the microwave signal used to probe the resonator and classical noise in digital amplifiers as well as intrinsic uncertainty required by the Heisenberg principle. For example, Figure 2.4 shows measurement data taken on the IBM Lima superconducting quantum computer. In this thesis, we use the (I, Q)-signal directly as an input to our algorithm rather than classifying the state to reduce the impact of measurement errors.

2.3 Qiskit pulse

Quantum control hardware must be programmed to implement the voltage pulses and measurements described in the previous sections. Qiskit is an open source framework for programming quantum computers [56]. The Qiskit Pulse library allows pulse-level control of quantum devices [57]. Users can specify the continuous time-dynamics of operations and measurements. A pulse is a time-series of complex amplitudes $\{a_m\}_{m=0}^{N_{\text{samp}}-1}$ called samples. Each sample in a pulse is mixed with a local oscillator to give

$$\text{Re}\left[e^{-i\omega_d dt}a_m\right] \tag{2.49}$$

where the length dt of the time step is device dependent. For instance, the IBM Lima superconducting quantum computer has $dt = 0.22222$ nanoseconds. The total gate time is $t_g = N_{\text{samp}}dt$ (about 35.6 nanoseconds on IBM Lima for single-qubit gates). The overall control signal

$$c(t) = \begin{cases} \text{Re}[a_m]\cos(\omega_d t) + \text{Im}[a_m]\sin(\omega_d t) & \text{if } mdt \leq t \leq (m+1)dt \\ & \text{for } m = 0, \ldots, N_{\text{samp}} - 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.50}$$

is sent along the drive line (called a channel in Qiskit Pulse) of the specified qubit. In this thesis, we use Qiskit Pulse to define custom pulses for gate design experiments on IBM quantum devices.

Chapter 3

Optimal control for quantum gate design

Quantum optimal control encompasses a set of analytic, numerical, and machine learning strategies for achieving the goal of steering a quantum system from an initial state to a target state. In this chapter, we give an overview of existing analytic and numerical quantum optimal control methods for gate design and argue that reinforcement learning is a better approach. We proceed by describing the reinforcement learning algorithm. In the last section, we review other proposed reinforcement learning algorithms for quantum gate design.

3.1 Analytic solutions and numerical optimization

Existing strategies for quantum gate design can be divided into two categories: analytic solutions and numerical methods.

Analytic solutions for quantum control are generally difficult to find; however some do exist [8–12]. These solutions may be found by analyzing simple models or through more intensive procedures such as Lie group decomposition [58] which yields exact results but involves decomposing the desired operation into a product of basic

unitary operators and is often unfeasible for practical systems. For single-qubit gates on superconducting devices, an analytic solution based on the Derivative Removal by Adiabatic Gate (DRAG) formula [8] is the industry standard. Recall the form of the control envelope in (2.24). A first-order DRAG gate starts with a Gaussian pulse

$$c^x(t) = A \exp\left(\frac{\left(t - \frac{t_g}{2}\right)^2}{2\sigma^2} \right) - B \tag{3.1}$$

on the first control quadrature where σ is the standard deviation, A is chosen such that the correct rotation (e.g. π for an X gate) is implemented, and B enforces the pulse to start and end at zero. To cancel out leakage, an additional derivative term $c^y(t) = \gamma \dot{c}^x(t)$ is employed on the second quadrature. Analytic solutions like DRAG are easy to implement but are based on simplified models. Their effectiveness can be improved by experimentally calibrating parameters such as the amplitude and γ factor. In this thesis, we compare our experimental results to calibrated DRAG gates [49]. For the remainder of the thesis, we refer to the calibrated first-order DRAG gates in [49] as the "default gates" since they are used on IBM quantum devices.

Many attempts have also been made to develop improved control schemes using numerical methods. Tools from classical optimal control theory including variational methods (i.e., the Krotov approach [13]), the Pontryagin maximum principle [14], and convergent iterative algorithms [15] have been extended to the quantum case [16]. Such gradient-free methods are useful for calibration or optimization when the control pulses are defined by a limited number of free parameters but converge too slowly for more complex problems. Gradient-based numerical methods such as Gradient Ascent Pulse Engineering (GRAPE) [7], Gradient Optimization of Analytic conTrols (GOAT) [17] and Chopped RAndom Basis (CRAB) [18] have also been proposed for quantum gate design. The gradient is difficult to calculate in experiments because

the cost function is noisy and may be discontinuous. As a result, control parameters are determined in simulations requiring complete system modelling. The high computational costs and the sub-optimality of controls designed using simplified models contribute to the limited adoption of these numerical methods for engineering quantum gates in realistic settings. For this reason, we do not compare our experimental results to numerical optimization methods.

3.2 The reinforcement learning algorithm

Reinforcement learning is an alternative approach for optimal control that requires no prior information about the system (i.e., it is model free). Reinforcement learning is a machine learning algorithm wherein an agent interacts with a system and iteratively updates a control policy based on its observations. The entire process is modelled as a controlled Markov Decision Process (MDP). Let $\mathbb{S} \subset \mathbb{R}^n$ be the space of states for the system and $\mathbb{U}$ the set of possible actions. At each step, the system is in some state $s_j \in \mathbb{S}$ and the agent decides on an action $u_j \in \mathbb{U}$ according to a policy. The policy $\pi(\cdot|s_j)$ is a conditional probability distribution over the possible actions in $\mathbb{U}$ given the current state. The system moves into the next state s_{j+1} via a stochastic transition kernel $\mathcal{T}(\cdot|s_j, u_j)$. After observing the next state, the agent receives a corresponding reward $r_j(s_j, u_j, s_{j+1})$. The objective of the controller is to maximize the infinite-horizon discounted expected reward

$$J_\beta(s_0, \pi) = E_{s_0}^{\mathcal{T},\pi}\left[\sum_{j=0}^{\infty} \beta^j r_j(s_j, u_j, s_{j+1})\right] \tag{3.2}$$

over the set of admissible policies π, where $0 < \beta < 1$ is a discount factor, and $E_{s_0}^{\mathcal{T},\pi}$ denotes the expectation for initial state s_0 and transition kernel $\mathcal{T}$ under policy π.

The standard reinforcement learning algorithm is Q-learning [22]. Q-learning

involves tracking the "value" of taking an action u_j given the current state s_j. The Q-value is stored in a table indexed by states and actions. Given an initial table Q_0, the value of each state-action pair is updated according to the Bellman equation

$$Q_{j+1}(s_j, u_j) = Q_j(s_j, u_j) + \alpha_j(s_j, u_j)\Big[r(s_j, u_j, s_{j+1}) + \beta \max_{v \in \mathbb{U}} Q_j(s_{j+1}, v) - Q_j(s_j, u_j)\Big] \tag{3.3}$$

as the agent explores the environment [22]. The coefficient α_j is a hyperparameter called the learning rate and determines how quickly the agent adapts to changes in the environment. Under mild conditions on the learning rate, the algorithm converges to a fixed point denoted Q_*, which satisfies

$$Q_*(s, u) = E\left[r(s, u, s') + \beta \max_{v \in \mathbb{U}} Q_*(s', v)\Big| s, u\right]. \tag{3.4}$$

A policy π which satisfies

$$\max_{u \in \mathbb{U}} Q_*(s, u) = Q_*(s, \pi(s)) \tag{3.5}$$

is an optimal policy (see Theorem 4 in [21] and the main Theorem in [22]).

The general Q-learning algorithm was conceived for finite action and state spaces. For large and/or continuous state spaces, storing the Q-value in a table is not an option. To overcome this challenge, one might quantize the state space [59] or use function approximation [60]. In this work, we approximate the Q-table using a neural network, in a strategy that has been termed "deep reinforcement learning" [61,62]. A neural network [63] is composed of nodes which take an input x and pass it through a linear function $y = w_j x + b_j$. The result y is subsequently passed to an activation function $f : \mathbb{R} \to \mathbb{R}$ to add non-linearity. The nodes are arranged in layers: an input

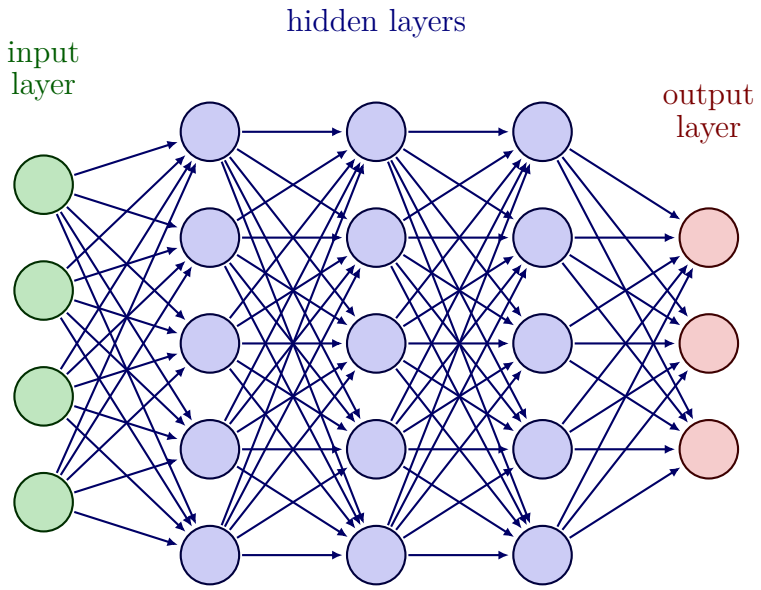


Figure 3.1: A neural network diagram

layer, one or more hidden layers, and an output layer. The nodes in one layer feed their outputs as inputs to the next layer as shown in Figure 3.1. In deep reinforcement learning, the state s_j is the input to the neural network and the output is a probability distribution $\pi(\cdot|s_j)$ over the action space $\mathbb{U}$. The network is represented by a set of parameters $\theta = \{(w_j, b_j)\}$ which are updated in a manner that approximates the Bellman equation.

Additionally, the Q-learning algorithm is "off-policy", meaning the agent learns the value of a different policy (the greedy policy) than the one it is following. Given the complex environment of the quantum computer, we choose instead to use an "on-policy" algorithm in which the agent learns the value of the policy as it is following it. Off-policy learning requires less resources to train and prioritizes learning a general policy quickly. In contrast, on-policy learning allows the agent to explore the environment more and leads to a policy that is more adapted to specific conditions.

In this thesis, we modify the deep REINFORCE algorithm [64] for our purposes.

Given a trajectory

$$\tau_\theta = (s_0, u_0, s_1, u_1, \ldots, s_{N_{\text{ep}}}) \tag{3.6}$$

of states and actions resulting from N_{ep} episodes where the agent has interacted with the environment and followed the policy π_θ parameterized by θ, we can write the expected reward as a function of θ. That is

$$\begin{aligned} R(\theta) &= J_\beta(s_0, \pi_\theta) && (3.7)\\ &= E_{s_0}^{\mathcal{T},\pi_\theta}\left[\sum_{j=0}^{N_{\text{ep}}-1} \beta^j r_j(s_j, u_j, s_{j+1})\right] && (3.8)\\ &= \sum_{j=0}^{N_{\text{ep}}-1} P(s_j, u_j|\tau_\theta)\beta^j r_j && (3.9) \end{aligned}$$

where

$$P(s_j, u_j|\tau_\theta) = P(s_0, u_0, s_1, u_1, \ldots, u_j, s_j|\pi_\theta) \tag{3.10}$$

is the probability of (s_j, u_j) occurring given the trajectory τ_θ. Taking the gradient yields

$$\begin{aligned} \nabla_\theta R(\theta) &= \sum_{j=0}^{N_{\text{ep}}-1} P(s_j, u_j|\tau_\theta)\nabla_\theta \log P(s_j, u_j|\tau_\theta)\left(\beta^j r_j\right) && (3.11)\\ &= E_{s_0}^{\mathcal{T},\pi_\theta}\left[\nabla_\theta \log P(s_j, u_j|\tau_\theta)\left(\beta^j r_j\right)\right]. && (3.12) \end{aligned}$$

In an experiment, we approximate the expectation value by sampling thus

$$\nabla_\theta R(\theta) \approx \sum_{j=0}^{N_{\text{ep}}-1} \nabla_\theta \log P(s_j, u_j|\tau_\theta)\left(\beta^j r_j\right). \tag{3.13}$$

Examining the conditional probability further, we have

$$\nabla_\theta \log P(s_j, u_j|\tau_\theta) = \sum_{k=0}^{j} \nabla_\theta \log \pi_\theta(u_k|s_k) \tag{3.14}$$

so altogether

$$\nabla_\theta R(\theta) \approx \sum_{j=0}^{N_{\text{ep}}-1} \beta^j r_j \left(\sum_{k=0}^{j} \nabla_\theta \log \pi_\theta(u_k|s_k) \right) . \tag{3.15}$$

Using a standard gradient descent

$$\theta_{i+1} = \theta_i + \alpha \nabla_\theta R(\theta) \tag{3.16}$$

we can update the network parameters

$$\theta_{i+1} = \theta_i + \alpha \sum_{j=0}^{N_{\text{ep}}-1} \nabla_\theta \log \pi_\theta(u_j|s_j) \left(\sum_{k=j+1}^{N_{\text{ep}}} \beta^k r_k \right) \tag{3.17}$$

where we have exchanged the order of summation for clarity. Recall that $\pi_\theta(u_k|s_k)$ is the value of one node in the output layer of a neural network parameterized by θ given the input s_k. Calculating $\nabla_\theta \log \pi_\theta(u_k|s_k)$ is achieved via a well-studied process called back-propagation [65] based on the Liebniz chain rule. While deep reinforcement learning is not guaranteed to converge to an optimal solution, we will present ample empirical evidence in the next section that it can be used to solve quantum control problems [24–26, 28, 30, 31, 33–37, 39–43, 66–70].

3.3 Reinforcement learning for quantum gate design

In this section, we review the uses of reinforcement learning for quantum gate design to-date. Many reinforcement learning algorithms have been proposed to solve quantum control problems in areas ranging from Hamiltonian engineering [28] to quantum metrology [25]. Theoretical algorithms make use of simulated environments to provide full access to the state of the quantum system [24, 25, 28, 31, 33–37, 41]. Of these

proposals, several specifically target unitary gate design [33, 38, 71]. In all cases, the agent has access to the exact unitary operator specifying the Schrödinger evolution of the system. In [33, 38], a simplified Hamiltonian model is used while Niu et al. simulate a gmon environment which mimics noisy control actuation and incorporates leakage errors [71]. The agent receives a reward based on gate infidelity which is inaccessible in experiments. In simulation, these algorithms are able to achieve improvements in gate fidelity [33, 38, 71] and gate time [33, 71] over other gate synthesis strategies. However, these proposals are not compatible with training on real hardware and thus suffer from model bias. More realistic reinforcement learning set-ups for quantum control only provide access to fidelities and/or expectation values for some observables [26, 30, 40, 66–69]. Specifically for gate design, Shindi et al. proposed probing a gate with a series of input states [66]. The reward incorporates the average fidelity between the output states and the target states. Such algorithms require prohibitive amounts of averaging in experiments so they are also confined to numerical simulations.

The next step is to perform reinforcement learning using stochastic measurement outcomes or low-sample estimators of physical observable [39, 42, 43, 70]. Most recently, some experiments have been carried out using reinforcement learning directly on noisy quantum hardware [42, 43]. Baum et al. trained a deep reinforcement learning agent on a superconducting computer for error-robust gateset design [43]. The agent was able to learn novel pulse shapes up to three times faster than the benchmark gates with slightly lower error-per-gate and improvements maintained without re-calibration for up to 25 days. The search space was restricted to 8 and 10 segment piece-wise constant (PWC) operations for one- and two-qubit gates respectively [43]. Despite the small search space, the optimization algorithm was inefficient. The training was completed in batches and the reward was a weighted mean over fidelities

estimated using full state tomography [43]. Subsequently, Reuer et al. developed a latency-based deep reinforcement learning agent implemented via an FPGA – a customizable integrated circuit – capable of using real-time feedback at the microsecond time scale [42]. They demonstrated its effectiveness with a state preparation experiment on a superconducting qubit. In this thesis, we look ahead to a future where real-time feedback for quantum control, enabled by hardware such as the FPGA developed in [42], is common. We improve upon the work in [43] by allowing greater flexibility in pulse shapes, training directly on the readout signal to reduce the impact of measurement errors, specifically targeting leakage errors, and relying on real-time feedback to speed-up the learning process.

Chapter 4

Our deep reinforcement learning algorithm for fast quantum gate design

We are now ready to describe our deep reinforcement learning algorithm for fast quantum gate design. First, we present the steps in our algorithm. The second half of this chapter deals with design choices and hyperparameters.

4.1 Algorithm description

We seek to design a PWC control pulse with up to N_{seg} segments of equal length $d\tau = t_g/N_{\text{seg}}$ where t_g is the maximum duration of the gate. Our reinforcement learning agent consists of two neural networks to decide sequentially on the x- and y-quadrature of the control pulse. At each time step, the agents select the amplitudes of the next segment based on real-time feedback about the state of the system. The agents receive rewards throughout the construction of the pulse (rather than just at the end) which opens the possibility to design faster gates.

The network parameters θ_i^x, θ_i^y are updated periodically during the training. Each training iteration for $i = 0, \ldots, N_{\text{iter}} - 1$ consists of $N_{\text{ep}} \leq N_{\text{seg}}$ episodes counted using the index j. The ratio $N_{\text{seg}}/N_{\text{ep}}$ is an integer that sets how many times the neural network parameters are updated before a full waveform is constructed. We use a second index $k = k(i, j)$ to track the pulse segment. The k-th segment of the control pulse takes the form

$$c_k(t) = c_k^x \cos(\omega_d t) + c_k^y \sin(\omega_d t) \tag{4.1}$$

for $kd\tau \leq t < (k+1)d\tau$ and can be represented by the constant amplitudes (c_k^x, c_k^y). Figure 4.1 is a graphic showing how the different indices are related for $N_{\text{seg}} = 20$ and $N_{\text{ep}} = 5$ (and $N_{\text{samp}} = 160$ total samples when converting to Qiskit Pulse) for the first four training iterations.

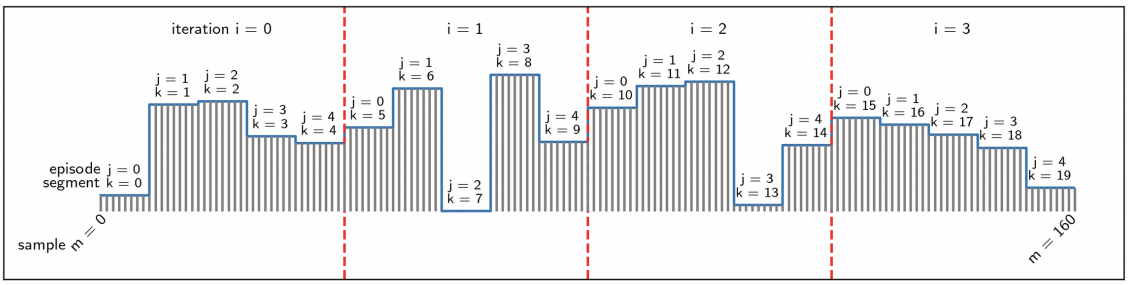


Figure 4.1: A graphic illustrating the different index counters used in our reinforcement learning algorithm.

The j-th input to the first neural network is a state $s_j^x = (\langle I_j \rangle, \langle Q_j \rangle, k)$ composed of the average (I, Q) signal over N_{shot} measurements and the current segment index k. The output of the agent is a probability distribution over the action space $\mathbb{U}^x$ which we denote $\pi_{\theta_i^x}(\cdot|s_j^x)$ to indicate the dependence on the neural network parameters θ_i^x. The agent samples an action $u_j^x \in \mathbb{U}^x$ according to $\pi_{\theta_i^x}(\cdot|s_j^x)$. The state for the second agent $s_j^y = (u_j^x, \mathcal{L}_j)$ is formed of the amplitude u_j^x on the first quadrature and the leakage population

$$\mathcal{L}_j = |\langle 2| U_{k+1} |0\rangle|^2 \tag{4.2}$$

estimated from the (I, Q) readout data, where U_{k+1} represents the operation of evolving the qubit by the first $k+1$ segments of the waveform. The agent now samples an action $u_j^y \in \mathbb{U}^y$ according to the output $\pi_{\theta_i^y}(\cdot|s_j^y)$ of the second network. We update the wave amplitudes $c_{k+1}^x = u_j^x$ and $c_{k+1}^y = u_j^y$ where $k = k(i,j)$ is the segment index stored in s_j^x, the j-th state of the system.

To train the agent, we require a reward function for each network. Let $c_T = (\langle I_T \rangle, \langle Q_T \rangle) \pm \sigma_T$ be the expected average measurement result after applying the target gate to an initial state $|\psi_i\rangle$ (calibrated experimentally). During training, we penalize the distance of the observed measurement signal from $(\langle I_T \rangle, \langle Q_T \rangle)$ which encapsulates both a failure to steer the qubit into the desired state and leakage into higher energy levels. We use the reward function

$$r_j^x \left(s_{j+1}^x, c_T \right) = \min \left\{ 1 - \lambda k, \frac{\sigma_T}{\|(\langle I_{j+1} \rangle, \langle Q_{j+1} \rangle) - (\langle I_T \rangle, \langle Q_T \rangle)\|} - \lambda k \right\}$$

to train the first network, where the term λk penalizes the length of the control pulse for some coefficient $\lambda \in \mathbb{R}$. For the second network, we compensate low leakage populations with the reward

$$r_j^y \left(s_{j+1}^y \right) = \max \left\{ 0, \mathcal{L}_{\max} - \mathcal{L}_{j+1} \right\} \tag{4.3}$$

where $\mathcal{L}_{\max}$ sets a limit on the allowable leakage. A summary of our deep reinforcement learning algorithm is shown in Algorithm 1.

4.1.1 Initialization

The initial state s_0^x and leakage population $\mathcal{L}_0$ are estimated by measuring the qubit in its initial state $|\psi_0\rangle$ before performing any gate operations. The initial policies $\pi_{\theta_0}^x, \pi_{\theta_0}^y$ are generated based on the default gate [49] by pre-training the agents. First,

Algorithm 1 Deep reinforcement learning for fast quantum gate design

Require: initial state s_0^x, leakage $\mathcal{L}_0$, and parameters θ_0^x, θ_0^y

Set $k = 0$
for each iteration $i = 0, ..., N_{\text{iter}} - 1$ **do**
  for each episode $j = 0, ..., N_{\text{ep}} - 1$ **do**
    if $k = N_{\text{seg}}$ **then**
      Re-initialize waveform $k = 0$
      Re-initialize state $s_j^x = s_0^x$
      Re-initialize leakage $\mathcal{L}_j = \mathcal{L}_0$
    end if
    Select next action u_j^x according to policy $\pi_{\theta_i^x}(\cdot|s_j^x)$
    Set $s_j^y = (u_j^x, \mathcal{L})$
    Select next action u_j^y according to policy $\pi_{\theta_i^y}(\cdot|s_j^y)$
    Evolve qubit by first $k+1$ segments of waveform
    Measure qubit N_{shot} times to get $(\langle I_{j+1}\rangle, \langle Q_{j+1}\rangle)$
    Update $k \to k+1$
    Set $s_{j+1}^x = (\langle I_{j+1}\rangle, \langle Q_{j+1}\rangle, k)$
    Calculate reward r_j^x
    Estimate leakage population $\mathcal{L}_{j+1}$
    Calculate reward r_j^y
    Reset qubit to $|0\rangle$
  end for
  Send trajectory $\left(s_0^x, u_0^x, r_0^x, s_1^x, u_1^x, r_1^x, \ldots, r_{N_{\text{ep}}-1}^x\right)$ to first network
  Update neural network parameters $\theta_i^x \to \theta_{i+1}^x$
  Send trajectory $\left(s_0^y, u_0^y, r_0^y, s_1^y, u_1^y, r_1^y, \ldots, r_{N_{\text{ep}}-1}^y\right)$ to second network
  Update neural network parameters $\theta_i^y \to \theta_{i+1}^y$
end for

we discretize a calibrated DRAG pulse into an N_{seg} PWC pulse where the amplitudes (c_k^x, c_k^y) of each segment are the nearest action from $\mathbb{U}^x$ and $\mathbb{U}^y$ for the first and second quadratures respectively. For $k = 0, \ldots, N_{\text{seg}} - 1$, we measure the average $(\langle I \rangle, \langle Q \rangle)$ signal and leakage $\mathcal{L}$ after the first $k + 1$ segments of the waveform. Let $q = x$ or y depending to which quadrature we are referring. The agent receives a reward (denoted p for pre-training) based on the mean squared error

$$p_k^q(s_k^q) = \frac{1}{|\mathbb{U}^q|} \sum_{u \in \mathbb{U}^q} \left| \pi_{\theta_k^q}(u|s_k^q) - \mathcal{N}_{c_k^q}(u) \right|^2 \tag{4.4}$$

between the policy $\pi_{\theta_k^q}(\cdot|s_k^q)$ output by the neural network and a Gaussian distribution

$$\mathcal{N}_{c_k^q}(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}\left(u - c_k^q\right)^2} \tag{4.5}$$

over the respective action space $\mathbb{U}^q$ centered at the desired amplitude c_k^q. The pre-training is performed for several iterations over the waveform. The pre-training can be executed efficiently because the reward and action are independent of the state of the system so the circuits can all be run before updating the neural networks. At the end of the pre-training, the agents' policies $\pi_{\theta_0}^x$ and $\pi_{\theta_0}^y$ favour a calibrated DRAG pulse.

Our initial policy obtained in pre-training captures information about the system dynamics using the default gate; however, our algorithm should not be considered an optimization of the DRAG pulse which has been proposed in other works [8, 20, 72]. The pre-training ensures the exploration begins in a near-optimal region of the solution space and the agents subsequently learn novel pulse shapes.

4.2 Design choices

Having introduced our algorithm, we now elaborate on our design choices. The greatest feature of our algorithm is the ability to create gates of any length from 1 to N_{seg}. Other reinforcement learning algorithms [33,38,39,43,66,71] calculate a reward at the end of a full pulse so their gates have a fixed length. Our intermediate reward scheme lets the agents explore the landscape of shorter pulses. In fact, we explicitly penalize the pulse length in the reward to encourage discovery of fast gates. Aided by real-time feedback, the agents quickly traverse the control landscape. The intermediate rewards have a secondary purpose: they allow the agent to track leakage throughout the gate operation. Even if the leaked population settles back into the computational subspace by the end of the gate, transitions through the $|2\rangle$ state can add undesirable non-Markovian noise [73]. Unlike other reinforcement learning algorithms, our agent minimizes leakage *during* the gate.

We also target leakage with our dual agent architecture. Leakage is a competing effect when attempting to design fast gates with reduced decoherence. Using two agents allows us to manage these challenges separately unlike Baum et al., who address leakage in [43] only using one agent which decides on both quadrature amplitudes. We are also inspired by the DRAG algorithm, where the leakage correction on the y-quadrature depends on the shape of the x-quadrature control. For this reason, we use u_j^x as an input to the second decision maker. Based on hardware parameters, we restrict the x-quadrature amplitudes to the set $\mathbb{U}^x = \{0.00, 0.01, \ldots, 0.19, 0.20\}$ and the y-quadrature amplitudes to $\mathbb{U}^y = \{-0.10, -0.09, \ldots, 0.09, 0.10\}$. Our algorithm converges faster by simultaneously training two agents each endowed with a small action space ($|\mathbb{U}^x| = |\mathbb{U}^y| = 21$) rather than training over a large joint action space $\mathbb{U}^x \times \mathbb{U}^y$ with size 441.

Our choice of state for the first network is motivated by measurement errors. We train directly on the observation signal (I, Q) resulting from probing the qubit as described in Section 2.2.3. Previous algorithms for gate design [33, 38, 43, 66, 71] have used the $|0\rangle$ and $|1\rangle$ populations imperfectly estimated from the location of the signal in the (I, Q)-plane (or tracked in simulation). Our algorithm reduces the impact of measurement errors because we avoid classifying the signal into $|0\rangle$ or $|1\rangle$. We choose a maximum reward of 1 once the average (I, Q) signal falls within a specified radius of the desired location. We aim to reward readout terms far from the decision boundary between states in the (I, Q)-plane but do not require the signal to land precisely on the target location. Our algorithm also has a low measurement overhead since we do not perform full state tomography. Recall the model for a controlled transmon qubit in Section 2.2.2. In particular, from eq. (2.30) we see that X and Y rotations are brought about by driving on independent quadratures. With a limit

$$\max_{u^y \in \mathbb{U}^y} |u^y| = 0.10 \tag{4.6}$$

on the second quadrature amplitude and a well-calibrated drive frequency ω_d, we may assume that any rotations outside the zy-plane are insignificant thus state tomography is unnecessary during training for a universal gateset composed of x-rotations.

The overall efficiency of our algorithm (real-time feedback, small action spaces, and low measurement overhead) makes it is easy to implement experimentally. The main advantage of reinforcement learning over other optimal control strategies is that it requires no model of the quantum system. Most reinforcement learning algorithms proposed to-date rely on numerically simulated environments [33, 38, 66, 71] thus cancelling out their model-free advantage. Unlike these, our algorithm can account for realistic noise in the qubits and for other errors such as over-rotation introduced by the classical drive lines.

4.3 Hyperparameters

Once the overarching design choices have been made, there are numerous hyperparameters that need to be specified. In machine learning, hyperparameters refer to parameters that control the learning process (as opposed to parameters whose value is learned by the training). Hyperparameters are selected to achieve maximal performance in a reasonable amount of time. For example, increasing the number of iterations N_{iter} allows the agent to converge further towards an optimal solution but lengthens the training time. The learning rate α is another important hyperparameter: if it is too small, the agent will not converge, but if it is too large, the agent may hone in quickly on a sub-optimal solution. Along with N_{iter} and α, the discount factor β is present in every discounted cost reinforcement learning algorithm. The discount factor makes the agent prioritize current rewards over future rewards.

The selection of appropriate hyperparameters also depends on the specific situation in which the algorithm is being employed. Here, we discuss hyperparameters in the context of experiments on IBM superconducting quantum computers. In our algorithm, the maximum allowable leakage $\mathcal{L}_{\max}$ is a hyperparameter in the reward function for the second agent (eq. (4.3)). We adjust the leakage threshold depending on the quality of the hardware. IBM quantum computers suffer from significant leakage so we set the threshold at $\mathcal{L}_{\max} = 5\%$. The number of shots N_{shot} is another hyperparameter that may be limited by hardware. Our access to IBM quantum devices allows at most 10000 shots per circuit; although, this is enough for good statistics. As well, we set the maximum gate time $t_g = 35.6$ ns based on the single-qubit gate time on IBM quantum devices, since we want our gates to be at least as fast as the default.

In addition, some hyperparameters are constrained by the choices made for other

hyperparameters. In our algorithm, the number of segments N_{seg} in the PWC control pulse cannot exceed the total gate time t_g divided by the sample length dt. Further, the number of episodes N_{ep} should be a factor of N_{seg} to ensure the policy gets updated periodically throughout the construction of the gate, with one of these updates occurring at the end of the pulse.

The architecture of each neural network is also defined by hyperparameters: the number of layers, the number of nodes in each layer, and the activation functions. The design of neural networks for machine learning is well-studied [74] but there is no formula for building a good network. For each agent, we use a fully-connected feed-forward network containing a single hidden layer with just 12 neurons. A simple network cannot represent a very complex policy, but by limiting the number of parameters, we allow our agent to converge more quickly. We use the rectified linear unit (ReLU) activation function

$$f(y) = \begin{cases} y_i & \text{if } y_i \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

for the hidden layer. At the output layer, we use the softmax function

$$g(y_1, \ldots, y_n) = \left(\frac{e^{y_1}}{\sum_{j=1}^{n} e^{y_j}}, \ldots, \frac{e^{y_n}}{\sum_{j=1}^{n} e^{y_j}} \right) =: (\pi(u_1|s), \ldots, \pi(u_n|s)) \tag{4.8}$$

to convert the vector of outputs $(y_1, \ldots, y_n)$ into a probability distribution $\pi(\cdot|s)$ where s is the state input to the first layer of the network and the action space $\mathbb{U} = (u_1, \ldots, u_n)$ is ordered.

The hyperparameters for our deep reinforcement learning algorithm are summarized in Table 4.1. The last column in Table 4.1 contains our hyperparameter choices for our experiments (i.e., context dependent values). In general, reinforcement learn-

ing is very sensitive to hyperparameter tuning [74]. We select hyperparameters based on hardware constraints and our goal to *efficiently* discover fast quantum gates, but we leave it to future work to optimize the hyperparameters through methods such as grid search and Bayesian updates.

Hyperparameter	**Symbol**	**Constraints**	**Choice for PoC**
Number of iterations	N_{iter}	–	200
Learning rate	α	–	0.002
Discount factor	β	–	0.98
Maximum allowable leakage	$\ell_{\max}$	–	5%
Number of shots	N_{shot}	≤ 10000	2000
Maximum gate time	t_g	–	35.6 ns
Gate length penalty	λ	–	0
Number of segments	N_{seg}	$\leq t_g/dt$	20
Number of episodes	N_{ep}	$N_{\text{ep}}/N_{\text{seg}} \in \mathbb{N}$	5
Number of hidden layers	–	–	1
Number of neurons in hidden layer	–	–	12
Activation functions	f, g	–	ReLU, Softmax
Size of first action space	$\|\mathbb{U}^x\|$	–	21
Size of second action space	$\|\mathbb{U}^y\|$	–	21

Table 4.1: A table of algorithm hyperparameters for our deep reinforcement learning algorithm for fast quantum gate design. Some hyperparameters are limited by hardware or other hyperparameter choices as shown in the **Constraints** column. The **Choice for PoC** column presents the values for hyperparameters used in proof-of-concept experiments on the IBM Lima quantum computer.

Chapter 5

Proof-of-concept for the X and $\sqrt{X}$ gates

We conduct a proof-of-concept by designing X and $\sqrt{X}$ gates of various durations on the IBM Lima quantum computer using the Qiskit Pulse library [56, 57]. In our proof-of-concept experiments, we optimize these gates only for state preparation from the ground state, but we explain how to take an average over different initial states to elevate the optimization beyond state transfer. In this chapter, we describe our experiments and analyze the results.

5.1 Fast quantum gates

Using the hyperparameters specified in Table 4.1, we run our algorithm to create fast quantum gates on IBM Lima. Once the training is complete, the agents can create gates of any number of segments from 1 to N_{seg} with the final network parameters $\theta_f^x = \theta^x_{N_{\text{iter}}-1}, \theta_f^y = \theta^y_{N_{\text{iter}}-1}$. To create a gate of length N_g segments, the agents sequentially select the amplitudes just as they did during training. However, there is no longer a need for training iterations or episodes. The agent records the states

s_k^x, s_k^y after the first k segments of the waveform for $k = 0, \ldots, N_g - 1$ and proposes the next amplitudes $c_{k+1}^x = u_k^x$ and $c_{k+1}^x = u_k^y$ according to the policies $\pi_{\theta_f}^x(\cdot|s_k^x), \pi_{\theta_f}^y(\cdot|s_k^y)$. The process for optimizing a gate is summarized in Algorithm 2.

Algorithm 2 Fast quantum gate design with a trained agent

Require: initial state s_0^x, leakage $\mathcal{L}_0$, and trained parameters θ_f^x, θ_f^y
1: **for** each segment $k = 0, ..., N_g - 1$ **do**
2: Select next action u_k^x according to policy $\pi_{\theta_f^x}(\cdot|s_k^x)$
3: Set $s_k^y = (u_k^x, \mathcal{L}_k)$
4: Select next action u_k^y according to policy $\pi_{\theta_f^y}(\cdot|s_k^y)$
5: Evolve qubit by first $k+1$ segments of waveform
6: Measure qubit N_{shot} times to get $(\langle I_{k+1}\rangle, \langle Q_{k+1}\rangle)$
7: Set $s_{k+1}^x = (\langle I_{k+1}\rangle, \langle Q_{k+1}\rangle, k+1)$
8: Estimate leakage population $\mathcal{L}_{k+1}$
9: Reset qubit to $|0\rangle$
10: **end for**
11: Return final pulse amplitudes $\left((u_0^x, u_0^y), \ldots, (u_{N_g-1}^x, u_{N_g-1}^y)\right)$

The policy is probabilistic and the state evolution is stochastic, so it is best to run the optimization several times and take the best result. A similar approach is used in [43], where the average gate over several optimizations is used. To assess the success of a gate U, we measure the leakage rate defined in eq. (4.2) and the state fidelity

$$\mathcal{F} = |\langle\psi_T| U |0\rangle|^2 \tag{5.1}$$

where $|\psi_T\rangle$ is the target state. The target state for an X gate is $|\psi_T\rangle = |1\rangle$ and for a $\sqrt{X}$ is

$$|\psi_T\rangle = \frac{|0\rangle - i\,|1\rangle}{\sqrt{2}}\,. \tag{5.2}$$

The state fidelity in eq. (5.1) is one measure of success; however, to completely design a gate, our agent should be trained on a diverse set of initial states to maximize the

gate fidelity. One definition of gate fidelity is

$$\mathcal{F}_G = \frac{1}{n(n+1)} \left[\mathrm{Tr}(MM^\dagger) + |\mathrm{Tr}(M)|^2 \right]. \tag{5.3}$$

where n is the size of the Hilbert space and $M = U_T^\dagger U$ is the product between the conjugate of the desired gate operation U_T and the actual gate operation U. The squared overlap between the actual outcome $U|\psi_0\rangle$ and the desired final state $|\psi_T\rangle = U_T|\psi_0\rangle$ averages to $\mathcal{F}_G$ when different initial states $|\psi_0\rangle$ are considered [75]. In this proof-of-concept, our experiments are limited to state preparation from the initial state $|0\rangle$ due to hardware availability, but our algorithm can be used on any initial state.

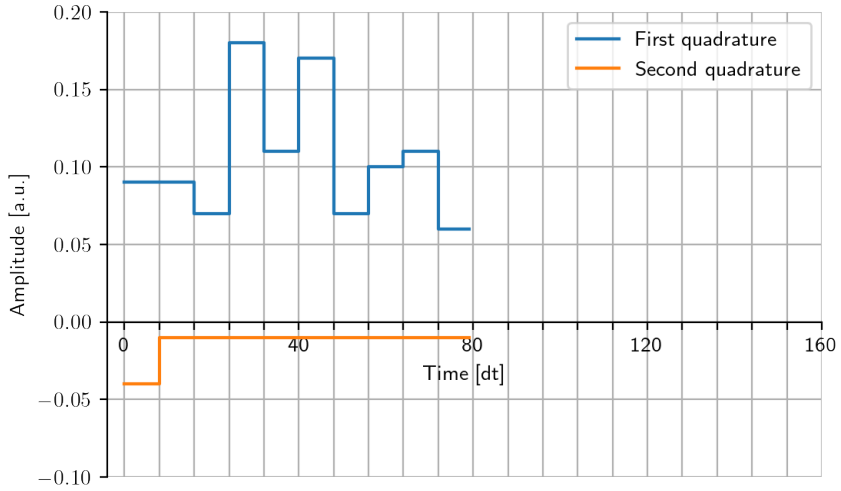
We test gates of length 20 segments ($t_g \approx 35.6$ ns) – the same length as the default gate –, 15 segments ($t_g \approx 26.7$ ns), and 10 segments ($t_g \approx 17.8$ ns). The waveforms for fast X and $\sqrt{X}$ gates are shown in Figures 5.1 and 5.3 respectively. The full-length optimized X and $\sqrt{X}$ gates alongside the default gates are depicted in Figures 5.2 and 5.4. After just 200 training iterations, the faster gates begin to match the performance of the default gate, with fidelities and leakage rates the same or only slightly worse. Our full-length optimized gates begin to achieve even slightly higher state fidelities and lower leakage rates than the calibrated DRAG pulses. The fidelities and leakage rates are summarized in Table 5.1. The statistics are gathered using 2000 shots and the margins of error are estimated by taking the standard deviation across five 2000 shot experiments.

Due to limited access to hardware, we do not train our agent beyond 200 iterations nor do we optimize the learning rate, neural network structure, number of segments, or other hyperparameters. Our algorithm has not fully converged after the first 200 training iterations. During training, we periodically perform the optimization procedure and test the performance of the resulting 100 proposed gates. As a measure

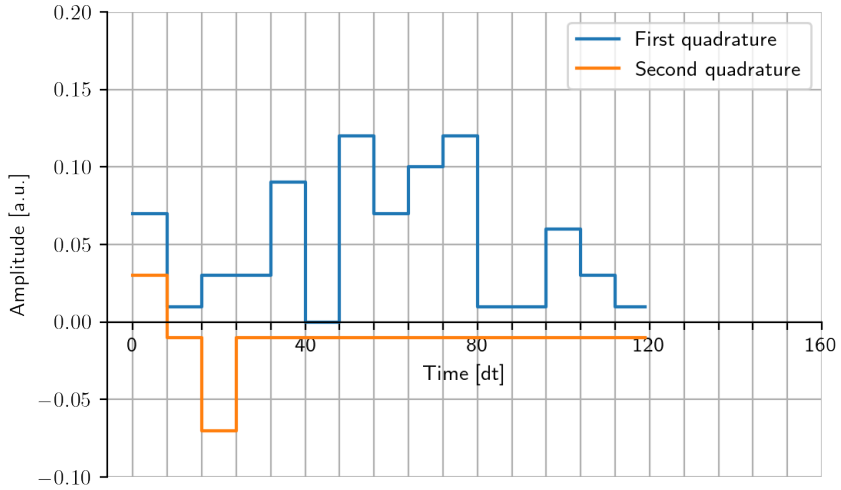
Gate	**Duration [ns]**	**Fidelity [%]** (± 0.6)	**Leakage [%]** (± 0.3)
Optimized X 2× faster	17.8	90.4	3.8
Optimized X 1.33× faster	26.7	92.2	4.3
Optimized X	35.6	92.3	3.5
Default X	35.6	91.7	4.2
Optimized $\sqrt{X}$ 2× faster	17.8	96.5	3.5
Optimized $\sqrt{X}$ 1.33× faster	2.67	96.8	2.9
Optimized $\sqrt{X}$	35.6	97.5	2.4
Default $\sqrt{X}$	35.6	96.8	2.5

Table 5.1: A table of the state fidelities (5.1) and leakage rates (4.2) for the optimized X and $\sqrt{X}$ gates of different lengths learned by our deep reinforcement learning algorithm and for the default gates.

of convergence, we check how many gates out of these 100 have fidelity greater than 90%. Figure 5.5 shows this statistic as the training proceeds. With more training iterations, we expect the quality of the optimized gates increase further before levelling off; although, convergence to an optimal solution is not guaranteed. We anticipate achieving more significant advantage by tuning the algorithm hyperparameters, including the number of training iterations.

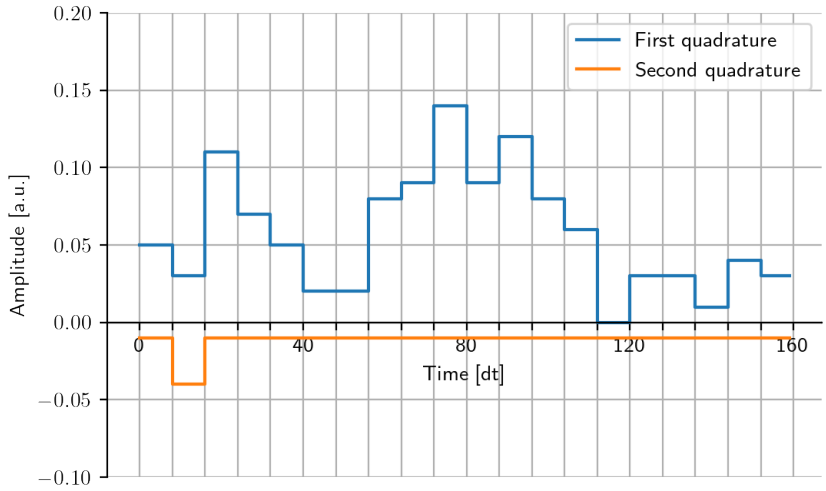


(a) Optimized X gate 2× faster than default time.

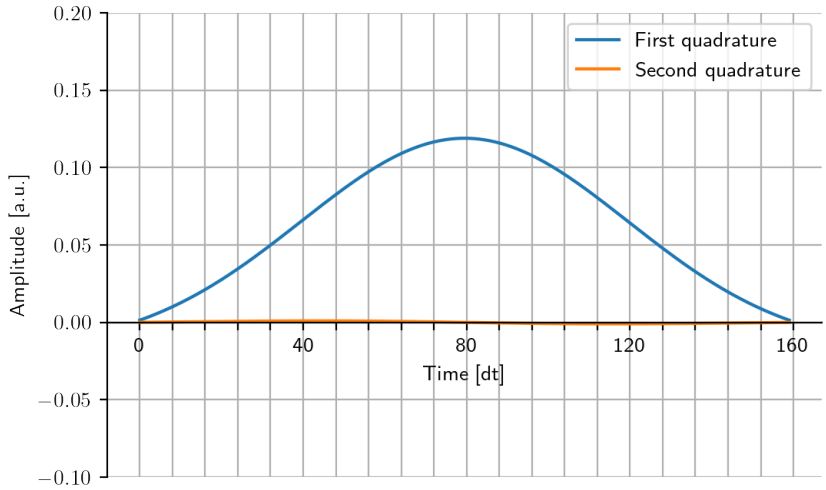


(b) Optimized X gate 1.33× faster than default time.

Figure 5.1: Plots showing optimized control pulses for fast single-qubit gates on the IBM Lima quantum computer. The time unit on the x-axis is in units of $dt = 0.222222$ ns, a device-dependent parameter specifying the maximum sampling rate of the waveform generator. Fig. (a) is a pulse for an X gate with duration 17.8 ns (2× faster than the default gate) learned by our deep reinforcement learning algorithm. Fig. (b) is a pulse for an X gate with duration 26.7 ns (1.33× faster than the default gate) learned by our deep reinforcement learning algorithm.

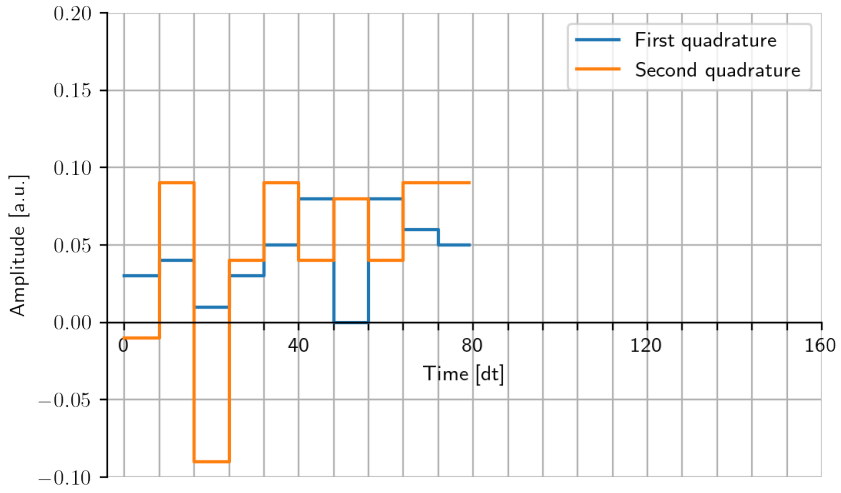


(a) Full-length optimized X gate.

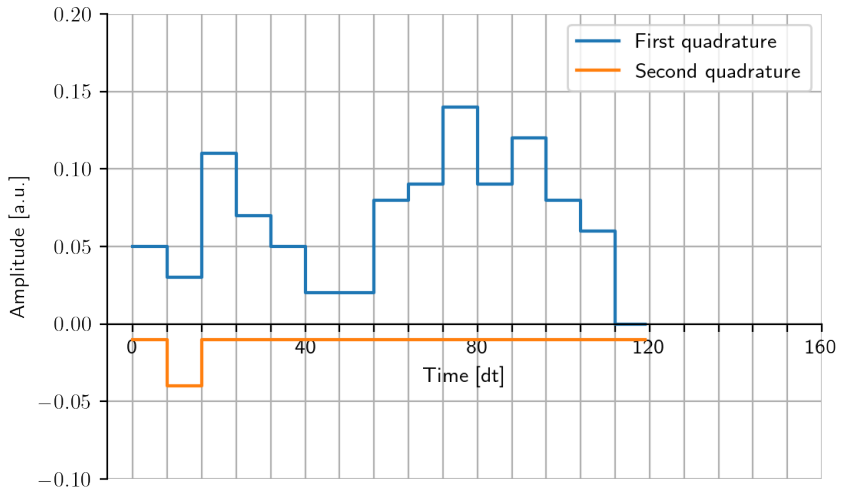


(b) Default X gate.

Figure 5.2: Plots showing optimized control and default pulses for standard length single-qubit gates on the IBM Lima quantum computer. The time unit on the x-axis is in units of $dt = 0.222222$ ns, a device-dependent parameter specifying the maximum sampling rate of the waveform generator. Fig. (a) is a pulse for an X gate with duration 35.6 ns learned by our deep reinforcement learning algorithm. Fig. (b) is a calibrated DRAG pulse for an X gate with duration 35.6 ns.

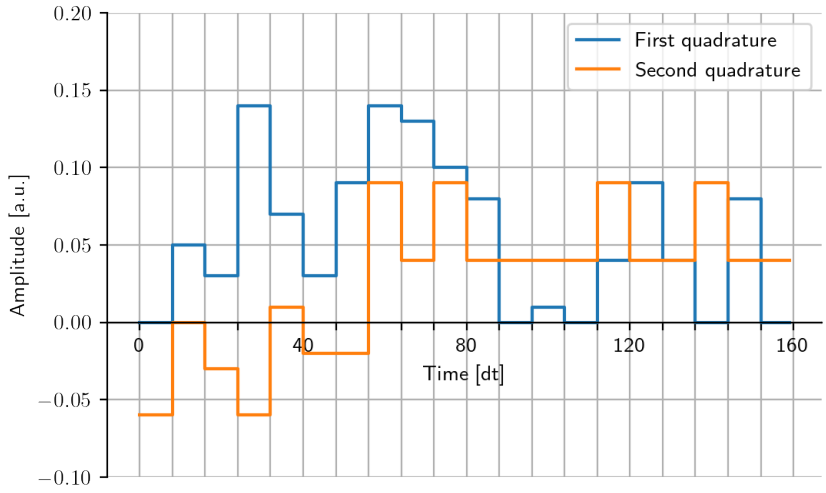


(a) Optimized $\sqrt{X}$ gate 2× faster than default time.

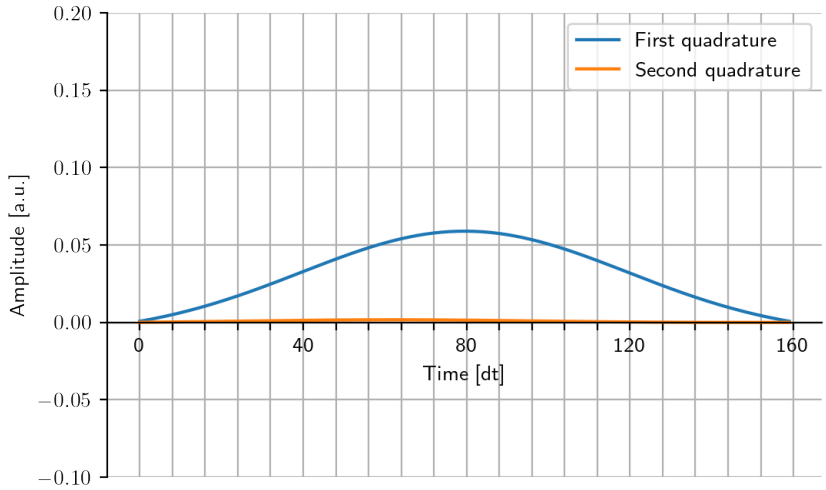


(b) Optimized $\sqrt{X}$ gate 1.33× faster than default time.

Figure 5.3: Plots showing optimized control pulses for fast single-qubit gates on the IBM Lima quantum computer. The time unit on the x-axis is in units of $dt = 0.222222$ ns, a device-dependent parameter specifying the maximum sampling rate of the waveform generator. Fig. (a) is a pulse for an $\sqrt{X}$ gate with duration 17.8 ns (2× faster than the default gate) learned by our deep reinforcement learning algorithm. Fig. (b) is a pulse for an $\sqrt{X}$ gate with duration 26.7 ns (1.33× faster than the default gate) learned by our deep reinforcement learning algorithm.



(a) Full-length optimized $\sqrt{X}$ gate.



(b) Default $\sqrt{X}$ gate.

Figure 5.4: Plots showing optimized control and default pulses for standard length single-qubit gates on the IBM Lima quantum computer. The time unit on the x-axis is in units of $dt = 0.222222$ ns, a device-dependent parameter specifying the maximum sampling rate of the waveform generator. Fig. (a) is a pulse for an $\sqrt{X}$ gate with duration 35.6 ns learned by our deep reinforcement learning algorithm. Fig. (b) is a calibrated DRAG pulse for an X gate with duration 35.6 ns .

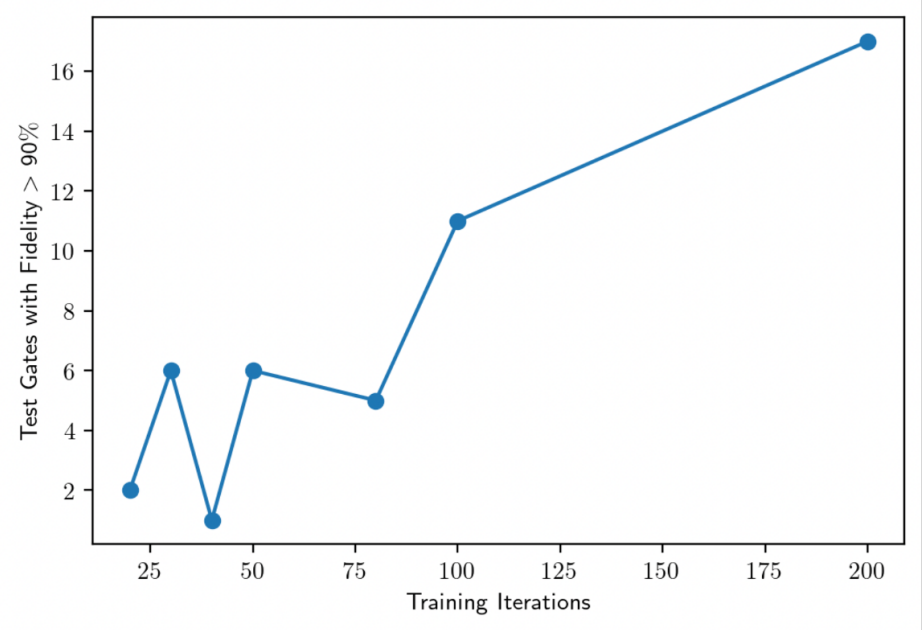


Figure 5.5: A plot of the number of optimized gates out of 100 that have fidelity greater than 90% as the number of training iterations increases. The gates were optimized by our deep reinforcement learning agent for fast quantum gate design.

5.2 Robustness over time

We also test the robustness of our agent and pulse over time. We perform the same optimization procedure described in the previous section using the trained agents a total of 30 days after finding the network parameters θ_f^x and θ_f^y. Figure 5.6 shows the new optimized X gate. The X gate calibrated by our agent has approximately the same fidelity $\mathcal{F} = 89.2 \pm 0.6\%$ but lower leakage $\mathcal{L} = 4.3 \pm 0.3\%$ than the default gate, whose statistics are $\mathcal{F} = 88.9 \pm 0.6\%$ and $\mathcal{L} = 5.3 \pm 0.3\%$. After an initial training period, our algorithm can be used to efficiently calibrate gates on superconducting qubits.

Although the agent is robust over time, the original optimized waveform is not. The previously optimized X gate is no longer effective, with a fidelity of just $74.2 \pm 0.6\%$ and leakage rate $6.4 \pm 0.3\%$. Therefore, our agent proposes a waveform that

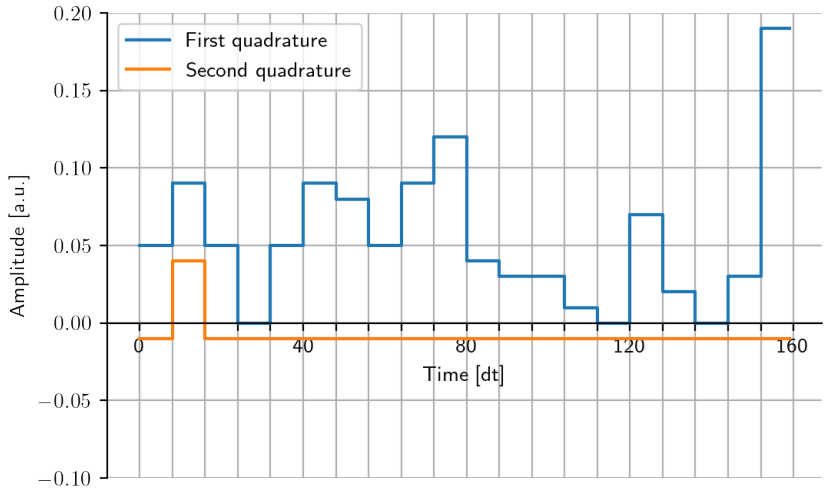


Figure 5.6: A plot showing an optimized control pulse for an X gate on the IBM Lima quantum computer. The pulse was learned by our deep reinforcement learning agent with no additional training 30 days after initially running the algorithm. The time unit on the x-axis is in units of $dt = 0.222222$ ns, a device-dependent parameter specifying the maximum sampling rate of the waveform generator.

adapts to current hardware conditions. It is known that qubit parameters such as resonance frequencies and decoherence rates change slowly on time scales of hours [76]. The fact that our agent can adapt to these changes without retraining is one of the strengths of our method. This should be contrasted to the reinforcement learning algorithm proposed in [43], that led to waveforms that are more robust over time.

Chapter 6

Generalization to two-qubit gates and other hardware platforms

The proof-of-concept in the previous chapter is for single-qubit gates on transmon qubits. To complete a universal gate set for quantum computing, we also require an entangling gate. In this chapter, we describe the extension of our deep reinforcement learning algorithm to two-qubit gates. As well, we provide some insights into how our algorithm could be used on other hardware platforms such as trapped-ion quantum computers.

6.1 Two-qubit gates

A small modification to the state space and reward function generalizes our algorithm to the design of two-qubit gates. On superconductcing transmon qubits, two-qubit interactions are generated via the cross-resonance gate which we described in Section 2.2.2. The industry standard CR pulse is a rounded square pulse with Gaussian rise and a derivative-pulse correction on the second quadrature [53]. To avoid spurious cross-talk, a simultaneous cancellation tone is applied to the target qubit. Baum et al. used RL to design an improved CR gate which did not require this extra pulse [43].

We take the same strategy as Baum et al., so our agent can learn an improved CR pulse shape using separate action spaces $\mathbb{U}^x$ and $\mathbb{U}^y$ as in the single-qubit case. The state $\tilde{s}_j^x = (\langle I_j^1 \rangle, \langle Q_j^1 \rangle, \langle I_j^2 \rangle, \langle Q_j^2 \rangle, k)$ contains the average readout signal from each qubit. We use the reward

$$\tilde{r}_j^x(\tilde{s}_{j+1}^x, c_T^1, c_T^2) = \frac{1}{2}\left(r_j^x(\langle I_{j+1}^1 \rangle, \langle Q_{j+1}^1 \rangle, k, c_T^1) + r_j^x(\langle I_{j+1}^2 \rangle, \langle Q_{j+1}^2 \rangle, k, c_T^2)\right) \tag{6.1}$$

where r_j^x is defined in (4.3) and c_T^1, c_T^2 are the expected measurement locations for the first and second qubit respectively. For the second control quadrature, the leakage population is summed over both qubits to give the same state and reward as the single-qubit case. The CR gate completes a universal gateset $\left\{CR, R_Z(\theta), X, \sqrt{X}\right\}$ when combined with arbitrary virtual Z rotations [50] and the single-qubit gates shown in our proof-of-concept.

6.2 Other quantum hardware platforms

Our proof-of-concept was carried out on superconducting qubits; however, the proposed reinforcement learning algorithm is general and can be used on other quantum hardware with some adjustments. Besides superconducting qubits, leading candidates for quantum information processing technologies include: ion traps [44], quantum dots [77], and neutral atoms [78]. Just like superconducting circuits, gate operations on these proposed quantum systems are subject to errors including decoherence and leakage out of the qubit subspace. To take advantage of the features of our algorithm to combat these competing effects, any adaptation should have the following properties:

1. a control architecture that permits PWC pulses on independent quadratures, and

2. training directly on the readout signal rather than classifying the qubit state.

Other aspects of the algorithm such as the action space, state space, and hyperparameters are flexible.

Let us study ion traps as an example. In trapped ion quantum computing, ions are confined using electromagnetic fields. In this paradigm, a qubit is represented by the stable electronic states of an ion and gate operations are implemented via laser pulses. The laser pulse can be represented by a PWC pulse with two independent quadratures at a specific drive frequency, as desired. The amplitudes that make up the action spaces $\mathbb{U}^x, \mathbb{U}^y$ could be different than the superconducting case depending on the laser generator settings. The qubit is measured by detecting photons emitted when the ion decays into its ground state (a phenomena called fluorescence). The state of the qubit is inferred from the photon counts, similar to how the transmon qubit state is estimated from the (I, Q) signal, but with higher accuracy. Let $\langle n_j \rangle$ be the average photon count over N_{shot} measurements in the j-th time step. The first agent could use the state $s_j^x = (\langle n_j \rangle, k)$ as an input to the neural network. Two-qubit gates are engineered by coupling the state of individual qubits to the collective vibrational mode of a linear array of qubits [79, 80]. The adaptation of our algorithm to two-qubit gates on ion traps would involve expanding the state space to include both qubits (same as the superconducting case) and possibly the joint state of the array as well. The typical entangling gate is the Cirac-Zoller CX gate [79]. This is a sequence of laser Ramsey pulses [81] rather than a square cross-resonance pulse, but still involves applying a pulse that is resonant on only one qubit at a time.

By changing the state and action spaces and hyperparameters to fit the hardware specifications, our deep reinforcement learning algorithm can be used to design fast quantum gates while minimizing leakage on other types of quantum computers.

Chapter 7

Conclusions and future work

7.1 Conclusion

We have created a deep reinforcement learning algorithm for fast quantum gate design using real-time feedback based on the readout signal from noisy hardware. Reinforcement learning is model free, as opposed to other gate synthesis strategies which require precise models that cannot capture the stochastic dynamics of the qubit. Our dual-agent architecture allows us to target competing goals of decreasing leakage and creating faster gates to reduce decoherence. Our proposed algorithm reduces the impact of measurement errors by training directly on the readout signal from the superconducting resonator coupled to the qubit. The low measurement overhead means that our algorithm can be trained on hardware, rather than a numerical simulation.

We carried out a proof-of-concept with X and $\sqrt{X}$ gates of different durations on IBM superconducting quantum computers. Our agent proposed novel control pulses which are two times faster than default gates and match their performance in terms of state fidelity and leakage after just 200 training iterations. Our agent also created gates of the same duration as the default which offer slight improvements in state fidelity and leakage. The proof-of-concept focuses on state preparation from

the ground state but we explain how our agent can take an average over different initial states to improve overall gate fidelity. Our agent has not converged after 200 training iterations, but we still obtain useful results. We also showed that our trained agent is robust over time. So far, we had limited access to hardware and only ran the algorithm with one specific set of hyperparameters. As is well known, the performance of reinforcement learning is greatly improved with fine tuning of hyperparameters [74]. We expect optimization of the algorithm hyperparameters to lead to much more significant advantage.

The improved gate operations created by our deep reinforcement algorithm for fast quantum gate design open the way for more extensive applications on near term and future quantum devices with reduced decoherence and leakage which cannot be fixed by most error correction algorithms.

7.2 Future work

There is a need for more evaluation and experimentation to reveal the full capabilities of our deep reinforcement learning algorithm for fast quantum gate design. A first step for future research will be conducting a proof-of-concept experiment to optimize two-qubit gates. More testing will be required for all gates (single- and two-qubit) that make up a universal gateset. This will include gathering other performance metrics such as the error-per-gate estimated with interleaved randomized benchmarking [82] as done in [43]. As well, the robustness of both the optimized gates and the trained agent can be tested at different time scales (not just 30 days) and across other qubits and quantum computers.

These experiments will be enabled by increased access to hardware, in terms of both more time on quantum processors and the ability to use custom control

hardware. Our algorithm can be implemented on an FPGA such as the one used for state preparation experiments in [42]. This implementation can be used to improve the convergence time of our algorithm. Our proof-of-concept experiments involved wait times in a queue to access IBM Lima and other delays from communicating data, so we could not reliably measure convergence time.

With these metrics (fidelity, leakage rate, error-per-gate, robustness over time, robustness across qubits and quantum computers, and convergence time) in mind, the algorithm hyperparamters can be optimized. We emphasize again that hyperparameter tuning is vital to improving the performance of reinforcement learning [74]. This involves running the algorithm repeatedly while making minor adjustments to the hyperparameters. The FPGA implementation will ensure that each iteration takes less than 10 minutes wall-clock time, as was seen in [42], so it will be possible to test many hyperparameter configurations.

We leave these considerations and experiments to future work.

Bibliography

[1] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, 2018. [Online]. Available: https://doi.org/10.22331/q-2018-08-06-79

[2] D. Gottesman, "An introduction to quantum error correction and fault-tolerant quantum computation," *arXiv*, 2009. [Online]. Available: https://arxiv.org/abs/0904.2557

[3] M. McEwen, D. Kafri, Z. Chen, J. Atalaya, K. J. Satzinger, C. Quintana, P. V. Klimov, D. Sank, C. Gidney, A. G. Fowler, F. Arute, K. Arya, B. Buckley, B. Burkett, N. Bushnell, B. Chiaro, R. Collins, S. Demura, A. Dunsworth, C. Erickson, B. Foxen, M. Giustina, T. Huang, S. Hong, E. Jeffrey, S. Kim, K. Kechedzhi, F. Kostritsa, P. Laptev, A. Megrant, X. Mi, J. Mutus, O. Naaman, M. Neeley, C. Neill, M. Niu, A. Paler, N. Redd, P. Roushan, T. C. White, J. Yao, P. Yeh, A. Zalcman, Y. Chen, V. N. Smelyanskiy, J. M. Martinis, H. Neven, J. Kelly, A. N. Korotkov, A. G. Petukhov, and R. Barends, "Removing leakage-induced correlated errors in superconducting quantum error correction," *Nature Communications*, vol. 12, no. 1, p. 1761, 2021. [Online]. Available: https://www.nature.com/articles/s41467-021-21982-y

[4] C. J. Wood and J. M. Gambetta, "Quantification and characterization of leakage errors," *Physical Review A*, vol. 97, p. 032306, 2018. [Online]. Available:

https://link.aps.org/doi/10.1103/PhysRevA.97.032306

[5] M. Suchara, A. W. Cross, and J. M. Gambetta, "Leakage suppression in the toric code," *arXiv*, 2014. [Online]. Available: https://arxiv.org/abs/1410.8562

[6] A. G. Fowler, "Coping with qubit leakage in topological codes," *Physical Review A*, vol. 88, p. 042308, 2013. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.88.042308

[7] N. Khaneja, T. Reiss, C. Kehlet, T. Schulte-Herbrüggen, and S. J. Glaser, "Optimal control of coupled spin dynamics: design of nmr pulse sequences by gradient ascent algorithms," *Journal of Magnetic Resonance*, vol. 172, no. 2, pp. 296–305, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1090780704003696

[8] F. Motzoi, J. M. Gambetta, P. Rebentrost, and F. K. Wilhelm, "Simple pulses for elimination of leakage in weakly nonlinear qubits," *Physical Review Letters*, vol. 103, p. 110501, 2009. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.103.110501

[9] N. Khaneja, R. Brockett, and S. J. Glaser, "Time optimal control in spin systems," *Physical Review A*, vol. 63, p. 032308, 2001. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.63.032308

[10] U. Boscain, G. Charlot, J.-P. Gauthier, S. Guérin, and H.-R. Jauslin, "Optimal Control in laser-induced population transfer for two- or three-level quantum systems," *Journal of Mathematical Physics*, vol. 43, no. 5, pp. 2107–2132, 2002. [Online]. Available: https://hal.science/hal-00383019

[11] A. Pechen, N. Ilín, F. Shuang, and H. Rabitz, "Quantum control by von neumann measurements," *Physical Review A*, vol. 74, p. 052102, 2006. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.74.052102

[12] S. Montangero, T. Calarco, and R. Fazio, "Robust optimal quantum gates for josephson charge qubits," *Physical Review Letters*, vol. 99, no. 17, 2007. [Online]. Available: https://doi.org/10.1103%2Fphysrevlett.99.170501

[13] V. F. Krotov, *Global Methods in Optimal Control Theory.* Boston, MA: Birkhäuser Boston, 1993, pp. 74–121.

[14] N. B. Dehaghani and F. L. Pereira, "High fidelity quantum state transfer by pontryagin maximum principle," *arXiv*, 2022. [Online]. Available: https://arxiv.org/abs/2203.04361

[15] W. Zhu and H. Rabitz, "A rapid monotonically convergent iteration algorithm for quantum optimal control over the expectation value of a positive definite operator," *The Journal of Chemical Physics*, vol. 109, no. 2, pp. 385–391, 1998. [Online]. Available: https://pubs.aip.org/aip/jcp/article/109/2/385/476865

[16] D. D'Alessandro and M. Dahleh, "Optimal control of two-level quantum systems," *IEEE Transactions on Automatic Control*, vol. 46, no. 6, pp. 866–876, 2001. [Online]. Available: https://ieeexplore.ieee.org/document/928587

[17] S. Machnes, E. Assémat, D. Tannor, and F. K. Wilhelm, "Tunable, flexible, and efficient optimization of control pulses for practical qubits," *Physical Review Letters*, vol. 120, p. 150401, 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.120.150401

[18] P. Doria, T. Calarco, and S. Montangero, "Optimal control technique for many-body quantum dynamics," *Physical Review Letters*, vol. 106, no. 19, 2011. [Online]. Available: https://doi.org/10.1103%2Fphysrevlett.106.190501

[19] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary computation*, vol. 4, no. 1, pp. 1–32, 1996.

[20] J. Kelly, R. Barends, B. Campbell, Y. Chen, Z. Chen, B. Chiaro, A. Dunsworth, A. Fowler, I.-C. Hoi, E. Jeffrey, A. Megrant, J. Mutus, C. Neill, P. O'Malley, C. Quintana, P. Roushan, D. Sank, A. Vainsencher, J. Wenner, T. White, A. Cleland, and J. M. Martinis, "Optimal quantum control using randomized benchmarking," *Physical Review Letters*, vol. 112, no. 24, 2014. [Online]. Available: https://doi.org/10.1103%2Fphysrevlett.112.240504

[21] J. N. Tsitsiklis, "Asynchronous stochastic approximation and q-learning," *Machine Learning*, vol. 16, no. 3, pp. 185–202, 1994. [Online]. Available: https://link.springer.com/article/10.1023/A:1022689125041

[22] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine Learning*, vol. 8, no. 3, pp. 279–292, 1992. [Online]. Available: https://link.springer.com/article/10.1007/BF00992698

[23] C. Szepesvari and M. L. Littman, "A unified analysis of value-function-based reinforcement-learning algorithms," *Neural Computation*, vol. 11, pp. 2017–2060, 1999. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/10578043/

[24] F. Metz and M. Bukov, "Self-correcting quantum many-body control using reinforcement learning with tensor networks," *arXiv*, 2022. [Online]. Available: https://arxiv.org/abs/2201.11790

[25] Y. Qiu, M. Zhuang, J. Huang, and C. Lee, "Efficient and robust entanglement generation with deep reinforcement learning for quantum metrology," *New Journal of Physics*, vol. 24, no. 8, p. 083011, 2022. [Online]. Available: https://doi.org/10.1088%2F1367-2630%2Fac8285

[26] V. V. Sivak, A. Eickbusch, H. Liu, B. Royer, I. Tsioutsios, and M. H. Devoret, "Model-free quantum control with reinforcement learning," *Physical Review X*, vol. 12, p. 011059, 2022. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.12.011059

[27] C. Jiang, Y. Pan, Z.-G. Wu, Q. Gao, and D. Dong, "Robust optimization for quantum reinforcement learning control using partial observations," *Physical Review A*, vol. 105, p. 062443, 2022. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.105.062443

[28] P. Peng, X. Huang, C. Yin, L. Joseph, C. Ramanathan, and P. Cappellaro, "Deep reinforcement learning for quantum hamiltonian engineering," *arXiv*, 2021. [Online]. Available: https://arxiv.org/abs/2102.13161

[29] M.-Z. Ai, Y. Ding, Y. Ban, J. D. Martín-Guerrero, J. Casanova, J.-M. Cui, Y.-F. Huang, X. Chen, C.-F. Li, and G.-C. Guo, "Experimentally realizing efficient quantum control with reinforcement learning," *arXiv*, 2021. [Online]. Available: https://arxiv.org/abs/2101.09020

[30] S.-F. Guo, F. Chen, Q. Liu, M. Xue, J.-J. Chen, J.-H. Cao, T.-W. Mao, M. K. Tey, and L. You, "Faster state preparation across quantum phase transition assisted by reinforcement learning," *Physical Review Letters*, vol. 126, no. 6, 2021. [Online]. Available: https://doi.org/10.1103%2Fphysrevlett.126.060401

[31] Z. An, H.-J. Song, Q.-K. He, and D. L. Zhou, "Quantum optimal control of multilevel dissipative quantum systems with reinforcement learning," *Physical Review A*, vol. 103, p. 012404, 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.103.012404

[32] I. Khalid, C. A. Weidner, E. A. Jonckheere, S. G. Schirmer, and F. C. Langbein, "Reinforcement learning vs. gradient-based optimisation for robust energy landscape control of spin-1/2 quantum networks," in *2021 60th IEEE Conference on Decision and Control (CDC)*. IEEE, 2021. [Online]. Available: https://doi.org/10.1109%2Fcdc45484.2021.9683463

[33] S. Daraeizadeh, S. P. Premaratne, and A. Y. Matsuura, "Designing high-fidelity multi-qubit gates for semiconductor quantum dots through deep reinforcement learning," in *2020 IEEE International Conference on Quantum Computing and Engineering (QCE)*. IEEE, 2020. [Online]. Available: https://doi.org/10.1109%2Fqce49297.2020.00014

[34] H. Ma, D. Dong, S. X. Ding, and C. Chen, "Curriculum-based deep reinforcement learning for quantum control," *arXiv*, 2020. [Online]. Available: https://arxiv.org/abs/2012.15427

[35] J.-J. Chen and M. Xue, "Manipulation of spin dynamics by deep reinforcement learning agent," *arXiv*, 2019. [Online]. Available: https://arxiv.org/abs/1901.08748

[36] R. Porotti, D. Tamascelli, M. Restelli, and E. Prati, "Coherent transport of quantum states by deep reinforcement learning," *Communications Physics*, vol. 2, no. 1, 2019. [Online]. Available: https://doi.org/10.1038%2Fs42005-019-0169-x

[37] X.-M. Zhang, Z. Wei, R. Asad, X.-C. Yang, and X. Wang, "When does reinforcement learning stand out in quantum control? a comparative study on state preparation," *NPJ Quantum Information*, vol. 5, 2019. [Online]. Available: https://www.nature.com/articles/s41534-019-0201-8#citeas

[38] Z. An and D. L. Zhou, "Deep reinforcement learning for quantum gate control," *Europhysics Letters*, vol. 126, no. 6, p. 60002, 2019. [Online]. Available: https://doi.org/10.1209%2F0295-5075%2F126%2F60002

[39] M. Bukov, "Reinforcement learning for autonomous preparation of floquet-engineered states: Inverting the quantum kapitza oscillator," *Physical Review B*, vol. 98, no. 22, 2018. [Online]. Available: https://doi.org/10.1103%2Fphysrevb.98.224305

[40] M. Bukov, A. G. R. Day, D. Sels, P. Weinberg, A. Polkovnikov, and P. Mehta, "Reinforcement learning in different phases of quantum control," *Physical Review X*, vol. 8, p. 031086, 2018. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevX.8.031086

[41] C. Chen, D. Dong, H.-X. Li, J. Chu, and T.-J. Tarn, "Fidelity-based probabilistic q-learning for control of quantum systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 5, pp. 920–933, 2014. [Online]. Available: https://doi.org/10.1109%2Ftnnls.2013.2283574

[42] K. Reuer, J. Landgraf, T. Fösel, J. O'Sullivan, L. Beltrán, A. Akin, G. J. Norris, A. Remm, M. Kerschbaum, J.-C. Besse, F. Marquardt, A. Wallraff, and C. Eichler, "Realizing a deep reinforcement learning agent discovering real-time feedback control strategies for a quantum system," *arXiv*, 2022. [Online]. Available: https://arxiv.org/abs/2210.16715

[43] Y. Baum, M. Amico, S. Howell, M. Hush, M. Liuzzi, P. Mundada, T. Merkh, A. R. Carvalho, and M. J. Biercuk, "Experimental deep reinforcement learning for error-robust gate-set design on a superconducting quantum computer," *Physical Review X Quantum*, vol. 2, no. 4, 2021. [Online]. Available: https://doi.org/10.1103%2Fprxquantum.2.040324

[44] C. D. Bruzewicz, J. Chiaverini, R. McConnell, and J. M. Sage, "Trapped-ion quantum computing: Progress and challenges," *Applied Physics Reviews*, vol. 6, no. 2, p. 021314, 2019. [Online]. Available: https://doi.org/10.1063%2F1.5088164

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

[46] 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*, vol. 6, no. 2, p. 021318, 2019. [Online]. Available: https://doi.org/10.1063%2F1.5089550

[47] B. Josephson, "Possible new effects in superconductive tunnelling," *Physics Letters*, vol. 1, no. 7, pp. 251–253, 1962. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/0031916362913690

[48] B. D. Josephson, "Coupled superconductors," *Reviews of Modern Physics*, vol. 36, pp. 216–220, 1964. [Online]. Available: https://journals.aps.org/rmp/abstract/10.1103/RevModPhys.36.216

[49] J. M. Chow, L. DiCarlo, J. M. Gambetta, F. Motzoi, L. Frunzio, S. M. Girvin, and R. J. Schoelkopf, "Optimized driving of superconducting artificial atoms for improved single-qubit gates," *Physical Review A*, vol. 82, no. 4, 2010. [Online]. Available: https://doi.org/10.1103%2Fphysreva.82.040305

[50] D. C. McKay, C. J. Wood, S. Sheldon, J. M. Chow, and J. M. Gambetta, "Efficient z gates for quantum computing," *Physical Review A*, vol. 96, p. 022330, 2017. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.96.022330

[51] Forschungszentrum Juelich Institute for Advanced Simulation, "Quantum information processing lecture notes," in *44 IFF Spring school 2013*, D. DiVincenzo, Ed., Germany, 2013, pp. B4.1–B4.44. [Online]. Available: http://inis.iaea.org/search/search.aspx?orig_q=RN:46123072

[52] G. S. Paraoanu, "Microwave-induced coupling of superconducting qubits," *Physical Review B*, vol. 74, p. 140504, 2006. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevB.74.140504

[53] S. Sheldon, E. Magesan, J. M. Chow, and J. M. Gambetta, "Procedure for systematically tuning up cross-talk in the cross-resonance gate," *Physical Review A*, vol. 93, p. 060302, 2016. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevA.93.060302

[54] J. Gambetta, A. Blais, D. I. Schuster, A. Wallraff, L. Frunzio, J. Majer, M. H. Devoret, S. M. Girvin, and R. J. Schoelkopf, "Qubit-photon interactions in a cavity: Measurement-induced dephasing and number splitting," *Physical Review A*, vol. 74, no. 4, 2006. [Online]. Available: https://doi.org/10.1103%2Fphysreva.74.042318

[55] E. Magesan, J. M. Gambetta, A. D. Córcoles, and J. M. Chow, "Machine learning for discriminating quantum measurement trajectories and improving readout," *Physical Review Letters*, vol. 114, p. 200501, 2015. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.114.200501

[56] Qiskit contributors, "Qiskit: An open-source framework for quantum computing," 2023. [Online]. Available: https://qiskit.org/

[57] D. C. McKay, T. Alexander, L. Bello, M. J. Biercuk, L. Bishop, J. Chen, J. M. Chow, A. D. Córcoles, D. Egger, S. Filipp, J. Gomez, M. Hush, A. Javadi-Abhari, D. Moreda, P. Nation, B. Paulovicks, E. Winston, C. J. Wood, J. Wootton, and J. M. Gambetta, "Qiskit backend specifications for openqasm and openpulse experiments," *arXiv*, 2018. [Online]. Available: https://arxiv.org/abs/1809.03452

[58] D. D'Alessandro, *Introduction to Quantum Control and Dynamics*. Chapman and Hall, 2007.

[59] R. Gray and D. Neuhoff, "Quantization," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2325–2383, 1998.

[60] A. Gadjiev and A. Ghorbanalizadeh, "Approximation of analytical functions by sequences of k-positive linear operators," *Journal of Approximation Theory*, vol. 162, no. 6, pp. 1245–1255, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021904510000249

[61] D. Bertsekas and J. N. Tsitsiklis, *Neuro-dynamic programming*. Athena Scientific, 1996.

[62] Y. Li, "Deep reinforcement learning: An overview," *arXiv*, 2018. [Online]. Available: https://arxiv.org/abs/1701.07274

[63] I. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application," *Journal of Microbiological Methods*, vol. 43, no. 1, pp. 3–31, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0167701200002013

[64] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, 1992. [Online]. Available: https://link.springer.com/article/10.1007/BF00992696

[65] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986. [Online]. Available: https://www.nature.com/articles/323533a0

[66] O. Shindi, Q. Yu, P. Girdhar, and D. Dong, "Model-free quantum gate design and calibration using deep reinforcement learning," *arXiv*, 2023. [Online]. Available: https://arxiv.org/abs/2302.02371

[67] E.-J. Kuo, Y.-L. L. Fang, and S. Y.-C. Chen, "Quantum architecture search via deep reinforcement learning," *arXiv*, 2021. [Online]. Available: https://arxiv.org/abs/2104.07715

[68] M. M. Wauters, E. Panizon, G. B. Mbeng, and G. E. Santoro, "Reinforcement-learning-assisted quantum optimization," *Physical Review Research*, vol. 2, p. 033446, 2020. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevResearch.2.033446

[69] A. García-Sáez and J. Riu, "Quantum observables for continuous control of the quantum approximate optimization algorithm via reinforcement learning," *arXiv*, vol. abs/1911.09682, 2019. [Online]. Available: https://arxiv.org/abs/1911.09682

[70] S. Borah, B. Sarma, M. Kewming, G. J. Milburn, and J. Twamley, "Measurement-based feedback quantum control with deep reinforcement learning for a double-well nonlinear potential," *Physical Review Letters*, vol.

127, p. 190403, 2021. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.127.190403

[71] M. Y. Niu, S. Boixo, V. N. Smelyanskiy, and H. Neven, "Universal quantum control through deep reinforcement learning," *NPJ Quantum Information*, vol. 5, p. 33, 2019. [Online]. Available: https://www.nature.com/articles/s41534-019-0141-3#Bib1

[72] M. Werninghaus, D. J. Egger, F. Roy, S. Machnes, F. K. Wilhelm, and S. Filipp, "Leakage reduction in fast superconducting qubit gates via optimal control," *NPJ Quantum Information*, vol. 7, no. 1, 2021. [Online]. Available: https://doi.org/10.1038%2Fs41534-020-00346-2

[73] J. J. Wallman, M. Barnhill, and J. Emerson, "Robust characterization of leakage errors," *New Journal of Physics*, vol. 18, no. 4, p. 043021, 2016. [Online]. Available: https://doi.org/10.1088%2F1367-2630%2F18%2F4%2F043021

[74] M. Feurer and F. Hutter, *Hyperparameter Optimization*. Cham: Springer International Publishing, 2019, pp. 3–33.

[75] L. H. Pedersen, N. M. Møller, and K. Mølmer, "Fidelity of quantum operations," *Physics Letters A*, vol. 367, no. 1-2, pp. 47–51, 2007. [Online]. Available: https://doi.org/10.1016%2Fj.physleta.2007.02.069

[76] S. Schlör, J. Lisenfeld, C. Müller, A. Bilmes, A. Schneider, D. P. Pappas, A. V. Ustinov, and M. Weides, "Correlating decoherence in transmon qubits: Low frequency noise by single fluctuators," *Physical Review Letters*, vol. 123, p. 190502, 2019. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.123.190502

[77] L. M. K. Vandersypen and M. A. Eriksson, "Quantum computing with semiconductor spins," *Physics Today*, vol. 72, no. 8, pp. 38–45, 2019. [Online]. Available: https://doi.org/10.1063/PT.3.4270

[78] L. Henriet, L. Beguin, A. Signoles, T. Lahaye, A. Browaeys, G.-O. Reymond, and C. Jurczak, "Quantum computing with neutral atoms," *Quantum*, vol. 4, p. 327, 2020. [Online]. Available: https://doi.org/10.22331%2Fq-2020-09-21-327

[79] J. I. Cirac and P. Zoller, "Quantum computations with cold trapped ions," *Physical Review Letters*, vol. 74, pp. 4091–4094, 1995. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevLett.74.4091

[80] F. Schmidt-Kaler, H. Häffner, M. Riebe, S. Gulde, G. P. T. Lancaster, T. Deuschle, C. Becher, C. F. Roos, J. Eschner, and R. Blatt, "Realization of the cirac–zoller controlled-not quantum gate," *Nature*, vol. 422, no. 6930, pp. 408–411, 2003. [Online]. Available: https://www.nature.com/articles/nature01494

[81] N. F. Ramsey, "A molecular beam resonance method with separated oscillating fields," *Physical Review Letters*, vol. 78, pp. 695–699, 1950. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRev.78.695

[82] E. Magesan, J. M. Gambetta, B. R. Johnson, C. A. Ryan, J. M. Chow, S. T. Merkel, M. P. da Silva, G. A. Keefe, M. B. Rothwell, T. A. Ohki, M. B. Ketchen, and M. Steffen, "Efficient measurement of quantum gate error by interleaved randomized benchmarking," *Physical Review Letters*, vol. 109, no. 8, 2012. [Online]. Available: https://doi.org/10.1103%2Fphysrevlett.109.080505