Fast and Efficient Creation of an Universal Quantum Gate Set Using Reinforcement Learning Methods

Pablo Darío Conte

M.Sc. Degree Candidate in Quantum Computing and Quantum Technologies

Faculty Advisor: **Emmanuel Paspalakis**

Members of the Examination Committee: **Ioannis Thanopulos** and **Dionisis Stefanatos**

Department of Electrical and Computer Engineering

Democritus University of Thrace (DECE)

Institute of Nanoscience and Nanotechnology

National Center of Scientific Research "Demokritos" (INN-D)

March 2025

DEMOCRITUS UNIVERSITY OF THRACE | DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

MSc in QUANTUM COMPUTING AND QUANTUM TECHNOLOGIES

NATIONAL CENTRE FOR SCIENTIFIC RESEARCH "DEMOKRITOS"

Contents

List of Figures **vi**

List of Tables **ix**

1 Introduction **2**

1.1 Overview of Quantum Computing 2

1.2 The Role of Universal Gate Sets in Quantum Control 2

1.3 Reinforcement Learning in Quantum Control 3

1.4 Objectives of the Thesis 3

1.5 Contributions of the Work 4

1.6 Structure of the Thesis 4

2 Fundamentals of Quantum Control and Reinforcement Learning **6**

2.1 Basics of Quantum Mechanics and Quantum Gates 6

2.2 Quantum Control: Concepts and challenges 7

2.2.1 Challenges in Quantum Control 7

2.3 Overview of Reinforcement Learning in Quantum Systems 7

2.3.1 Key Components of RL 7

2.4 The Role of Markov Decision Processes (MDPs) in Quantum Control 8

3 Universal Quantum Gate Set: Theoretical Foundations **10**

3.1 The H, T and CNOT Gates 10

3.1.1 Hadamard (H) Gate . . . 10
3.1.2 T Gate . . . 11
3.1.3 Controlled-NOT (CNOT) Gate . . . 11
3.2 Universality and Minimality in Gate Sets . . . 11
3.3 Control Optimization Challenges . . . 12
3.4 Practical Applications of the H-T-CNOT Set . . . 12
3.5 Time Evolution Implementations . . . 12
3.5.1 Hadamard Gate (H) . . . 13
3.5.2 T Gate . . . 14
3.5.3 Controlled-NOT (CNOT) Gate . . . 15

4 Reinforcement Learning for Quantum Gate Control Optimization 18
4.1 Quantum Control as an MDP . . . 18
4.2 State, Action, and Reward Spaces in Quantum Gate Design . . . 19
4.2.1 State Space . . . 19
4.2.2 Action Space . . . 19
4.2.3 Reward Function . . . 20
4.3 Algorithms for Quantum Control: A Comparative Framework . . . 21
4.3.1 Monte Carlo Methods . . . 21
4.3.2 Temporal Difference (TD) Methods . . . 21
4.3.3 Policy Gradient Methods . . . 21
4.3.4 Actor-Critic Architectures . . . 22
4.3.5 Actor Architectures . . . 22
4.4 Exploration vs. Exploitation in Quantum Systems . . . 22
4.5 General Approach . . . 22

5 Reinforcement Learning Agents: An Overview 24
5.1 Monte Carlo Methods for Quantum Control . . . 24

5.2 Temporal Difference (TD) Methods for Policy Optimization 25
5.2.1 Q-Learning 25
5.2.2 Sarsa 25
5.2.3 Deep Q-Networks (DQN) 26
5.2.4 Double Deep Q-Networks (DDQN) 26
5.2.5 Double Dueling Deep Q-Network (DDDQN) 26
5.3 Policy Gradient Methods: Direct Policy Search 27
5.3.1 REINFORCE Algorithm 27
5.3.2 REINFORCE with Baseline 28
5.4 Actor-Critic Architectures and Extensions 28
5.4.1 Deep Deterministic Policy Gradient (DDPG) 28
5.4.2 Twin Delayed Deep Deterministic Policy Gradient (TD3) 28
5.4.3 Proximal Policy Optimization (PPO) 29
5.5 Actor Architectures and Extensions 29
5.5.1 Critic-Free Reinforcement Learning 29
5.6 Experimenting with Multiple RL Agents 30

6 Implementation and Simulation **31**

6.1 Quantum Control Hamiltonian for the H-T-CNOT Gate Set 31
6.1.1 Connection to Quantum Gates: Hadamard and T Gates 33
6.1.2 Coupling Strength interactions 35
6.2 Implementation of RL Agents for Quantum Gate Optimization 37
6.2.1 State, Action, and Reward Spaces 37
6.2.2 Reward Shaping 37
6.2.3 Quantum State Representation from Unitary Matrices 39
6.2.4 Action Space: Control of Hamiltonian parameters 41
6.2.5 Quantum Evolution in the RL Environment 42
6.3 Double Dueling Deep Q-Network (DDDQN) Implementation 43

6.4 Twin Delayed Deep Deterministic Policy Gradient (TD3) Implementation . . 44
6.5 Proximal Policy Optimization (PPO) Implementation 45
6.6 Group Policy Relative Optimization (GRPO) Implementation 45
6.7 Comparison with GRAPE and CRAB 46
6.7.1 Gradient Ascent Pulse Engineering (GRAPE) 46
6.7.2 Chopped Random Basis (CRAB) 46
6.7.3 Loss Function for GRAPE and CRAB 47
6.7.4 Reinforcement Learning vs. GRAPE and CRAB 48
6.8 Parameterization and Tuning of RL Algorithms 49
6.8.1 Dropout 51
6.8.2 Implementation Environment 52
6.9 Challenges and Lessons Learned 54

7 Results and Analysis **55**
7.1 Performance Metrics: Fidelity and Efficiency 55
7.2 Comparison of RL Algorithms for Quantum Gate Optimization 56
7.2.1 Single-Qubit Gates (H and T) 56
7.2.2 Two-Qubit Gate (CNOT) 74
7.3 Training Convergence Analysis 85
7.3.1 Convergence Speed and Computational Resource Usage 85
7.3.2 Learning Stability 85
7.4 Validation and Benchmarking 85

8 Applications and Extensions **97**
8.1 Applications of the H-T-CNOT Gate Set in Quantum Algorithms 97
8.1.1 Quantum Fourier Transform (QFT) 97
8.1.2 Grover's Search Algorithm 98
8.1.3 Variational Quantum Algorithms (VQAs) 98

8.2 Scalability of RL Approaches to Multi-Qubit Systems 99
8.2.1 Extending RL to N-Qubit Systems 99
8.2.2 RL-Based Gate Compilation for Large Circuits 99
8.3 Extensions to Fault-Tolerant Quantum Computing 100
8.3.1 Optimizing Quantum Error Correction (QEC) 100
8.3.2 RL for Logical Gate Optimization 100
8.3.3 RL-Based Quantum Compilation for Fault-Tolerant Devices 100
8.4 Potential Future Directions 100
8.5 Summary of Applications and Extensions 101

9 Conclusion and Future Work **102**
9.1 Summary of Contributions 102
9.2 Limitations of the Current Work 103
9.3 Directions for Future Research 104
9.3.1 Hardware Deployment of RL-Based Quantum Control 104
9.3.2 Hybrid Quantum-Classical RL Methods 104
9.3.3 Scalability to Large Quantum Systems 104
9.3.4 Adaptive RL for Noise-Resilient Quantum Gates 104
9.4 Final Remarks 105

A Appendix **106**
A.1 Algorithms 106
A.2 Hamiltonian derivation 113

Bibliography **116**

List of Figures

3.1 Unitary Propagation 17

7.1 DDDQN Agent H-Gate Fidelities and Total Reward 56

7.2 Discrete PPO Agent H-Gate Fidelities and Total Reward 57

7.3 Continuous PPO Agent H-Gate Fidelities and Total Reward 57

7.4 Discrete GRPO Agent H Gate Fidelities and Total Reward 58

7.5 Continuous GRPO Agent H Gate Fidelities and Total Reward 58

7.6 TD3 Agent H Gate Fidelities and Total Reward 59

7.7 H Gate Normalized Maximum Fidelity Trajectory Log Infidelity for every agent 60

7.8 DDDQN H Gate Control Pulses 62

7.9 Discrete PPO H Gate Control Pulses 63

7.10 Continuous PPO H Gate Control Pulses 63

7.11 Discrete GRPO H Gate Control Pulses 64

7.12 Continuous GRPO H Gate Control Pulses 64

7.13 TD3 H Gate Control Pulses 65

7.14 DDDQN Agent T-Gate Fidelities and Total Reward 66

7.15 Discrete PPO Agent T-Gate Fidelities and Total Reward 66

7.16 Continuous PPO Agent T-Gate Fidelities and Total Reward 67

7.17 Discrete GRPO Agent T-Gate Fidelities and Total Reward 67

7.18 Continuous GRPO Agent T-Gate Fidelities and Total Reward 68

7.19 TD3 Agent T-Gate Fidelities and Total Reward 68
7.20 T-Gate Normalized Maximum Fidelity Trajectory Log Infidelity for every agent 69
7.21 DDDQN T Gate Control Pulses 70
7.22 Discrete PPO T Gate Control Pulses 71
7.23 Continuous PPO T Gate Control Pulses 71
7.24 Discrete GRPO T Gate Control Pulses 72
7.25 Continuous GRPO T Gate Control Pulses 72
7.26 TD3 T Gate Control Pulses 73
7.27 DDDQN Agent CNOT-Gate Fidelities and Total Reward 74
7.28 Discrete PPO Agent CNOT-Gate Fidelities and Total Reward 74
7.29 Continuous PPO Agent CNOT-Gate Fidelities and Total Reward 75
7.30 Discrete GRPO Agent CNOT-Gate Fidelities and Total Reward 75
7.31 Continuous GRPO Agent CNOT-Gate Fidelities and Total Reward 76
7.32 TD3 Agent CNOT-Gate Fidelities and Total Reward 76
7.33 CNOT-Gate Normalized Maximum Fidelity Trajectory Log Infidelity for every agent 78
7.34 DDDQN CNOT Gate Control Pulses 79
7.35 Discrete PPO CNOT Gate Control Pulses 80
7.36 Continuous PPO CNOT Gate Control Pulses 81
7.37 Discrete GRPO CNOT Gate Control Pulses 82
7.38 Continuous GRPO CNOT Gate Control Pulses 83
7.39 TD3 CNOT Gate Control Pulses 84
7.40 H Gate Optimal Control 86
7.41 T Gate Optimal Control 87
7.42 CNOT Gate Optimal Control 88
7.43 H Gate Optimal Pulses 89
7.44 T Gate Optimal Pulses 90

7.45 CNOT Gate GRAPE AdamW Optimal Pulses 91
7.46 CNOT Gate GRAPE LBFGS Optimal Pulses 92
7.47 CNOT Gate CRAB Optimal Pulses 93
7.48 H Gate Log Infidelity Comparison 94
7.49 T Gate Log Infidelity Comparison 95
7.50 CNOT Gate Log Infidelity Comparison 96

List of Tables

2.1 DRL for Quantum Control Applications [1] 9

7.1 H-Gate Fidelities Comparison 60

7.2 T Gate Fidelities Comparison 69

7.3 CNOT Gate Fidelities Comparison 77

Abbreviations

A3C: Asynchronous Advantage Actor Critic

CGRPO: Continuous GRPO

CNOT: Controlled NOT

CPPO: Continuous PPO

CRAB: Chopped Random Basis

CUDA: Compute Unified Device Architecture

DDPG: Deep Deterministic Policy Gradient

DGRPO: Discrete GRPO

DPPO: Discrete PPO

DQN: Deep Q-Network

DRL: Deep Reinforcement Learning

F: Process Fidelity

GAE: Generalized Advantage Estimation

GPU: Graphic Process Unit

GRAPE: Gradient Ascent Pulse Engineering

GRPO: Group Relative Policy Optimization

H: Hadamard

LBFGS: Limited Broyden–Fletcher–Goldfarb–Shanno

MC: Monte Carlo

MDP: Markov Decision Process

PPO: Proximal Policy Optimization

RL: Reinforcement Learning

RWA: Rotation Wave Approximation

TD: Temporal Difference

TD3: Twin Delayed Deep Deterministic Policy Gradient

Abstract

The design and implementation of universal quantum gate sets are foundational to the advancement of quantum computing, enabling the realization of complex quantum algorithms and error correction protocols. This thesis explores the use of reinforcement learning (RL) methods to efficiently construct a universal quantum gate set comprising the Hadamard (H), the $\frac{\pi}{8}$ (T), and controlled-NOT (CNOT) gates. These gates, recognized for their minimality and universality, serve as essential building blocks for arbitrary quantum operations.

The problem is formulated as a control optimization task, where various RL agents are deployed to determine the optimal Rabi Frequency, Detuning and coupling strength timing of quantum pulses to implement these gates with high fidelity. The investigation includes an evaluation of multiple reinforcement learning algorithms to assess their performance in balancing computational efficiency, physical constraints, and scalability across single- and multi-qubit systems.

The proposed approach is validated through numerical simulations, demonstrating the ability of RL techniques to automate and enhance the design of universal gate sets. This work contributes to the growing synergy between machine learning and quantum technologies, offering a flexible and scalable framework for optimizing quantum control. The findings highlight the potential of RL-based methodologies in advancing practical and robust implementations of quantum computing.

Keywords: *Reinforcement Learning, Machine Learning, Deep Learning, Quantum Control, Quantum Computing, Quantum Technologies.*

Acknowledgements

First, I express my deepest gratitude to my advisor, Prof. Emmanuel Paspalakis, for his invaluable guidance, encouragement, and unwavering support throughout this journey. His expertise and insights have been instrumental in shaping this thesis and, undoubtedly, my understanding of quantum technologies.

I am also deeply grateful to the members of my examination committee, Prof. Ioannis Thanopulos and Prof. Dionisis Stefanatos, for their valuable feedback, which have significantly improved the quality of this work.

I would like to thank the professors, colleagues, and peers at Democritus University of Thrace (DUTh) and the Institute of Nanoscience and Nanotechnology (INN-D) for their lectures, collaboration, and shared passion for research.

To my partner of heart, Jessica, I extend my deepest thanks for their unwavering belief in me, their constant encouragement, and their understanding during this challenging journey. Her support has been my source of strength and motivation throughout this endeavor.

Furthermore, I wish to express my sincere gratitude to the Colegio de Matemáticas Bourbaki, under the distinguished leadership of Prof. Carlos Alfonso Ruiz Guido, for their generous support which partially funded my studies at this institution. Their commitment to fostering advanced research and education has been instrumental in my academic journey and I am deeply thankful for the opportunities they have provided.

Finally, I am grateful to the wider scientific community and pioneers of quantum science whose visionary work has inspired and guided this research.

Chapter 1

Introduction

1.1 Overview of Quantum Computing

Quantum computing represents a paradigm shift in computation, leveraging quantum mechanical principles such as superposition, entanglement, and interference to solve problems that are intractable for classical computers [2, 3]. Unlike classical bits, which exist in binary states (0 or 1), quantum bits (qubits) can exist in superpositions of states, enabling exponential parallelism. This potential makes quantum computing particularly promising for applications in supply chain, cryptography, optimization, chemistry, and the simulation of quantum systems [3]. The latter is especially crucial for understanding the microscopic structure of nature as it behaves quantum mechanically.

1.2 The Role of Universal Gate Sets in Quantum Control

Universal gate sets are fundamental to quantum computing, similar to universal logic gates in classical computation [2]. A gate set is universal if any quantum operation can be approximated with arbitrary accuracy by sequences of gates of the set [4]. The Hadamard (H),

the $\frac{\pi}{8}$ phase (T) and the controlled-NOT (CNOT) gates form one of these minimal universal gate sets, widely recognized for their simplicity and effectiveness [5]. However, achieving high-fidelity implementations of these gates is challenging due to decoherence, noise, and hardware imperfections [6].

1.3 Reinforcement Learning in Quantum Control

Reinforcement learning (RL), a subset of machine learning, provides an effective framework to address complex control problems [7]. In the context of quantum control, RL techniques frame the implementation of quantum gates as a Markov Decision Process (MDP), enabling optimization of control parameters such as pulse amplitude, detuning, phase, coupling strength and timing [6]. These methods excel at navigating high-dimensional parameter spaces and adapting to system-specific constraints, making them ideal for designing high-fidelity universal quantum gates [8, 9] or scaling quantum circuits [10].

1.4 Objectives of the Thesis

This thesis aims to investigate the application of RL methods for the efficient design and implementation of a set of universal quantum gates consisting of the Hadamard (H), $\frac{\pi}{8}$ phase (T) and CNOT gates. Specifically, this work seeks to:

- Formulate quantum gate control as a reinforcement learning problem.
- Deploy and compare the performance of various RL agents for optimizing quantum control.
- Validate the proposed methods through numerical simulations and analyze their scalability to multi-qubit systems.

1.5 Contributions of the Work

This thesis contributes to the intersection of quantum computing and machine learning by proposing a scalable framework for automated quantum gate design using reinforcement learning, evaluates the effectiveness of multiple RL algorithms in achieving high-fidelity implementations of the H-T-CNOT gate, and highlights the trade-offs between computational efficiency and physical constraints in quantum control.

1.6 Structure of the Thesis

The remainder of this thesis is organized as follows.

Chapter 2 introduces the basics of quantum mechanics and quantum gates, discusses key concepts and challenges of quantum control, and provides an overview of reinforcement learning (RL) principles. This chapter also explains the formulation of quantum control problems as Markov Decision Processes (MDPs).

Chapter 3 focuses on the theoretical aspects of the Hadamard (H), T, and controlled-NOT (CNOT) gates. The author discusses their universality and minimality in quantum computing and highlights the challenges in optimizing their control.

Chapter 4 describes the formulation of quantum gate control as an MDP. The author discusses the state, action, and reward spaces in the context of quantum gate design and explores the comparative framework of various RL algorithms.

Chapter 5 examines the implementation of multiple RL methods, including Monte Carlo, Temporal Difference (TD), Policy Gradient, and actor-critic approaches. This chapter also discusses the design and deployment of RL agents for quantum control.

Chapter 6 details the implementation of the RL framework for the H-T-CNOT gate set, including the simulation environments, parameterization, and tuning of RL algorithms. Highlights the quantum control Hamiltonian used in the experiments.

Chapter 7 analyzes the performance of RL algorithms using metrics such as fidelity

and efficiency. Comparison of different RL approaches and discussion of trade-offs between computational and physical constraints. Presents key observations and insights from the simulations.

Chapter 8 discusses the practical applications of the H-T-CNOT gate set in quantum algorithms and explores the scalability of RL-based approaches to multi-qubit systems. Highlights potential extensions to fault-tolerant quantum computing.

Chapter 9 summarizes the thesis contributions, discusses the limitations of the current work, and provides directions for future research.

Chapter 2

Fundamentals of Quantum Control and Reinforcement Learning

2.1 Basics of Quantum Mechanics and Quantum Gates

Quantum mechanics forms the foundation of quantum computing, where the state of a quantum system is represented as a vector in a complex Hilbert space [2]. We note that we take $\hbar = 1$ for all the equations in this thesis. Then, the evolution of quantum states is governed by the Schrödinger equation expressed as follows:

$$i\frac{\partial}{\partial t}\left|\psi(t)\right\rangle = \hat{H}\left|\psi(t)\right\rangle, \tag{2.1}$$

where $\hat{H}$ is the Hamiltonian operator that dictates the system's dynamics.

Quantum gates operate on qubits and are represented by unitary matrices. For example: the Hadamard Gate (H) creates a superposition from a basis state, the T Gate (T) introduces a $\frac{\pi}{8}$ phase shift and the Controlled-NOT Gate (CNOT) acts on two qubits, flipping the second qubit if the first is $|1\rangle$.

The universality of these gates lies in their ability to approximate any quantum operation when combined [4, 5, 11].

2.2 Quantum Control: Concepts and challenges

Quantum control is the process of manipulating quantum systems to achieve specific objectives [12, 13], such as implementing gates with high fidelity. Control is achieved by tuning system parameters, such as the amplitude, phase, and timing of external pulses, which influence the Hamiltonian of the system. Several methods exist to control quantum systems such as the Stimulated Raman Adiabatic Passage [14] and optimal control methods such as GRAPE, CRAB, Krotov's [15, 16], and many others.

2.2.1 Challenges in Quantum Control

Quantum systems are sensitive to interactions with their environment, known as decoherence, which can degrade the fidelity of operations [6]. Furthermore, for multi-qubit systems, the number of parameters to be optimized grows exponentially. Moreover, limited precision and bandwidth in real-world quantum devices restrict the control field's complexity.

2.3 Overview of Reinforcement Learning in Quantum Systems

Reinforcement learning (RL) is a machine learning paradigm in which an agent learns to make decisions by interacting with an environment to maximize a cumulative reward [7]. In the context of quantum control, RL provides a flexible framework for optimizing control pulses to achieve desired quantum operations.

2.3.1 Key Components of RL

In our reinforcement learning framework for quantum control, the agent is responsible for learning and making decisions, while the environment simulates the quantum system. The current state represents the configuration of the quantum system at any given moment,

and the action corresponds to a control parameter that is applied to influence the system. Finally, the reward is a scalar value that quantifies the performance of the control strategy, typically in terms of the fidelity of the desired operation. RL algorithms, such as Q-learning, Deep Q-Networks (DQN), and Proximal Policy Optimization (PPO), have shown promise in optimizing quantum gate design [9, 8].

2.4 The Role of Markov Decision Processes (MDPs) in Quantum Control

The problem of quantum gate optimization can be formulated as a Markov Decision Process (MDP). Furhter, MDP and quantum system start to show connection through RL [17]. In this formulation, the **state space** represents possible configurations of the quantum system (e.g., the Bloch sphere for single qubits or the real and imaginary components of unitary matrices for more complex systems), the **action space** defines the control parameters (e.g., pulse amplitude, phase, and duration), the **transition dynamics** encapsulate how the quantum system evolves under a given action, and the **reward function** quantifies the performance of a gate implementation, typically based on fidelity. Mathematically, an MDP is defined as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ the action space, $\mathcal{P}$ the transition probability function, R the reward function, and γ the discount factor. By solving a MDP, RL algorithms can discover optimal policies to maximize gate fidelity while adhering to physical constraints [6, 18]. Several agents have been deployed to solve different quantum control problems as the following table shows.

Table 2.1: DRL for Quantum Control Applications [1]

Quantum state preparation	Hamiltonian-control	DQN, Policy-gradient, PPO
Quantum gate control	Hamiltonian-control	DQN, Policy-gradient
Extreme spin squeezing	Hamiltonian-control	PPO
Quantum metrology	Hamiltonian-control	A3C, DDPG
Quantum compiler	Gate-control	DQN
Quantum state engineering	Measurement-based	DQN
Quantum state stabilization	Measurement-based	DQN, PPO
Quantum error correction	Measurement-based	Policy-gradient, PPO

Chapter 3

Universal Quantum Gate Set: Theoretical Foundations

3.1 The H, T and CNOT Gates

Universal quantum computation requires a set of gates capable of approximating any unitary operation to arbitrary precision [2]. The Hadamard (H), $\pi/4$ Phase Shift (T), and controlled-NOT (CNOT) gates are a minimal universal set that can achieve this.

3.1.1 Hadamard (H) Gate

The Hadamard gate creates superposition states by transforming a qubit from a computational basis to an equal-weight superposition:

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

This gate is essential for initializing quantum states and implementing quantum algorithms such as the quantum Fourier transform [2].

3.1.2 T Gate

The T gate, also known as the $\pi/4$ gate, applies a phase shift to the state $|1\rangle$:

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}.$$

Together with the H gate, the T gate enables the construction of any single-qubit operation [5].

3.1.3 Controlled-NOT (CNOT) Gate

The CNOT gate operates on two qubits, flipping the target qubit if the control qubit is in the $|1\rangle$ state:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

The CNOT gate is necessary to entanglement (not sufficient), which is a fundamental resource in quantum computation [4].

3.2 Universality and Minimality in Gate Sets

A gate set is considered universal if any unitary operation can be approximated to arbitrary precision using a finite sequence of gates in the set [11]. The H, T, and CNOT gates form a universal set because the H and T gates can construct any single-qubit operation and the CNOT gate enables two-qubit interactions, which are necessary for entanglement and multi-qubit computation [4]. The minimality of this set is significant because fewer gate types simplify hardware implementations. The ability to approximate an arbitrary unitary

matrix using the H, T, and CNOT gates has been rigorously proven [11].

3.3 Control Optimization Challenges

Despite their universality, implementing the H, T, and CNOT gates with high fidelity presents challenges. Quantum gates are sensitive to environmental interactions, leading to errors during operations [6]. Precise control of the amplitude, phase, and timing of pulses is required to realize unitary transformations. Longer gate times increase the likelihood of decoherence, necessitating optimization to reduce execution time. Recent advances in reinforcement learning have shown promise in addressing these challenges by optimizing control parameters to achieve high-fidelity gate operations [9, 8].

3.4 Practical Applications of the H-T-CNOT Set

The H, T and CNOT gates are the foundation for the implementation of key quantum algorithms. Shor's efficient integer factorization relies on modular exponentiation and the quantum Fourier transform, both of which can be constructed using the H-T-CNOT set [19]. Grover's quadratic speed-up database search is implemented using Hadamard gates for state initialization and CNOT gates for diffusion operators [20]. Going further, quantum error correction codes, such as the surface code, use these gates for encoding, syndrome extraction, and correction [3].

3.5 Time Evolution Implementations

The first important point that we should keep in mind is that time is just a parameter in quantum mechanics, not an operator [21]. In many quantum control schemes, the desired quantum gate is generated by letting the system evolve under a chosen Hamiltonian for a specific time. In what follows, we describe how to obtain the Hadamard (H), T (or $\pi/4$) and

controlled-NOT (CNOT) gates by appropriate choices of Hamiltonians and evolution times.

3.5.1 Hadamard Gate (H)

The Hadamard gate is given (up to a global phase) by

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

A common strategy is to implement it as a rotation by π about the axis

$$\hat{n} = \frac{1}{\sqrt{2}}(\hat{x} + \hat{z}),$$

which lies halfway between the x and z axes on the Bloch sphere.

A typical one-qubit Hamiltonian is written as

$$H = \frac{1}{2}\Big(\Delta\,\sigma_z + \Omega_x\,\sigma_x\Big).$$

For the Hadamard gate, one often chooses

$$\Omega_x = \Delta = 1,$$

so that

$$H = \frac{1}{2}(\sigma_z + \sigma_x).$$

The corresponding time evolution operator is

$$U(t) = e^{-iHt}.$$

A calculation shows that if the pulse duration is set to

$$t = \frac{\pi}{\sqrt{2}},$$

the resulting operator is (up to a global phase) equivalent to the Hadamard gate. In other words,

$$e^{-i\left(-\frac{1}{2}(\sigma_z+\sigma_x)\right)\frac{\pi}{\sqrt{2}}} \propto H.$$

3.5.2 T Gate

The T gate is a single-qubit phase gate defined by

$$T = \begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix}.$$

A convenient way to generate the T gate is to use a Hamiltonian that produces a z-rotation. For example, consider the Hamiltonian

$$H = \frac{1}{2}\sigma_z.$$

Then, the time evolution operator is

$$U(t) = e^{-iHt} = e^{-i\frac{t}{2}\sigma_z} = \begin{pmatrix} e^{-it/2} & 0 \\ 0 & e^{it/2} \end{pmatrix}.$$

Removing the global phase $e^{-it/2}$ (which is physically irrelevant), we have

$$U(t) \propto \begin{pmatrix} 1 & 0 \\ 0 & e^{it} \end{pmatrix}.$$

To generate the T gate we need the relative phase to be $\pi/4$, so we set

$$t = \frac{\pi}{4}.$$

Thus,

$$e^{-i\frac{\pi}{8}\sigma_z} \propto T.$$

More precisely, one obtains

$$U\left(\frac{\pi}{4}\right) = e^{-i\frac{\pi}{8}\sigma_z} = e^{-i\pi/8}\begin{pmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{pmatrix},$$

which implements the T gate up to a global phase.

3.5.3 Controlled-NOT (CNOT) Gate

The controlled-NOT (CNOT) gate is a two-qubit gate defined by

$$\text{CNOT} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X,$$

where the target qubit is flipped (via the Pauli X operator) if and only if the control qubit is in the state $|1\rangle$.

A standard approach to generate a controlled gate is to use a Hamiltonian of the form

$$H_{\text{int}} = |1\rangle\langle 1| \otimes h,$$

so that the time evolution operator becomes

$$U(t) = e^{-iH_{\text{int}}t} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes e^{-iht}.$$

To implement the CNOT gate, choose

$$h = \frac{\pi}{2} X,$$

and set the evolution time $t = 1$ (with $g = 1$ in dimensionless units). Then,

$$U = e^{-i\left(|1\rangle\langle 1| \otimes \frac{\pi}{2} X\right)} = |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes e^{-i\frac{\pi}{2}X}.$$

Since

$$e^{-i\frac{\pi}{2}X} = -iX,$$

(up to a phase), we have

$$U \propto |0\rangle\langle 0| \otimes I + |1\rangle\langle 1| \otimes X,$$

which is the CNOT gate (ignoring the unimportant phase factor).

By selecting appropriate Hamiltonians and evolution times, we can generate many fundamental quantum gates via time evolution. These implementations illustrate how time evolution under properly chosen Hamiltonians can realize essential quantum operations.

For example, in the following image we present the landscape of a unitary propagated from the identity matrix to achieve the H-Gate. This evolution is shown for various time steps and incremental intervals, with the driving amplitude and detuning both set to -4.

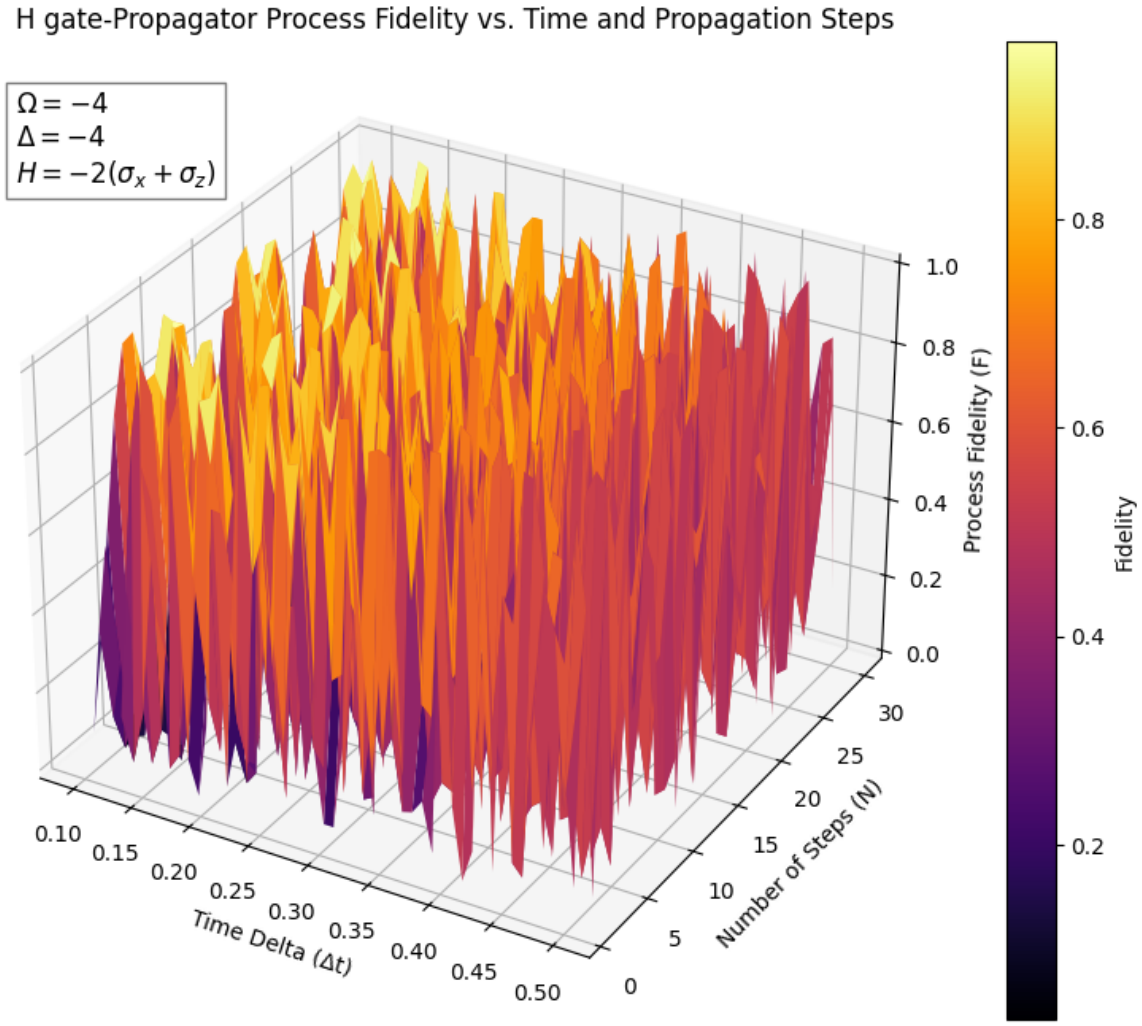

Figure 3.1: Unitary Propagation

Chapter 4

Reinforcement Learning for Quantum Gate Control Optimization

4.1 Quantum Control as an MDP

The optimization of quantum gates can be formulated as a Markov Decision Process (MDP), where the goal is to maximize the fidelity of a quantum operation. An MDP is represented as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, R, \gamma)$:

- $\mathcal{S}$: The state space represents the quantum system's configuration, such as the Bloch sphere for single qubits.
- $\mathcal{A}$: The action space defining the control parameters, such as pulse amplitude, phase, and timing.
- $\mathcal{P}$: The transition probabilities describe how the quantum system evolves under a specific action.
- R: The reward function, which measures the fidelity of the desired quantum gate.
- γ: The discount factor, controlling the agent's focus on immediate versus long-term rewards [7].

The agent's objective is to find an optimal policy π^* that maps states to actions, maximizing the expected cumulative reward:

$$J(\pi) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right].$$

4.2 State, Action, and Reward Spaces in Quantum Gate Design

4.2.1 State Space

The state space represents the configuration of the quantum system. For a single qubit: Points on the Bloch sphere defined by angles (θ, ϕ). For multiple qubits: Tensor products of individual qubit states, forming a high-dimensional Hilbert space [2]. In the present work, the state is defined by the real and imaginary elements of the target unitary (H, T, or CNOT).

4.2.2 Action Space

Actions correspond to control parameters applied to the quantum system, such as adjusting the amplitude, frequency, and phase of the control pulses and applying discrete gate operations or continuous transformations of the Hamiltonian.

For example, applying a rotation $R_x(\theta)$ about the x-axis modifies the state to:

$$R_x(\theta) = \exp\left(-i\frac{\theta}{2}\sigma_x\right),$$

where σ_x is the Pauli-X matrix.

4.2.3 Reward Function

The reward function will be related to the fidelity of the gate implemented, ensuring that the reinforcement learning agent is optimized for high-accuracy quantum operations.

Gate Fidelity

The fidelity of the implemented gate is defined as

$$F(U,T) = \frac{1}{d^2}\left|\mathrm{Tr}(T^\dagger U)\right|^2, \tag{4.1}$$

where T is the target gate, U is the realized gate, d is the dimension of the Hilbert space [22, 2]. A fidelity close to **1** indicates a high gate accuracy.

Average Process Fidelity

While the gate fidelity measures how well a specific unitary transformation is achieved, a more robust measure is the average fidelity of the process, which quantifies the precision of the gate implemented across all possible input states [22]. The average process fidelity $\bar{F}$ is given by:

$$\bar{F} = \frac{d + F(U,T)d}{d+1}, \tag{4.2}$$

where$d$ is the dimension of the Hilbert space and F is the process fidelity calculated in the previous equation.

This metric will be used in conjunction with the fidelity to track the evolution of the unitary.

4.3 Algorithms for Quantum Control: A Comparative Framework

Several reinforcement learning algorithms have been applied to quantum control problems [1]. Each offers unique strengths and trade-offs.

4.3.1 Monte Carlo Methods

Monte Carlo methods estimate the value of states or actions by sampling episode returns:

$$V(s) = \mathbb{E}[G_t \mid S_t = s],$$

where $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k}$. These methods are effective for episodic tasks, but may struggle with high-dimensional state spaces [7].

4.3.2 Temporal Difference (TD) Methods

TD methods combine Monte Carlo sampling with bootstrapping, updating value estimates based on temporal differences:

$$V(s) \leftarrow V(s) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right].$$

Q-Learning and Sarsa are popular TD methods for estimating action value [7].

4.3.3 Policy Gradient Methods

Policy gradient methods optimize a parameterized policy $\pi_\theta(a \mid s)$ by directly maximizing the expected return:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a \mid s) G_t\right].$$

The REINFORCE algorithm and its variants, such as REINFORCE with baseline, are widely used for continuous control [23].

4.3.4 Actor-Critic Architectures

Actor-critic methods combine value-based and policy-based approaches. The actor updates the policy, while the critic estimates the value function, reducing the variance in the policy updates [24]. Examples include Deep Deterministic Policy Gradient (DDPG) [25] and Proximal Policy Optimization (PPO) [26].

4.3.5 Actor Architectures

Actor methods use only policy-based approaches. Examples include Group Relative Policy Optimization (GRPO) [27] recently developed on top of PPO by Startup DeepseekAI.

4.4 Exploration vs. Exploitation in Quantum Systems

Reinforcement learning balances exploration (trying new actions) and exploitation (choosing the best-known action). Techniques such as ϵ-greedy policies and entropy regularization are crucial in quantum control to avoid local optima and achieve globally optimal gate fidelities [7] or greedy strategies to prepare quantum states [28].

4.5 General Approach

The Reinforcement Learning Algorithm for the agents deployed in this work should work along this line:

Algorithm 1 General

Require: Initial model parameters θ

for episode = 1, 2, ... EPISODES **do**
  Reset the quantum environment state
  for $k = 1, 2, \ldots, N$ **do**
    Choose first action according to the agent strategy to define the pulse
    Evolve the unitary by the pulse and time
    Compute the step reward based on fidelity between target and current unitary
    Estimate the next unitary
    Choose next action according to the agent strategy
    Store trajectory $(s_{k-1}, a_{k-1}, s_k, r_k, d_k)$
  end for
  Calculate Discounted Rewards
  Calculate Loss Function
  Update the model parameters θ
end for

The agent may have their nuances, like GAE estimation, updating Actor and Critic networks, off- or on-policy, etc., but this outline should give the reader an idea of how these agents interact with the environment.

Chapter 5

Reinforcement Learning Agents: An Overview

5.1 Monte Carlo Methods for Quantum Control

Monte Carlo (MC) methods estimate action or state values by averaging sampled returns from episodes. In quantum control, they can be used to optimize pulse sequences for gate implementation:

$$V(s) = \mathbb{E}[G_t \mid S_t = s], \tag{5.1}$$

where $G_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k}$ is the return. Monte Carlo methods are particularly suitable for episodic tasks in which the reward can be calculated after full trajectory simulations [7].

Although MC methods are computationally inexpensive, their reliance on complete episodes limits their applicability in quantum control scenarios with high-dimensional state-action spaces [6].

5.2 Temporal Difference (TD) Methods for Policy Optimization

Temporal Difference (TD) methods combine Monte Carlo sampling with bootstrapping to estimate value functions. They are widely used in reinforcement learning for their ability to update estimates incrementally during training:

$$V(s) \leftarrow V(s) + \alpha \left[R_{t+1} + \gamma V(S_{t+1}) - V(S_t) \right]. \tag{5.2}$$

5.2.1 Q-Learning

Q-Learning is an off-policy TD control algorithm that updates the action-value function using:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]. \tag{5.3}$$

In quantum control, Q-Learning can optimize discrete control parameters, but its convergence slows in continuous or high-dimensional spaces [7].

5.2.2 Sarsa

Sarsa is an on-policy variant of TD methods that updates based on the action taken by the policy:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \right]. \tag{5.4}$$

It ensures smoother exploration, but faces similar challenges to Q-Learning in complex quantum systems [6].

5.2.3 Deep Q-Networks (DQN)

DQN extends Q-Learning by approximating the Q-value function with a deep neural network. The network is trained to minimize the temporal difference error:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')}\left[\left(R_{t+1} + \gamma \max_a Q(s', a; \theta^-) - Q(s, a; \theta)\right)^2\right], \tag{5.5}$$

where θ^- represents the weights of a target network that is periodically updated to stabilize training [29]. In quantum control, DQN is used for tasks with discrete action spaces, such as selecting among predefined control pulses. It offers improved convergence compared to tabular Q-learning, but requires significant computational resources for high-dimensional systems.

5.2.4 Double Deep Q-Networks (DDQN)

DDQN addresses the overestimation bias in DQN by decoupling the action selection and evaluation processes:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s')}\left[\left(R_{t+1} + \gamma Q(s', \arg\max_a Q(s', a; \theta); \theta^-) - Q(s, a; \theta)\right)^2\right]. \tag{5.6}$$

This approach improves stability and accuracy in estimating Q-values [30]. In quantum control, DDQN performs better in noisy environments and is less prone to suboptimal policy convergence compared to DQN. For example, modified DQN architectures were implemented to control single- and double-qubit systems [31, 32]

5.2.5 Double Dueling Deep Q-Network (DDDQN)

Double Dueling Deep Q-Network (DDDQN) is an extension of DDQN that incorporates the dueling network architecture. The dueling architecture improves action-value estimation by separating the value and advantage functions in the Q-network, making learning more

efficient in environments where some actions have little impact. Unlike standard Q-learning networks that directly estimate $Q(s, a)$, DDDQN decomposes it into:

$$Q(s, a) = V(s) + A(s, a) - \frac{1}{|\mathcal{A}|} \sum_{a'} A(s, a'), \tag{5.7}$$

where $V(s)$ is the state-value function, which estimates the importance of being in a state and $A(s, a)$ is the advantage function, which determines the relative benefit of taking action a in the state s.

Several works have demonstrated the use of DQN's and their variants to achieve high-fidelity gates [33].

5.3 Policy Gradient Methods: Direct Policy Search

Policy gradient methods directly optimize the policy $\pi_\theta(a \mid s)$ by maximizing the expected return:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(a \mid s) G_t \right]. \tag{5.8}$$

This approach is particularly effective for continuous control problems in quantum systems [23].

5.3.1 REINFORCE Algorithm

The REINFORCE algorithm uses the policy gradient theorem to optimize the policy:

$$\Delta\theta \propto \nabla_\theta \log \pi_\theta(a \mid s) G_t. \tag{5.9}$$

Its simplicity makes it ideal for single-qubit gates, though it may suffer from high variance in multi-qubit systems.

5.3.2 REINFORCE with Baseline

To reduce variance, a baseline $b(s)$ is introduced:

$$\Delta\theta \propto \nabla_\theta \log \pi_\theta(a \mid s) \left[G_t - b(s)\right]. \tag{5.10}$$

Common baselines include the value function $V(s)$ estimated by a neural network [7].

5.4 Actor-Critic Architectures and Extensions

Actor-critic methods combine the strengths of value-based and policy-based approaches. The actor updates the policy, while the critic estimates the value function.

5.4.1 Deep Deterministic Policy Gradient (DDPG)

DDPG is designed for continuous action spaces and leverages a deterministic policy:

$$a = \mu_\theta(s). \tag{5.11}$$

The critic evaluates the Q-value $Q(s, a)$, and both the actor and the critic are updated using neural networks [25]. DDPG has proven effective for multi-qubit systems where precise control is required.

5.4.2 Twin Delayed Deep Deterministic Policy Gradient (TD3)

TD3 improves DDPG by addressing its sensitivity to overestimation and instability. Key modifications include:

- Double Critics: Two critic networks are used, and the smaller Q-value is chosen during updates:

$$y = R + \gamma \min_{i=1,2} Q_i(s', \mu(s')).$$

- Delayed Policy Update The actor is updated less frequently than the critics to improve stability.

- Target Policy Smoothing: Adds noise to the target actions to prevent the use of sharp Q-value spikes [34].

TD3 is particularly effective for noisy quantum control environments, where the precise tuning of continuous parameters is critical.

5.4.3 Proximal Policy Optimization (PPO)

PPO simplifies trust-region policy optimization by clipping the policy ratio:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right], \tag{5.12}$$

where $r_t(\theta)$ is the probability ratio and $\hat{A}_t$ is the advantage estimate [26].

5.5 Actor Architectures and Extensions

Traditional actor-critic methods rely on both an actor network to optimize policy selection and a critic network to estimate value functions. However, recent breakthroughs, such as DeepSeekMath [27], have demonstrated that the critic network can be omitted while maintaining stable policy learning.

5.5.1 Critic-Free Reinforcement Learning

DeepSeekMath introduced a variation of policy optimization in which the critic network is neglected, leveraging only the actor for decision making. This approach significantly reduces computational overhead while still achieving high performance in complex reasoning tasks. The key advantages of critic-free architectures include reduced computational complexity by the absence of a critic network which decrease training time and memory usage, improved

stability which benefits from direct policy optimization without relying on bootstrapped value estimates and the approach scales efficiently to high-dimensional spaces, such as multi-qubit quantum control, where value estimation may introduce unnecessary variance.

5.6 Experimenting with Multiple RL Agents

To identify how different RL algorithms behave for quantum gate optimization, multiple agents were deployed, such as TD methods for discrete control optimization, Actor-critic (Policy-Value) methods for discrete and continuous control and Critic-free methods for discrete and continuous control. The experiments focused on Fidelity improvement across gate types (H, T, and CNOT) and convergence rates and stability.

The reader may check within the section 'Algorithms' within the appendix how these algorithms work.

Chapter 6

Implementation and Simulation

6.1 Quantum Control Hamiltonian for the H-T-CNOT Gate Set

The control of quantum gates is governed by the time-dependent Schrödinger equation (2.1) where $\hat{H}(t)$ is the Hamiltonian of the system, composed of:

$$\hat{H}(t) = \hat{H}_0 + \sum_k u_k(t)\hat{H}_k, \tag{6.1}$$

where $\hat{H}_0$ is the drift Hamiltonian (natural system evolution), $\hat{H}_k$ represents external control fields and $u_k(t)$ are the time-dependent control amplitudes optimized using reinforcement learning.

The optimization of these control fields aims to implement high-fidelity Hadamard (H), T, and CNOT gates, forming a universal quantum gate set [2].

The Hamiltonian for a single-qubit system interacting with an external control field is generally expressed as

$$\hat{H}(t) = \frac{1}{2}\Big(\omega_0\,\hat{\sigma}_z + \Omega_x(t)\,\hat{\sigma}_x + \Omega_y(t)\,\hat{\sigma}_y\Big), \tag{6.2}$$

where ω_0 is the natural resonance frequency of the qubit, $\Omega_x(t)$ and $\Omega_y(t)$ are external driving fields (time-dependent), and $\hat{\sigma}_x$, $\hat{\sigma}_y$, $\hat{\sigma}_z$ are the Pauli matrices.

Applying the Rotating Wave Approximation (RWA), which neglects rapidly oscillating terms, the Hamiltonian in the rotating frame simplifies to

$$\hat{H}_{\text{RWA}}(t) = \frac{1}{2}\Big(\Delta\,\hat{\sigma}_z + \Omega(t)e^{i\phi}\,\hat{\sigma}_+ + \Omega(t)e^{-i\phi}\,\hat{\sigma}_-\Big), \tag{6.3}$$

where $\Delta = \omega - \omega_0$ is the detuning between the control field frequency ω and the qubit frequency ω_0, $\Omega(t)$ is the Rabi frequency, which controls the strength of the interaction, $\hat{\sigma}_\pm = \frac{1}{2}(\hat{\sigma}_x \pm i\hat{\sigma}_y)$ are the raising and lowering operators and ϕ is the phase of the control field, which determines the rotation axis in the Bloch sphere representation.

Just to clairfy, the raising and lowering operators are defined as

$$\hat{\sigma}_\pm = \frac{1}{2}\left(\hat{\sigma}_x \pm i\,\hat{\sigma}_y\right).$$

For clarity, the Pauli matrices are given by

$$\hat{\sigma}_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \hat{\sigma}_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \hat{\sigma}_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

Then,

$$\hat{\sigma}_+ = \frac{1}{2}\left(\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + i\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}\right) = \frac{1}{2}\left(\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} + \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\right) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix},$$

and

$$\hat{\sigma}_- = \frac{1}{2}\left(\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} - i\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}\right) = \frac{1}{2}\left(\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} - \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}\right) = \begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix}.$$

6.1.1 Connection to Quantum Gates: Hadamard and T Gates

By tuning the control field parameters $\Omega(t)$ and ϕ, specific quantum gates can be implemented.

Hadamard Gate (H)

The Hadamard gate is given by:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{6.4}$$

To realize this gate using Hamiltonian evolution we can apply a controlled external field with controlled detuning by introducing an appropriately detuned control field, we can engineer an effective Hamiltonian that rotates the state vector to the Hadamard basis or apply a controlled external field with phase control and controlled detuning by precisely tuning the phase ϕ of the driving field along with detuning, a more robust implementation of the Hadamard transformation can be achieved.

T Gate

The T gate, also known as the $\pi/8$-gate, is defined as:

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}. \tag{6.5}$$

This corresponds to a phase shift. To realize this gate using Hamiltonian evolution, we can control the detuning where the qubit undergoes a phase evolution over time, craft a combination of a driving field and detuning allows for a more precise implementation, reducing sensitivity to frequency drifts or adjust the phase ϕ of the external field along with detuning allows for a fine-tuned phase accumulation, making the gate more robust against hardware imperfections.

CNOT Gate (Controlled-NOT)

The controlled-NOT (CNOT) gate is a fundamental two-qubit operation defined as:

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{6.6}$$

The CNOT gate flips the state of the target qubit ($|1\rangle \leftrightarrow |0\rangle$) only when the control qubit is in the $|1\rangle$ state. This operation is essential for entanglement generation and universal quantum computation. To realize this gate using Hamiltonian evolution we can control the coupling in which the interaction strength between the two qubits is modulated via coupling terms control an external field with controlled coupling where an external driving field is applied with precise detuning to control the two-qubit interaction or control an external field with phase control and controlled coupling where a fully controlled external field, including phase tuning, is applied to optimize the CNOT implementation.

A way to implement the CNOT gate in a coupled qubit system is through the cross-coupling or off-diagonal anisotropic exchange interaction Hamiltonian:

$$H_{\text{CNOT}} = \frac{1}{2} J_{zx} \hat{\sigma}_z^{(c)} \hat{\sigma}_x^{(t)}, \tag{6.7}$$

where J_{zx} is the coupling strength between the control qubit (c) and the target qubit (t) which applies a phase flip on the first qubit and flips the second qubit, $\hat{\sigma}_z^{(c)}$ applies a conditional phase shift based on the control qubit state and $\hat{\sigma}_x^{(t)}$ flips the target qubit if the control qubit is in $|1\rangle$.

For an alternative implementation using controlled detuning and controlled drive, we can use the Hamiltonian:

$$H_{\text{CNOT, detuning, driving}} = \frac{1}{2}(\Delta_c \hat{\sigma}_z^{(c)} + \Delta_t \hat{\sigma}_z^{(t)} + \Omega_c \hat{\sigma}_x^{(c)} + \Omega_t \hat{\sigma}_x^{(t)} + J_{zx} \hat{\sigma}_z^{(c)} \hat{\sigma}_x^{(t)}) \tag{6.8}$$

For an implementation using an external field with phase control, an additional driving term is introduced:

$$H_{\text{CNOT, detuning, driving, phase}} = \frac{1}{2}(\Delta_c \hat{\sigma}_z^{(c)} + \Delta_t \hat{\sigma}_z^{(t)} + \Omega_c e^{i\phi} \hat{\sigma}_x^{(c)} + \Omega_t e^{i\phi} \hat{\sigma}_x^{(t)} + J_{zx} \hat{\sigma}_z^{(c)} \hat{\sigma}_x^{(t)}) \tag{6.9}$$

In practical hardware implementations, such as superconducting qubits the CNOT gate can be realized using tunable couplers or microwave-driven cross-resonance interactions, where reinforcement learning (RL) can optimize pulse shaping for high-fidelity entangling operations [35].

6.1.2 Coupling Strength interactions

Interactions between qubits are described by coupling terms in the Hamiltonian that act on different spin components. For example, consider the following types of interactions:

The cross-coupling terms between the components x and z are given by

$$J_{zx}\, \hat{\sigma}_z^{(c)} \otimes \hat{\sigma}_x^{(t)}$$

and

$$J_{xz}\, \hat{\sigma}_x^{(c)} \otimes \hat{\sigma}_z^{(t)}.$$

In the first term, the z-component of the control qubit interacts with the x-component of the target qubit. This means that the state of the control qubit along the z-axis can influence transitions or rotations of the target qubit in the x-direction. In the second term, the roles are reversed. Another important interaction is the standard Ising coupling, which is given

by

$$J_{zz}\,\hat{\sigma}_z^{(c)} \otimes \hat{\sigma}_z^{(t)}.$$

This term couples the z-components of both qubits. It tends to favor an alignment or anti-alignment of the qubit states along the z-axis, depending on the sign of J_{zz}. Such an interaction is central to models of magnetism and quantum computation, as it can be used to generate entanglement or implement logical operations between qubits.

In addition, there is the x–x coupling,

$$J_{xx}\,\hat{\sigma}_x^{(c)} \otimes \hat{\sigma}_x^{(t)},$$

which couples the x-components of the two qubits. This interaction is common in models like the XX model, where it can lead to simultaneous spin flips along the x-axis and is a mechanism for establishing quantum correlations between qubits.

In summary, the different coupling terms produce distinct effects on the qubits:

$J_{zx}\,\hat{\sigma}_z^{(c)} \otimes \hat{\sigma}_x^{(t)}$	:	Flips the target qubit correlated with the phase of the control qubit.
$J_{xz}\,\hat{\sigma}_x^{(c)} \otimes \hat{\sigma}_z^{(t)}$	:	Shifts the target qubit correlated with the driving of the control qubit.
$J_{zz}\,\hat{\sigma}_z^{(c)} \otimes \hat{\sigma}_z^{(t)}$	:	Both qubits interact via their detuning.
$J_{xx}\,\hat{\sigma}_x^{(c)} \otimes \hat{\sigma}_x^{(t)}$	:	Both interact via their driving.

Each of these interactions plays a unique role in determining the dynamics and correlations in a two-qubit system.

6.2 Implementation of RL Agents for Quantum Gate Optimization

To optimize the gate control process, four reinforcement learning algorithms were implemented: Double Dueling Deep Q-Network (DDDQN), Twin Delayed Deep Deterministic Policy Gradient (TD3), Proximal Policy Optimization (PPO) and Group Relative Policy Optimization (GRPO).

6.2.1 State, Action, and Reward Spaces

The RL environment was defined as follows. The state is represented by the complete quantum state, including all the real and imaginary parts of the elements of the unitary operator. The action is defined as a set of amplitudes, detuning, phases, and coupling (if needed). The reward is connected to the fidelity F between the implemented gate and the target gate, calculated as in Equation (4.2).

6.2.2 Reward Shaping

To improve the sensitivity of the reward function, particularly in high-fidelity regimes, a more convenient metric is the logarithm of infidelity:

$$L = \log_{10}(1 - F(U, V)). \tag{6.10}$$

This transformation provides several advantages. One advantage is improved numerical differentiation. When the fidelity is very close to 1, the absolute differences in fidelity become small, but taking the logarithm amplifies these differences. Additionally, logarithmic scaling results in smoother gradient behavior, helping to avoid vanishing gradients during RL optimization. Finally, the reward signal is enhanced since even small improvements in fidelity (for example, from 0.999 to 0.9999) lead to significant reward differences, facilitating

easier learning for the RL agent.

By defining the RL environment as a discretized quantum evolution problem, reinforcement learning agents can optimize quantum control parameters to achieve high-fidelity quantum gate synthesis.

Thus, the RL agent maximizes the logarithmic reward, effectively minimizing infidelity and improving gate accuracy. In addition, a penalization of the time steps in the pulse trajectory will be applied and a bonus to reach the fidelity threshold. Moreover, we take the aboslute value of the logaritm of infidelity to push the agent to get positive rewards. As a consequence, the reward will be the following:

$$R = -L\left(1 - \frac{t}{T_{\max}}\right), \tag{6.11}$$

where F is the fidelity of the implemented quantum gate, t is the current time step and $T_{\max}$ is the total number of allowed steps.

Termination Condition

The episode terminates if:

$$t = T_{\max}. \tag{6.12}$$

At the final time step, an additional reward scaling factor is introduced:

$$R \leftarrow R \times 5, \quad \text{if } F \geq F_{\text{threshold}}. \tag{6.13}$$

where $F_{\text{threshold}}$ is the predefined fidelity threshold.

This design ensures that the agent is incentivized to reach high-fidelity quantum gates efficiently while discouraging unnecessarily long episodes. Some works note the importance of reward shaping techniques [36].

6.2.3 Quantum State Representation from Unitary Matrices

The quantum state representation is defined by all the elements of the unitary evolution operator, including its real and imaginary components. The unitary matrix for a given quantum gate can be expressed as:

$$U = \begin{bmatrix} a+ib & c+id \\ e+if & g+ih \end{bmatrix}, \tag{6.14}$$

where each entry consists of real and imaginary parts.

For an n-qubit system, the full unitary representation contains 2^{2n} complex elements, meaning we extract and store the real and imaginary parts separately to fully describe the transformation.

Hadamard Gate (H)

The Hadamard gate is defined by the unitary matrix:

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{6.15}$$

Since all elements are purely real, the real and imagginary elements of H are:

$$H_{\text{real}} = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad H_{\text{imag}} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}. \tag{6.16}$$

T Gate ($\pi/4$-Gate)

The T gate introduces a complex phase to the $|1\rangle$ state:

$$T = \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix}. \tag{6.17}$$

Splitting it into real and imaginary parts:

$$T_{\text{real}} = \begin{bmatrix} 1 & 0 \\ 0 & \cos(\pi/4) \end{bmatrix}, \quad T_{\text{imag}} = \begin{bmatrix} 0 & 0 \\ 0 & \sin(\pi/4) \end{bmatrix}. \tag{6.18}$$

CNOT Gate (Controlled-NOT)

The CNOT gate acts on two qubits, flipping the target qubit if the control qubit is in the $|1\rangle$ state:

$$\text{CNOT} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{6.19}$$

Since CNOT is a purely real gate, its decomposition is:

$$CNOT_{\text{real}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \quad CNOT_{\text{imag}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}. \tag{6.20}$$

Extracting Real and Imaginary Components

For reinforcement learning applications, the environment extracts and flattens the real and imaginary components into a single state vector:

$$S = \big[U_{\text{real}}, U_{\text{imag}}\big]. \tag{6.21}$$

This ensures that the RL agent has full access to the entire unitary transformation, allowing it to optimize the implementation of quantum gates efficiently.

6.2.4 Action Space: Control of Hamiltonian parameters

The action space in the reinforcement learning (RL) framework defines the set of control parameters that the agent can manipulate to optimize the implementation of the quantum gate. The agent modifies the time-dependent Hamiltonian by selecting the appropriate control parameters at each time step.

Continuous and Discrete Control Actions

The RL action space can be defined in two ways: In a continuous action space, the agent directly selects the amplitude, detuning, and coupling of control pulses from a continuous range. In contrast, in a discrete action space, the agent chooses from a predefined set of control pulses with fixed parameter values.

Control Parameters for the H, T, and CNOT Gates

For each target gate, the agent optimizes a set of control parameters:

Hadamard Gate (H) and T Gate

- Control Parameters: Rabi frequency $\Omega_x(t)$, and detuning $\Delta(t)$.
- Actions: Adjust Ω_x for qubit rotations and control Δ to maintain resonance. The action space is continuous or discrete depending on the agent, where the discrete set is {-4, 0, 4} for both parameters

CNOT Gate

- Control Parameters: Coupling strength J_{zx}, target qubit Rabi frequency Ω_x^t and detuning Δ_t, and control qubit Rabi frequency Ω_x^c and detuning Δ_c.
- Actions: Tune J_{zx} for entangling operation, adjust $\Omega_x^c \Omega_x^t$ for qubit rotations and control Δ_c, Δ_t to maintain resonance. The action space is continuous or discrete depending

on the agent, where the discrete set is {-4, 0, 4} for single qubit parameters and {-4, -2, 2, 4} for the coupling strength.

The agent explores the action space through epsilon-greedy exploration (for discrete actions) or Gaussian noise perturbations (for continuous actions), ensuring both exploitation of learned strategies and discovery of new optimal control sequences.

Effect of Actions on Quantum Evolution

The agent selects a set of control parameters $\{u_k(t)\}$, which modify the time evolution operator:

$$U(t + \Delta t) = e^{-i\hat{H}(t)\Delta t}U(t). \tag{6.22}$$

The objective of RL is to adjust actions such that $U(t)$ converges to the target quantum gate T with high fidelity.

6.2.5 Quantum Evolution in the RL Environment

The system follows the Schrödinger equation:

$$i\frac{d}{dt}U(t) = \hat{H}(t)U(t), \tag{6.23}$$

where $U(t)$ is the time evolution operator that describes how quantum states evolve and $\hat{H}(t)$ is the time-dependent Hamiltonian.

The environment state is represented by the discretized time evolution:

$$U(t + \Delta t) = e^{-i\hat{H}(t)\Delta t}U(t). \tag{6.24}$$

Since the goal is to optimize a quantum gate operation, the evolution starts from the identity operator:

$$U(0) = \mathbb{I}. \tag{6.25}$$

At each time step n, the agent selects a control action, modifying the Hamiltonian parameters, and the evolution operator updates accordingly:

$$U_{n+1} = e^{-i\hat{H}_n \Delta t} U_n. \tag{6.26}$$

Fixed Time Step Evolution

The environment discretizes time evolution using a fixed time step Δt, which ensures consistency in the numerical integration of quantum dynamics, stable reinforcement learning training by avoiding issues with variable time steps, and efficient simulation using matrix exponentiation techniques, such as those implemented in SciPy. In this method, a scaling and squaring algorithm is combined with a Padé approximant: the input matrix is first scaled to reduce its norm, the exponential is then computed via a rational Padé approximation, and finally the result is repeatedly squared to reverse the scaling. This approach is well established for its efficiency and numerical stability.

6.3 Double Dueling Deep Q-Network (DDDQN) Implementation

DDDQN extends DDQN by incorporating the dueling network architecture, which separately estimates the state-value function $V(s)$, representing the overall usefulness of being in a state and the advantage function $A(s, a)$, which determines the relative benefit of selecting action a in state s.

The final Q-value is computed as:

$$Q(s,a) = V(s) + A(s,a) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a'), \tag{6.27}$$

which ensures better action selection and faster convergence [37].

DDDQN was implemented with a dueling network architecture with separate value and advantage estimators, a target network updated periodically to stabilize training, an epsilon-greedy exploration strategy, with epsilon decaying over time, and a Replay buffer with prioritized experience replay (PER) to improve sample efficiency [38].

DDDQN was applied to discrete control settings where a predefined set of pulse sequences was optimized.

6.4 Twin Delayed Deep Deterministic Policy Gradient (TD3) Implementation

TD3 is designed for continuous action spaces and improves DDPG by mitigating Q-value overestimation. One key modification is the use of double critic networks, where two Q-networks are employed and the minimum Q-value is selected:

$$y = R + \gamma \min_{i=1,2} Q_i(s', \mu(s')). \tag{6.28}$$

In addition, the actor network is updated less frequently than the critics, a strategy known as delayed policy updates. Furthermore, target policy smoothing is applied by adding noise to the target actions to prevent overfitting to sharp Q-value spikes [34].

6.5 Proximal Policy Optimization (PPO) Implementation

PPO is a policy gradient method that optimizes policies while preventing drastic updates by using a clipped objective:

$$L^{CLIP}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}\big(r_t(\theta), 1-\epsilon, 1+\epsilon\big)\hat{A}_t \right) \right], \tag{6.29}$$

where $r_t(\theta)$ denotes the probability ratio and $\hat{A}_t$ is the advantage estimate [26].

PPO was implemented using an actor-critic neural network architecture, where the actor outputs a policy distribution, and the critic estimates the value function. In addition, generalized advantage estimation (GAE) [39] was employed to reduce the variance in policy updates, combined with Monte Carlo estimation of returns. The clipped surrogate objective further prevents large policy updates that might destabilize training. PPO was applied to both discrete and continuous action spaces, and it was also implemented for RYY Gates [40].

6.6 Group Policy Relative Optimization (GRPO) Implementation

GRPO is a policy optimization algorithm that uses the relative performance of a group of candidate policies to guide more robust policy updates. In GRPO, the advantage function is estimated relative to a baseline of the group, which normalizes the reward signal and helps reduce the variance in the estimates of the policy gradients. This relative measure enables the algorithm to be less sensitive to large fluctuations in absolute returns and promotes more stable learning dynamics. GRPO is applicable to both discrete and continuous action spaces.

6.7 Comparison with GRAPE and CRAB

To validate the effectiveness of reinforcement learning (RL) methods for quantum gate optimization, the results will be compared with two well-established numerical optimal control techniques: Gradient Ascent Pulse Engineering (GRAPE) and Chopped Random Basis (CRAB) optimization.

6.7.1 Gradient Ascent Pulse Engineering (GRAPE)

GRAPE is a widely used numerical optimal control algorithm to find time-dependent control pulses that maximize the fidelity of a target quantum gate [41]. Optimizes the control function $u_k(t)$ by iteratively updating the pulse parameters using the gradient ascent:

$$u_k^{(n+1)}(t) = u_k^{(n)}(t) + \eta \frac{\delta F}{\delta u_k(t)}, \tag{6.30}$$

where η is the learning rate, F is the fidelity of the implemented gate and $\frac{\delta F}{\delta u_k(t)}$ is the gradient of fidelity with respect to the control field.

Its primary advantage is the rapid convergence to high-fidelity solutions due to the use of analytic gradients, which provides accurate updates during the optimization process. This makes GRAPE very effective for systems with relatively smooth control landscapes. However, GRAPE can be sensitive to local minima, and its performance often depends on having a good initial guess. Additionally, as the system size increases, the computation of gradients can become increasingly expensive.

6.7.2 Chopped Random Basis (CRAB)

CRAB is an optimization method that restricts the control function $u_k(t)$ to a finite basis of functions (such as Fourier or wavelet basis) and optimizes their coefficients using non-gradient-based techniques like Nelder-Mead or evolutionary algorithms [42]. The control field is expressed as:

$$u_k(t) = \sum_{m=1}^{M} c_m f_m(t), \tag{6.31}$$

where c_m are the optimization parameters and $f_m(t)$ are predefined basis functions (e.g. sine and cosine functions). CRAB is particularly useful in experimental quantum control, where pulse smoothness and bandwidth constraints must be respected. It allows flexible optimization without requiring gradient information, which makes it robust against experimental imperfections and noise. This randomness can help the optimization process avoid getting trapped in local minima, making the method more robust in complex or rugged control landscapes. A drawback of CRAB is that it may require a larger number of iterations to reach a high-fidelity solution, and its performance can be variable due to the stochastic nature of the basis. Moreover, in high-dimensional parameter spaces, CRAB might be less efficient unless the basis and other hyperparameters are carefully tuned.

6.7.3 Loss Function for GRAPE and CRAB

Instead of the fidelity, which will be our final metric, we define a loss function to measure the distance between a computed unitary U and a target unitary T, after canceling out any global phase difference. The procedure is as follows:

1. Trace Overlap: First, compute the trace overlap between T and U:

$$\text{overlap} = \text{Tr}\left(U_{\text{target}}^{\dagger} U\right).$$

This complex number contains information about the global phase difference between the two unitaries.

2. Global Phase Extraction: To extract the phase, define

$$\phi = \arg\Big(\text{overlap} + \epsilon\Big),$$

where $\epsilon = 10^{-8}$ is a small constant added to avoid numerical issues (e.g., division by zero).

3. Phase Cancellation: Cancel the global phase by computing the phase factor

$$e^{-i\phi},$$

and then define the phase-aligned unitary as

$$U_{\text{aligned}} = U\, e^{-i\phi}.$$

4. Loss Calculation: Next, define the difference between the target unitary and the aligned unitary:

$$\Delta = U_{\text{target}} - U_{\text{aligned}}.$$

The loss is given by the square of the Frobenius norm of this difference:

$$\mathcal{L} = \|\Delta\|_F^2 = \left\|U_{\text{target}} - U\, e^{-i\phi}\right\|_F^2.$$

This loss function is used in the GRAPE algorithm to optimize the control pulses so that the computed unitary U approaches the target T, up to a physically irrelevant global phase.

6.7.4 Reinforcement Learning vs. GRAPE and CRAB

While GRAPE and CRAB provide powerful numerical optimization techniques, RL-based quantum control offers several advantages. For example, RL supports adaptive learning by dynamically discovering optimal pulse sequences rather than relying on static optimization procedures. In addition, RL is more robust to noise because it can generalize across different hardware noise models, whereas GRAPE and CRAB typically optimize for specific conditions. Finally, RL enables exploration beyond local optima; unlike GRAPE, which relies on local gradient updates, RL explores a broader range of control solutions.

The results obtained from the optimization of the quantum gate based on RL will be benchmarked against GRAPE and CRAB to assess fidelity, robustness, and computational efficiency.

6.8 Parameterization and Tuning of RL Algorithms

Each RL agent was run with the following parameters:

General

Parameter	**Value**
HIDDEN_FEATURES	64 (H,T) — 256 (CNOT)
NUM_HIDDEN_LAYERS	4
DROPOUT	0.1
BATCH_SIZE	64 (H,T) — 256 (CNOT)
GAMMA	0.95
TOTAL_TIME	1
MAX_STEPS	10 (H, T) — 5 (CNOT)
TIME_DELTA	1/9 (H) — 1/10 (T) — 1/5 (CNOT)

Optimizer

Parameter	**Value**
OPTIMIZER_TYPE	AdamW
SCHEDULER_LEARNING_RATE	0.001
WEIGHT_DECAY	1e-3

Loss

Parameter	Value	
LOSS_TYPE	MSE	(MSE/HUBER)
HUBER_DELTA_MAX	10.0	
HUBER_DELTA_MIN	1.0	
HUBER_DELTA_DECAY_RATE	1e-4	

Train

Parameter	Value
EPISODES	200000
PATIENCE	30000
FIDELITY_THRESHOLD	0.999
WINDOW_SIZE	100

DDDQN

Parameter	Value
MEMORY_SIZE	100000
MAX_EPSILON	1.0
MIN_EPSILON	1e-10
EPSILON_DECAY_RATE	1e-3
TARGET_UPDATE	100

PPO/GRPO

Parameter	Value
CLIP_EPSILON	0.2
ENTROPY_COEFF	0.01
VALUE_COEFF	0.5
LAMBDA	0.95
EPOCHS_PPO	4
TIMESTEPS	70 (H,T) — 80 (CNOT)

TD3

Parameter	Value
NOISE	0.1
POLICY_NOISE	0.2
TAU	0.005
NOISE_CLIP	0.5

6.8.1 Dropout

Dropout is a regularization technique used to prevent overfitting in neural networks by randomly disconnecting nodes during training, which effectively reduces weight magnitudes and performs model averaging. Although originally developed for classical supervised learning, a quantum analog of dropout has been proposed to reduce circuit complexity [40]. In the context of deep reinforcement learning (DRL) for quantum control, dropout is used to emulate systematic errors by randomly disconnecting nodes, rather than explicitly modeling error parameters in the reward function. Once the model converges, dropout is disabled to obtain a robust control pulse, enhancing gate performance and robustness without human intervention [43].

6.8.2 Implementation Environment

The implementation of quantum gate optimization was carried out on a Linux-based system with a dedicated NVIDIA GPU, using Python 3.12.7 and PyTorch 2.6.0 for deep learning computations. The experiments were designed to leverage CUDA acceleration for efficient training of the RL agents. The simulations were conducted using PyTorch [44] to implement deep reinforcement learning agents. All agents were written from scratch and do not rely on any other high-level libraries. PyTorch was integrated to optimize quantum control parameters and evaluate the performance of reinforcement learning models.

System Configuration

The computational setup used for the implementation is as follows:

- Operating system: Linux Kernel 6.11.5-1-liquorix-amd64
- Processor: x86_64, 64-bit architecture
- CPU Cores: 6 physical cores, 12 logical threads
- CPU Frequency: 2601.00 MHz
- Total RAM: 31.00 GB

Software Stack

The RL framework and quantum control simulations were implemented using the following software versions:

- Python Version: 3.12.7
- PyTorch Version 2.6.0+cu126
- CUDA Version: 12.6

- cuDNN Version: 9.0.1
- NVIDIA Compiler (nvcc): CUDA 12.6, Build cuda_12.6.r12.6/compiler.34714021_0

GPU Configuration

A dedicated NVIDIA GPU was used to accelerate the reinforcement learning training process. The GPU specifications are:

- GPU Model: NVIDIA GeForce GTX 1650 Ti
- Driver Version: 560.35.03
- GPU Memory: 4 GB
- CUDA Cores: GTX 1650 Ti features 1024 CUDA cores.

Training Hardware Acceleration

The reinforcement learning agents were trained with GPU acceleration using PyTorch's CUDA back-end, allowing parallelized matrix operations and efficient tensor computations. The NVIDIA GTX 1650 Ti system was utilized for gradient calculations and deep reinforcement learning model updates.

Parallelization and Computational Considerations

Large training batches were processed using GPU acceleration to accelerate the training of RL agents. Memory management was optimized to ensure that GPU memory usage did not exceed 4GB, which maintained stability during training. Additionally, due to the limited memory of the GTX 1650 Ti, efficient memory allocation was essential to prevent the execution of very large models.

Reproducibility

All simulations were performed on the specified system, and the configurations were saved to ensure reproducibility. The same software stack and hardware setup was maintained throughout the project to prevent system-dependent variations in performance.

6.9 Challenges and Lessons Learned

Several challenges were encountered: Several challenges were encountered. Interactions with the environment proved costly because it was necessary to propagate the identity operator with each new set of actions. In addition, the computational complexity was significant, with PPO and TD3 requiring higher computational resources; moreover, computing single-qubit gates is less expensive than computing two-qubit gates.

The source code for this work is available at the GitHub repository: `https://github.com/contepablod/Quantum_Control_Reinforcement_Learning`.

Chapter 7

Results and Analysis

7.1 Performance Metrics: Fidelity and Efficiency

To evaluate the effectiveness of reinforcement learning agents, two main performance metrics were considered. The first is fidelity, F (4.1) which measures the accuracy of the implemented quantum gate compared to the target gate and is defined. The second metric is logarithmic infidelity L (6.10) that is directly related to fidelity. Moreover, related to fidelity, we also have the average process fidelity $\bar{F}$ (4.2). In addition, efficiency was measured in terms of computational complexity, training time, and convergence rate. The performance of DDDQN, PPO, GRPO, and TD3 was analyzed for the implementation of the H, T, and CNOT gates in different quantum control environments. Note that DDDQN uses a discrete set of actions (similar to DPPO and DGRPO), while CPPO, CGRPO, and TD3 use a continuous set of actions. Furthermore, the logarithm of infidelity is displayed as a positive value in the graphs, as the reward is calculated accordingly.

7.2 Comparison of RL Algorithms for Quantum Gate Optimization

Each reinforcement learning agent was evaluated on the basis of convergence speed and stability, as well as the fidelity achieved across multiple quantum gates.

7.2.1 Single-Qubit Gates (H and T)

The optimization of single-qubit gates focused on pulse control for the Hadamard (H) and T gates.

Hadamard Gate

We present the fidelities achieved by the different agents and thus their control pulse outcomes, respectively.

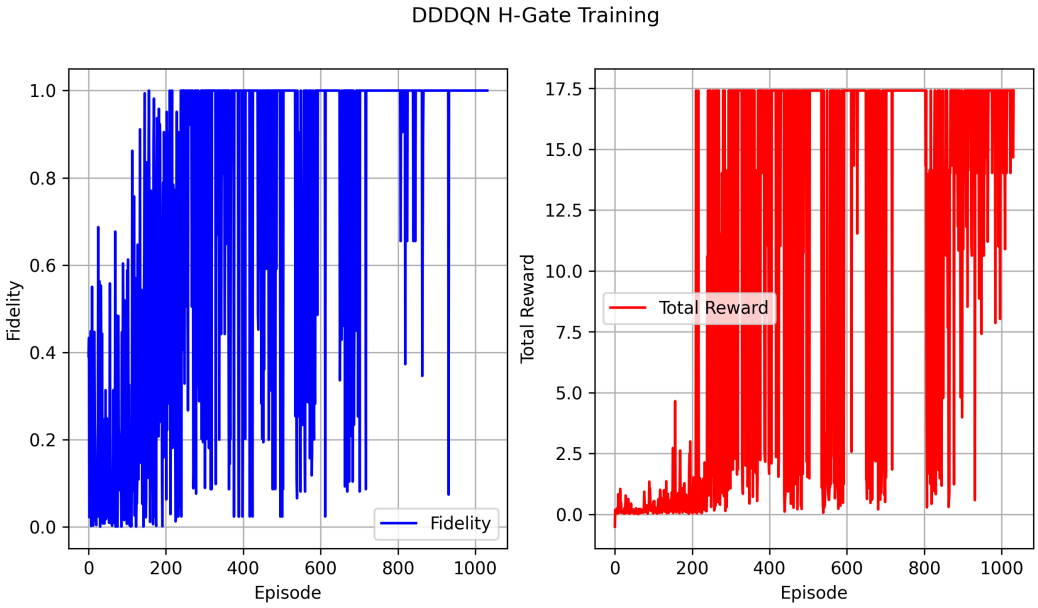

Figure 7.1: DDDQN Agent H-Gate Fidelities and Total Reward

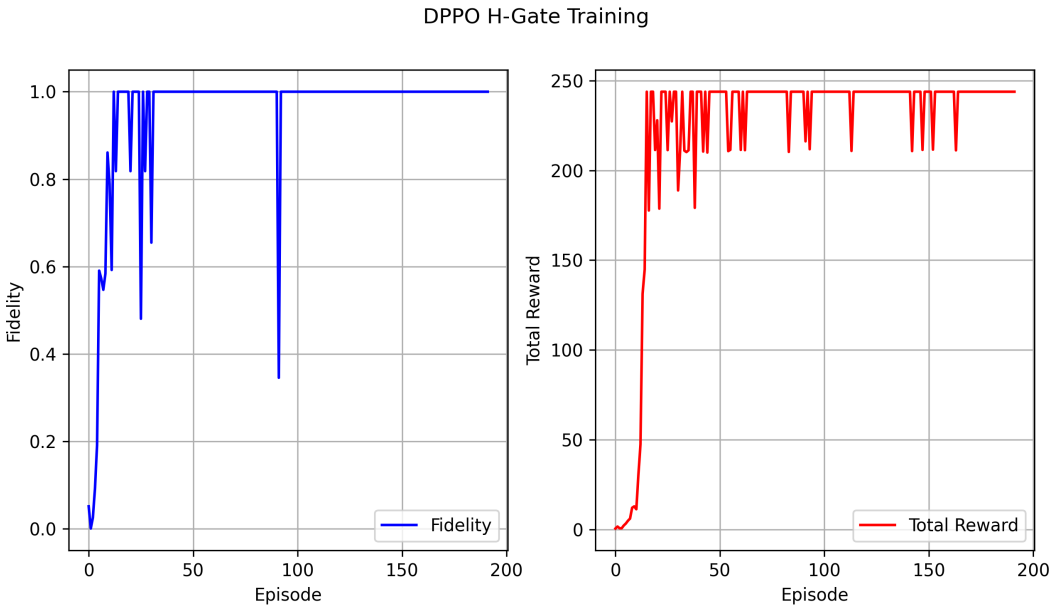

Figure 7.2: Discrete PPO Agent H-Gate Fidelities and Total Reward

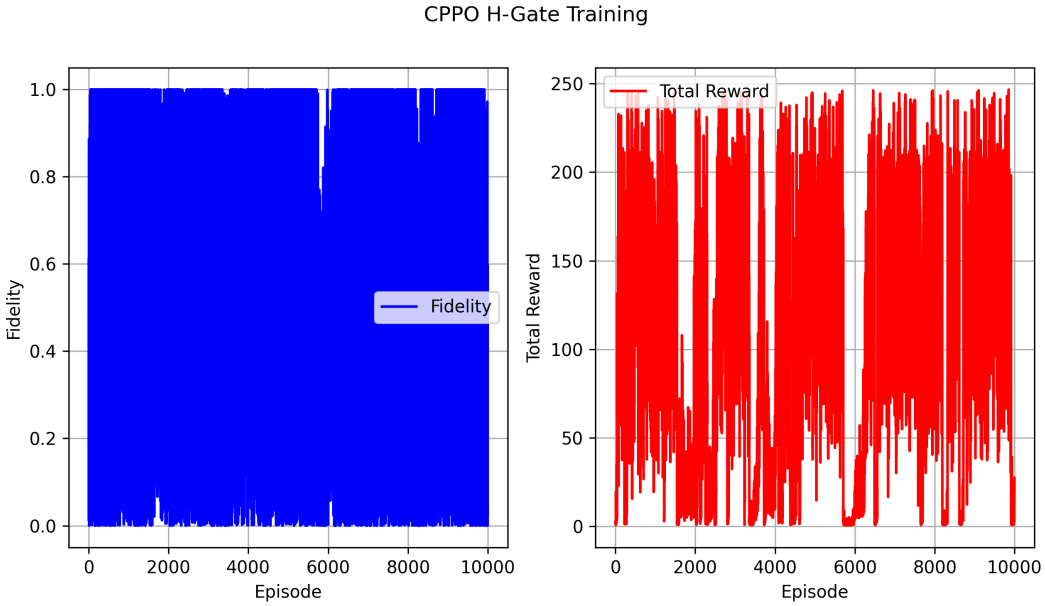

Figure 7.3: Continuous PPO Agent H-Gate Fidelities and Total Reward

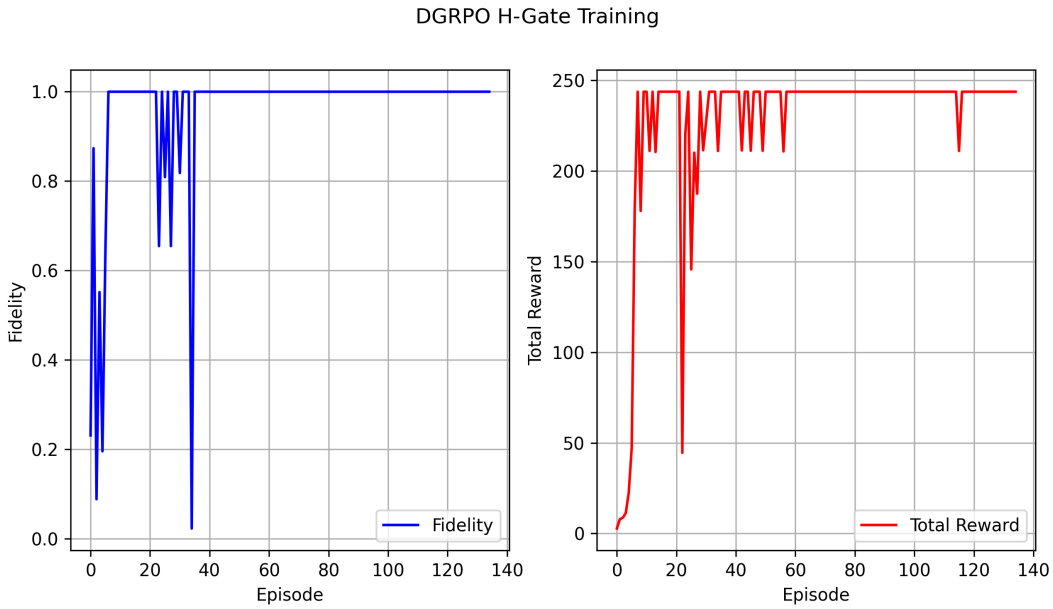

Figure 7.4: Discrete GRPO Agent H Gate Fidelities and Total Reward

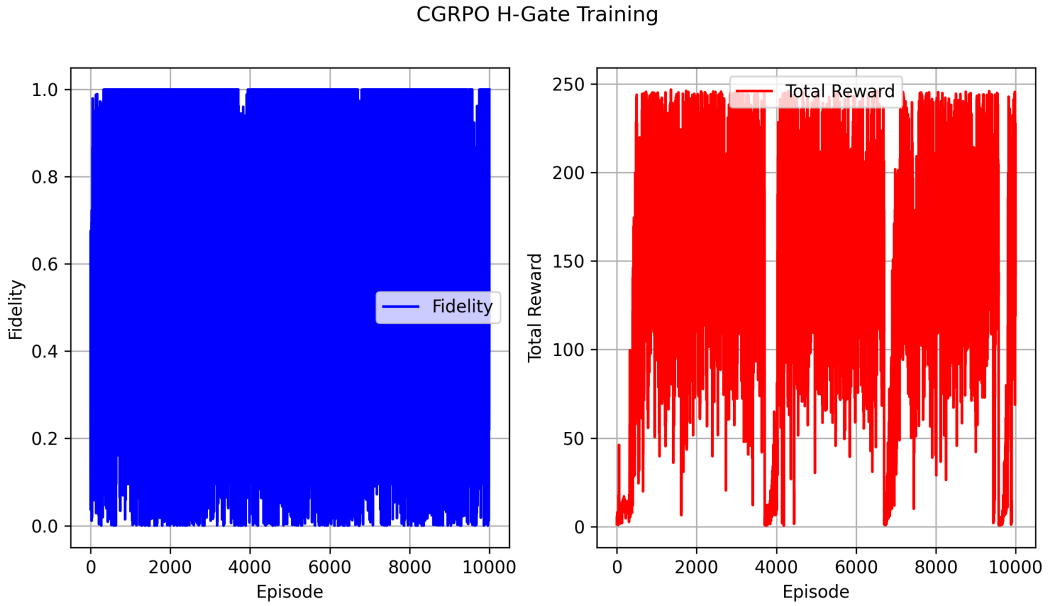

Figure 7.5: Continuous GRPO Agent H Gate Fidelities and Total Reward

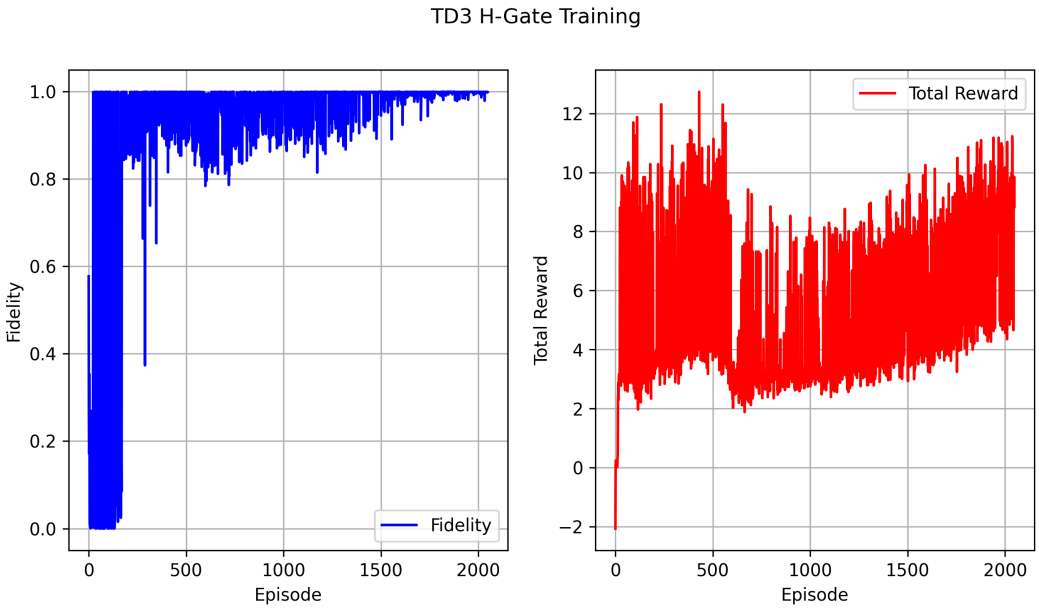

Figure 7.6: TD3 Agent H Gate Fidelities and Total Reward

Now, we extract the trajectory for each agent for maximum fidelity. In the following plot, agents with a discrete set of actions are depicted using dashed lines, while those with a continuous set of actions are represented by solid lines. In addition, a circular marker denotes a deterministic agent, and a square marker indicates a stochastic one.

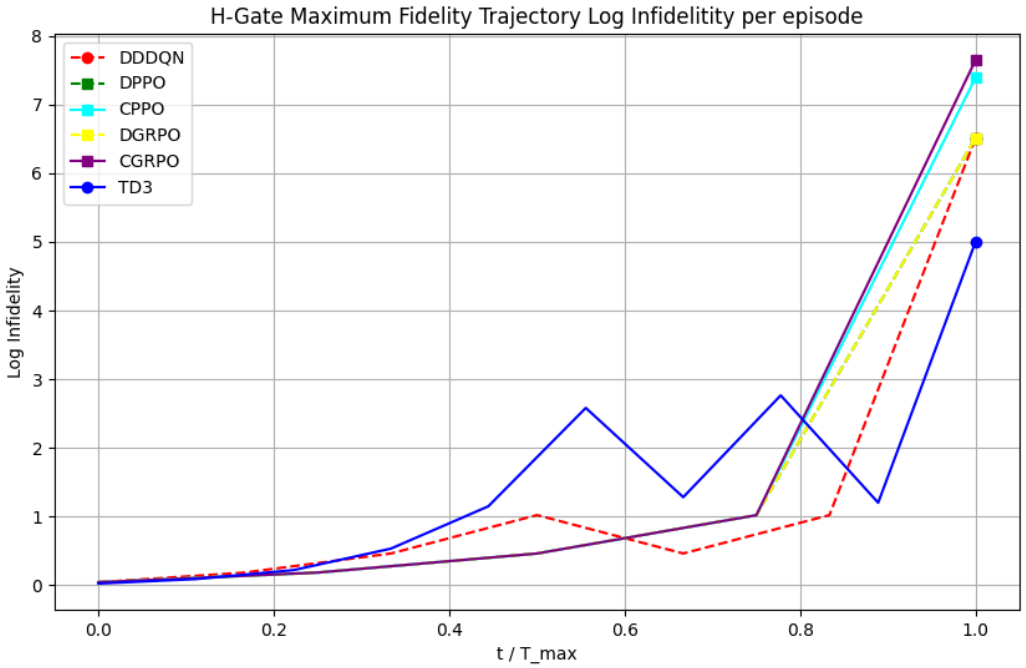

Figure 7.7: H Gate Normalized Maximum Fidelity Trajectory Log Infidelity for every agent

We show the maximum fidelity, log infidelity, average fidelity and time steps values for each agent.

	Agent	Max Fidelity	Max Log Infidelity	Avg Fidelity	Time Steps
5	TD3	0.999990	5.005089	0.999993	10
1	DPPO	1.000000	6.516003	1.000000	5
3	DGRPO	1.000000	6.516003	1.000000	5
0	DDDQN	1.000000	6.516003	1.000000	7
2	CPPO	1.000000	7.395267	1.000000	5
4	CGRPO	1.000000	7.650909	1.000000	5

Table 7.1: H-Gate Fidelities Comparison

Our simulations revealed that several agents, specifically TD3, DPPO, DGRPO and DDDQN, exhibit robust performance in their respective discrete and continuous control settings, achieving average fidelities of **1.000000**. This behavior is consistent with the literature that indicates that deterministic policies often provide enhanced stability in control tasks [45, 46]. DGRPO and DPPO, in their discrete control pulse settings, also performed in the same line achieving fidelities of **1.000000**, respectively.

The performance of continuous stochastic agents, such as CGRPO and CPPO, reflects a more complex dynamic. Continuous action formulations in Proximal Policy Optimization can more effectively leverage gradient-based updates for fine-grained control. However, inherent stochasticity, in which the policy may select different actions in identical states, can introduce instability during training, a well-known trade-off in policy gradient methods.

CPPO, CGRPO and TD3 both work in continuous action spaces, but they differ significantly in how they update policies and handle variance in the learning process. Some reasons why PPO might struggle to converge when optimizing parameters while TD3 succeeds are: PPO is an on-policy method, meaning it updates its policy based solely on trajectories generated from the current policy. This can lead to high variance in the gradient estimates. In contrast, TD3 is an off-policy method, which reuses past experiences. This can lead to more stable and sample-efficient updates, particularly in continuous control problems. Further, PPO uses a clipped surrogate objective to ensure that policy updates do not change the policy too much in one step. While this stabilizes training in many cases, it can also make the learning process overly conservative. If the reward landscape is steep or has narrow optima, the clipping may prevent PPO from making the necessary adjustments to converge. Moreover, TD3 relies on deterministic policy gradients, which can be more effective in continuous settings where fine-tuning of parameters is required. The deterministic nature of the policy update, along with techniques like target policy smoothing and double Q-learning, helps TD3 avoid overestimation bias and oscillations in value estimates. PPO's on-policy updates might suffer if the exploration noise is not well-tuned, leading to insufficient explo-

ration of the continuous space. TD3 incorporates noise directly into the target policy (target policy smoothing) to encourage better exploration and more robust learning, which can be particularly beneficial when optimizing fine control parameters.

In the following page, we show the optimal control pulses for the minimum time step length plotted above.

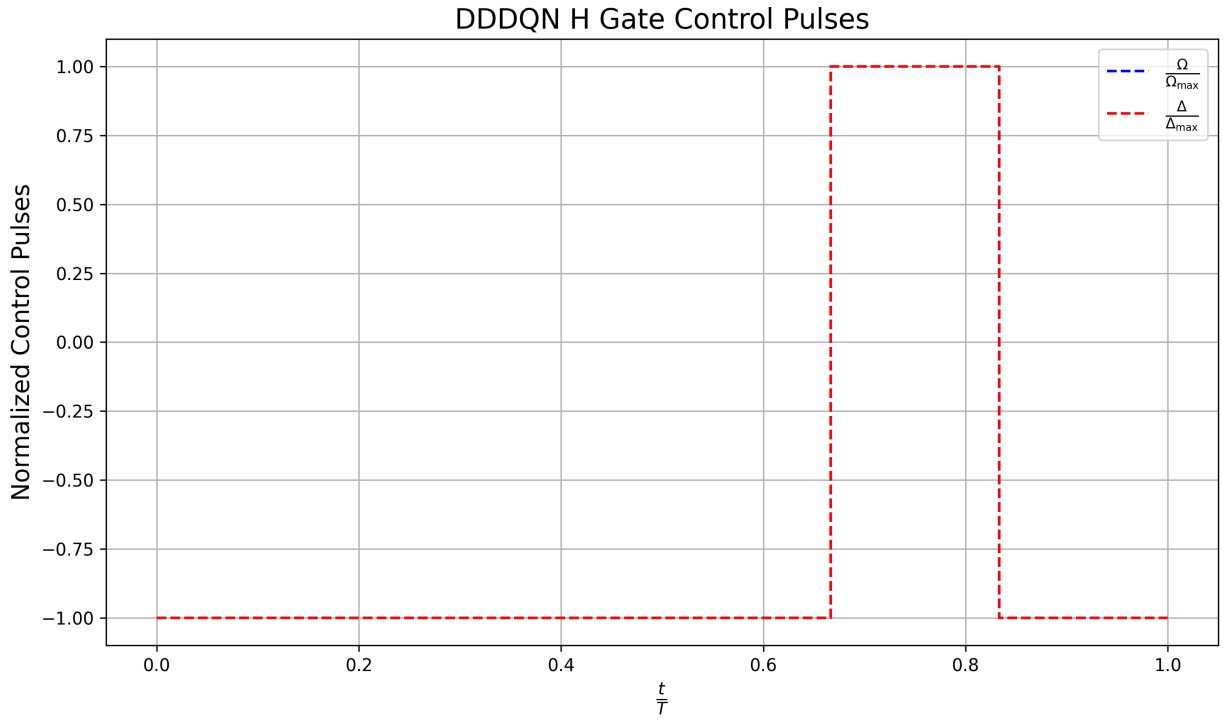

Figure 7.8: DDDQN H Gate Control Pulses

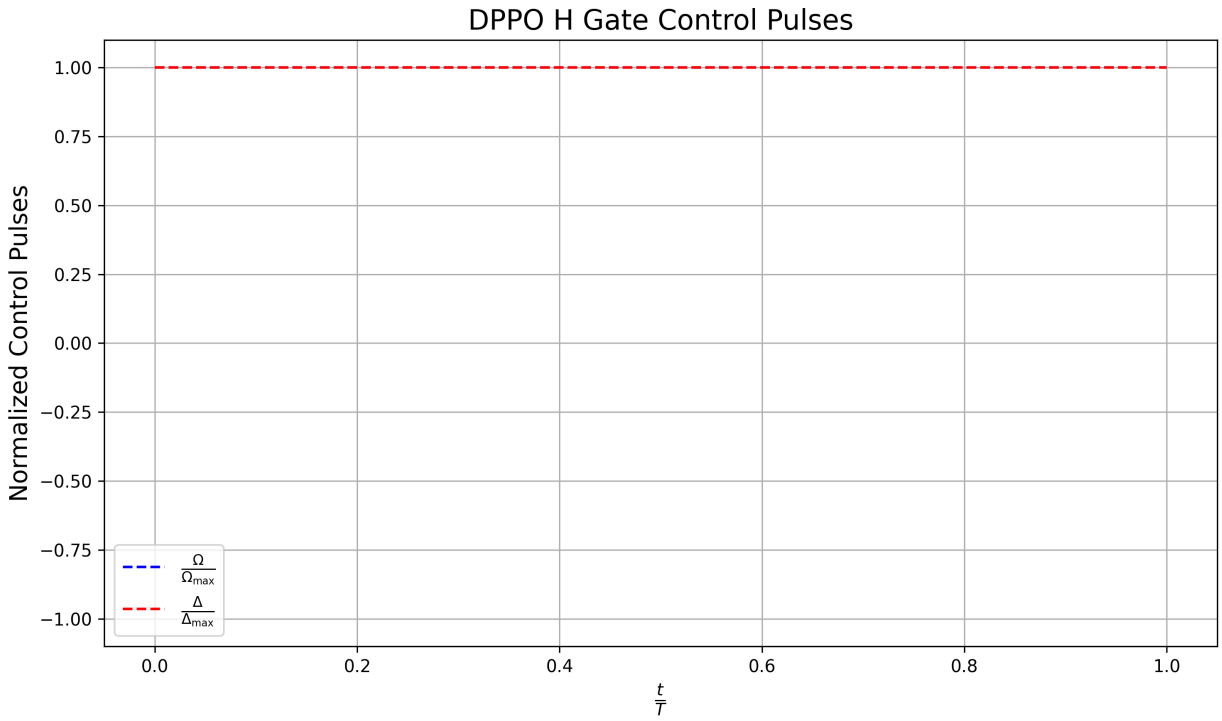

Figure 7.9: Discrete PPO H Gate Control Pulses

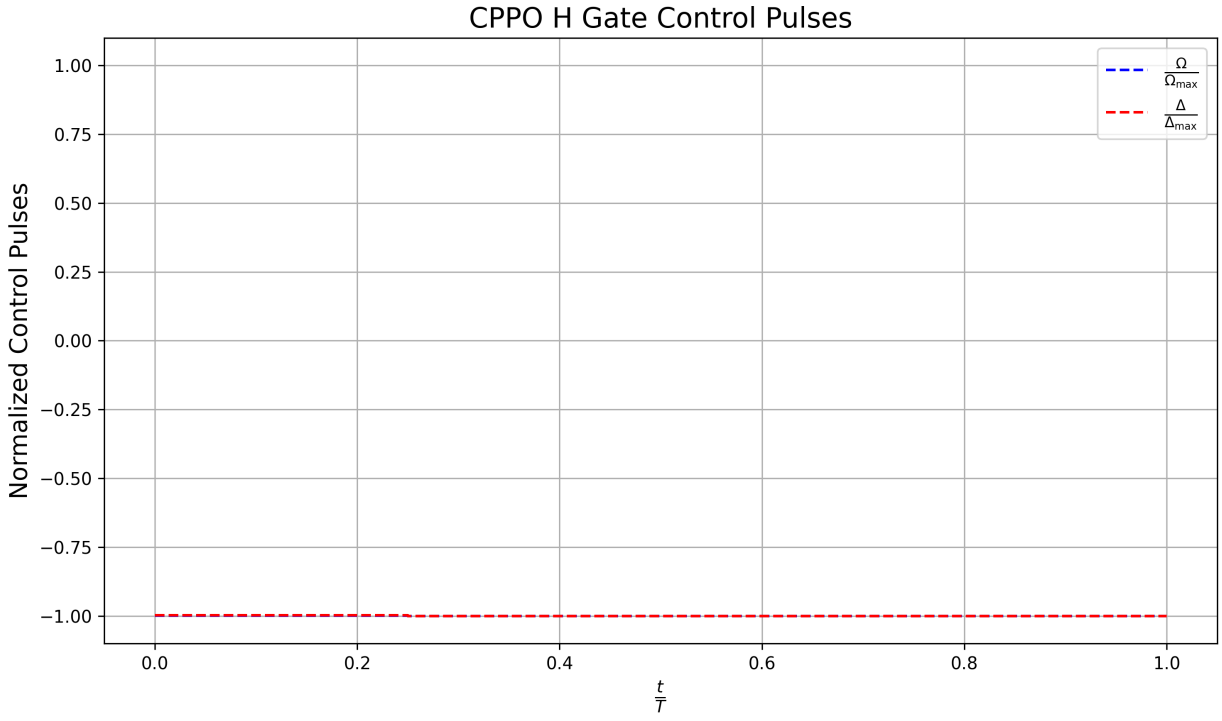

Figure 7.10: Continuous PPO H Gate Control Pulses

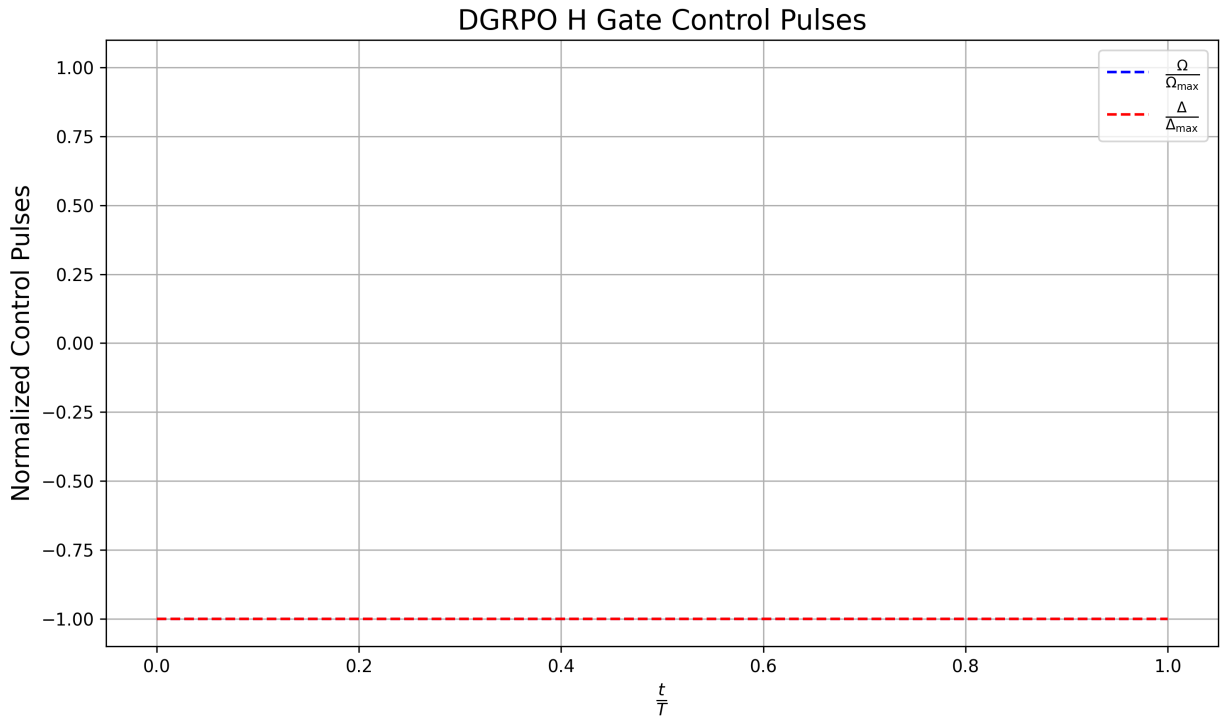

Figure 7.11: Discrete GRPO H Gate Control Pulses

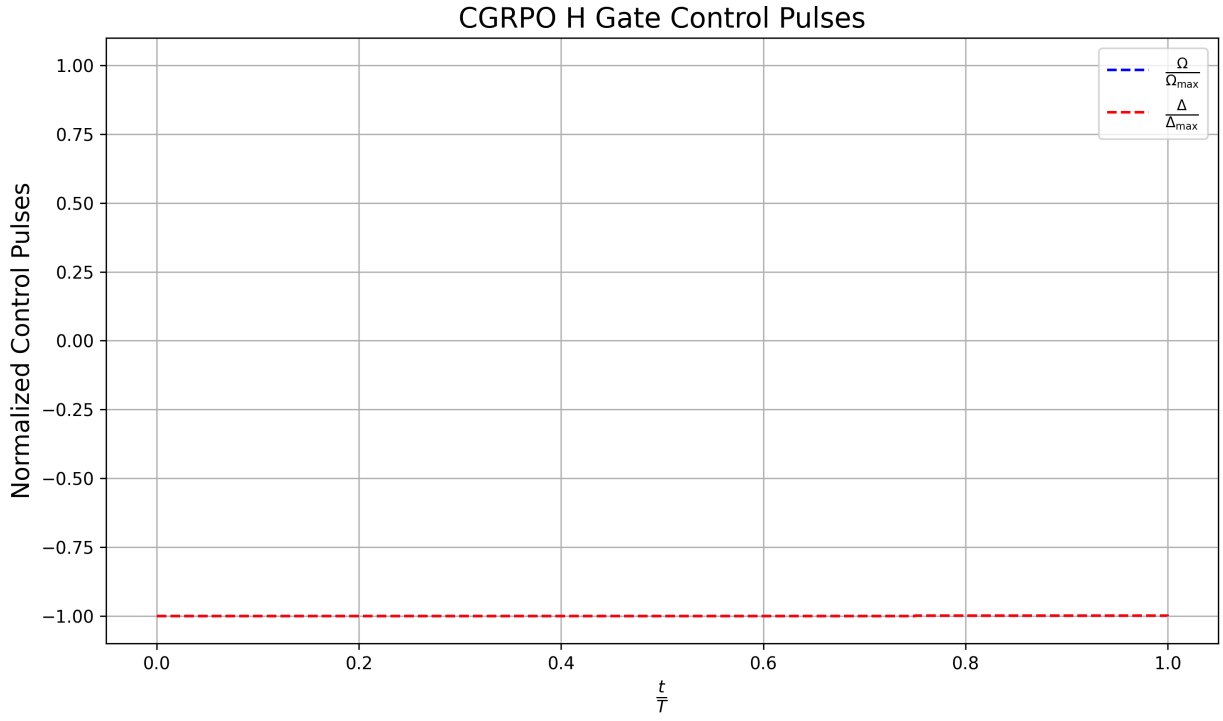

Figure 7.12: Continuous GRPO H Gate Control Pulses

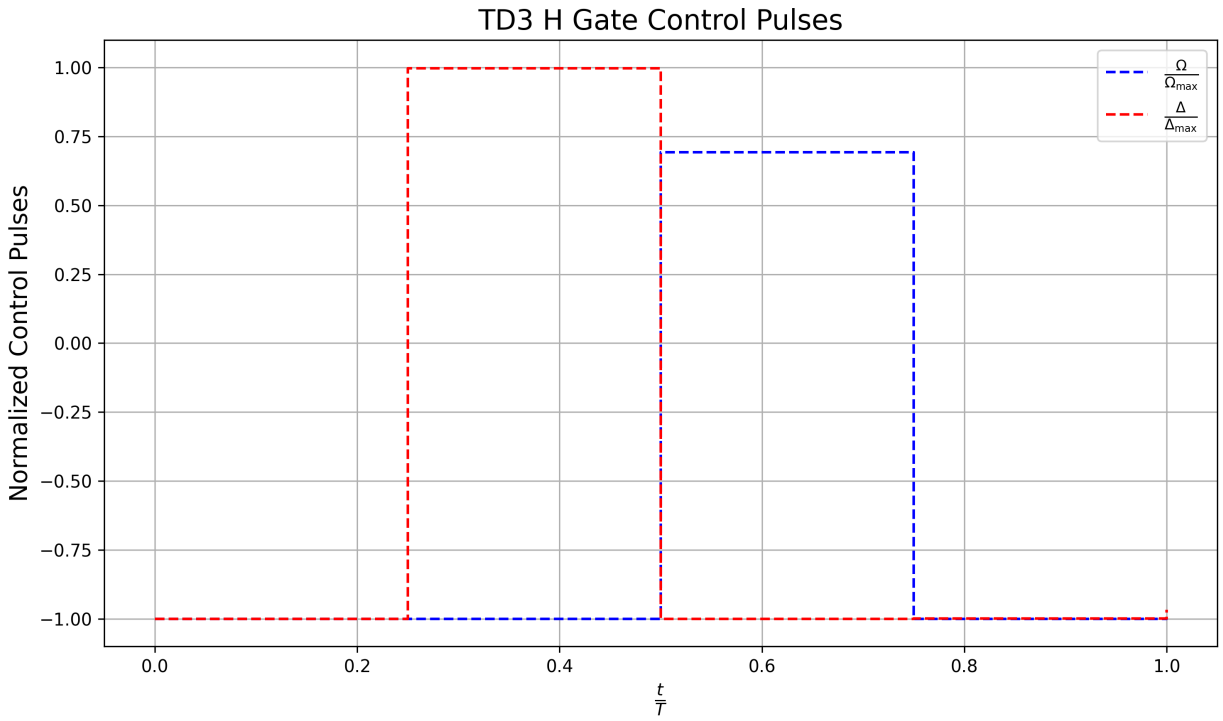

Figure 7.13: TD3 H Gate Control Pulses

T Gate

Therefore, we present the fidelities achieved by the different agents, and thus, their control pulse outcomes, respectively, for the T gate.

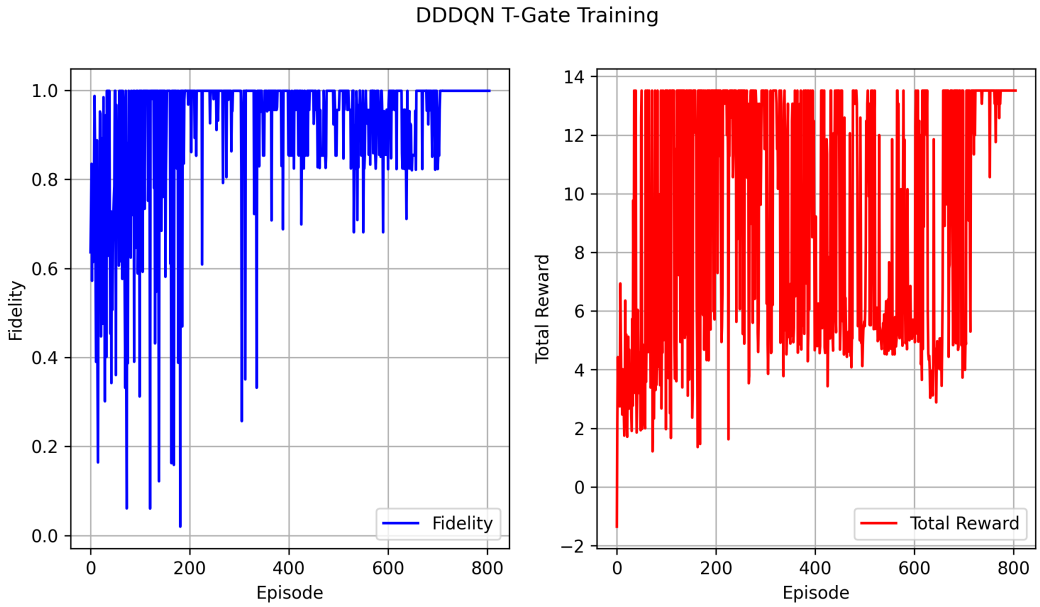

Figure 7.14: DDDQN Agent T-Gate Fidelities and Total Reward

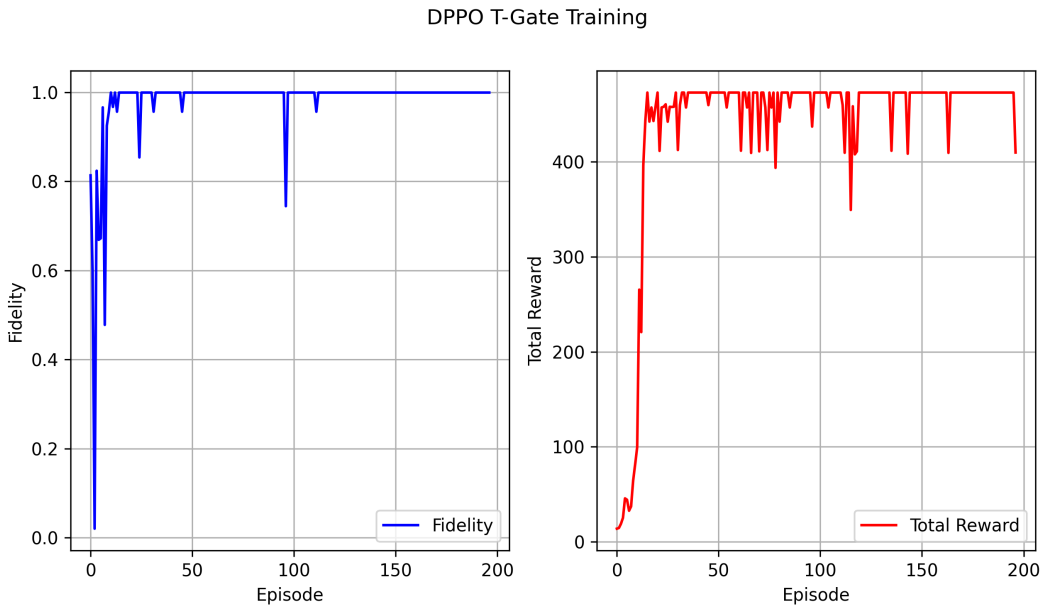

Figure 7.15: Discrete PPO Agent T-Gate Fidelities and Total Reward

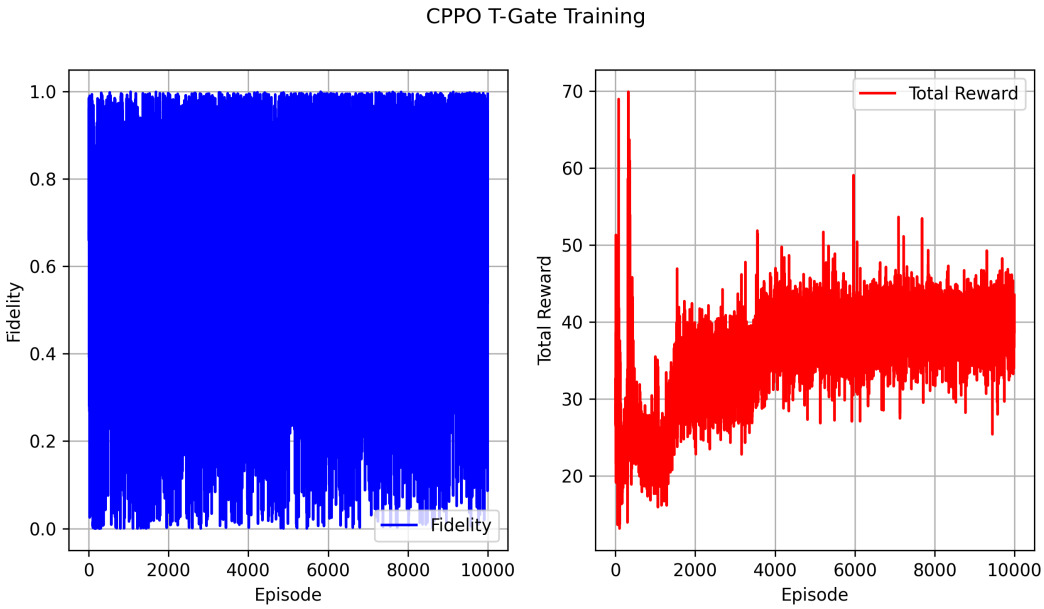

Figure 7.16: Continuous PPO Agent T-Gate Fidelities and Total Reward

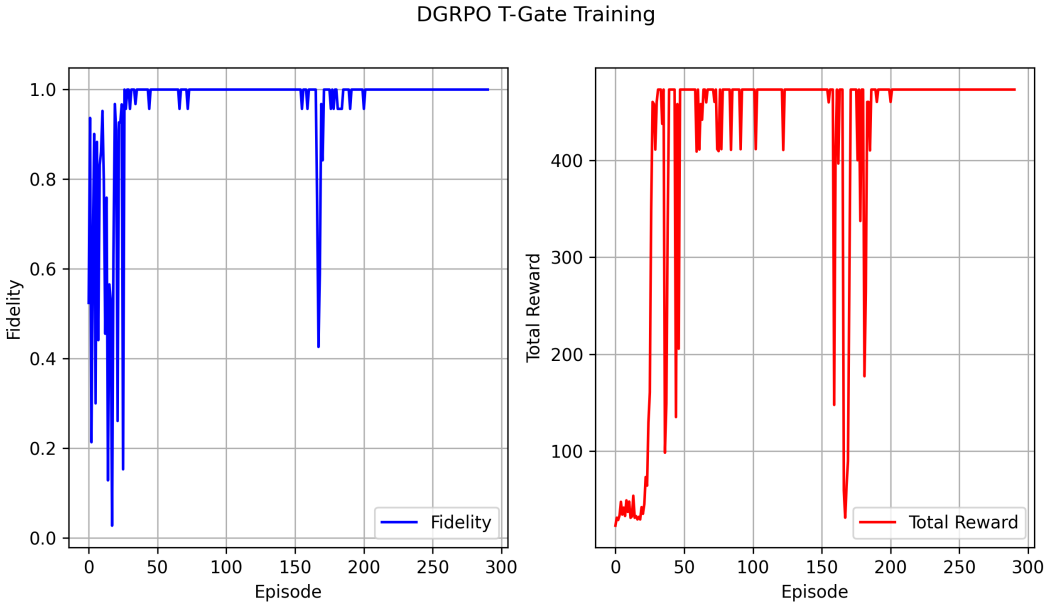

Figure 7.17: Discrete GRPO Agent T-Gate Fidelities and Total Reward

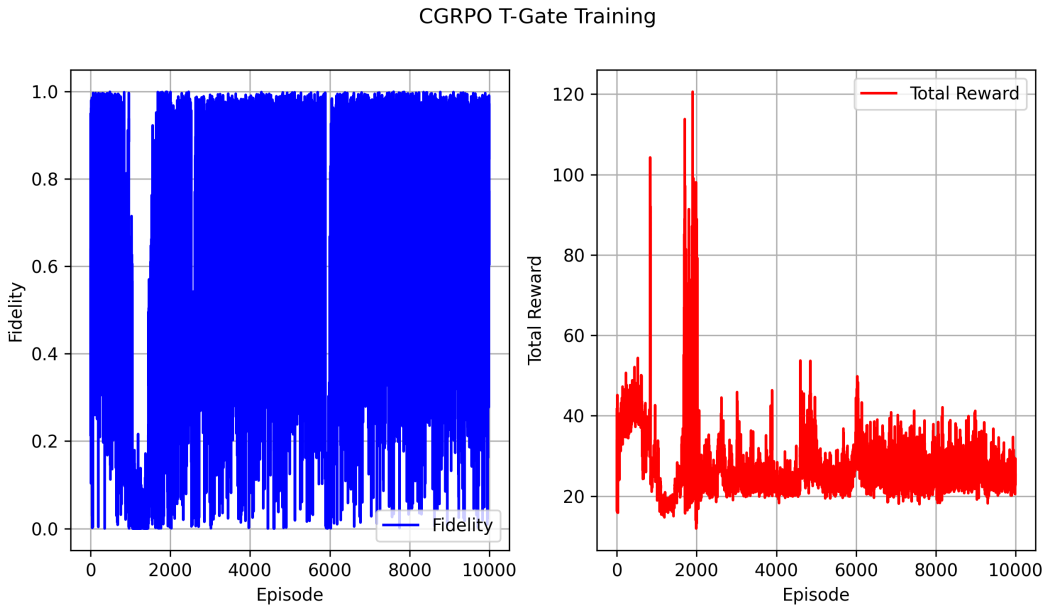

Figure 7.18: Continuous GRPO Agent T-Gate Fidelities and Total Reward

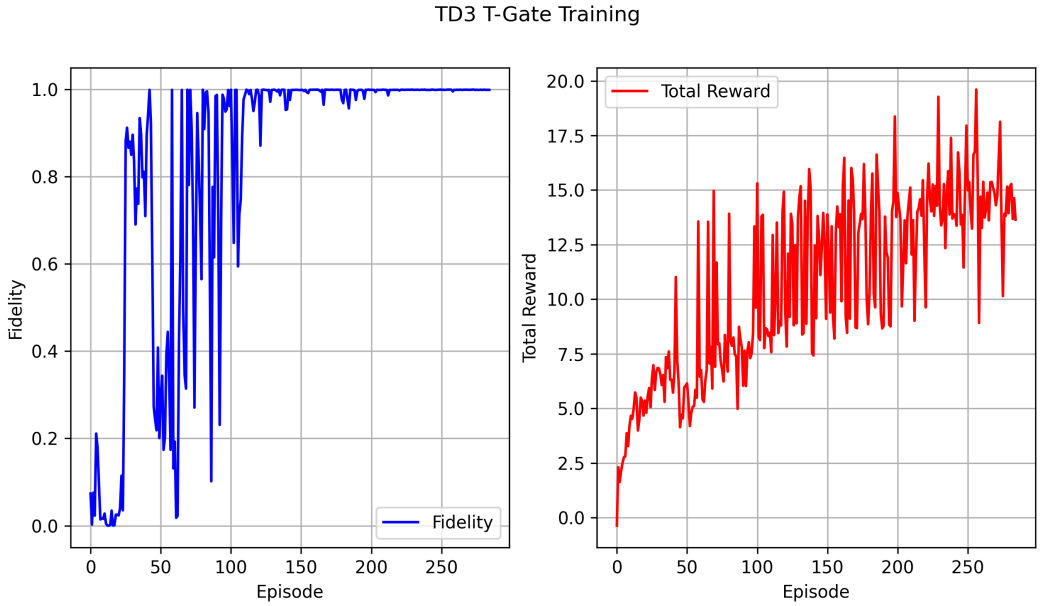

Figure 7.19: TD3 Agent T-Gate Fidelities and Total Reward

The comparison of log infidelity for the maximum fidelity trajectory is presented.

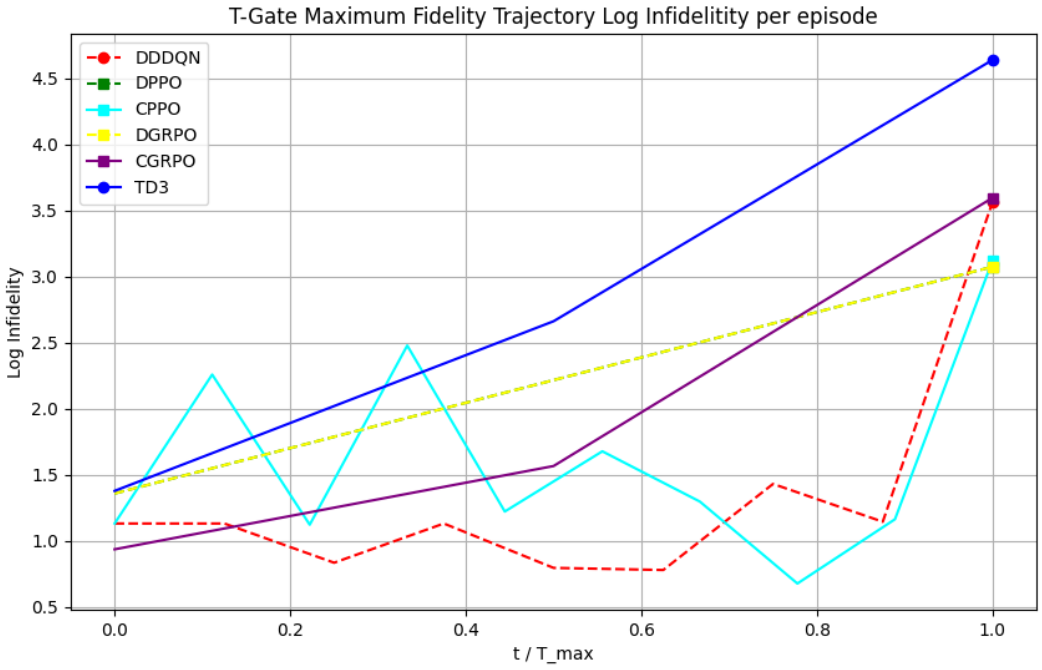

Figure 7.20: T-Gate Normalized Maximum Fidelity Trajectory Log Infidelity for every agent

We show the maximum fidelity, log infidelity, average fidelity and time step values for each agent.

	Agent	Max Fidelity	Max Log Infidelity	Avg Fidelity	Time Steps
1	PPO-D	0.999156	3.073450	0.999437	2
3	GRPO-D	0.999156	3.073450	0.999437	2
2	PPO-C	0.999240	3.119451	0.999494	10
0	DDDQN	0.999726	3.562547	0.999817	9
4	GRPO-C	0.999746	3.594970	0.999831	3
5	TD3	0.999977	4.638075	0.999985	3

Table 7.2: T Gate Fidelities Comparison

As in the H-Gate, the simulations revealed the same pattern in deterministic approaches: stable learning achieving high fidelities. For DPPO and DGRPO, we also have a robust learning curve. Again, continuous stochastic methods struggle to achieve robust learning, yielding a noisy curve.

We show the optimal control pulses obtained by the agents. For some agents achieving stable learning curves, there is no driving, and the detuning has a fixed value. This is advantageous, as the gate matrix contains only diagonal terms, with the off-diagonal elements being zero.

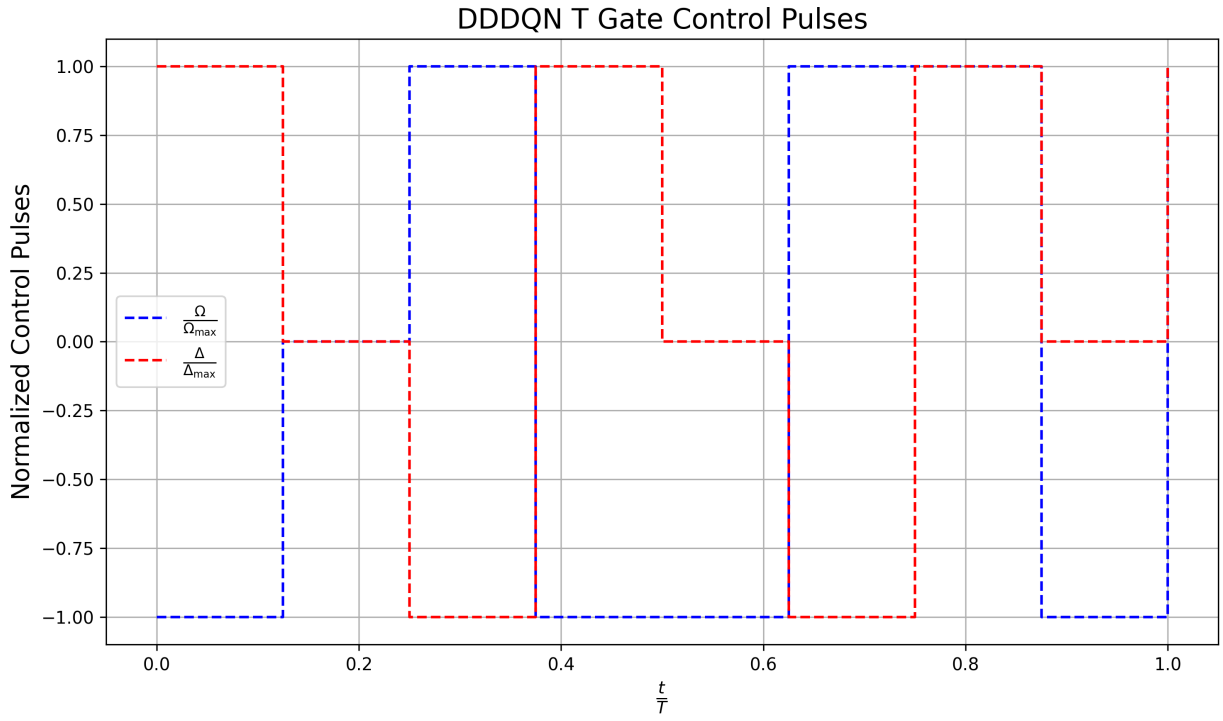

Figure 7.21: DDDQN T Gate Control Pulses

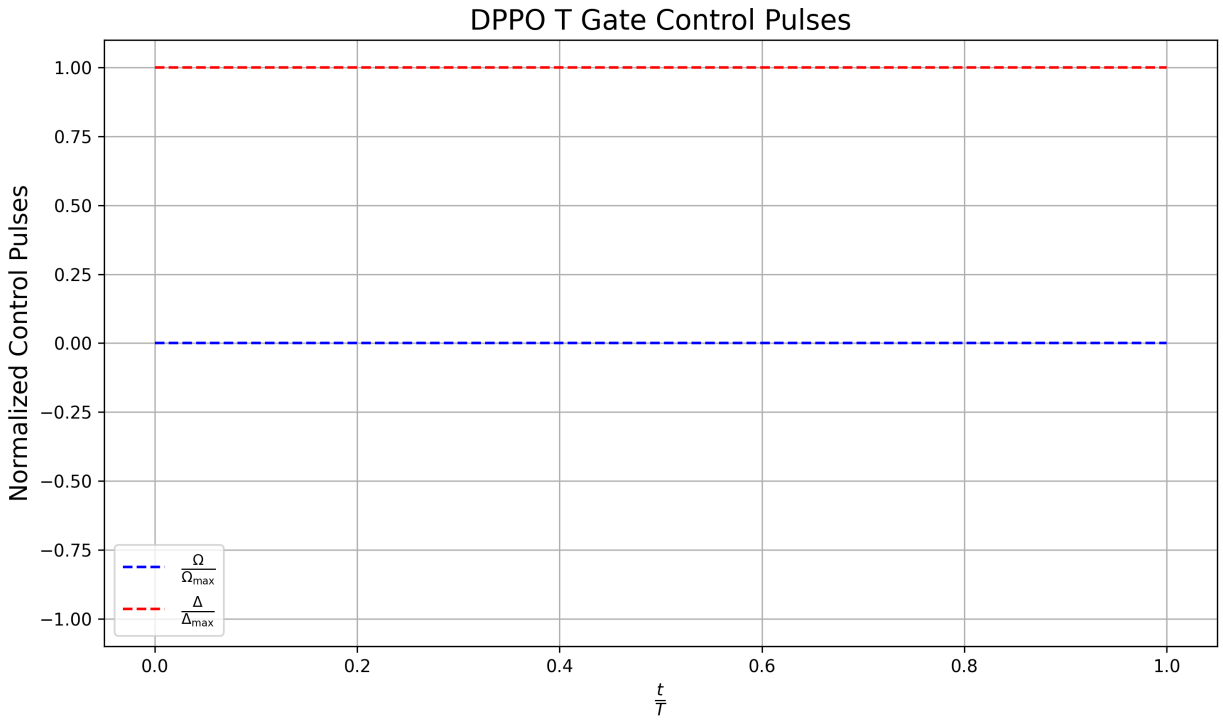

Figure 7.22: Discrete PPO T Gate Control Pulses

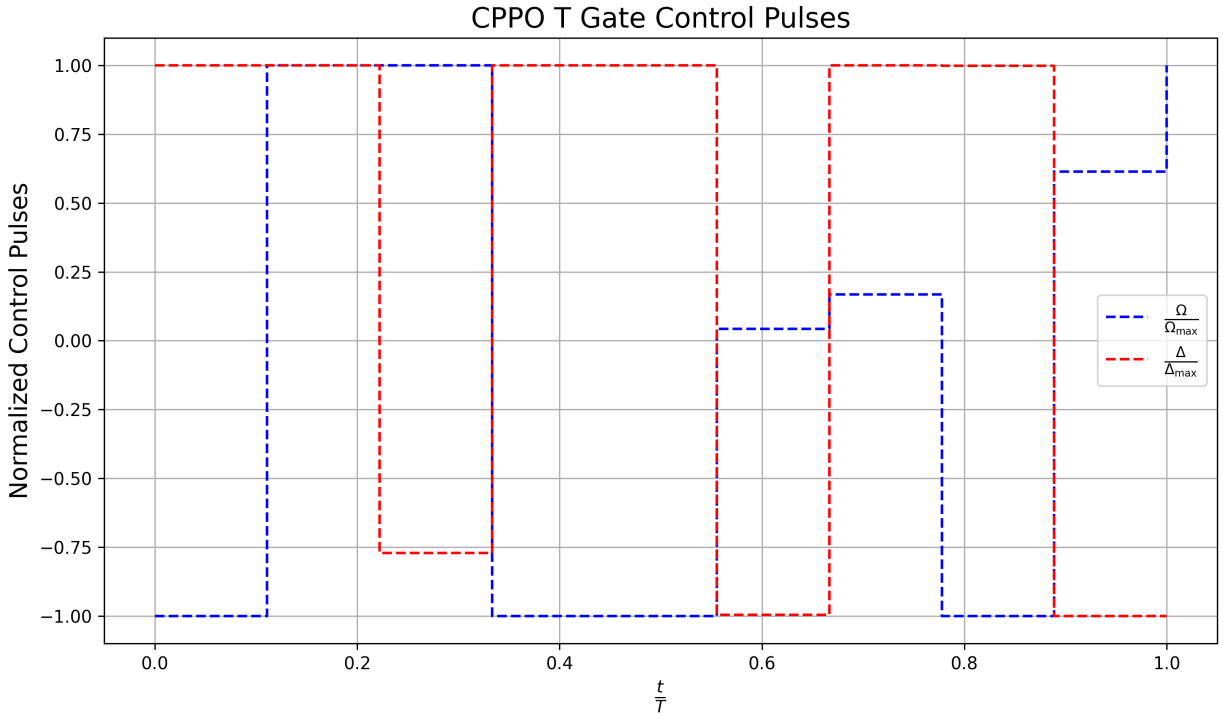

Figure 7.23: Continuous PPO T Gate Control Pulses

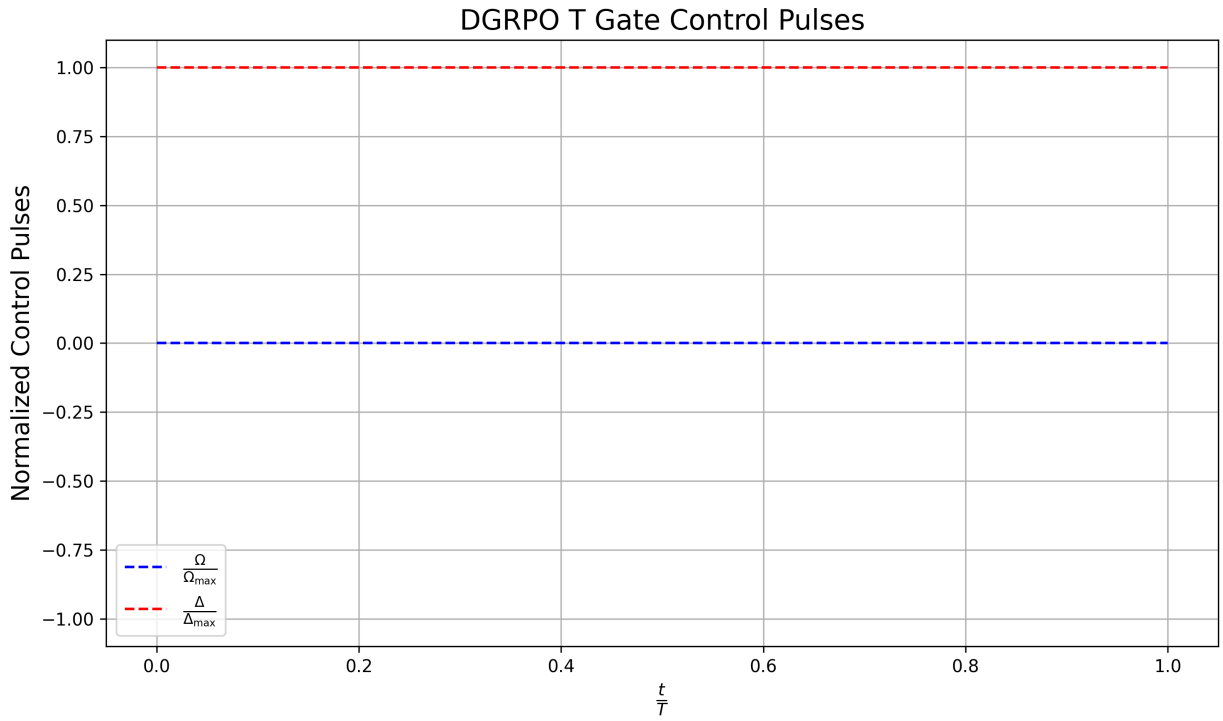

Figure 7.24: Discrete GRPO T Gate Control Pulses

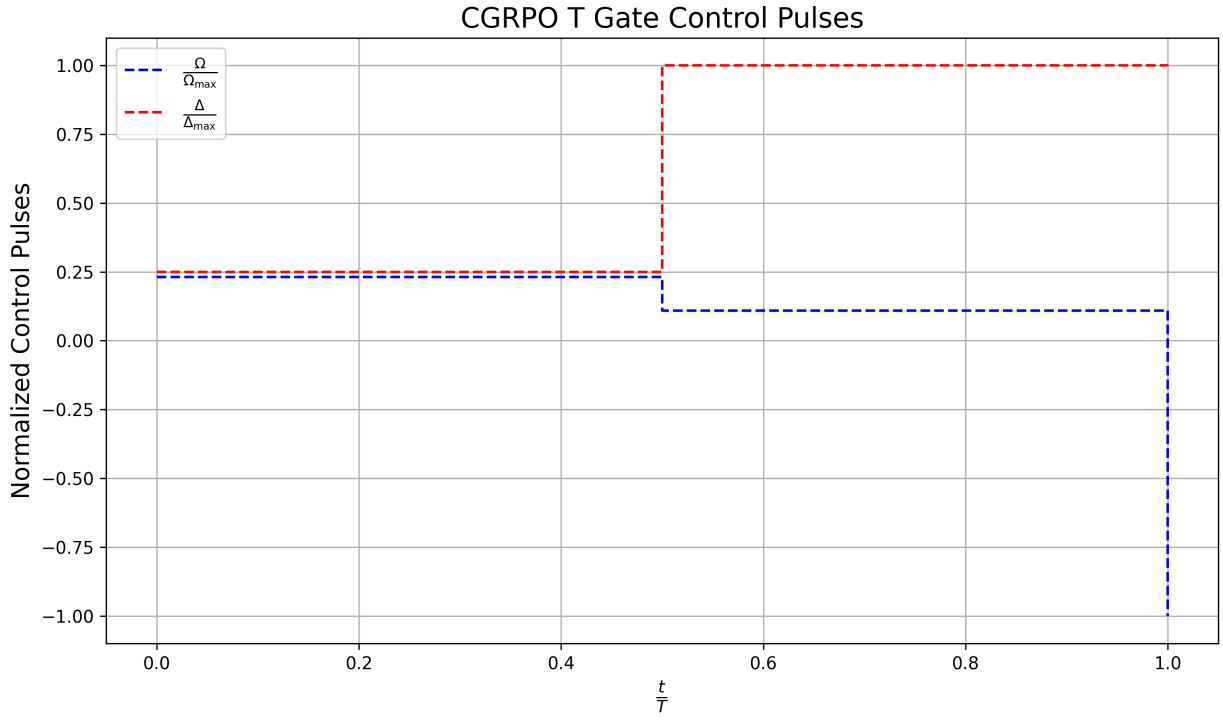

Figure 7.25: Continuous GRPO T Gate Control Pulses

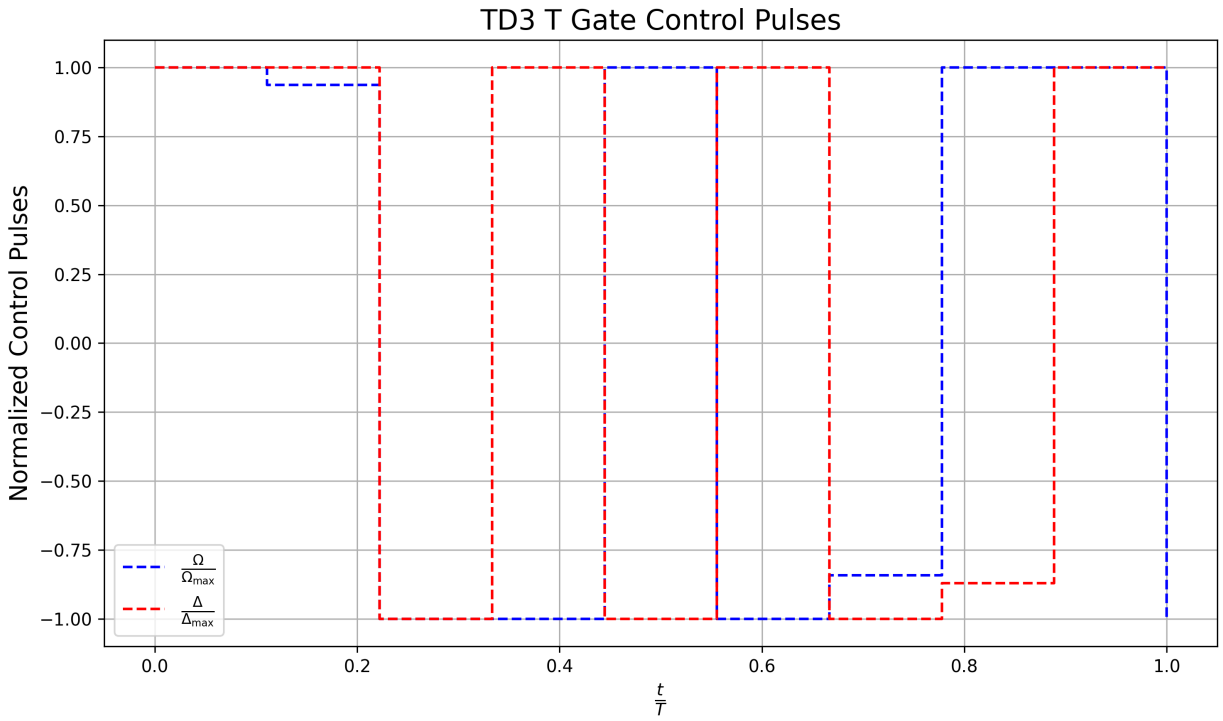

Figure 7.26: TD3 T Gate Control Pulses

7.2.2 Two-Qubit Gate (CNOT)

The CNOT gate optimization introduced additional challenges due to entangling operations and multi-qubit interactions. As we stated before, Rabi frequency and detuning of the control and target qubits will be controlled. Also, the coupling strength J_{zx}.

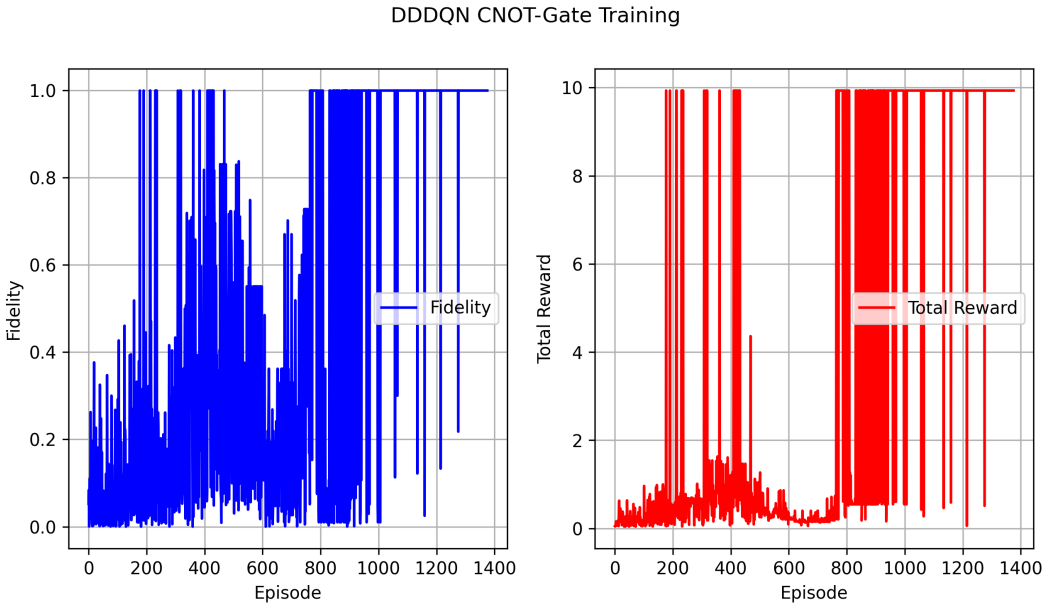

Figure 7.27: DDDQN Agent CNOT-Gate Fidelities and Total Reward

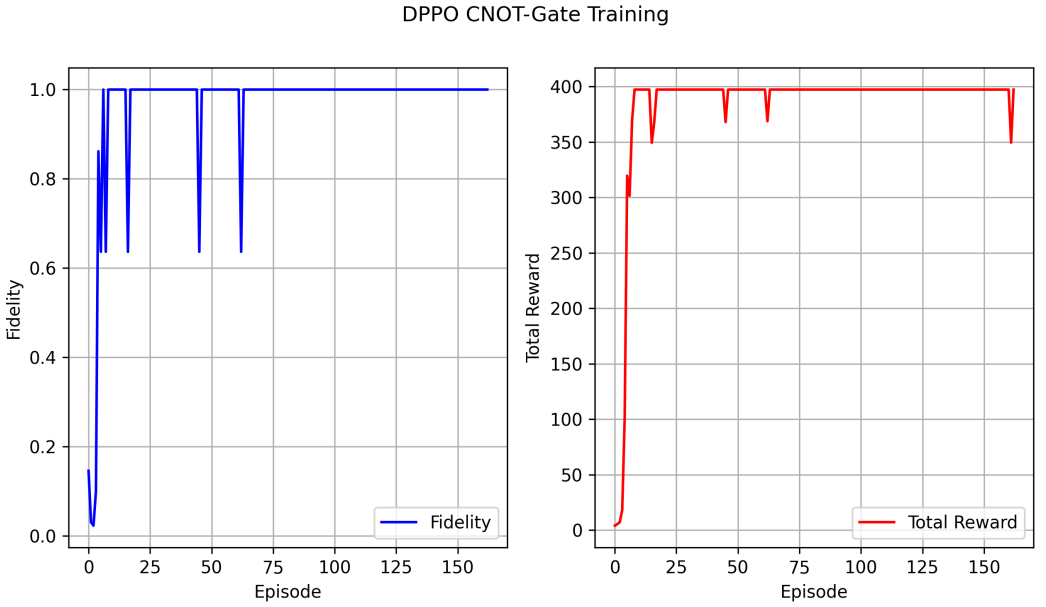

Figure 7.28: Discrete PPO Agent CNOT-Gate Fidelities and Total Reward

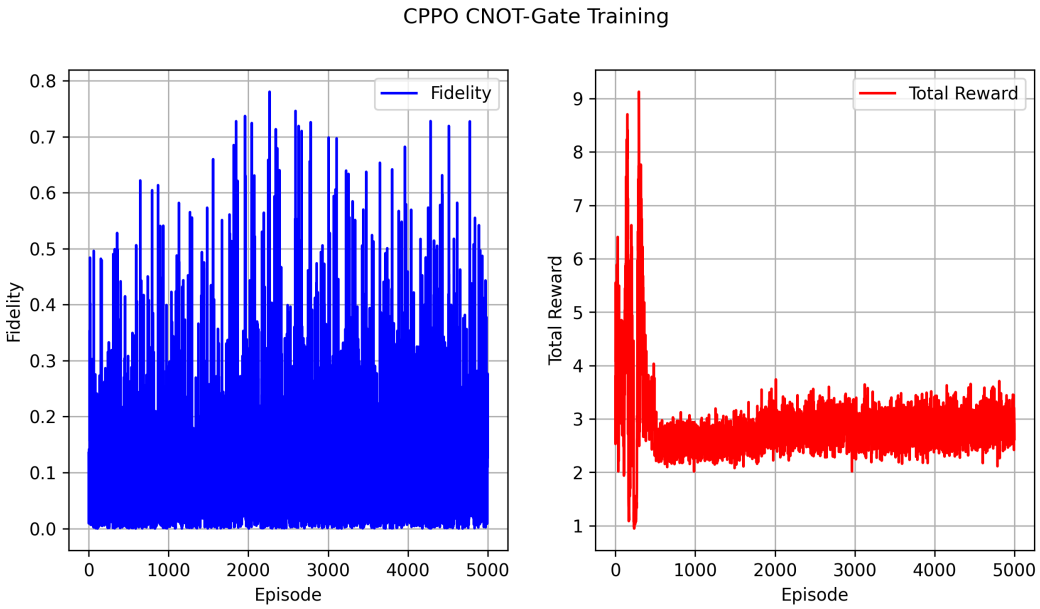

Figure 7.29: Continuous PPO Agent CNOT-Gate Fidelities and Total Reward

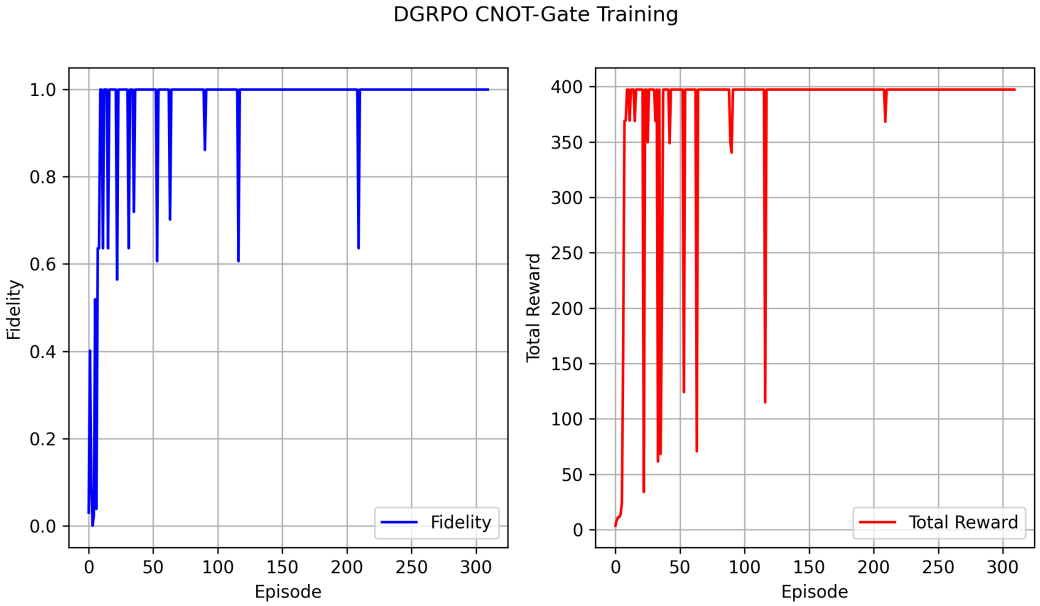

Figure 7.30: Discrete GRPO Agent CNOT-Gate Fidelities and Total Reward

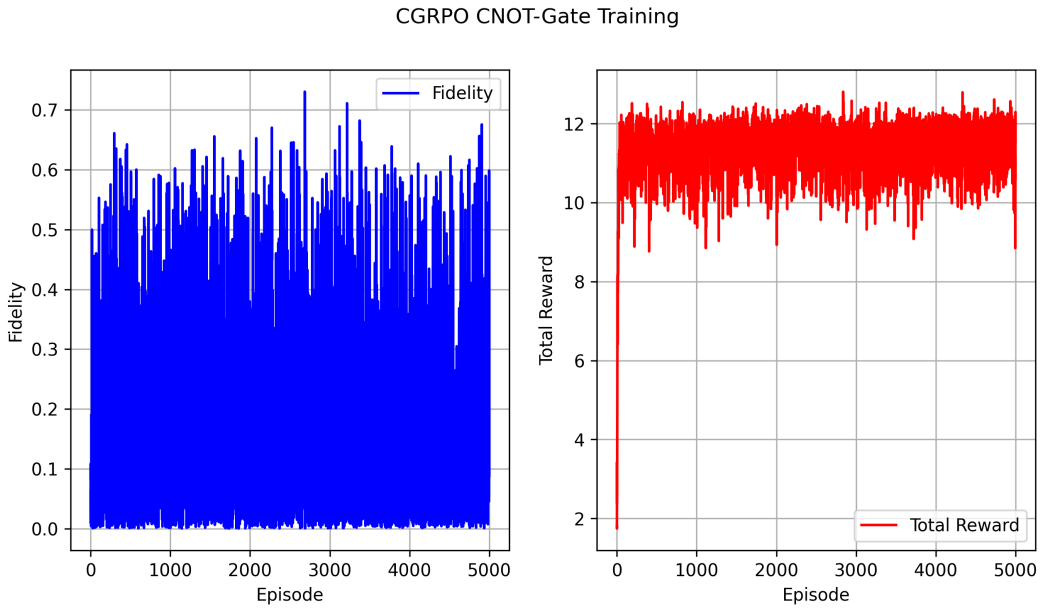

Figure 7.31: Continuous GRPO Agent CNOT-Gate Fidelities and Total Reward

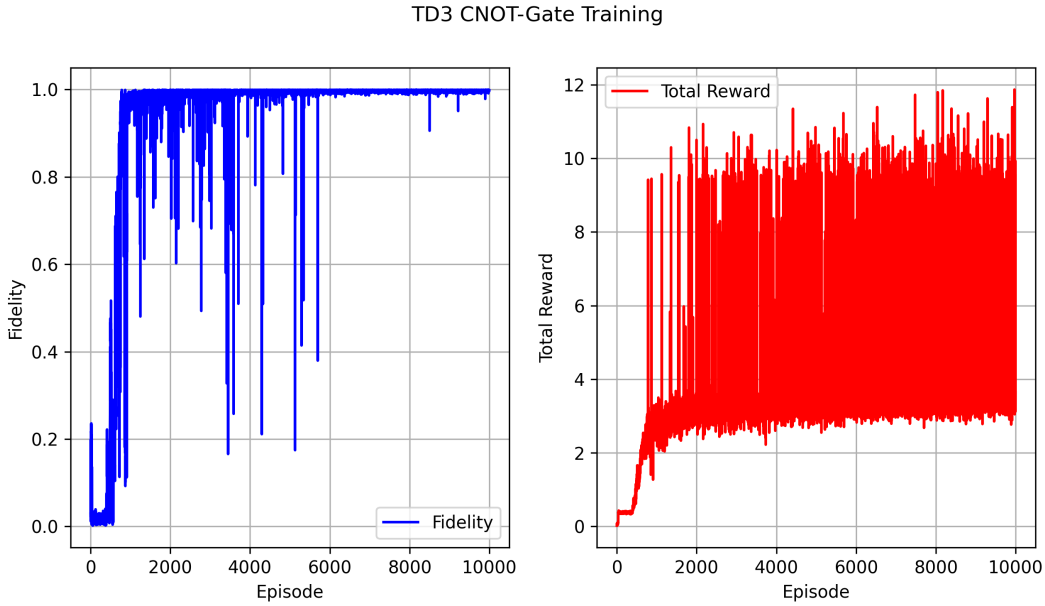

Figure 7.32: TD3 Agent CNOT-Gate Fidelities and Total Reward

We show the maximum fidelity, log infidelity, and average fidelity and time step values for each agent.

	Agent	Max Fidelity	Max Log Infidelity	Avg Fidelity	Time Steps
4	GRPO-C	0.730429	0.569327	0.784343	5
2	PPO-C	0.780537	0.658638	0.824429	5
0	DDDQN	0.999361	3.194187	0.999488	2
1	PPO-D	0.999361	3.194187	0.999488	2
3	GRPO-D	0.999361	3.194187	0.999488	2
5	TD3	0.999860	3.852707	0.999888	4

Table 7.3: CNOT Gate Fidelities Comparison

The comparison of log infidelity for the maximum fidelity trajectory is presented for the CNOT-Gate:

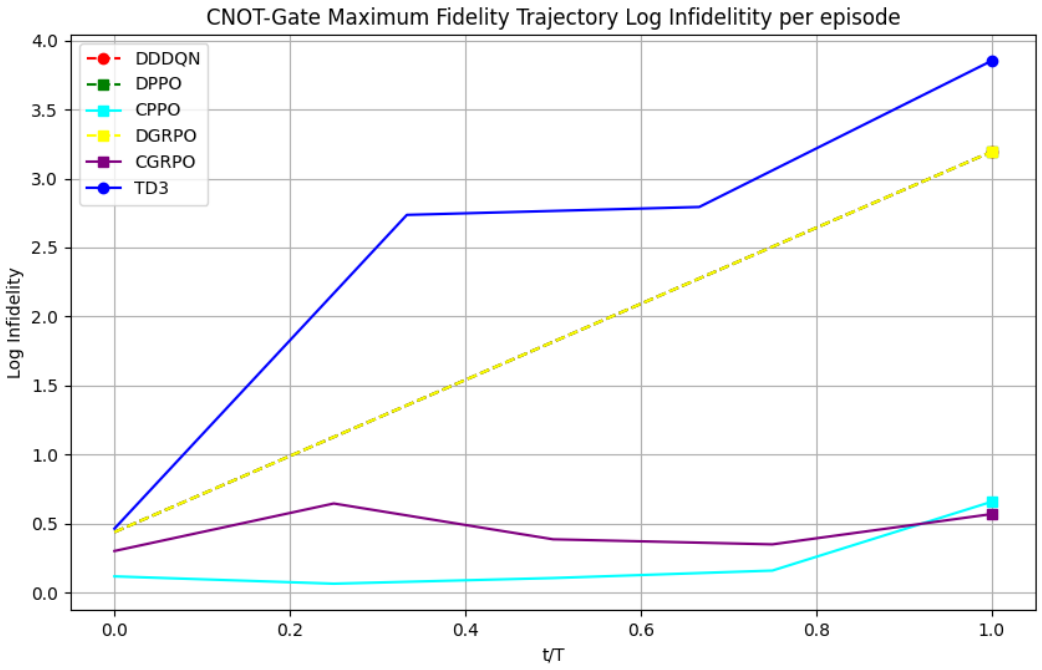

Figure 7.33: CNOT-Gate Normalized Maximum Fidelity Trajectory Log Infidelity for every agent

As in single-qubit gates, the simulations revealed the same pattern in the discrete action set: stable learning achieving high fidelities. Again, continuous stochastic methods methods failed once more to produce a stable and robust learning curve.

We show the optimal control pulses obtained by the agents. Since we have chosen to control the coupling strength J_{zx}, which flips the phase in the control qubit and the state in the target qubit, the pulses consist of detuning only for the control qubit, driving only for the target qubit, and a fixed J_{zx} pulse. This setup is aligned with the selected coupling strength. The dynamic becomes more intriguing if we choose to control J_{zx}, which flips the phase of both qubits.

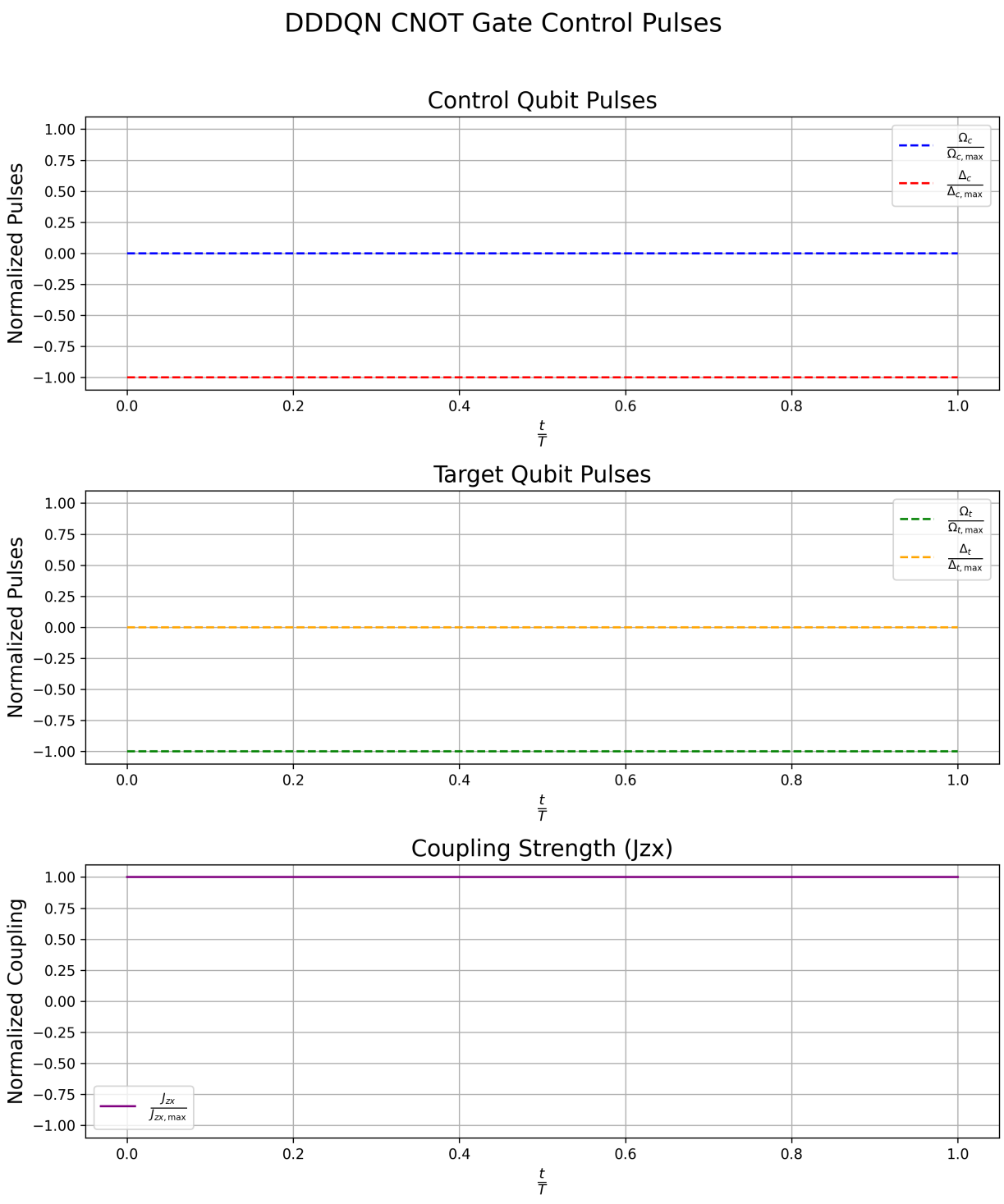

Figure 7.34: DDDQN CNOT Gate Control Pulses

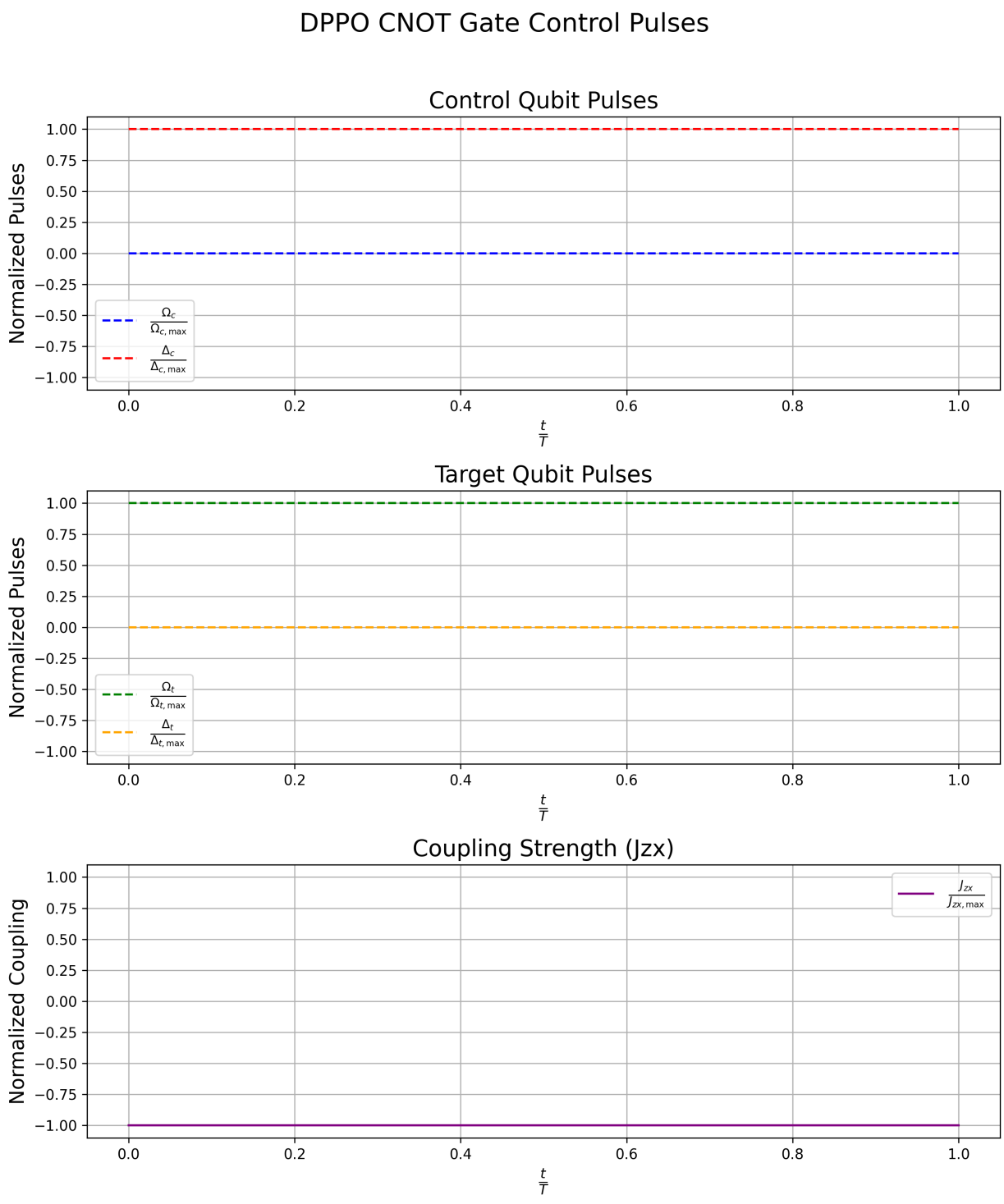

Figure 7.35: Discrete PPO CNOT Gate Control Pulses

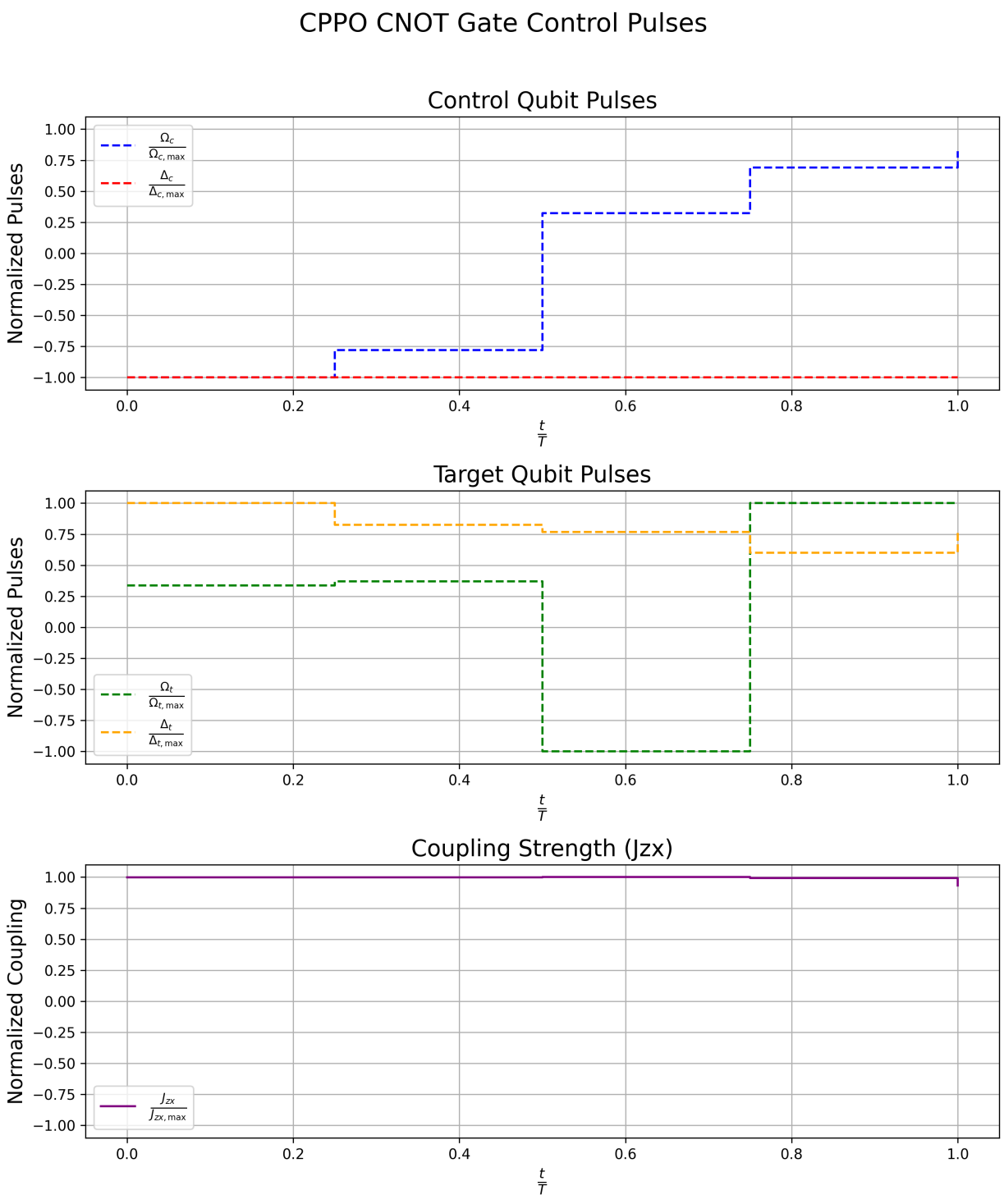

Figure 7.36: Continuous PPO CNOT Gate Control Pulses

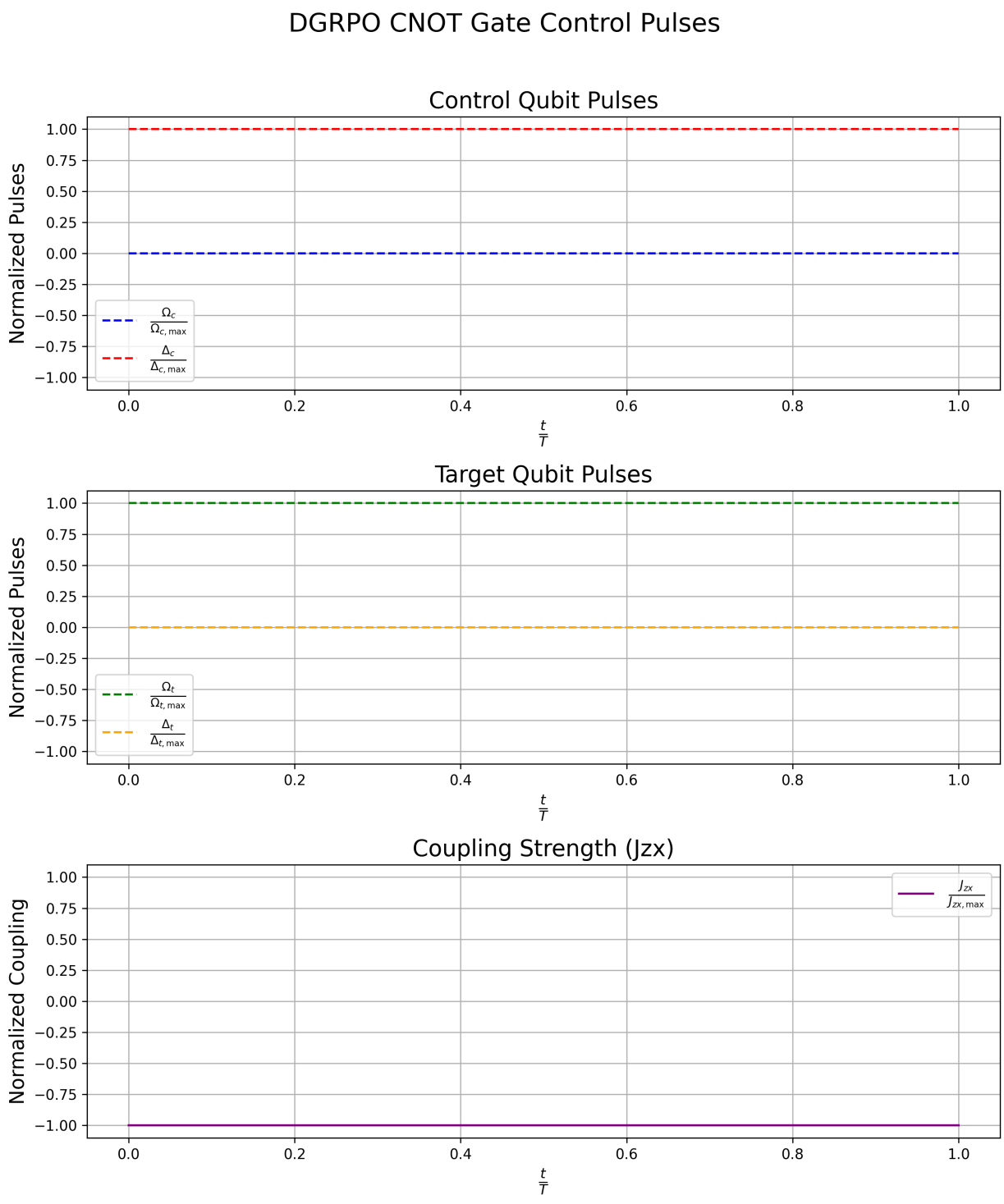

Figure 7.37: Discrete GRPO CNOT Gate Control Pulses

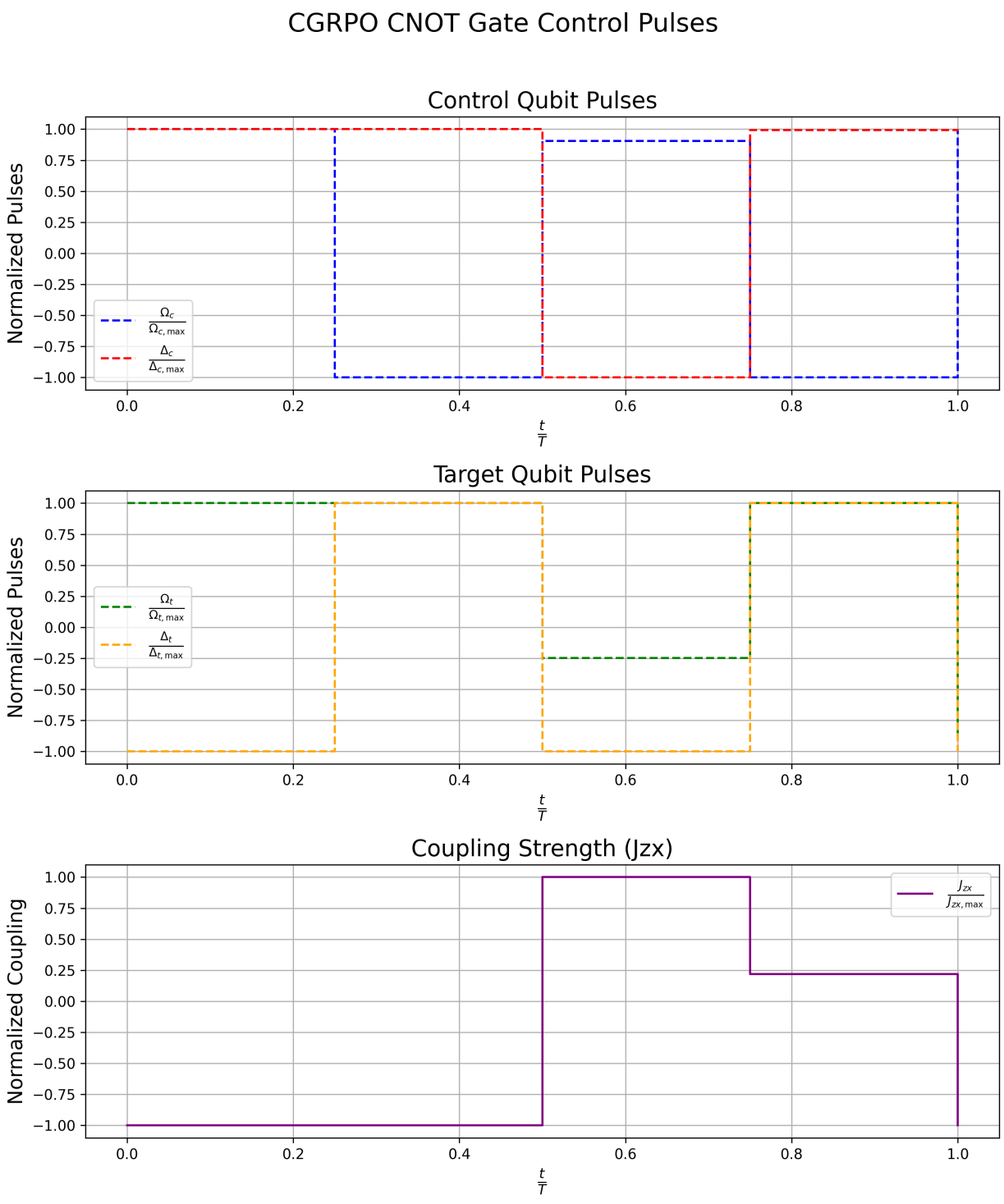

Figure 7.38: Continuous GRPO CNOT Gate Control Pulses

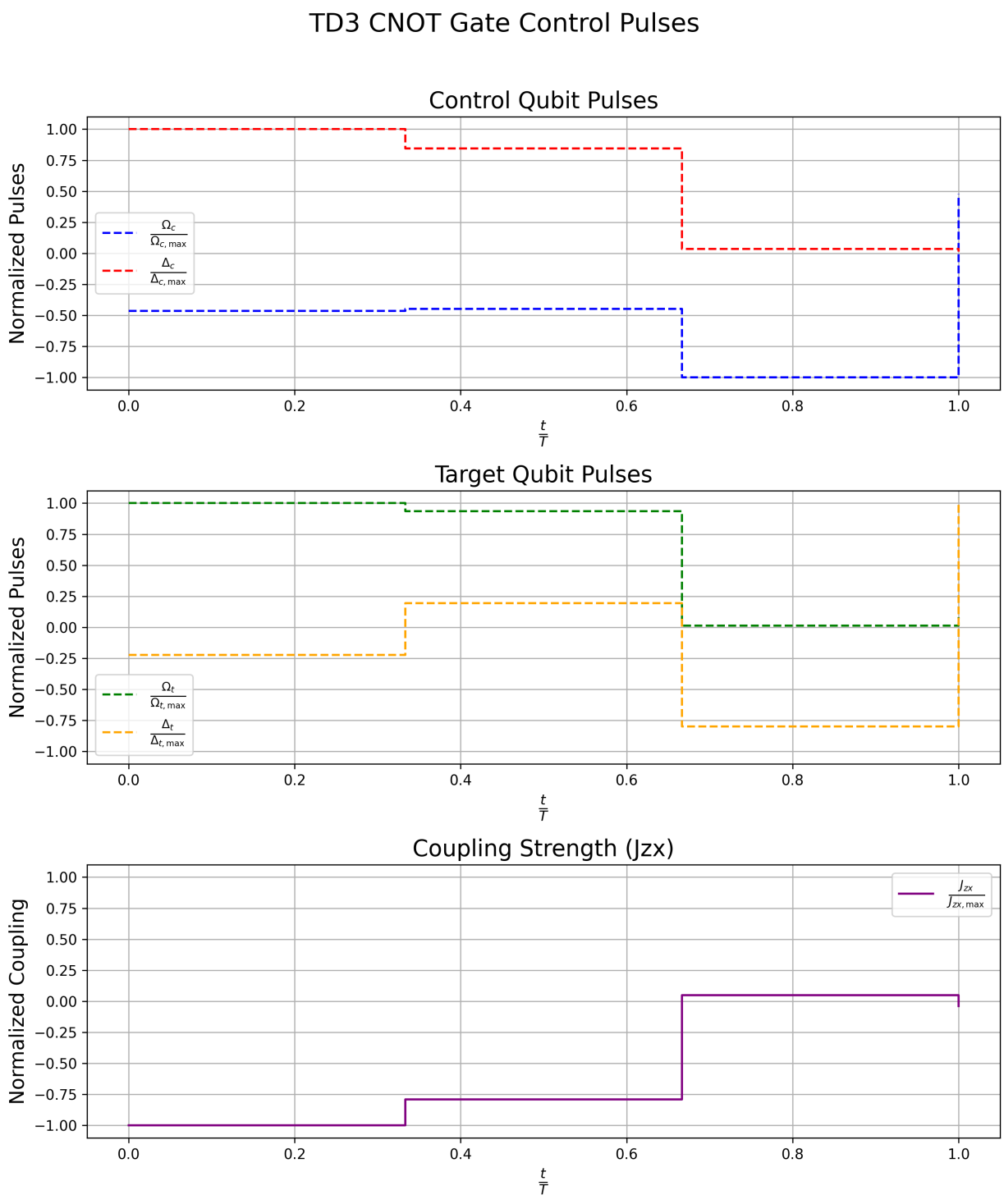

Figure 7.39: TD3 CNOT Gate Control Pulses

7.3 Training Convergence Analysis

The training process was analyzed in terms of the number of training episodes needed to achieve the convergence and stability of the learning curves.

7.3.1 Convergence Speed and Computational Resource Usage

DDDQN, DPPO and DGRPO required the least number of episodes to achieve high fidelities. TD3 converged in the second place. CPPO and CGRPO as they struggled to achieve stable learning curves, they required more time than deterministic or discrete approaches.

7.3.2 Learning Stability

TD3 exhibited minor fluctuations due to the approximation errors of the function. DDDQN, DPPO and DGRPO showed the most stable learning due to their deterministic way of solving the problem. DGRPO and GRPO were in line with the deterministic albeit stochastic results due to categorical sampling in a finite and small discrete set of actions. CPPO and CGRPO struggle due to General Advance Estimation (GAE) and stochasticity.

7.4 Validation and Benchmarking

To ensure the accuracy of the RL-based quantum control, the results were compared with GRAPE and CRAB. GRAPE was run twice with a first-order optimizer, such as AdamW, and a second-order optimizer, such as LBFGS. We present the results below.

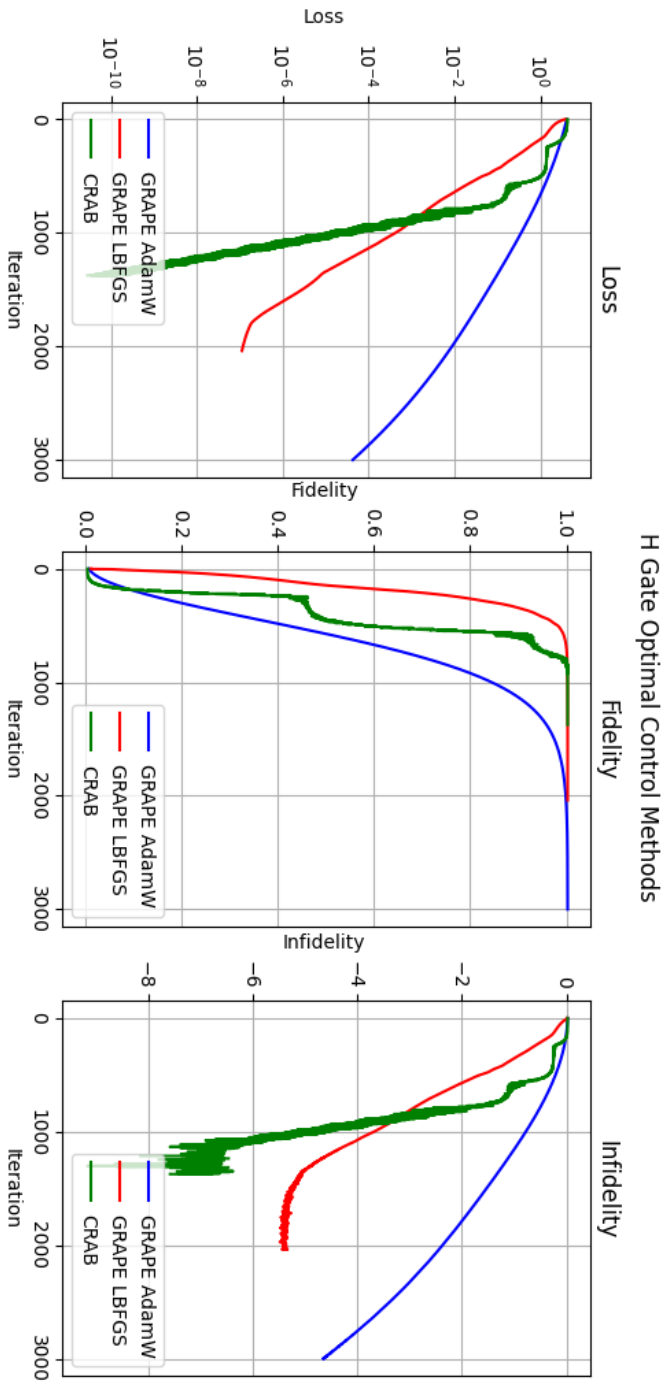

Figure 7.40: H Gate Optimal Control

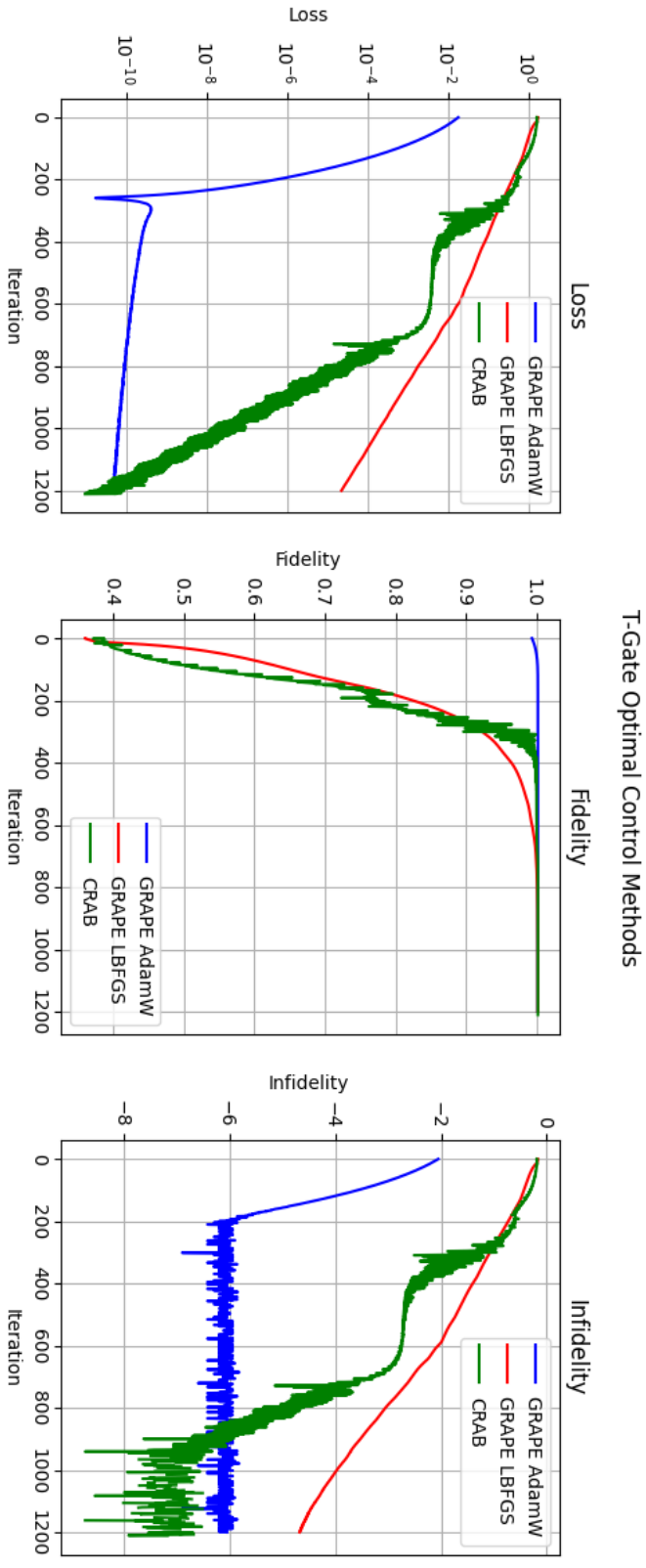

Figure 7.41: T Gate Optimal Control

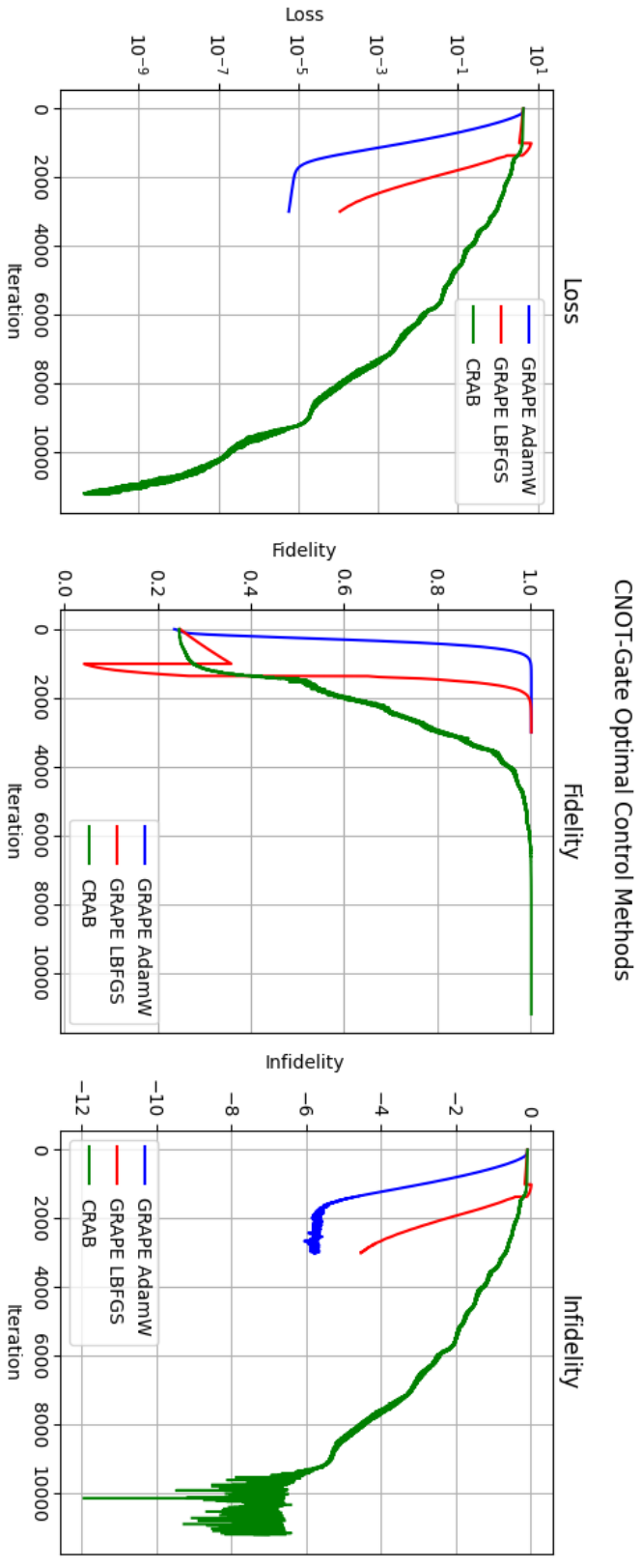

Figure 7.42: CNOT Gate Optimal Control

In addition, we present the result of the control pulses.

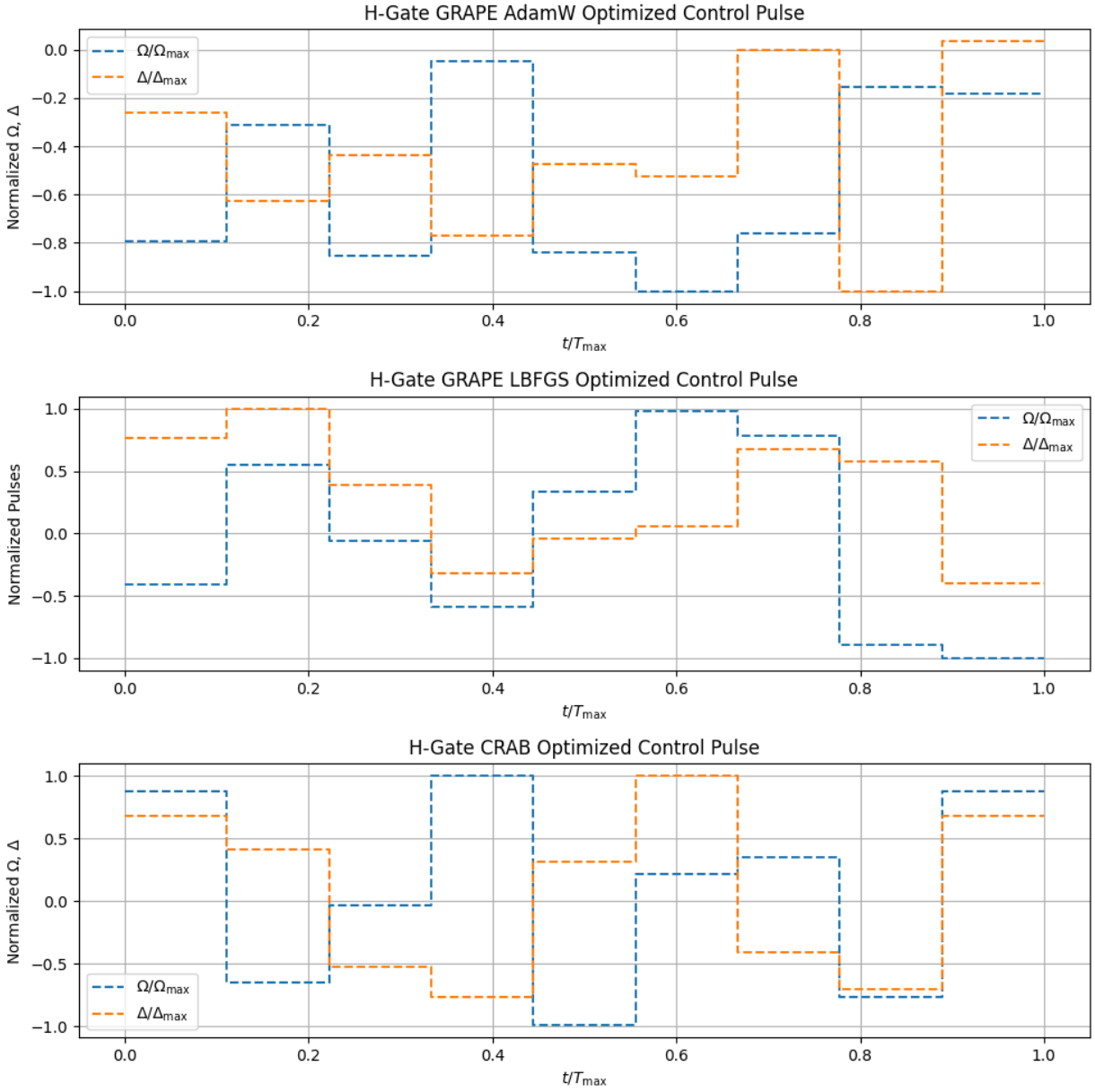

Figure 7.43: H Gate Optimal Pulses

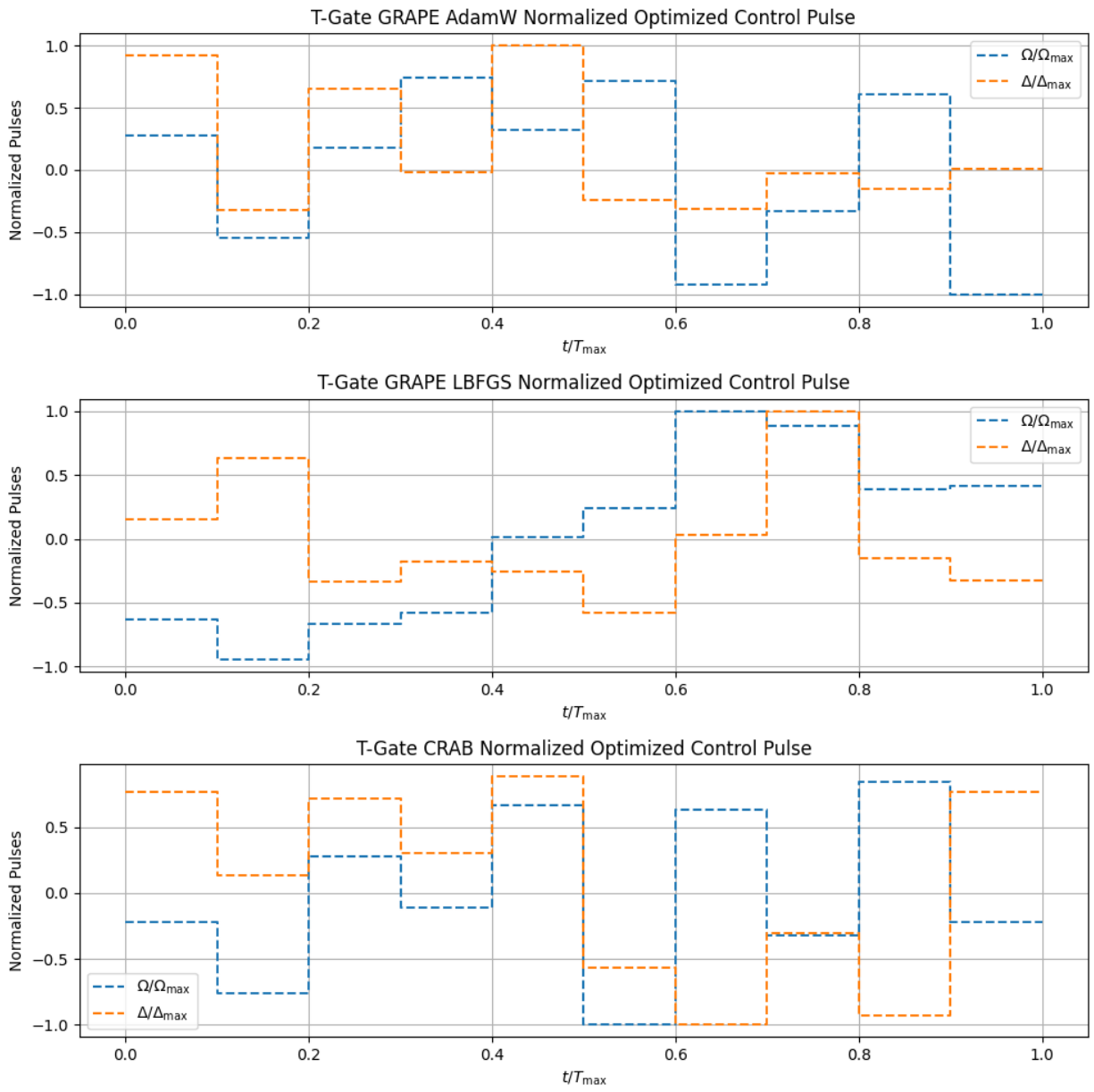

Figure 7.44: T Gate Optimal Pulses

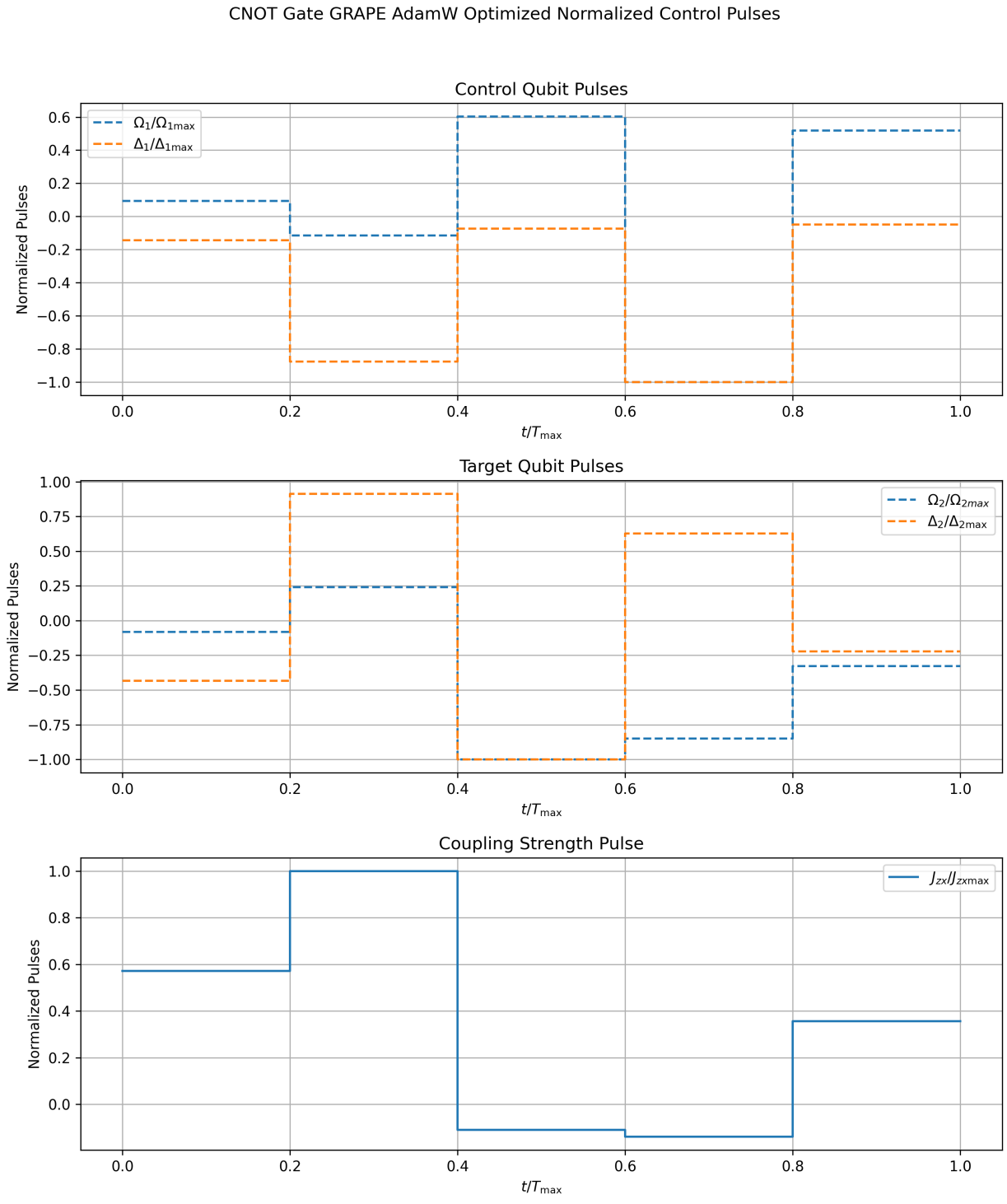

Figure 7.45: CNOT Gate GRAPE AdamW Optimal Pulses

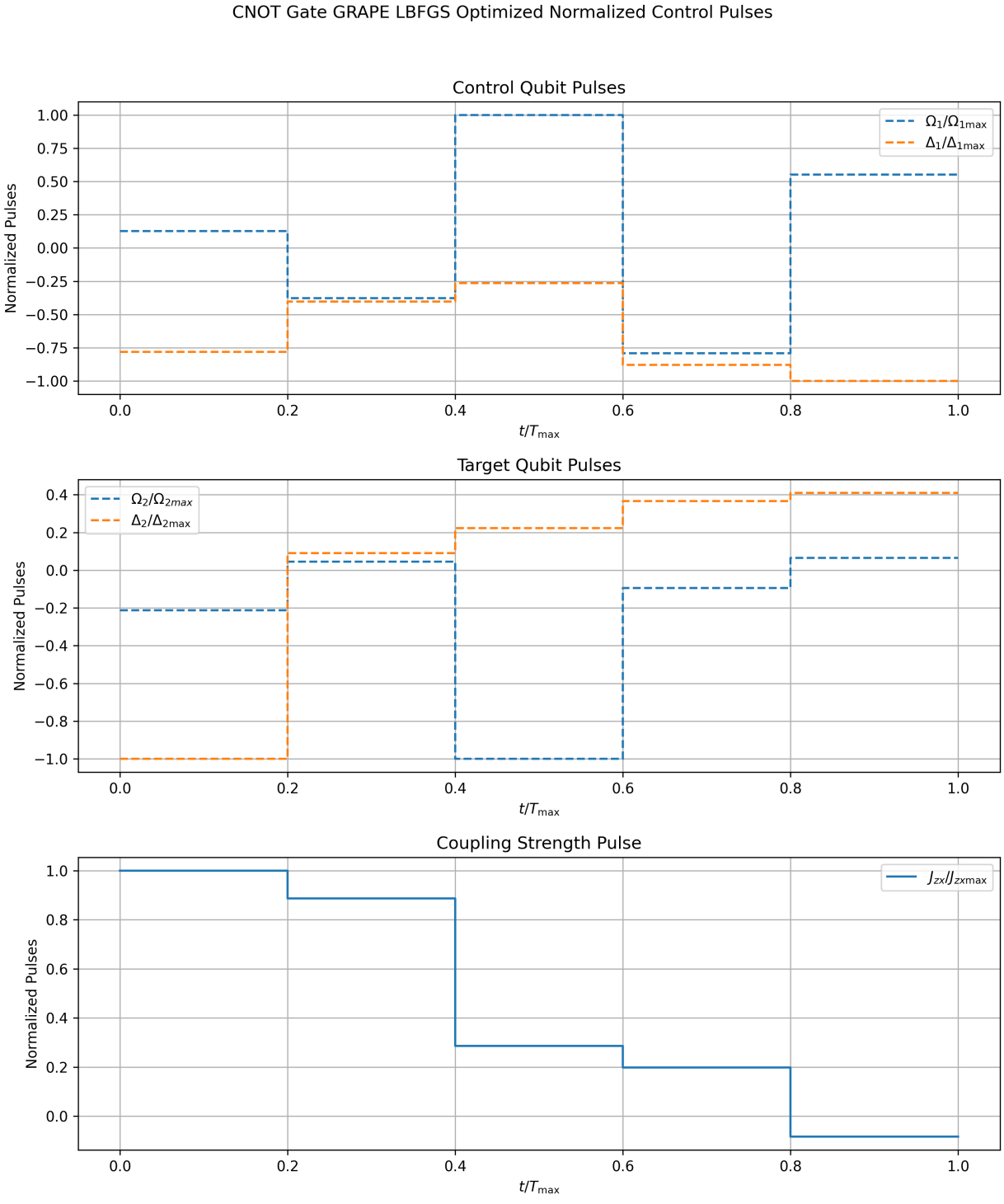

Figure 7.46: CNOT Gate GRAPE LBFGS Optimal Pulses

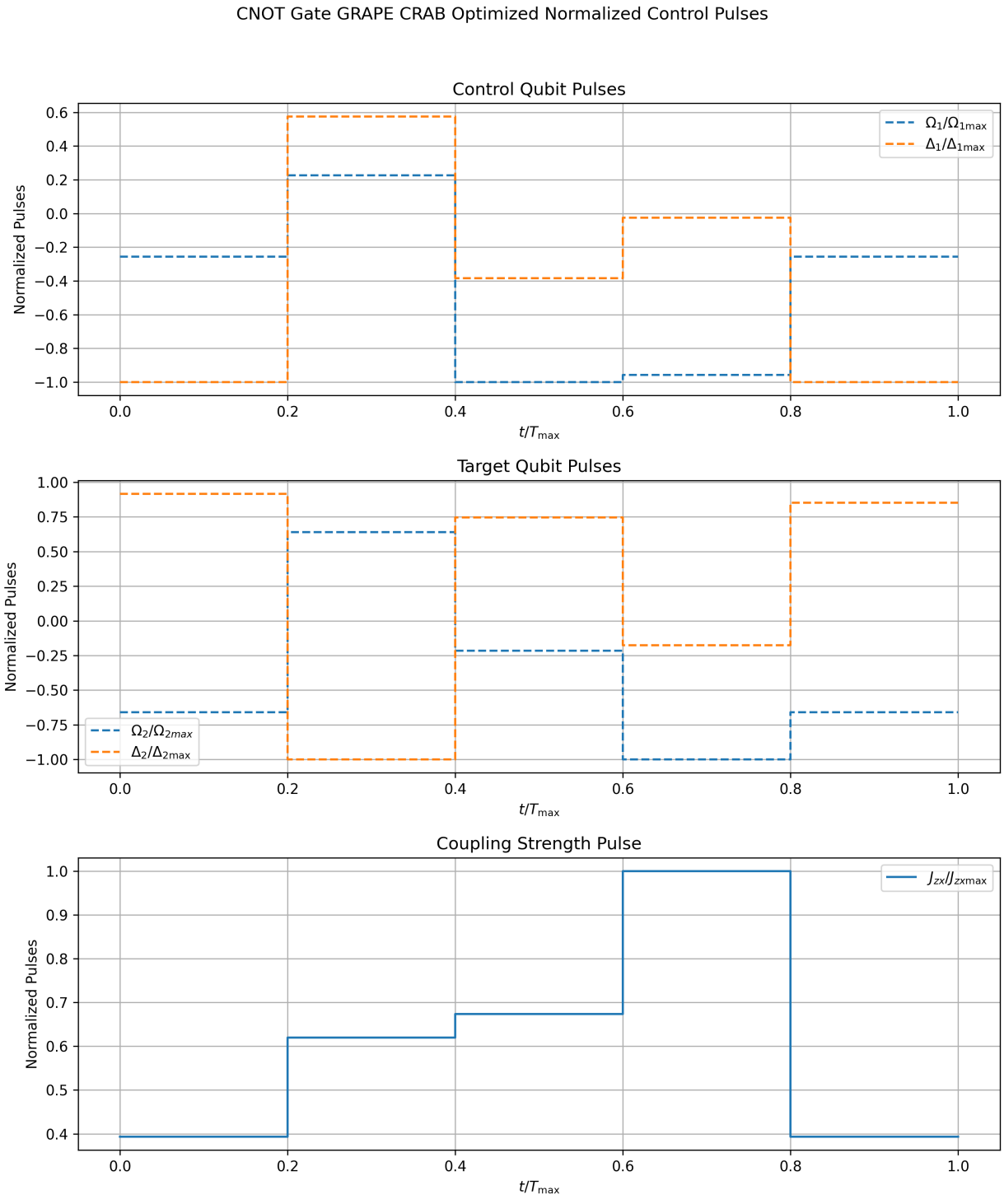

Figure 7.47: CNOT Gate CRAB Optimal Pulses

We can benchmark the different agents with these methods. The 100 rolling mean window of the logarithm fidelity is plotted.

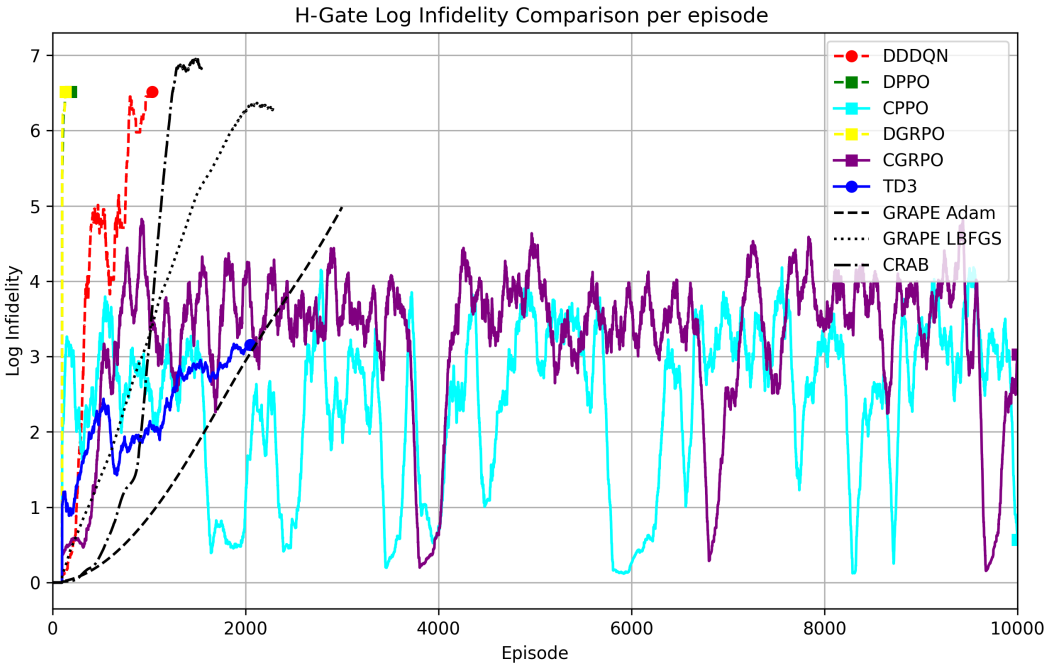

Figure 7.48: H Gate Log Infidelity Comparison

Here, we can see that for the first 2000 episodes, DDDQN, DPPO and DRGPO outperform the optimal sequences given by GRAPE and CRAB.

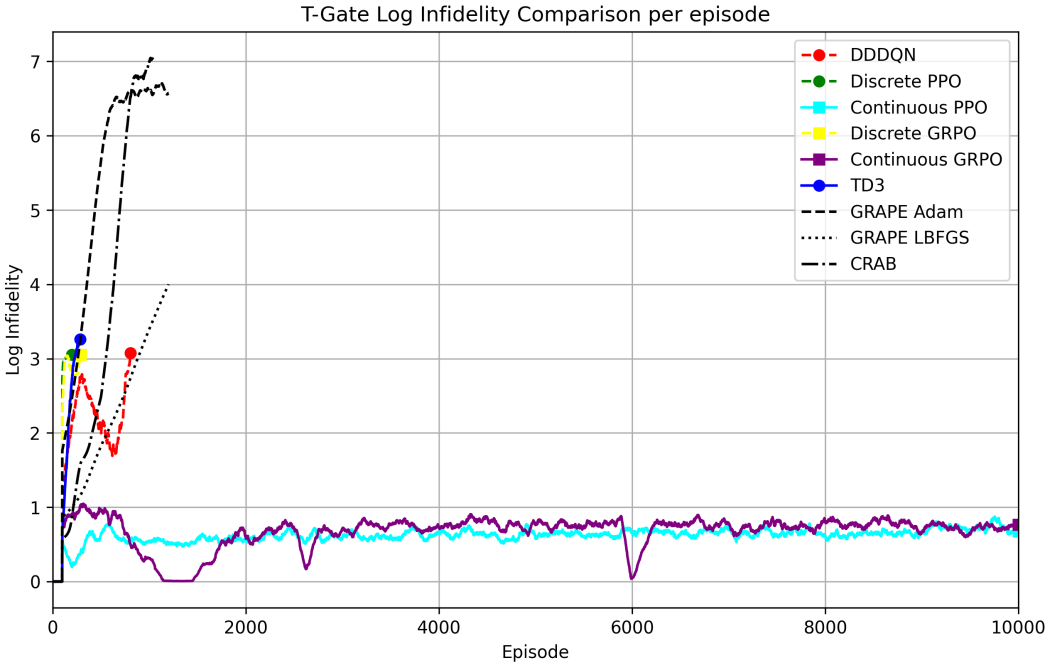

Figure 7.49: T Gate Log Infidelity Comparison

In this plot, DDDQN, TD3 and DPPO match the performance of GRAPE and CRAB for the first 1000 episodes.

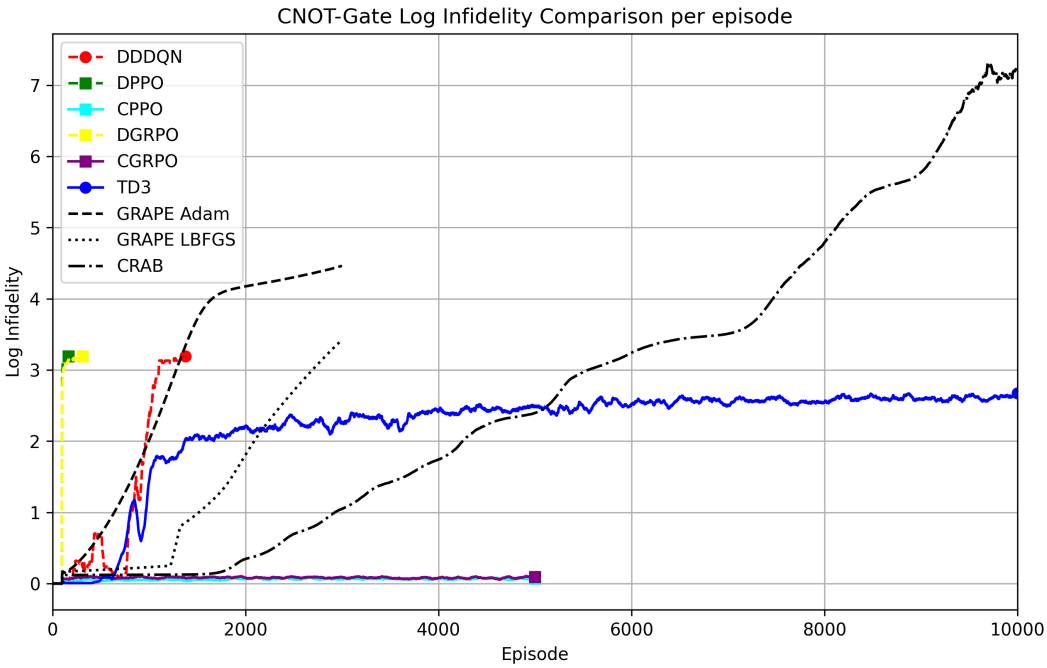

Figure 7.50: CNOT Gate Log Infidelity Comparison

For the CNOT Gate, DDPO and DGRPO outperform the optimal control methods. DDDQN matches the performance of GRAPE ADAM and outperforms the others. TD3 is in line with these methods for the first 1000 episodes. There is room for improvement here.

Chapter 8

Applications and Extensions

8.1 Applications of the H-T-CNOT Gate Set in Quantum Algorithms

The universal gate set comprising the Hadamard (H), T, and CNOT gates is fundamental to quantum computing. By optimizing these gates using reinforcement learning, several quantum algorithms can benefit from high-fidelity implementations.

8.1.1 Quantum Fourier Transform (QFT)

The Quantum Fourier Transform (QFT) is a core component of quantum algorithms like Shor's algorithm [19]. QFT relies on a combination of Hadamard gates to create superpositions and controlled phase gates, which can be synthesized using T and CNOT gates. High-fidelity implementations of the H-T-CNOT set improve error robustness in QFT circuits, enhancing performance in quantum factorization. Furthermore, we can see the Hadamard

gate as a QFT for a single qubit. In matrix form, the QFT is represented as

$$F_N = \frac{1}{\sqrt{N}} \begin{pmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & \omega & \omega^2 & \cdots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \cdots & \omega^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{N-1} & \omega^{2(N-1)} & \cdots & \omega^{(N-1)^2} \end{pmatrix}.$$

For a single-qubit circuit, we have $n = 1$ and $N = 2^1 = 2$. In this case, the root of unity is

$$\omega = e^{2\pi i/2} = e^{\pi i} = -1.$$

Thus, the QFT for one qubit is given by

$$\mathrm{QFT}_1 = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},$$

which is equivalent to the Hadamard gate.

8.1.2 Grover's Search Algorithm

Grover's algorithm [20] achieves quadratic speedup in unstructured search problems. The optimized gate set enhances oracle function construction, where CNOT gates play a critical role and diffusion operators, which rely on Hadamard and T gates.

Better gate implementations reduce decoherence effects, making the search process more reliable on near-term quantum devices.

8.1.3 Variational Quantum Algorithms (VQAs)

Variational algorithms such as the Quantum Approximate Optimization Algorithm (QAOA) and Variational Quantum Eigensolver (VQE) depend on parameterized quantum circuits

that extensively use CNOT gates for entanglement generation and single-qubit rotations using H and T gates.

8.2 Scalability of RL Approaches to Multi-Qubit Systems

One major challenge in quantum computing is scalability. RL-based control optimization can be extended to multi-qubit architectures while addressing exponential growth in state space, for example, with TD3 which allows continuous optimization in high-dimensional control problems and error mitigation in larger circuits where fidelity is lost due to crosstalk and noise can be minimized using RL-learned error-resilient gate implementations.

8.2.1 Extending RL to N-Qubit Systems

To scale RL-based quantum gate optimization beyond 1-qubit and 2-qubit operations, we may find that hierarchical RL (HRL) methods can be explored to learn multi-qubit control hierarchically and Multi-agent RL (MARL) [47] which can be applied where each qubit is controlled by an independent RL agent. This allows for adaptive gate synthesis across different quantum hardware platforms.

8.2.2 RL-Based Gate Compilation for Large Circuits

Current quantum devices impose hardware constraints such as limited connectivity between qubits and high error rates in long-depth circuits. Reinforcement learning can be applied for gate compilation optimization [48, 49, 50], where RL agents learn optimal decomposition strategies that reduce overall gate depth. and minimize the number of CNOT gates (which have the highest error rates up to now).

8.3 Extensions to Fault-Tolerant Quantum Computing

In the long term, RL-based control optimization can be integrated with fault-tolerant quantum computing (FTQC) frameworks.

8.3.1 Optimizing Quantum Error Correction (QEC)

Fault-tolerant quantum computing error correction codes (QECCs) such as the Surface Code [51] and Steane Code [52]. RL-based techniques can enhance QEC by optimizing syndrome extraction circuits to improve measurement reliability, learning error-resilient quantum gates by reducing logical error rates and adaptive error correction policies, where an RL agent dynamically adjusts correction strategies based on measured syndromes.

8.3.2 RL for Logical Gate Optimization

Logical gates, such as T-gates in fault-tolerant architectures, require costly operations like magic state distillation. RL-based control could learn optimal scheduling of distillation and teleportation operations, minimize resource overhead in logical gate synthesis, and adapt gate sequences to hardware-specific noise models.

8.3.3 RL-Based Quantum Compilation for Fault-Tolerant Devices

Since logical operations have high overhead costs, RL can: optimize logical gate scheduling to minimize circuit depth, reduce qubit idle times in fault-tolerant systems, and adapt gate routing strategies for hardware-specific constraints.

8.4 Potential Future Directions

Several open research questions arise from the use of reinforcement learning in quantum control. One issue concerns the generalization of trained RL models to different quantum

hardware platforms, such as superconducting qubits [53], trapped ions, or neutral atoms. Another question involves the development of hybrid RL-Quantum Optimal Control approaches, where reinforcement learning is combined with methods such as GRAPE [41] or CRAB [42]. Furthermore, it remains to be seen whether RL can dynamically adjust gate sequences in response to real-time noise conditions [54]. Finally, exploring Lyapunov-based Reinforcement Learning as a potential scheme for these problems is a promising direction [55, 56].

One promising direction is to integrate RL with Quantum Approximate Control (QAC), where RL policies are trained to adaptively compensate for hardware noise.

8.5 Summary of Applications and Extensions

The optimized H-T-CNOT gate set benefits core quantum algorithms such as QFT, Grover search, and VQE. Reinforcement learning (RL) approaches can scale to multi-qubit systems through hierarchical and multi-agent learning. Furthermore, fault-tolerant quantum computing may leverage RL to improve error correction, logical gate synthesis, and compilation. Future research should focus on generalizing between different hardware, developing noise-adaptive RL policies, and exploring hybrid control methods [57]. Furthermore, quantum compilers have been implemented using RL approaches, further demonstrating the importance of RL in quantum control [58].

Chapter 9

Conclusion and Future Work

9.1 Summary of Contributions

This thesis explored the application of reinforcement learning (RL) methods to optimize the implementation of quantum gates, specifically for the universal gate set H-T-CNOT. On the one hand, several works showed that RL promises good performance in these systems [59]. However, achieving the high fidelities gate is challenging, but the scientific community pushes results even better as time passes [60].

The main contributions of this work include the formulation of quantum gate optimization as a reinforcement learning problem. In this approach, reinforcement learning is applied to quantum control by defining an environment in which the state space captures the evolution of the quantum system, the action space comprises the control pulse parameters, and the reward function is based on the fidelity of the quantum gate process. The thesis deployed RL agents for quantum gate synthesis by implementing and comparing several RL methods. Specifically, it explored the use of Double Dueling Deep Q-Network (DDDQN) for discrete action optimization, Twin Delayed Deep Deterministic Policy Gradient (TD3) for continuous pulse control, as well as Proximal Policy Optimization (PPO) and Group Policy Relative Optimization (GRPO) for hybrid discrete-continuous learning. Performance evalu-

ation and benchmarking were performed on single-qubit (Hadamard and T) and two-qubit (CNOT) gates. The results indicate that the deterministic agents performed better than the stochastic ones, with DGRPO and DPPO exhibiting deterministic behavior when the action set was both finite and small. Moreover, these deterministic agents matched or outperformed the solutions obtained from numerical optimal control approaches such as CRAB and GRAPE.

These findings contribute to the growing field of machine learning for quantum control, demonstrating the potential of RL to improve quantum computation.

9.2 Limitations of the Current Work

Despite its success, this thesis has several limitations that highlight areas for improvement. The complexity of training is a significant challenge for RL-based quantum gate optimization. Scalability also remains an issue. Although RL performs well on single-qubit and two-qubit systems, applying it to multiqubit architectures is challenging because of state space explosion in high-dimensional quantum systems, increased hardware-specific noise models, and longer training times as the number of qubits grows. Moreover, RL agents face exploration versus exploitation trade-offs. Stochastic agents and continuous action spaces, such as PPO and GRPO, struggle with exploration efficiency because they do not use memory replay and instead collect trajectories as they interact with the environment, a process that is very expensive in terms of computation. Addressing these limitations will be crucial in future research.

9.3 Directions for Future Research

Several avenues can extend the findings of this thesis.

9.3.1 Hardware Deployment of RL-Based Quantum Control

Future work should focus on the implementation of RL-optimized quantum gates on real quantum hardware. For example, one may consider using IBM Quantum Processors via Qiskit [61] or testing model generalization on trapped ion and neutral atom platforms.

9.3.2 Hybrid Quantum-Classical RL Methods

Another promising direction is to develop hybrid quantum-classical RL methods. An approach is quantum reinforcement learning (QRL), which integrates quantum circuits into RL architectures to reduce computational overhead. Alternatively, RL can be combined with traditional optimal control techniques such as GRAPE and CRAB [41, 42] to achieve high-precision quantum control.

9.3.3 Scalability to Large Quantum Systems

Scalability remains a significant challenge for multi-qubit systems. Future research could explore multi-agent reinforcement learning, where independent RL agents control different qubits, or hierarchical reinforcement learning, which decomposes complex quantum circuits into subtasks that can be optimized separately.

9.3.4 Adaptive RL for Noise-Resilient Quantum Gates

Quantum noise remains a fundamental challenge for quantum computing. Future work could focus on developing RL-driven error mitigation techniques that enable RL agents to dynamically adjust gate parameters to compensate for decoherence. Another promising direction is to extend the RL policies to adapt control sequences in real time based on

measured noise fluctuations in quantum hardware. In addition, reinforcement learning can be applied to improve fault-tolerant operations by optimizing quantum error correction (QEC) strategies.

9.4 Final Remarks

This thesis demonstrated the effectiveness of learning to optimize the control of quantum gates. The results showed that, although CPPO and CGRPO struggled to achieve stable learning despite obtaining good fidelities, TD3 performed well in continuous control, and DDDQN, DPPO and DGRPO were effective in discrete control, albeit limited by the granularity of the action space. The study also highlighted the potential of reinforcement learning for quantum computing, particularly in optimizing control pulses for quantum gates.

In the future, the integration of machine learning and quantum computing will be essential to advance quantum technologies. Reinforcement learning provides a promising framework for adaptive, hardware-efficient quantum control, paving the way for scalable, and robust quantum computing.

Appendix A

Appendix

A.1 Algorithms

In this section, we provide a detailed pseudocode for the various training algorithms used throughout our study. These include DDDQN, discrete and continuous variants of PPO and GRPO, as well as the TD3 algorithm. Each algorithm is presented with a clear, step-by-step outline to facilitate reproducibility and provide deeper insight into the implementation details of our experimental framework.

Algorithm 2 DDDQN Training Algorithm

Require: Initial network parameters θ, replay buffer D, target network parameters θ^-

for episode = 1, 2, ... EPISODES **do**

  Reset the quantum environment state

  for $k = 1, 2, \ldots, N$ **do**

    Choose action with ϵ-greedy strategy to define the pulse

    Evolve the unitary by the pulse and time

    Compute the step reward based on fidelity between target and current unitary

    Estimate the next state

    Store trajectory state: s_{k-1}, action: a_{k-1}, next state: s_k, step reward: r_k, dones: d_k

  end for

  Sample $\{(s, a, r, s')\} \sim D$

  Compute target: $y \leftarrow r + \gamma \max_{a'} Q(s', a'; \theta^-)$

  Update θ by minimizing loss:

$$L(\theta) = \frac{1}{2} \left(Q(s, a; \theta) - y \right)^2$$

  Periodically update target network: $\theta^- \leftarrow \theta$

end for

Algorithm 3 Discrete PPO Training Algorithm

Require: Initial policy parameters θ, old policy θ_{old}

for episode = 1, 2, ... EPISODES **do**

  Reset state

  Reset Rollout

  for $k = 1, 2, \ldots, N$ **do**

    Choose action by sampling from a **Categorical Distribution**

    Evolve the unitary by the pulse and time

    Compute the step reward based on fidelity between target and current unitary

    Estimate the next state

    Store trajectory state: s_{k-1}, action: a_{k-1}, next state: s_k, step reward: r_k, dones: d_k

  end for

  Compute advantages $\hat{A}_t$ using GAE

  for each update epoch **do**

    Compute ratio:

$$r_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

    Compute surrogate loss:

$$L^{CLIP} = \min\left(r_t\hat{A}_t, \text{clip}(r_t, 1-\epsilon, 1+\epsilon)\hat{A}_t\right)$$

    Compute value loss:

$$L^{VF} = \frac{1}{T}\sum_{t=1}^{T}\left(V_\theta(s_t) - V_t^{\text{target}}\right)^2$$

    Update θ by maximizing L^{CLIP} plus an entropy bonus while minimizing L^{VF}

  end for

  Set $\theta_{\text{old}} \leftarrow \theta$

end for

Algorithm 4 Continuous PPO Training Algorithm

Require: Initial policy parameters θ

for episode = 1, 2, ... EPISODES **do**

  Reset state

  Reset Rollout

  for $k = 1, 2, \ldots, N$ **do**

    Choose action by sampling from a **Normal Distribution**

    Evolve the unitary by the pulse and time

    Compute the step reward based on fidelity between target and current unitary

    Estimate the next state

    Store trajectory state: s_{k-1}, action: a_{k-1}, next state: s_k, step reward: r_k, dones: d_k

  end for

  Compute advantages $\hat{A}_t$ using GAE

  for each update epoch **do**

    Compute ratio:

$$r_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

    Compute surrogate loss:

$$L^{CLIP} = \min\left(r_t\hat{A}_t, \text{clip}(r_t, 1-\epsilon, 1+\epsilon)\hat{A}_t\right)$$

    Compute value loss:

$$L^{VF} = \frac{1}{T}\sum_{t=1}^{T}\left(V_\theta(s_t) - V_t^{\text{target}}\right)^2$$

    Update θ by maximizing L^{CLIP} plus an entropy bonus while minimizing L^{VF}

  end for

  Set $\theta_{\text{old}} \leftarrow \theta$

end for

Algorithm 5 Discrete GRPO Training Algorithm

Require: Initial policy parameters θ

for episode = 1, 2, ... EPISODES **do**

  Reset state

  Reset Rollout

  for $k = 1, 2, \ldots, N$ **do**

    Choose action by sampling from a **Categorical Distribution**

    Evolve the unitary by the pulse and time

    Compute the step reward based on fidelity between target and current unitary

    Estimate the next state

    Store trajectory state: s_{k-1}, action: a_{k-1}, next state: s_k, step reward: r_k, dones: d_k

  end for

  Compute group advantages $\hat{A}_t$

  for each update epoch **do**

    Compute ratio:

$$r_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

    Compute surrogate loss:

$$L^{CLIP} = \min\left(r_t\hat{A}_t, \text{clip}(r_t, 1-\epsilon, 1+\epsilon)\hat{A}_t\right)$$

    Update θ by maximizing L^{CLIP} plus an entropy bonus

  end for

  Set $\theta_{\text{old}} \leftarrow \theta$

end for

Algorithm 6 Continuous GRPO Training Algorithm

Require: Initial policy parameters θ

for episode = 1, 2, ... EPISODES **do**

  Reset state

  Reset Rollout

  for $k = 1, 2, \ldots, N$ **do**

    Choose action by sampling from a **Normal Distribution**

    Evolve the unitary by the pulse and time

    Compute the step reward based on fidelity between target and current unitary

    Estimate the next state

    Store trajectory state: s_{k-1}, action: a_{k-1}, next state: s_k, step reward: r_k, dones: d_k

  end for

  Compute group advantages $\hat{A}_t$

  for each update epoch **do**

    Compute ratio:

$$r_t = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

    Compute surrogate loss:

$$L^{CLIP} = \min\left(r_t\hat{A}_t, \text{clip}(r_t, 1-\epsilon, 1+\epsilon)\hat{A}_t\right)$$

    Update θ by maximizing L^{CLIP} plus an entropy bonus

  end for

  Set $\theta_{\text{old}} \leftarrow \theta$

end for

Algorithm 7 TD3 Training Algorithm

Require: Initial actor parameters θ, critic networks with parameters ϕ_1, ϕ_2, replay buffer D

for episode = 1, 2, ... EPISODES **do**

  Reset the quantum environment state

  for $k = 1, 2, \ldots, N$ **do**

    Choose action with ϵ-greedy strategy to define the pulse

    Evolve the unitary by the pulse and time

    Compute the step reward based on fidelity between target and current unitary

    Estimate the next state

    Store trajectory state: s_{k-1}, action: a_{k-1}, next state: s_k, step reward: r_k, dones: d_k

  end for

  Sample $\{(s, a, r, s')\} \sim D$

  Compute target action: $\tilde{a} \leftarrow \pi_\theta(s') + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma)$ (clip $\tilde{a}$)

  Compute target Q-value:

$$y \leftarrow r + \gamma \min\{Q_{\phi_1}(s', \tilde{a}), Q_{\phi_2}(s', \tilde{a})\}$$

  Update critic networks by minimizing:

$$L(\phi_i) = \frac{1}{N} \sum (Q_{\phi_i}(s, a) - y)^2, \quad i = 1, 2$$

  Update Actor and Critic Networks by Delayed Updates with τ parameter

end for

A.2 Hamiltonian derivation

We introduce how the Hamiltonian for a single qubit is derived [62].

We define

$$P_{nm} = |\psi_n\rangle\langle\psi_m|,$$

and, using the completeness relation,

$$P_{11} + P_{22} = I.$$

The total Hamiltonian is given by

$$H(t) = H_0 + H_I(t),$$

where

$$H_0 = E_1\, P_{11} + E_2\, P_{22},$$

with E_1 and E_2 being the eigenenergies of the free Hamiltonian operator H_0.

After some straightforward operator algebra, the interaction term of the Hamiltonian can be written as

$$H_I(t) = -d_{12}\left(P_{12} + P_{21}\right)\epsilon(t)\cos\big[\omega t + \phi(t)\big],$$

where we have assumed a centrosymmetric quantum system with zero permanent dipole matrix elements. The electric dipole matrix elements are defined as

$$d_{nm} = \langle\psi_n|\,\vec{d}\cdot\hat{\epsilon}\,|\psi_m\rangle,$$

with $d_{11} = d_{22} = 0$ and $d_{12} = d_{21}$.

Defining the Rabi frequency as

$$\Omega(t) = -\frac{d_{21}\,\epsilon(t)}{\hbar} \quad (\Omega \in \mathbb{R}),$$

the interaction Hamiltonian becomes

$$H_I(t) = \hbar\,\Omega(t)\cos(\omega t)\big(P_{12} + P_{21}\big).$$

In the *interaction picture* the interaction Hamiltonian transforms as

$$H_I^{(i)}(t) = e^{\frac{iH_0 t}{\hbar}}\, H_I(t)\, e^{-\frac{iH_0 t}{\hbar}},$$

and, upon substituting H_0 and $H_I(t)$ into (A.2), one obtains

$$H_I^{(i.p.)}(t) = \frac{\hbar\,\Omega(t)}{2}\Big[\Big(e^{-i(\omega_0+\omega)t - i\phi(t)} + e^{-i(\omega_0-\omega)t + i\phi(t)}\Big)P_{21} + \Big(e^{i(\omega_0+\omega)t + i\phi(t)} + e^{i(\omega_0-\omega)t - i\phi(t)}\Big)P_{12}\Big],$$

where

$$\omega_0 = \frac{E_2 - E_1}{\hbar}.$$

Under the rotating wave approximation (RWA) the rapidly oscillating counter-rotating terms $\exp\big[\pm i(\omega_0+\omega)t\big]$ can be neglected. Thus, the interaction Hamiltonian in the interaction picture reduces to

$$H_{I,\mathrm{RWA}}^{(i)}(t) = \frac{\hbar\,\Omega(t)}{2}\left[P_{21}\, e^{i\Delta_1 t + i\phi(t)} + P_{12}\, e^{-i\Delta_1 t - i\phi(t)}\right],$$

where the *bare detuning* is defined as

$$\Delta_1 = \omega - \omega_0.$$

The state vector of the system can be expressed as

$$|\psi(t)\rangle = a_1(t)\,|\psi_1\rangle + a_2(t)\,|\psi_2\rangle.$$

From the time-dependent Schrödinger equation (TDSE) we obtain

$$i\hbar\,\dot{a}_1(t) = \frac{\hbar\,\Omega(t)}{2}\,e^{-i\Delta_1 t - i\phi(t)}\,a_2(t), \qquad i\hbar\,\dot{a}_2(t) = \frac{\hbar\,\Omega(t)}{2}\,e^{i\Delta_1 t + i\phi(t)}\,a_1(t).$$

With a suitable unitary transformation or a change of variables to new probability amplitudes $b_1(t)$ and $b_2(t)$, these equations reduce to

$$i\hbar\,\dot{b}_1(t) = \frac{\hbar\,\Delta(t)}{2}\,b_1(t) + \frac{\hbar\,\Omega(t)}{2}\,b_2(t), \qquad i\hbar\,\dot{b}_2(t) = \frac{\hbar\,\Omega(t)}{2}\,b_1(t) - \frac{\hbar\,\Delta(t)}{2}\,b_2(t),$$

where the time-dependent detuning is given by

$$\Delta(t) = \omega - \omega_0 + \dot{\phi}(t).$$

Introducing $\hbar$ in all terms, the final Hamiltonian used in the present study—and common in many quantum technology applications—is

$$H_{\mathrm{RWA}} = \frac{\hbar}{2}\begin{pmatrix} \Delta(t) & \Omega(t) \\ \Omega(t) & -\Delta(t) \end{pmatrix}.$$

Bibliography

[1] H. Ma, B. Qi, I. R. Petersen, R.-B. Wu, H. Rabitz, and D. Dong, "Machine learning for estimation and control of quantum systems," Mar. 2025. *arXiv preprint arXiv:2503.03164v1.*

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

[3] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[4] A. Barenco, C. H. Bennett, R. Cleve, D. P. DiVincenzo, N. Margolus, P. Shor, T. Sleator, J. A. Smolin, and H. Weinfurter, "Elementary gates for quantum computation," *Physical Review A*, vol. 52, no. 5, p. 3457, 1995.

[5] P. O. Boykin, T. Mor, M. Pulver, V. Roychowdhury, and F. Vatan, "On universal and fault-tolerant quantum computing," *arXiv preprint arXiv:quant-ph/9906054v1.*

[6] M. Bukov, A. G. Day, D. Sels, P. Weinberg, A. Polkovnikov, and P. Mehta, "Reinforcement learning in different phases of quantum control," *Physical Review X*, vol. 8, no. 3, p. 031086, 2018.

[7] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction.* MIT Press, 2nd ed., 2018.

[8] P. Palittapongarnpim, P. Wittek, E. Zahedinejad, S. Vedaie, and B. C. Sanders, "Learning in quantum control: High-dimensional global optimization for noisy quantum dynamics," *Neurocomputing*, vol. 268, pp. 116–126, 2017.

[9] M. Y. Niu, S. Boixo, V. N. Smelyanskiy, and H. Neven, "Universal quantum control through deep reinforcement learning," *NPJ Quantum Information*, vol. 5, no. 1, p. 33, 2019.

[10] J. Olle, O. M. Yevtushenko, and F. Marquardt, "Scaling the automated discovery of quantum circuits via reinforcement learning with gadgets," Mar. 2025. *ArXiv preprint arXiv:2503.11638v1.*

[11] M. A. Nielsen and C. M. Dawson, "Universality for quantum computation with partially defined quantum gates," *arXiv preprint quant-ph/9906054*, 1999.

[12] D. d'Alessandro, *Introduction to Quantum Control and Dynamics*. CRC Press, 2021.

[13] C. W. Duncan, P. M. Poggi, M. Bukov, N. T. Zinner, and S. Campbell, "Taming quantum systems: A tutorial for using shortcuts-to-adiabaticity, quantum optimal control, & reinforcement learning," Jan. 2025. *ArXiv preprint arXiv:2501.16436v1.*

[14] N. V. Vitanov, A. A. Rangelov, B. W. Shore, and K. Bergmann, "Stimulated raman adiabatic passage in physics, chemistry, and beyond," *Reviews of Modern Physics*, vol. 89, no. 1, p. 015006, 2017.

[15] M. Goerz, D. Basilewitsch, F. Gago-Encinas, M. G. Krauss, K. P. Horn, D. M. Reich, and C. Koch, "Krotov: A python implementation of krotov's method for quantum optimal control," *SciPost Physics*, vol. 7, no. 6, p. 080, 2019.

[16] U. Boscain, M. Sigalotti, and D. Sugny, "Introduction to the Pontryagin maximum principle for quantum optimal control," *PRX Quantum*, vol. 2, no. 3, p. 030203, 2021.

[17] T. H. Su, S. Shresthamali, and M. Kondo, "Quantum framework for reinforcement learning: integrating markov decision process, quantum arithmetic, and trajectory search," Dec. 2024. arXiv prepint arXiv:2412.18208v1.

[18] J. O. Ernst, A. Chatterjee, T. Franzmeyer, and A. Kuhn, "Reinforcement learning for quantum control under physical constraints," *arXiv preprint arXiv:2501.14372*, 2025.

[19] P. W. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pp. 124–134, IEEE, 1994.

[20] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pp. 212–219, ACM, 1996.

[21] J. J. Sakurai and J. Napolitano, *Modern Quantum Mechanics*. Boston: Addison-Wesley, 2nd ed., 2011.

[22] M. A. Nielsen, "A simple formula for the average gate fidelity of a quantum dynamical operation," *Physics Letters A*, vol. 303, no. 4, pp. 249–252, 2002.

[23] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[24] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 12, pp. 1008–1014, 1999.

[25] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[26] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[27] Z. Shao, P. Wang, Q. Zhu, R. Xu, J. Song, X. Bi, H. Zhang, M. Zhang, Y. K. Li, Y. Wu, and D. Guo, "Deepseekmath: Pushing the limits of mathematical reasoning in open language models," *arXiv preprint arXiv:2402.03300*, 2024.

[28] R.-H. He, H.-D. Liu, S.-B. Wang, J. Wu, S.-S. Nie, and Z.-M. Wang, "Universal quantum state preparation via revised greedy algorithm," *Quantum Sci. Technol.*, vol. 6, p. 045021, 2021.

[29] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[30] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, pp. 2094–2100, AAAI Press, 2016.

[31] O. Shindi, Q. Yu, and D. Dong, "A modified deep q-learning algorithm for control of two-qubit systems," in *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, (Melbourne, Australia), IEEE, Oct 2021.

[32] O. Shindi, Q. Yu, P. Girdhar, and D. Dong, "A modified deep q-learning algorithm for optimal and robust quantum gate design of a single qubit system," in *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, 2022.

[33] O. Shindi, Q. Yu, P. Girdhar, and D. Dong, " Model-Free Quantum Gate Design and Calibration Using Deep Reinforcement Learning ," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 01, pp. 346–357, 2024.

[34] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proceedings of the 35th International Conference on Machine Learning (ICML)*, vol. 80, pp. 1587–1596, PMLR, 2018.

[35] H. N. Nguyen, F. Motzoi, M. Metcalf, K. B. Whaley, M. Bukov, and M. Schmitt, "Reinforcement learning pulses for transmon qubit entangling gates," *Mach. Learn.: Sci. Technol.*, vol. 5, p. 025066, 2024.

[36] H. Yu and X. Zhao, "Deep reinforcement learning with reward design for quantum control," *IEEE Transactions on Artificial Intelligence*, vol. 5, p. 1087, Mar 2024.

[37] Z. Wang, N. de Freitas, and M. Lanctot, "Dueling network architectures for deep reinforcement learning," in *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, vol. 48, pp. 1995–2003, PMLR, 2016.

[38] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *4th International Conference on Learning Representations (ICLR)*, 2016.

[39] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proceedings of the 4th International Conference on Learning Representations (ICLR 2016)*, 2016.

[40] T.-N. Xu, Y. Ding, J. D. Martín-Guerrero, and X. Chen, "Robust two-qubit gate with reinforcement learning and dropout," *Phys. Rev. A*, vol. 110, p. 032614, Sep 2024.

[41] 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.

[42] T. Caneva, T. Calarco, and S. Montangero, "Chopped random-basis quantum optimization," *Physical Review A*, vol. 84, no. 2, p. 022326, 2011.

[43] T.-N. Xu, Y. Ding, J. D. Martín-Guerrero, and X. Chen, "Robust two-qubit gate with reinforcement learning and dropout," *Physical Review A*, vol. 110, p. 032614, 2024.

[44] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 8024–8035, 2019.

[45] Z. An and D. L. Zhou, "Deep reinforcement learning for quantum gate control," *Europhysics Letters (EPL)*, vol. 126, p. 60002, 2019.

[46] S. Hu, C. Chen, and D. Dong, "Deep reinforcement learning for control design of quantum gates," in *Proceedings of the 13th Asian Control Conference (ASCC 2022)*, (Jeju Island, Korea), May 2022.

[47] K.-C. Chen, S. Y.-C. Chen, C.-Y. Liu, and K. K. Leung, "Quantum-train-based distributed multi-agent reinforcement learning," 2024. *arXiv prepint arXiv:2412.08845*.

[48] A. Dubal, D. Kremer, S. Martiel, V. Villar, D. Wang, and J. Cruz-Benito, "Pauli network circuit synthesis with reinforcement learning," 2025. *arXiv preprint arXiv: 2503.14448*.

[49] Z. Wang, C. Feng, C. Poon, L. Huang, X. Zhao, Y. Ma, T. Fu, and X.-Y. Liu, "Reinforcement learning for quantum circuit design: Using matrix representations," 2025. *arXiv preprint arVix:2501.16509*.

[50] P. Altmann, J. Stein, M. Kölle, A. Bärligea, T. Gabor, T. Phan, S. Feld, and C. Linnhoff-Popien, "Challenges for reinforcement learning in quantum circuit design," 2024. *arXiv preprint arVix: 2312.11337*.

[51] A. G. Fowler, M. Mariantoni, J. M. Martinis, and A. N. Cleland, "Surface codes: Towards practical large-scale quantum computation," *Physical Review A*, vol. 86, p. 032324, 2012.

[52] A. Steane, "Error correcting codes in quantum theory," *Physical Review Letters*, vol. 77, no. 5, p. 793, 1996.

[53] Z. T. Wang, Q. Chen, Y. Du, Z. H. Yang, X. Cai, K. Huang, J. Zhang, K. Xu, J. Du, Y. Li, Y. Jiao, X. Wu, W. Liu, X. Lu, H. Xu, Y. Jin, R. Wang, H. Yu, and S. P. Zhao, "Quantum compiling with reinforcement learning on a superconducting processor," 2024. *arXiv preprint arXiv: 2406.12195*.

[54] I. Khalid, C. A. Weidner, E. A. Jonckheere, S. G. Schirmer, and F. C. Langbein, "Sample-efficient model-based reinforcement learning for quantum control," *Physical Review Research*, vol. 5, p. 043002, 2023.

[55] Y. Chow, O. Nachum, E. Duenez-Guzman, and M. Ghavamzadeh, "A lyapunov-based approach to safe reinforcement learning," in *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS 2018)*, (Montréal, Canada), 2018.

[56] S. C. Hou and X. X. Yi, "Quantum lyapunov control with machine learning," *Quantum Information Processing*, vol. 19, no. 8, 2020.

[57] 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," *PRX Quantum*, vol. 2, p. 040324, Nov 2021.

[58] Q. Chen, Y. Du, Y. Jiao, X. Lu, X. Wu, and Q. Zhao, "Efficient and practical quantum compiler towards multi-qubit systems with deep reinforcement learning," *Quantum Sci. Technol.*, vol. 9, p. 045002, 2024.

[59] D. Koutromanos, D. Stefanatos, and E. Paspalakis, "Control of qubit dynamics using reinforcement learning," *Information*, vol. 15, no. 5, p. 272, 2024.

[60] M. C. Smith, A. D. Leu, K. Miyanishi, M. F. Gely, and D. M. Lucas, "Single-qubit gates with errors at the 10^{-7} level," 2024. *arXiv preprint arXiv:2412.04421*.

[61] A. Javadi-Abhari, M. Treinish, K. Krsulich, C. J. Wood, J. Lishman, J. Gacon, S. Martiel, P. D. Nation, L. S. Bishop, A. W. Cross, B. R. Johnson, and J. M. Gambetta, "Quantum computing with qiskit," 2024. *arXiv preprint arXiv: 2405.08810*.

[62] D. Koutromanos, "Qubit control with reinforcement learning methods," Master's thesis, Democritus University of Thrace, Department of Electrical and Computer Engineering, Apr. 2024. *MSc. Thesis in Quantum Computing and Quantum Technologies*.