Automatic
Control &
Systems
Engineering.

A Comparative Study of SAC and PPO Algorithms in Reservoir Water Level Control Simulation

Muzhaffar Maruf Ibrahim

September 2024

Supervisor: Dr J Anthony Rossiter

A dissertation submitted in partial fulfilment of the requirements for the degree of MSc in Autonomous and Intelligent System

University of Sheffield

Control Systems Project and Dissertation
ACS6200

Muzhaffar Maruf Ibrahim

Supervisor: Dr J Anthony Rossiter

A report submitted in partial fulfilment of the requirements
for the degree of MSc in Autonomous and Intelligent System

in the

Department of Automatic Control and System Engineering

September 16, 2024

Declaration

All sentences or passages quoted in this document from other people's work have been specifically acknowledged by clear cross-referencing to author, work and page(s). Any illustrations that are not the work of the author of this report have been used with the explicit permission of the originator and are specifically acknowledged. I understand that failure to do this amounts to plagiarism and will be considered grounds for failure.

Name: Muzhaffar Maruf Ibrahim

Date: 16 September 2024

Abstract

This research investigates the performance of three newest version of algorithms in reinforcement learning (RL) as an advanced control strategy for water level control with single-tank system and quadruple-tank system, contrasting it with the conventional PID (proportional-integral-derivative) control within the framework. RL, which autonomously learns by interacting with its environment, is becoming increasingly popular for developing optimal controllers for complex, dynamic, and nonlinear processes. Unlike most RL studies that use open-source platforms like Python and OpenAI Gym, this research utilizes MATLAB's Reinforcement Learning Toolbox (introduced in R2024a) to design a water tank model using Transfer function and State Space Equation. The controller is trained using Soft Actor-Crtic (SAC), Proximal Policy Optimization (PPO), and Deep Deterministic Policy Gradient (DDPG) algorithm, with Simulink employed to simulate the water tank system and establish an experimental test bench for comparsion between them.

The findings indicate that the Soft Actor-Critic (SAC) algorithm deliver the best result in signal tracking, achieving high speed and low error relative to the reference signal when compared to other reinforcement learning (RL) algorithms. However, algorithms such as Proximal Policy Optimization (PPO) demonstrate superior performance in maintaining minimal steady-state error, despite requiring a significantly longer training period in the context of a single-tank system. In more complex scenarios, such as the quadruple-tank system, SAC proves to be superior due to its advantage in handling continuous action spaces, where PPO tends to diverge significantly.

Achieving success in machine learning necessitates the tuning of numerous hyperparameters, a process that is both time-consuming and labor-intensive. The practical insights derived from this research are corroborated by existing literature, underscoring the robustness and applicability of the findings.

Individual Contribution

This report delves into a detailed performance comparison between the latest iterations of two prominent Reinforcement Learning Algorithms: Soft Actor Critic (SAC) and Proximal Policy Optimization (PPO). These algorithms employ distinct approaches to value iteration within their critic networks to achieve the target water level. The primary contributions of this report are outlined below:

1. The design of control architectures and algorithms that incorporate reinforcement learning approaches into existing controller structures for single-tank systems and quadruple-tank systems.

2. This report presents a comprehensive comparison of various Reinforcement Learning (RL) algorithms, including the previous version using Deep Deterministic Policy Gradient (DDPG), the latest versions employing Soft Actor Critic (SAC) and Proximal Policy Optimization (PPO), and widely recognized Proportional-Integral-Derivative (PID) controller as a performance benchmark.

Acknowledgment

I would like to express my profound gratitude to Almighty Allah for His boundless mercy throughout my life, particularly in my academic journey.

I extend my heartfelt thanks to the Department of Automatic Control and System Engineering for granting me the opportunity to undertake this master program and for his unwavering support and guidance. Your support have been invaluable.

My sincere appreciation goes to my supervisors, Dr.J. Anthony Rossiter, for their intellectual support, guidance, and extensive feedback on my research.

I acknowledge with gratitude my senior colleagues and flatmates, Mr. Ridza Adhandra, Mr. Rizky Rizaldy, Ms. Kanya Citta Hanna Alifia, and Ms. Nursechafia, for their encouragement and support both within and outside the academic environment. To all my colleagues, past and present, the Indonesian Community, LPDP awardees, and department friends, thank you all.

I dedicate this research work to my family: my beloved parents, Mr.Rafli Zarmi and Mrs. Murniyetti; my dear sister, Rifda Rihhadatul Aisy, and my beloved pondok labu family, my lovely Nenek, Mami and Papi, Ibu and Ayah, Om Iman and Tante Ira, and for my beloved gramps who inspired me to pursue Master Degree. Your relentless prayers and support have been my strength. I love you all.

Muzhaffar Maruf Ibrahim September 2024

Contents

List of Figures

List of Tables

Abstract

This research investigates the performance of three newest version of algorithms in reinforcement learning (RL) as an advanced control strategy for water level control with single-tank system and quadruple-tank system, contrasting it with the conventional PID (proportional-integral-derivative) control within the framework. RL, which autonomously learns by interacting with its environment, is becoming increasingly popular for developing optimal controllers for complex, dynamic, and nonlinear processes. Unlike most RL studies that use open-source platforms like Python and OpenAI Gym, this research utilizes MATLAB's Reinforcement Learning Toolbox (introduced in R2024a) to design a water tank model using Transfer function and State Space Equation. The controller is trained using Soft Actor-Crtic (SAC), Proximal Policy Optimization (PPO), and Deep Deterministic Policy Gradient (DDPG) algorithm, with Simulink employed to simulate the water tank system and establish an experimental test bench for comparsion between them.

The findings indicate that the Soft Actor-Critic (SAC) algorithm deliver the best result in signal tracking, achieving high speed and low error relative to the reference signal when compared to other reinforcement learning (RL) algorithms. However, algorithms such as Proximal Policy Optimization (PPO) demonstrate superior performance in maintaining minimal steady-state error, despite requiring a significantly longer training period in the context of a single-tank system. In more complex scenarios, such as the quadruple-tank system, SAC proves to be superior due to its advantage in handling continuous action spaces, where PPO tends to diverge significantly.

Achieving success in machine learning necessitates the tuning of numerous hyperparameters, a process that is both time-consuming and labor-intensive. The practical insights derived from this research are corroborated by existing literature, underscoring the robustness and applicability of the findings.

Chapter 1

Introduction

Reinforcement Learning (RL) is a sophisticated branch of machine learning that seeks to replicate the learning processes observed in humans and animals. Unlike traditional machine learning techniques that rely on a fixed dataset, RL involves an agent that learns by interacting with its environment and receiving feedback in the form of rewards or penalties [55]. This dynamic learning process allows RL to adapt and improve its performance over time, making it particularly suitable for complex and changing environments and tries to overcome various limitations owned by supervised learning, unsupervised learning, and semi-supervised learning by combining previous ML methods into one unified algorithm through process modification learning. The main goal of RL in general is to provide machines with capabilities for learning through optimal mapping in a condition (environment) through action (referred as policy) by doing a search of trial-and-error guided by signals [55]. These two features of guided trial and error can be defined as delayed feedback, differentiate RL from various other ML approaches, and ultimately expand new frontiers of current knowledge. In some special scenarios, action has an effect not only on local areas but also on global rewards in the future.

RL has been employed in a variety of groundbreaking applications across different fields, where RL was used to program robotic hands [4]. These robotic hands were able to manipulate physical objects with unprecedented human-like dexterity, showcasing the potential of RL in robotics. Another significant application is in autonomous driving, where [65] CARMA program has explored the use of RL to develop self-driving cars that can navigate complex environments safely and efficiently. Additionally, RL has been studied for its potential in accelerating the design of new molecules, which could have far-reaching implications in fields such as pharmaceuticals and materials science [45]. Therefore, RL can solve non-linear and adaptive control due to large values and time parameters that vary greatly. Particularly, this can be an alternative to commonly used control systems such as PID control systems [26].

Traditional control strategies, such as Proportional-Integral-Derivative (PID) controllers, have been the cornerstone of industrial control systems for decades. Based on the report

by [13], PID controllers are used in over 95% of industrial control applications. However, these conventional strategies often require precise modeling of the system, which can be challenging and time-consuming. RL, on the other hand, offers a promising alternative by directly interacting with the system and learning optimal control policies through trial and error. This approach eliminates the need for accurate system modeling, potentially leading to more efficient and cost-effective control solutions.

While RL offers significant advantages, it also presents several challenges, including the need for extensive computational resources, as RL algorithms often require large amounts of data and processing power. Additionally, ensuring the safety and reliability of RL-based control systems in critical applications remains a key concern [55]. Future research should focus on addressing these challenges, exploring ways to make RL more efficient and robust, and expanding its applications to other areas of industrial control [41].

Building upon the research conducted by Rajesh Siraskar in 2014 [54], which investigates the potential of Reinforcement Learning (RL) to address the practical industrial challenge of maintaining water level in a single-tank system through the manipulation of a control valve, this research experimentally compares two RL algorithms grounded in distinct approximator object concepts: Soft Actor-Critic and Proximal Policy Optimization to a complex quadruple-tank system. These algorithms are employed to solve the stochastic policy optimization problem, formulated as a probability distribution with a complex dynamic system of Quadruple-Tank. Several research conducting the performance comparison for different applications have been conducted such as Quadruped Walking Gait Generation [25].

1.1 Motivation

In contemporary industrial plants, automatic control systems typically employ proportional-integral-derivative (PID) controllers, programmable logic controllers (PLCs), and field programmable gate arrays (FPGAs) [20, 71, 73, 58]. PLCs are predominantly utilized in safety systems to provide a rapid and dependable response, thereby preventing minor malfunctions from escalating into significant accidents. For non-safety systems, PID controllers or combinations of two types of controllers, such as proportional-integral controllers, are most commonly used. These controllers are designed to maintain system stability within a specified range.

Traditional methods for tuning PID controllers, such as Ziegler–Nichols [73], Cohen-Coon [40], and Astrom and Hagglund [74], have been widely implemented. However, these conventional techniques often require re-tuning before application to industrial processes due to potential issues like high overshoots, significant oscillations, and prolonged settling times in higher-order systems [54]. To address these limitations, advanced intelligent tuning methods have been introduced such as Harris hawks optimization (HHO) to optimize PID

controller parameters for an aircraft pitch control system [15], enhanced atom search optimization algorithm using simulated annealing (SA) [40], and tuning algorithms comparison between Ziegler–Nichols and particle swarm optimization [57]. For mobile robots, Ignacio Carlucho et al. proposed self-adaptive multiple PID controllers utilizing deep reinforcement learning (DRL) [11].

In recent years, the application of artificial intelligence (AI) techniques in control systems has gained significant attention across various industrial sectors [68]. Since the early 2000s, advancements in deep learning have been propelled by factors such as enhanced computational power, the exponential growth of data, and significant progress in deep learning research [19, 51]. Among these advancements, deep reinforcement learning (DRL) has emerged as a particularly noteworthy approach due to its human-like training mechanism, which relies on experiential learning through trial and error. This characteristic enables DRL-based controllers to undertake tasks that traditional controllers are incapable of performing, such as devising operational strategies, managing system operations, making real-time decisions based on current conditions, and optimizing overall operations. Consequently, DRL-based controllers have been successfully implemented in a wide array of applications, including robotics [20, 64], autonomous vehicles [46, 32], smart buildings [30], power management [30], the railway industry [70], wind turbines [50], traffic signal control [52], and nuclear power plants [14]. These advancements underscore the transformative potential of AI-driven control systems in enhancing the efficiency, reliability, and adaptability of industrial processes.

Despite significant advancements in developing AI-based controllers, their practical application in industry remains limited. This is primarily due to concerns regarding their ability to consistently demonstrate robustness and accuracy, as well as challenges related to regulatory compliance, particularly in terms of algorithm transparency. Nevertheless, AI-based controllers are anticipated to play a crucial role in the future of autonomous industrial controls, especially for large-scale control architecture with minimum participation of control engineers to establish the control law [54].

1.2 Aims and Objectives

The primary aim of this study is to evaluate and compare the performance characteristics of policy-based and value-based reinforcement learning strategies, as represented by PPO and SAC algorithms, within the domain of water level control in modeled reservoir systems of Quadruple-Tank. This study aims to investigate the performance of both SAC and PPO algorithms in regulating water levels within a multi-tank system to achieve desired setpoints. The following research objectives have been established and structured accordingly:

- Gain theoretical underpinnings and literature review for Soft Actor-Critic and Proximal Policy Optimization agent.

- Model an environment for a water reservoir system to evaluate and compare the control performance of Soft Actor-Critic (SAC) and Proximal Policy Optimization (PPO) algorithms.
- Conduct a series of experiments to compare SAC and PPO regarding control performance, learning efficiency, and policy adaptability.

1.3 Project Management

The project in Gantt Chart 1.1 will commence with a comprehensive initial discussion to establish its goals, scope, and potential challenges. Following this, a thorough literature review will be undertaken to identify existing research on Reinforcement Learning (RL) control applications, particularly those relevant to the project's focus on the single and quadruple-tank systems. Building upon this knowledge, the SAC and PPO algorithms will be implemented, which are recognized for their effectiveness in RL. A Simulink project will then be established to simulate these environments, which began the step of experimentation. A suitable reward function will be developed to guide the RL agents toward desired behaviors divided by Single-tank reward function and Quadruple-tank reward function. Once the simulation is up and running, experiments will be conducted to gather data on the performance of the RL agents resulting on training process data, developed agent per episode, and control performance for each algorithms. This data will be analyzed to evaluate the effectiveness of the control system, identify areas for improvement, and refine the algorithms as needed. The findings will be incorporated into a comprehensive report, which will summarize the project's key results and conclusions. In addition, a PowerPoint presentation will be prepared for the project presentation day, providing an overview of the project and highlighting its key findings. As the project nears completion, the final revisions will be made to the report, ensuring that it is clear, concise, and accurately reflects the project's outcomes. The final report will then be submitted, marking the successful conclusion of the project.

Due to unforeseen health challenges faced by the writer, the duration of the data preparation and data analysis phases has been adjusted to accommodate the necessary accommodations. These adjustments have necessitated a reevaluation of the project timeline and resource allocation. To mitigate the impact of these changes and ensure the project remains on track, several modifications have been implemented to the Reinforcement Learning hyper-parameters. These modifications aim to enhance training speed, control performance, and overall training effectiveness. By optimizing these hyper-parameters, the project seeks to maintain its original objectives and deliver the desired outcomes despite the unforeseen circumstances in Gannt Chart 1.2.

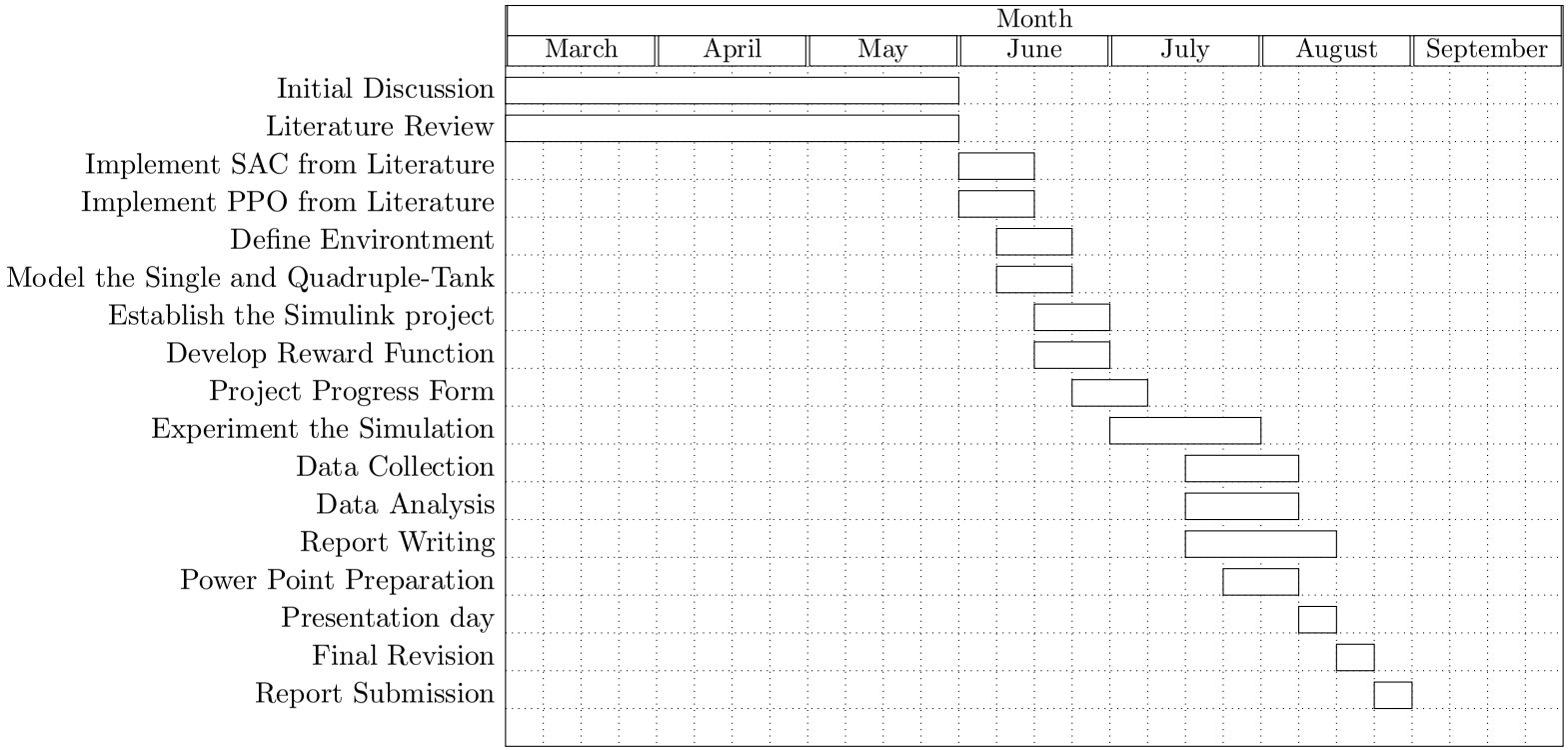

Figure 1.1: *Previous Gantt Chart*

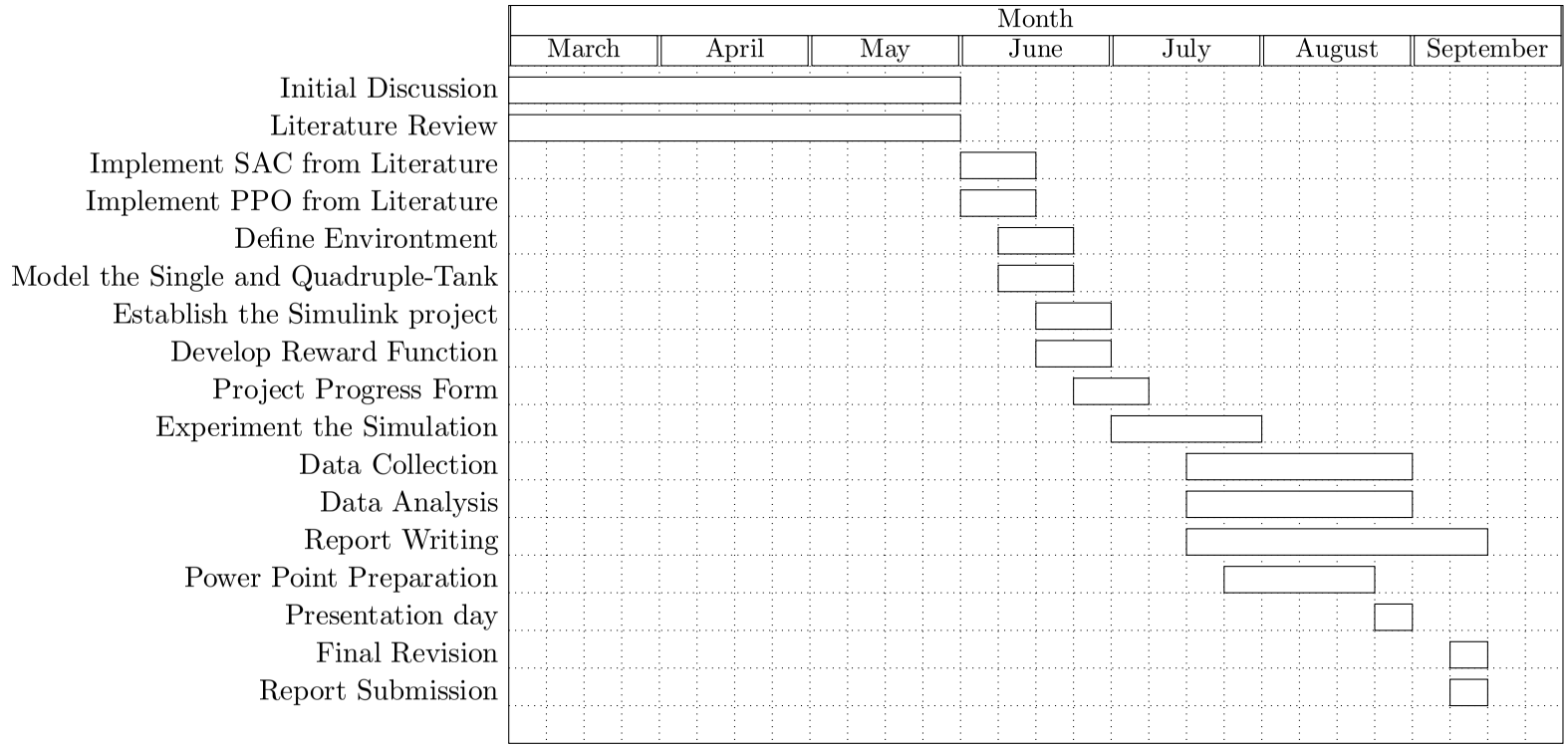

Figure 1.2: *Revised Gantt Chart*

1.4 Report Overview

In Chapter 2, the report presents a comprehensive literature review of cutting-edge optimal and adaptive control strategies aimed at optimizing system performance under uncertain conditions. It explores the challenges and ongoing research efforts related to model-free adaptive approaches. The chapter identifies reinforcement learning and approximate dynamic programming as promising model-free adaptive strategies, highlighting their ability to iteratively learn and optimize solutions to achieve desired performance costs. The chapter concludes with a discussion on open research areas that are further explored in subsequent chapters of the thesis.

Chapter 3 delves into the methodology for implementing Reinforcement Learning (RL) algorithms, specifically Deep Deterministic Policy Gradient (DDPG), Soft Actor Critic (SAC), and Proximal Policy Optimization (PPO). The chapter begins by detailing the process of constructing the simulation architecture, which encompasses the creation of the environment, the modeling of the industrial process, and the establishment of reinforcement learning protocols. This framework includes the architecture for a single-tank system, facilitating the execution of DDPG, SAC, PPO, and Proportional-Integral-Derivative (PID) controllers. Additionally, it covers the architecture for a quadruple-tank system, focusing on the implementation of SAC and PPO algorithms.

Chapter 4 provides a comprehensive analysis of the outcomes from the Reinforcement Learning (RL) training process applied to water level control problems. This chapter meticulously examines various critical variables, including the duration of the training process, the average reward obtained, and the total number of training episodes conducted. Additionally, it delves into the deployment phase of each trained algorithm, evaluating their performance based on several key metrics. These metrics include overshoot, rise time, settling time, peak value, and peak time, which are essential for assessing the responsiveness and stability of the control systems. The chapter aims to offer a detailed understanding of how different RL algorithms perform in practical scenarios, highlighting their strengths and potential areas for improvement. Through simulation examples and empirical data, the chapter demonstrates the effectiveness of the RL-based control strategies in maintaining optimal water levels, thereby providing valuable insights for future research and applications in the field of control engineering.

Finally, Chapter 5 offers concluding remarks on the proposed strategies and provides recommendations for future research directions.

Chapter 2

Background and Literature Review

This chapter reviews the latest optimal and adaptive control strategies for optimizing system performance amidst uncertainties. Most of these strategies rely on model-based approaches, which demand significant effort to develop high-fidelity models that accurately represent varying systems and operating conditions. However, for complex systems, these model-based methods often struggle to adapt to system variations. The chapter outlines the pros and cons of these techniques and explores recent research into data-driven adaptive strategies and reinforcement learning (RL). It also delves into the key topics and technical foundations of RL covered in this report. Additionally, the chapter reviews current research trends and applications of RL from three different application of RL Algorithms, including DDPG, SAC, and PPO.

2.1 Strategies for Optimal and Adaptive Control

Designing optimal controllers relies on having complete system information and setting limits on potential disturbances. A well-known example is the linear quadratic regulator (LQR), which is created offline by solving the Hamilton-Jacobi-Bellman (HJB) equations with full knowledge of the system dynamics [36]. In contrast, adaptive controllers use real-time system measurements to learn and adjust the controller's behavior in response to changes in system dynamics and operating conditions. The methods by which these controllers learn and adapt define various adaptive algorithms, some of which are explored in this chapter.

As researchers have gained a better understanding of these problems, newer adaptive algorithms have been developed to ensure the persistence of excitation and separation of time scales between adaptation and system dynamics. These newer adaptive schemes have shown promising applications, driven by the need to operate systems at varying optimal set-points for improved performance [14, 33, 59, 2].

This report focuses on adaptive algorithms that aim to achieve user-defined performance goals (both adaptive and optimal) while ensuring stability and convergence of the adaptation

process for practical use. The adaptive strategies discussed are broadly categorized into model-based and model-free approaches.

2.1.1 Model-Based

Model-based adaptive control techniques are conventionally classified as either indirect or direct adaptive schemes [75]. Indirect schemes make use of the system measurements to learn new system models via system identification techniques in closed loop [29]. The identified models are then used to adapt the system control law or modify its sensitivity. In contrast, direct model-based adaptive schemes make no efforts to identify new system models but instead use the system measurements to directly adapt parameterised system of controllers in the feed-forward or feedback path [37]. A popular direct adaptive technique is the model reference adaptive control (MRAC), which makes use of a reference model with desired performance characteristics running in tandem with the actual system. An adjustment mechanism compares the output of the reference model with that of the actual system and uses the generated error statistics to adapt the system controller.

Model-based adaptive schemes can be categorized based on their reliance on offline or online models. Offline models, such as gain scheduling, multiple model adaptive, and self-optimizing control schemes, are designed using comprehensive system data [22, 49, 28]. On the other hand, online models, like performance seeking control (PSC) and real-time optimization (RTO), are frequently utilized in aerospace and process industries. Both approaches require significant effort to develop high-fidelity models for effective adaptation.

Whilst the model-based adaptive schemes are considered matured judging by their long history of applications, their performance is limited to the known dynamics of the specific models used. This can be restrictive as the mathematical models are infeasible in fully approximating all the possible variations affecting the system performance. Perhaps, alternatives to these schemes are those that do not rely on explicit mathematical models of the system, but systematically adapt and control the system using obtained measurements.

2.1.2 Model-Free

Model-free adaptive schemes operate without relying on explicit mathematical models or prior system knowledge. They are inherently flexible, capable of handling uncertainties and variations by using real-time system measurements to directly adjust the controllers. Often referred to as data-driven or data-based control, these schemes avoid many of the drawbacks associated with model-based approaches, such as [3, 7]:

1. The necessity for costly models or detailed system knowledge.

2. Challenges in system identification when trying to achieve steady-state control while identifying system dynamics.

3. Complex manual tuning and stability issues linked to fixed model-based controller designs.

Moreover, model-free strategies enable the development of truly optimal and adaptive controllers by optimizing performance costs [18, 31]. Notable model-free strategies include unfalsified control (UC), simultaneous perturbation stochastic approximation (SPSA), iterative feedback tuning (IFT), virtual reference feedback tuning (VRFT), iterative learning control (ILC), extremum seeking control (ESC), and the class of reinforcement learning (RL) and approximate dynamic programming (ADP) [23].

Reinforcement Learning (RL) schemes offer the advantage of learning optimal behaviors over time by relying solely on observed system data. This allows them to address the limitations of other adaptive methods. Traditional adaptive schemes, such as MRAC, IFT, VRFT, and ILC, often assume that a model of the desired performance characteristics is available, either offline or online. This assumption restricts their flexibility in achieving true optimality, as they are bound by the predefined model characteristics. In contrast, RL adopts a novel approach that is entirely based on real-time interaction with the system, regardless of unknown dynamics or variations. This added flexibility makes RL a highly attractive option for optimal and adaptive control, and it has been successfully applied in various complex fields [31, 55, 34].

Reinforcement Learning (RL) has its limitations, primarily because it simplifies the control of complex systems by assuming a Markovian model, where the current state depends only on the previous state. This assumption necessitates extensive data collection to learn the optimal control policy. As a result, RL often relies on complex approximation methods, which can be challenging to ensure robustness and stability [10]. This report aims to extend RL theory to new frameworks that offer practical design and control solutions for complex dynamic systems, using the case of single-tank and quadruple-tank system. The remainder of this chapter will cover the key elements and foundational theories of RL used throughout the report.

2.2 Reinforcement Learning with Adaptive Dynamic Programming

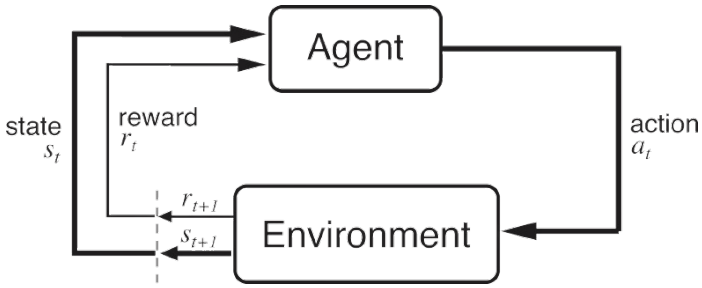

Figure 2.1: *Markov Decision Process [55]*

Common Learning systems can be divided into three main categories based on the type of feedback they use: supervised, unsupervised, and reinforcement learning. Supervised learning involves training a model to map inputs to outputs using labeled data. Unsupervised learning, on the other hand, identifies hidden patterns in data without any labeled inputs. Reinforcement learning (RL) is distinct in that it learns by interacting with the environment, using feedback from these interactions to improve future performance [21]. This approach is particularly relevant in practical applications, making RL a vibrant area of research with uses in fields such as computer science, artificial intelligence, operations research, robotics, and control systems.

In control systems, the mathematical application of Reinforcement Learning (RL) has been facilitated by approximate/adaptive dynamic programming (ADP) which optimizes performance costs and learns optimal control policies using only data collected along the system's trajectory. Both RL and ADP are fundamentally based on the principles of dynamic programming which mostly studied under the Markov decision process (MDP) framework [18] shown in Figure 2.1.

2.2.1 Adaptive Dynamic Programming

Dynamic programming (DP), introduced by Bellman [6], offers a structured approach to solving sequential decision problems (SDPs) optimally. SDPs involve making a series of decisions and observations, and they are prevalent in fields such as operations research and control engineering. The DP method is inherently recursive in equation 2.1 and has been utilized to address a wide range of problems with both continuous and discrete states and actions [47].

The goal in DP is then to optimize some desired cost function that is additive over time as result of visiting state s_0 to s_k and taking action a_0 to a_k and to find an optimal policy, denoted by π, that maximizes the expected cumulative discounted reward over time. A policy is a mapping from states to actions, $\pi : \mathcal{S} \rightarrow \mathcal{A}$ given as [55]:

$$
\begin{aligned}
v_\pi(s) &= E[G_t \mid S_t = s] \\
&= E\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right] \\
&= E\left[R_{t+1} + \gamma \sum_{k=0}^{\infty} \gamma^k R_{t+k+2} \mid S_t = s\right]
\end{aligned}
\tag{2.1}
$$

In the context of control systems, the goal of DP is to learn a control policy that effectively guides the agent towards desired outcomes without requiring explicit programming or detailed system modeling [18].

A Markov Decision Process (MDP) is formally defined by the following components [54]:

- **State space**: A set of states, denoted by $\mathcal{S}$, representing the possible configurations of the environment.
- **Action space**: A set of actions, denoted by $\mathcal{A}$, representing the choices available to the agent in each state.
- **Transition function**: A probability distribution $P(s'|s, a)$, which specifies the probability of transitioning from state s to state s' by taking action a.
- **Reward function**: A function $R(s, a, s')$ that assigns a scalar reward to each state-action-next state combination.
- **Start state**: An initial state $s_0 \in \mathcal{S}$ from which the agent begins its interaction with the environment.
- **Discount factor**: A scalar value $\gamma \in [0, 1)$ that determines the importance of future rewards relative to immediate rewards. A discount factor closer to 1 places more emphasis on long-term rewards, while a discount factor closer to 0 focuses on short-term rewards.
- **Horizon**: An optional parameter H that specifies the maximum number of time steps in the MDP. If the horizon is infinite, the MDP is considered to be an infinite horizon.

Dynamic programming offers a robust solution to optimization problems through the principle of optimality in the equation 2.2. This principle imposes that an optimal policy possesses the characteristic that, irrespective of prior decisions or actions, the subsequent decisions must form an optimal policy for the state resulting from those prior decisions [6]. This implies that the optimal control sequence can be decomposed into smaller, manageable stages [49]. The process initiates by determining the optimal decision for the final stage, referred to as the tail sub-problem, and then systematically addressing the preceding stages

until the entire problem is comprehensively solved. This methodical approach ensures a structured and efficient resolution of complex optimization challenges.

$$\begin{aligned} v_\pi(s) &= \max_{a \in A(s)} q_\pi(s,a) \\ &= \max_{a \in A(s)} \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a] \\ &= \max_{a \in A(s)} \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right] \\ &= \max_{a \in A(s)} \mathbb{E}_\pi[R_{t+1} + \gamma V(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_{a \in A(s)} \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_\pi(s')] \end{aligned} \tag{2.2}$$

2.2.2 Reinforcement Learning Framework

Reinforcement Learning (RL) and Adaptive Dynamic Programming (ADP) frameworks are extensively used to address the curse of dimensionality that limits traditional Dynamic Programming (DP) methods [47]. Initially, termed Adaptive Critic Designs (ACDs) approaches involve training a network known as the 'critic' to approximate the cost-to-go function in DP solutions [42]. These methods are also referred to as neuro-dynamic programming and critic global controllers [34, 17].

Unlike the backward-in-time iterative solutions of DP, ADP frameworks employ iterative forward-in-time techniques. These techniques use the Bellman optimality equation to formulate value and policy update equations, which are solved at each iteration step. Two well-known iterative forward-in-time methods are value iteration (VI) in equation 2.3 and policy iteration (PI) in equation 2.4.

$$\begin{aligned} v_{k+1}(s) &= \max_a E[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_k(s')] \end{aligned} \tag{2.3}$$

$$\pi(s) = \operatorname*{argmax}_a \sum_{s',r} p(s', r \mid s, a)(r + \gamma V(s')) \tag{2.4}$$

Value Iteration Methods: The Value Iteration (VI) method includes both value updates and policy updates, which progressively enhance policies and can be applied online to determine optimal control policies over time. For the VI method to converge, it is necessary that the cost equation 2.2 remains bounded and that updates 2.3 and 2.4 are performed infinitely often for each state. Algorithm 1 provides the fundamental framework

for the VI method.

Algorithm 1 Value Iteration (VI)

Input: initial policy $\mu_0(s)$
Initialize: $V_0(s)$ (arbitrary value function)
repeat
  for $s \in \mathcal{S}$ **do**
    $V_{k+1}(s) \leftarrow \mathcal{R}(s, a_k(s)) + \gamma \mathbb{E}_{s_{k+1} \sim p(s_{k+1}|s_k, a_k(s))} [V_k(s_{k+1})]$
  end for
  for $s \in \mathcal{S}$ **do**
    $\mu_{k+1}(s) \leftarrow \arg\min_{\mu(\cdot)} \left\{ \mathcal{R}(s, a(s)) + \gamma \mathbb{E}_{s_{k+1} \sim p(s_{k+1}|s_k, \mu(s))} [V_{k+1}(s_{k+1})] \right\}$
  end for
until convergence
Output: optimal policy $\mu^* = \mu_{k+1}$

Policy Iteration Methods: Unlike Value Iteration (VI) methods, Policy Iteration (PI) methods necessitate an initially admissible policy, which must be stabilizing and have a finite cost $V(\cdot)$. The PI methods then proceed by alternating between policy evaluation and policy update steps. Equations and act as consistency checks derived from the Bellman optimality equation and are resolved at each time step (k). When a policy is given, its value is determined by solving Equation 2.5 until it converges, which represents the policy evaluation phase. In systems with a finite state space, this phase is akin to solving a linear system of equations. Subsequently, an enhanced policy is calculated using Equation 2.6, marking the policy update phase.

Algorithm 2 Policy Iteration (PI)

Input: initial policy $\mu_0(s)$
Initialize: $V_0(s)$ (arbitrary value function)
repeat
  for $s \in \mathcal{S}$ **do**
    $V_{k+1}(s) \leftarrow \mathcal{R}(s, \mu_k(s)) + \gamma \mathbb{E}_{s_{k+1} \sim p(s_{k+1}|s_k, \mu_k(s))} [V_k(s_{k+1})]$
  end for
  for $x \in \mathcal{S}$ **do**
    $\mu_{k+1}(s) \leftarrow \arg\min_{\mu(\cdot)} \left\{ \mathcal{R}(s, \mu(s)) + \gamma \mathbb{E}_{s_{k+1} \sim p(s_{k+1}|s_k, \mu(s))} [V_{k+1}(s_{k+1})] \right\}$
  end for
until convergence
Output: optimal policy $\mu^* = \mu_{k+1}$

With this method, the algorithm computes a strictly improved policy and convergence to the optimal policy and value under the assumption that the system is controllable, as shown in [24],[48]. Algorithm 2 gives the basic template for the PI method. In general, the VI methods are less computationally demanding than the PI methods as they require only a one-step recursion in the value update step as opposed to solving the system of equations in the policy evaluation step. This characteristic would be shown in the results of the research.

However, the PI methods are known to converge in fewer iterations [48], [8].

$$\begin{aligned} v_{k+1}(s) &= \max_a E[R_{t+1} + \gamma v_k(S_{t+1}) \mid S_t = s, A_t = a] \\ &= \max_a \sum_{s',r} p(s', r \mid s, a)[r + \gamma v_k(s')] \end{aligned} \tag{2.5}$$

$$\pi(s) = \operatorname*{argmax}_a \sum_{s',r} p(s', r \mid s, a)(r + \gamma V(s')) \tag{2.6}$$

The iterative VI and PI methods function as online strategies due to their forward-in-time dynamic programming recursion, leveraging system dynamics and cost information to achieve optimal values. To facilitate model-free and forward-in-time online strategies, ADP employs function approximations to estimate the cost.

There are several techniques to estimate the approximated costs, including Monte Carlo (MC) and temporal difference (TD) methods [55]. MC methods are typically used in simulations, where learning happens episodically. This means that the approximated costs are updated based on data collected after a defined training period, starting from system initialization to certain terminal conditions. On the other hand, TD methods update incrementally and can be applied online, using data gathered along the system's trajectory. A hybrid method known as TD(λ) combines the incremental nature of TD with the episodic approach of MC [55]. This report will compare two algorithms which used the Temporal Difference (SAC) and Monte Carlo methods (PPO) which will be disucssed in the Result Chapter.

Actor-Critic Framework

Actor-critic frameworks leverage the advantages of both actor-only and critic-only frameworks by optimizing the policy space through the actor network and utilizing the low-variance value function approximations from the critic network. They employ two networks: the critic network, which approximates the value function, and the actor-network, which approximates the control policy.

The critic parameters, denoted as $\Phi_c(s)$ and the basis function with p_c features, are used to approximate the value function $V_m(s)$ as follows:

$$V^{(\mu)}(s) \approx \theta_{c,k}^{\top} \Phi_c(s) = \sum_{n=k}^{N} \gamma^{-n+k} R(s, a) \tag{2.7}$$

This equation represents the value function approximation for the critic network, which sums the discounted reward signals $R(s, a)$ starting from state s under a fixed policy $\mu(s)$.

Similarly, the actor network approximates the control policy as:

$$\mu(s) \approx \theta_{a,k}^T \Phi_a(s) \tag{2.8}$$

The critic network try to minimize the Bellman error which is given based on either a VI or PI recursion:

1. VI recursion: $e_{c,k} = R(s_k, a_k) + \gamma \theta_{c,k}^T \Phi(s_{k+1}) - \theta_{c,k+1}^T \Phi(s_k)$
2. PI recursion: $e_{c,k} = R(s_k, a_k) - \theta_{c,k+1}^T \Phi(s_k) - \gamma \Phi(s_{k+1})$

Following the update of the critic network, the actor network updates the control policy by minimising the critic estimates using gradient descent tuning as follows:

$$\theta_{a,k+1} = \theta_{a,k} - l_a \nabla_{\theta_a} V(s_k) \tag{2.9}$$

$$= \theta_{a,k} - l_a \nabla_{\theta_a} \theta_{c,k+1}^\top \Phi(s_k) \tag{2.10}$$

Evaluating the critic gradient estimates is crucial as it links the updates of the two networks. These estimates can be obtained during simulation or approximated using the policy gradient theorem [56]. The actor-critic update sequences generally lead to less oscillatory behavior, aiding convergence, as changes in the critic network are matched by small variations in the policy determined by the learning rate (λ).

The actor-critic framework is widely used in reinforcement learning (RL) methods. Applications include controlling discrete and continuous time dynamical systems, flight control systems, electrical power systems, dynamic energy management systems, and unmanned aerial vehicles. The next section will review recent applications of RL in control, which is the focus of this report.

Deep Deterministic Policy Gradient (DDPG)

The Deep Deterministic Policy Gradient (DDPG) algorithm is a model-free, off-policy reinforcement learning method that effectively learns optimal policies in continuous action spaces that employs an actor-critic architecture, where the actor-network generates continuous actions, and the critic network evaluates the quality of those actions defined by [41]. The algorithm leverages the strengths of both Deterministic Policy Gradient (DPG) and Deep Q-Networks (DQN) which learns a policy by directly optimizing a parameterized actor function $\mu_{\theta_\mu}(s)$ and determines the next action based on the current state. A critic function, $Q_{\theta_Q}(s, a)$, trained using the Bellman equation, estimates the expected future reward. By balancing exploration and exploitation, DDPG efficiently converges to near-optimal policies defined by [41].

The actor's parameters are adjusted to maximize the expected return, following a gradient update based on the chain rule in equation 2.11.

$$\nabla_{\theta_\mu} J(\theta_\mu) = \mathbb{E}_{s\sim\rho, a\sim\pi_{\theta_\mu}(s)} \left[\nabla_{\theta_\mu} \log \pi_{\theta_\mu}(a|s) Q_{\theta_Q}(s,a) \right] \tag{2.11}$$

$$= \mathbb{E}_{s\sim\rho} \left[\nabla_{\theta_\mu} \mu_{\theta_\mu}(s) \cdot \nabla_a Q_{\theta_Q}(s,a) \Big|_{a=\mu_{\theta_\mu}(s)} \right] \tag{2.12}$$

Where:

(i) θ_μ: Parameters of the actor function.

(ii) θ_Q: Parameters of the critic function.

(iii) $J(\theta_\mu)$: Expected return.

(iv) $\pi_{\theta_\mu}(a|s)$: Policy distribution.

(v) $\mu_{\theta_\mu}(s)$: Deterministic actor function.

(vi) $Q_{\theta_Q}(s,a)$: Q-value function.

(vii) ρ: Start state distribution.

Based on the reserach of by [61], The Off-Policy Actor-Critic (OffPAC) algorithm simplifies the learning process by focusing on a specific part of the gradient. Instead of considering the full gradient, OffPAC omits a term related to the action-value gradient in the DDPG Algorithm. This simplification is justified because it doesn't compromise the ability to find good solutions. OffPAC uses a behavior policy to generate experiences. These experiences are then used by a critic to estimate the value of different states. The actor, which controls the policy, is updated based on these experiences using a technique called stochastic gradient ascent. In order to address the challenge of using a behavior policy, OffPAC employs importance sampling. This technique adjusts the updates to account for the difference between the behavior policy and the target policy. This ensures that the actor learns effectively, even though the experiences were generated using a different policy.

Soft Actor-Critic (SAC)

The Soft Actor-Critic (SAC) algorithm is a model-free, online, off-policy, actor-critic reinforcement learning method designed to compute an optimal policy that not only maximizes the expected long-term reward but also the entropy of the policy defined by the paper from [12]. The policy entropy serves as a measure of the policy's uncertainty given the current state. A higher entropy value encourages the agent to explore more, preventing it from becoming overly focused on exploiting known rewards. By maximizing both the expected

cumulative long-term reward and the entropy, SAC effectively balances exploration and exploitation, enabling it to navigate complex environments and discover optimal solutions cited from [41].

The SAC algorithm optimizes the soft Q-function parameters to minimize the soft Bellman residual in Equation 2.13, which measures the difference between the predicted Q-value and the actual reward. The value function is implicitly parameterized by the soft Q-function parameters and can be optimized using stochastic gradient descent.

$$J_Q(\theta) = \mathbb{E}_{(s,a)\sim D}\left[\frac{1}{2}\left(Q_\theta(s_t,a_t) - \left(r(s_t,a_t) + \gamma \mathbb{E}_{s_{t+1}}[V_\theta(s_{t+1})]\right)\right)^2\right] \tag{2.13}$$

$$\nabla_\theta J_Q(\theta) = \mathbb{E}_{(s,a)\sim D}\left[\left(Q_\theta(s_t,a_t) - \left(r(s_t,a_t) + \gamma \mathbb{E}_{s_{t+1}}[V_\theta(s_{t+1})]\right)\right) \nabla_\theta Q_\theta(s_t,a_t)\right] \tag{2.14}$$

$$J_\pi(\phi) = \mathbb{E}_{s\sim D}\Big[\mathbb{E}_{a\sim\pi_\phi(a|s)}\left[\alpha \log(\pi_\phi(a|s)) - Q_\theta(s,a)\right]\Big] \tag{2.15}$$

The algorithm uses a target soft Q-function with parameters obtained from an exponentially moving average of the current soft Q-function weights which helps stabilize training [53]. The policy parameters are learned by directly minimizing the expected KL-divergence, a measure of the difference between the target policy and the current policy. To optimize the policy, the algorithm uses the reparameterization trick, which is a technique that avoids back-propagating the gradient through the policy and the target density network. This results in a lower variance estimator and improves the efficiency of the optimization process.

The contribution from entropy regularization starting with the recursive Bellman equation for the entropy-regularized Q-function with rewrite it using the definition of entropy in equation 2.16

$$Q^\pi(s,a) = \mathbb{E}\left[R(s,a,s') + \gamma\left(Q^\pi(s',a') - \alpha \log \pi(a'|s')\right)\right] \tag{2.16}$$

In the equation 2.17, the right-hand side (RHS) represents an expectation over the next states, which are sampled from the replay buffer, and the next actions, which are sampled from the current policy. To make this expectation computationally feasible, we approximate it using samples:

$$Q^\pi(s,a) \approx R(s,a,s') + \gamma\left(Q^\pi(s',a') - \alpha \log \pi(a'|s')\right) \tag{2.17}$$

a' is sampled from the current policy, ensuring that the policy remains updated with the latest information.

The Mean Squared Bellman Error (MSBE) loss for each Q-function in SAC is defined

as:

$$L(\phi, \mathcal{D}) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{D}}\left[\left(Q_\phi(s,a) - \left(r + \gamma\left(\min_{i=1,2} Q_{\phi_i}(s',a') - \alpha \log \pi(a'|s')\right)\right)\right)^2\right] \quad (2.18)$$

In this loss function, the target value $y(r, s')$ is given by:

$$y(r,s') = r + \gamma\left(\min_{i=1,2} Q_{\phi_i}(s',a') - \alpha \log \pi(a'|s')\right) \quad (2.19)$$

This objective ensures that the policy not only seeks to maximize the expected return but also maintains a level of randomness in its actions, which is beneficial for exploration.

The optimization process leverages the reparameterization trick, where actions are sampled as a deterministic function of the state, policy parameters, and independent noise. This can be represented as:

$$\pi^* = \arg\max_\pi \mathbb{E}_{s\sim d^\pi}\left[\mathbb{E}_{a\sim\pi}\left[Q^\pi(s,a) - \alpha \log \pi(a|s)\right]\right] \quad (2.20)$$

$$a(s,\xi) = \tanh\left(\mu(s) + \sigma(s) \odot \xi\right), \quad \xi \sim \mathcal{N}(0, I) \quad (2.21)$$

In this formulation, $(\mu(s))$ and $(\sigma(s))$ are the mean and standard deviation of the policy's action distribution, and (ξ) is a noise vector sampled from a standard normal distribution. The (tanh) function ensures that the actions remain within a bounded range.

By incorporating entropy regularization and using the reparameterization trick, the SAC algorithm effectively balances exploration and exploitation, leading to more robust and efficient learning.

Proximal Policy Optimization (PPO)

The state value function denoted $v_\pi(s)$, represents the expected future reward that will be obtained starting from state s, measured starting in that state s and following the policy π indefinitely.

Proximal Policy Optimization (PPO) updates policies via:

$$\theta_{k+1} = \arg\max_\theta \mathbb{E}\left[L(s,a,\theta_k,\theta)\right] \quad (2.22)$$

Typically, multiple steps of (usually minibatch) Stochastic Gradient Descent (SGD) are taken to maximize the objective. Here, L is given by:

$$L(s, a, \theta_k, \theta) = \min\left(r(\theta)(a|s)A^{\pi_{\theta_k}}(s, a), \text{clip}(r(\theta)(a|s), 1-\epsilon, 1+\epsilon)A^{\pi_{\theta_k}}(s, a)\right) \tag{2.23}$$

where

$$r(\theta)(a|s) = \frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} \tag{2.24}$$

and $A^{\pi_{\theta_k}}(s, a)$ denotes the advantage function at state s and action a under policy π_{θ_k}. The clip function is defined as:

$$\text{clip}(x, \text{lower\_bound}, \text{upper\_bound}) \tag{2.25}$$

which returns x if it's within the bounds [lower_bound, upper_bound], otherwise it returns the nearest bound.

The advantage function $A^{\pi_{\theta_k}}(s, a)$ is explained as follows:

$$A^{\pi_{\theta_k}}(s, a) = Q^{\pi_{\theta_k}}(s, a) - V^{\pi_{\theta_k}}(s) \tag{2.26}$$

where $Q^{\pi_{\theta_k}}(s, a)$ represents the expected return of taking action a in state s under policy π_{θ_k} and $V^{\pi_{\theta_k}}(s)$ represents the value of state s under policy π_{θ_k}.

Finally, there's an explanation about when this objective contributes to improving policies:

If $r(\theta)(a|s) > 1+\epsilon$ or $r(\theta)(a|s) < 1-\epsilon$, then:

$$L^{\text{clip}}(...) = \begin{cases} \min(r(\theta), 1+\epsilon) \cdot A & \text{if } A > 0 \\ \min(r(\theta), 1-\epsilon) \cdot A & \text{if } A < 0 \end{cases} \tag{2.27}$$

This indicates that when the probability ratio $r(\theta)(a|s)$ falls outside of $[1-\epsilon, 1+\epsilon]$, then $L^{\text{clip}}(...)$ depends on whether A is positive or negative.

2.3 Reinforcement Learning in Control System

This section reviews the recent progress in applying RL to control systems that are dynamic and complex, and that are relevant to this research. The initial research of RL implementation of RL began with the investigation of optimal control regulation of discrete-time system [17, 56, 60].

The reinforcement learning (RL) framework has been applied to the infinite horizon linear quadratic regulation (LQR) problem, where system dynamics are unknown or uncertain. Hagen et al. [60] introduced a Q-learning approach using policy iteration (PI) for the LQR problem, highlighting the necessity of the persistence of excitation (PE) condition for

convergence. A study comparing the RL framework with traditional control theory solutions for the LQR problem in industrial manufacturing processes found that both methods performed similarly, though the Q-learning RL approach required more exploratory noise for its approximations [60].

Further research has focused on enhancing the efficiency and extending RL frameworks to more practical control applications. Various adaptive critic algorithms (HDP, DHP, AD-HDP, and AD-DHP) for the LQR control problem [69], with convergence proofs provided by Landelius [35]. These algorithms were shown to converge to optimal LQR control parameters using only system measurements. For systems with terminal state penalties, a finite horizon control equivalent for discrete-time (DT) linear systems using RL frameworks was demonstrated [72]. Additionally, applications in the continuous-time (CT) domain for optimal state feedback control of linear systems have been proposed. Vrabie et al. [35] utilized a PI-based framework requiring an initially stabilizing policy, while Bian and Jiang proposed a value iteration (VI) based method that does not have this restriction [72].

For discrete-time (DT) nonlinear systems, Dierks and Jagannathan [72] introduced a reinforcement learning (RL) framework that updates policies over time using two neural networks (NN): the critic NN and the action NN. This framework addresses the infinite horizon control regulation problem, assuming an initially stabilizing policy. The two NNs are adjusted at regular intervals based on historical performance data of the nonlinear system. Under the conditions that NN weight estimation errors are uniformly ultimately bounded (UUB) and NN approximation errors are minimal, the control policy estimates were shown to asymptotically reach optimal values.

In a similar case, Bhasin [9] proposed an actor-critic RL framework for continuous-time (CT) nonlinear systems, employing system identification techniques to model system dynamics online. A persistence of excitation (PE) condition ensures the framework's convergence and guarantees UUB stability of the closed-loop system. Lv et al. [38] also suggested an identifier-critic RL framework for optimal control of CT nonlinear systems, utilizing a dual NN structure. The identifier NN learns the system dynamics model, while the critic NN approximates solutions to the nonlinear CT Hamilton-Jacobi-Bellman (HJB) equations, from which the system policy is derived.

Integral reinforcement learning (IRL), as described by Vrabie et al. [67, 66], facilitates the development of RL frameworks for optimal control of both CT linear and nonlinear systems without requiring a system dynamics model. In CT applications, the value function is represented as an integral of reward measurements, unlike the discrete summation in DT systems. The equivalent HJB equation in the CT domain includes the full system dynamics, making CT RL applications more challenging to solve. System measurements are sampled at fixed intervals to compute integral reinforcement signals, followed by a two-time scale asynchronous update process to sequentially adjust the weights of both the critic and actor networks.

A synchronous IRL method that updates both the critic and actor NNs simultaneously was demonstrated in [63], along with a PE condition for its convergence. Extensions of RL frameworks to handle input-constrained CT nonlinear system applications have been proposed in [39, 66, 63], with Modares et al. [43] contributing further advancements. Experience replay is used to ease the persistence of excitation (PE) conditions required for convergence. Other methods for continuous-time (CT) applications have utilized Euler discretization of the CT Bellman equations, making existing discrete-time reinforcement learning (DTRL) methods applicable [1].

In reinforcement learning (RL), tracking control problems aim to make system outputs follow desired reference trajectories. For instance, an infinite horizon linear quadratic tracking (LQT) control for discrete-time (DT) systems using an augmented RL state and reference dynamics formulation has been proposed. However, its practical application is limited due to the assumption that reference dynamics tend towards zero. An improved framework using a discounted tracking cost for DT linear systems was shown to be more practical, though it cannot guarantee zero steady-state error [63]. Extensions to DT nonlinear systems using actor-critic RL structures with neural networks have also been proposed, including applications to multiple-input multiple-output (MIMO) systems and finite horizon cases [5].

2.4 Summary

Based on the explanations above, Optimal and adaptive control are strategies designed to regulate systems in the most efficient and effective manner, often in dynamic or uncertain environments. These strategies aim to achieve desired system performance while minimizing costs or errors. Model-based control relies on a mathematical model of the system to be controlled, allowing for the design of control laws that optimize performance. Model-free control does not require an explicit model and learns from data, making it suitable for complex or unknown systems. Reinforcement learning (RL) with adaptive dynamic programming (ADP) is a powerful model-free control approach that uses iterative methods to learn optimal control policies. The value iteration and policy iteration frameworks are common RL algorithms, while the actor-critic framework combines value-based and policy-based methods for efficient learning. Given the distinct approaches of model-free and model-based methods, as well as the contrasting techniques of Monte Carlo and Temporal Difference, this research aims to conduct a comprehensive comparison of their performance in various control tasks.

Chapter 3

Methodology

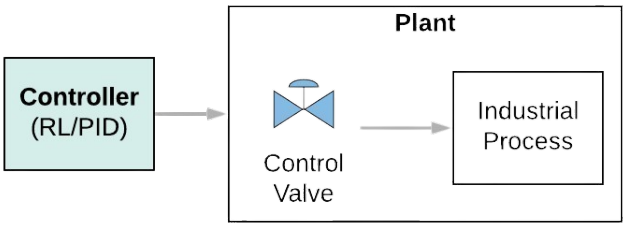

Figure 3.1: *Simulink Blocks Components [54]*

This section outlines the experimental framework, developed using MATLAB and Simulink, for the design and evaluation of Reinforcement Learning (RL) and Proportional-Integral-Derivative (PID) controllers as the benchmark. The core components of this setup are depicted in Figure 3.1 which draws heavily upon the foundational research presented in the 2021 paper, "Reinforcement Learning for Control of Valves" authored by Rajesh Siraskar.

The key components of our experimental setup include:

1. **PID Controller:** A PID controller, tuned using MATLAB's automated tuning feature.

2. **Reinforcement Learning Agent Training Setup:** A training environment for the RL agent, utilizing the Deep Deterministic Policy Gradient (DDPG), Soft-Actor Critic (SAC), and Proximal Policy Optimization (PPO) algorithm.

3. **Unified Framework for Controller Experimentation and Evaluation:** A comprehensive platform for conducting experiments and assessing the performance of both RL and PID controllers.

3.1 Modelling the System

The model representations of the single-tank system and the quadruple-tank system are illustrated in Figure 3.2 and Figure 3.6, respectively.

These two models are based on distinct mathematical principles. The single-tank system involves a constant inflow into the tank, which is followed by a constant outflow influenced by gravitational acceleration. The primary objective of the control strategy in this system is to maintain the voltage level V supplied to the input pump. This involves ensuring that the inflow and outflow rates are balanced to keep the tank's liquid level stable. The control mechanism must account for variations in the inflow rate and adjust the pump's voltage accordingly to maintain the desired liquid level.

On the other hand, the quadruple-tank system is more complex and consists of three main components: the tanks (x_1, x_2, x_3, x_4), the electrical pumps (u_1, u_2), and the control valves (γ_1, γ_2). Each tank in this system has its own dynamics, and the interaction between the tanks adds to the complexity of the control strategy. The electrical pumps are responsible for supplying the necessary flow to the tanks, while the control valves regulate the distribution of this flow among the tanks. The primary objective of the control strategy in the quadruple-tank system is to maintain the voltage level u supplied to the input pumps. This voltage must be constrained within the range $[0, \infty]$ volts to ensure safe and efficient operation. The control strategy must also consider the interactions between the tanks and adjust the pump voltages and valve positions to maintain the desired liquid levels in all four tanks.

3.1.1 Single-tank Model

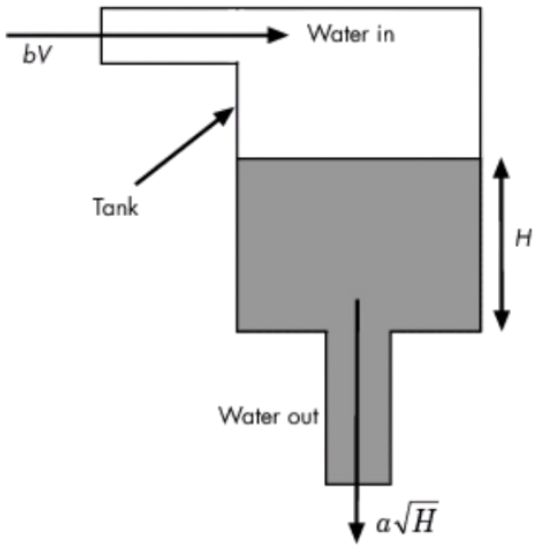

Figure 3.2: *System of measuring water level with constant outlet flow*

The control system performed in this process is the control of water level through the regulation of valve opening according to the water level measured in the tank. This is done by regulating the flow in (q_i) to the tank with the targeted water height being the desired water level/setpoint so that the system runs steadily and continuously. When the measured water level does not match the setpoint, the system will read the level transmitter and command the control valve to open the valve until the error between the setpoint and the measured water level is minimized to zero. This condition is known as reaching steady state. This process in a single-input single-output (SISO) system is also known as the water level control process.

When the flow in and flow out are equal, the water level in the tank does not change, a condition often referred to as steady state. This can be represented in the mass balance equation as

$$\frac{d(pV)}{dt} = pq_i(t) - pq_0(t) \tag{3.1}$$

The density in this equation is constant so it can be eliminated, leaving the volume variable. If expanded, the volume variable is the product of the area A and the height h of the tank.

$$\frac{d(Ah)}{dt} = q_i(t) - q_o(t) \tag{3.2}$$

Then, the area variable can be factored out, resulting in

$$A\frac{dh}{dt} = q_i(t) - q_o(t) \tag{3.3}$$

At steady state, the equation becomes

$$A\frac{dh}{dt} = q_i(t) - b \tag{3.4}$$

In this system, the outflow is not influenced by the tank height, hence q_o is not a function of time and can be replaced with a constant b.

$$A\frac{dh}{dt} = q_i(t) - b \tag{3.5}$$

At steady state,

$$A\frac{dh_s}{dt} = q_i(t) - b \tag{3.6}$$

Then, subtracting equation 3.5 from equation 3.4, we get

$$A\frac{d(h - h_s)}{dt} = q_i(t) - q_o(t) - (b - b) \tag{3.7}$$

Redefining the variables, we have

$$q(t) = q_i(t) - q_o(t) \tag{3.8}$$

Thus, we obtain a new equation with the parameters value can be seen in Table 3.1. The single-tank process, depicted in Figure 3.2, comprises single-tank which the Pump discharge water into tank 1 with the proportion of flow from the pump denoted by b and the outflow from the tank denoted as a. The primary control objective is to regulate the water levels in the tanks, denoted by H to the desired set points, h_s.

$$A\frac{dH(t)}{dt} = q(t) \tag{3.9}$$

$$A\frac{dH(t)}{dt} = bV - a\sqrt{H} \tag{3.10}$$

$$\frac{dH(t)}{dt} = \frac{bV}{A} - \frac{a\sqrt{H}}{A} \tag{3.11}$$

Parameter	b/A	a/A
Value	0.25	0.1

Table 3.1: *Single-tank process parameters*

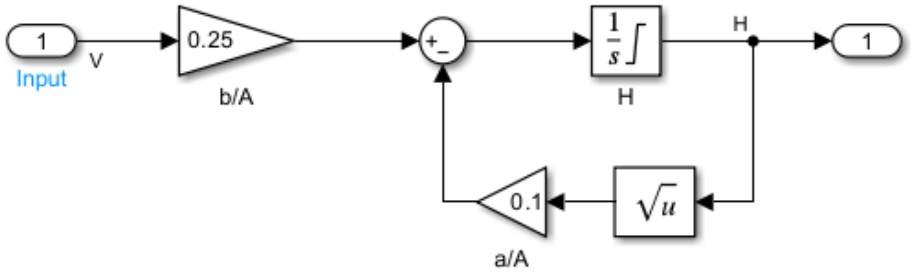

Figure 3.3: *Mathematical Modeling of Single Water Tank System*

3.1.2 Quadruple Tank Model

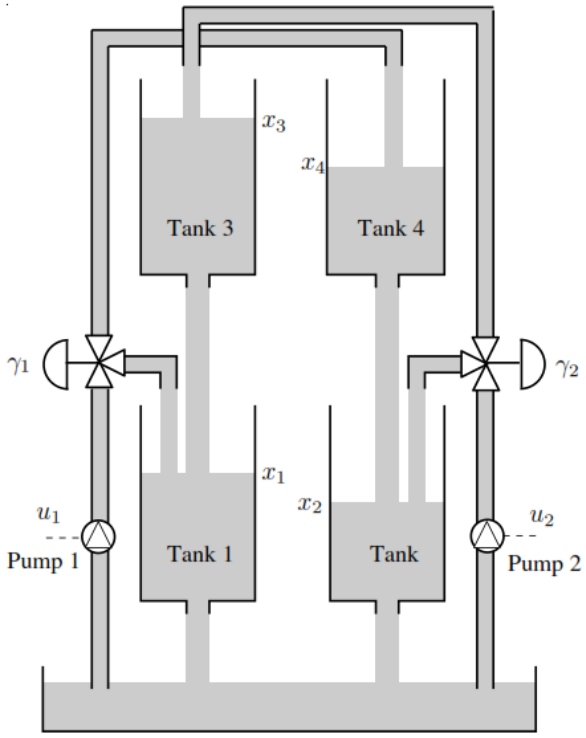

Figure 3.4: *Quadruple-Tank miniature plant*

The quadruple tank process, depicted in Figure 3.6, comprises four interconnected tanks and a water reservoir. Pump 1 discharges water into tanks 1 and 4, while Pump 2 delivers water to tanks 2 and 3. The combined flow from Pump 1 and Pump 2 is distributed between the upper and lower tanks through manually adjustable valves S_1 and S_2, with the proportion of flow from each pump denoted by γ_1 and γ_2 , respectively. The control objective is to regulate the water levels in four tanks, denoted by h_1, h_2, h_3, and h_4 to their desired set points, h_1s, h_2s, h_3s, and h_4s.

By applying mass balance and Bernoulli's principle, the following state-space equations can be derived to model the dynamics of the quadruple tank process given by [16]

$$\dot{x}_1 = -\frac{a_1}{\sqrt{2g}A_1}x_1 + \frac{a_3}{\sqrt{2g}A_1}x_3 + \frac{\gamma_1}{A_1}u_1 \tag{3.12}$$

$$\dot{x}_2 = -\frac{a_2}{\sqrt{2g}A_2}x_2 + \frac{a_4}{\sqrt{2g}A_2}x_4 + \frac{\gamma_2}{A_2}u_2 \tag{3.13}$$

$$\dot{x}_3 = -\frac{a_3}{\sqrt{2g}A_3}x_3 + \frac{(1-\gamma_2)}{A_3}u_2 \tag{3.14}$$

$$\dot{x}_4 = -\frac{a_4}{\sqrt{2g}A_4}x_4 + \frac{(1-\gamma_1)}{A_4}u_1. \tag{3.15}$$

a_i represent the cross-sectional area of the outlet hole for tank i, where $i = 1, 2, 3, 4$. A_i denote the cross-sectional area of tank i, where $i = 1, 2, 3, 4$. k_j signify the flow coefficient of Pump j, with $j = 1, 2$. This implies that $k_j v_j$ is the volumetric flow rate produced by Pump j.

The quadruple tank process is assumed as a minimum-phase system if the combined flow rate to the upper tanks is less than the combined flow rate to the lower tanks. Mathematically, this condition can be expressed as $(1 < \gamma_1 + \gamma_2 < 2)$, where γ_1 and γ_2 represent the fractions of flow from Pump 1 and Pump 2 directed to the upper tanks, respectively. Conversely, if the combined flow rate to the upper tanks is greater than or equal to the combined flow rate to the lower tanks, the process is considered non-minimum phase mentioned by [27].

The table below provides the parameters used for the quadruple-tank system which represents the parameters of cross-sectional areas A, fraction of water flow γ, and the coefficient of the pump of the tanks k which shown in Table 3.2. These values are crucial for understanding and analyzing the behavior of the system.

Parameter	$A_1(\text{cm}^2)$	$A_2(\text{cm}^2)$	$a_1(\text{cm}^2)$	$a_2(\text{cm}^2)$	k_1	k_2	γ_1	γ_2
Value	730	720	2.05	2.05	300	180	0.95	0.9

Table 3.2: *Quadruple tank process parameters*

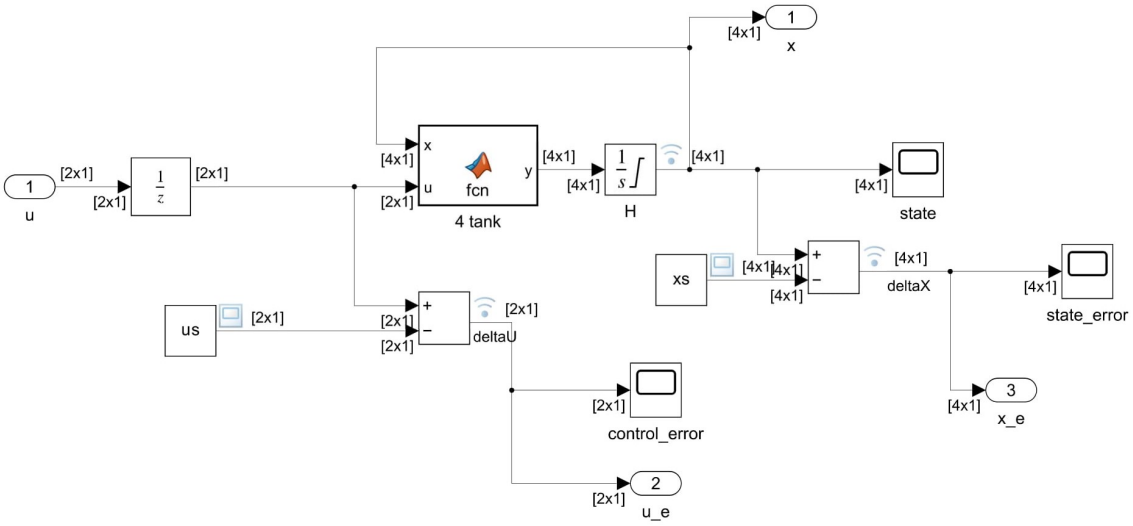

Figure 3.5: *Quadruple-Tank System Modelling in Simulink*

3.2 PID Controller Setup

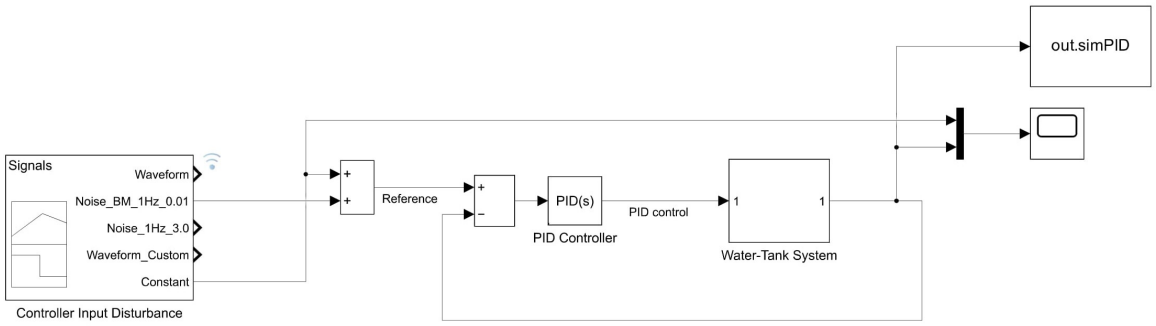

Figure 3.6: *Single-Tank controller setup*

In this research, PID would be utilized as the benchmark for the single-tank system in order to compare the stability and robustness of RL algorithms. The result would be shown in the result chapter.

PID is using a function of feedback error in time-domain

$$u(t) = K_p e + K_i \int e(t)dt + K_d \frac{de}{dt} \tag{3.16}$$

where u is the desired control signal and $e(t) = r(t) - y(t)$ is the tracking error, between the desired output r and the actual output y. This error signal is fed to the PID controller, and the controller computes both the derivative and the integral of this.

$$K_p + K_i \frac{1}{s} + K_d \frac{N}{(1 + N\frac{1}{s})} \tag{3.17}$$

The filter coefficient (N) sets the filter's pole position, which is crucial for reducing high-frequency noise amplification. It is advisable to keep (N) within the range of 2 to 20. When (N) exceeds 100, Equation 3.17 closely resembles the ideal form of Equation 3.16. MATLAB's auto-tuning feature was used to adjust the PID controller, resulting in the coefficients ($K_p = 1.82$), ($K_i = 0.24$), ($K_d = -1.72$), and ($N = 0.74$). The small value of (N) effectively minimizes the impact of the derivative term.

3.3 RL Controller Setup

Figure 3.7 and Figure 3.6 provides a comprehensive overview of the setup utilized for both the training and evaluation phases of the Reinforcement Learning (RL) controller within the Simulink environment. Training an RL agent is a complex process that involves extensive tuning of various hyper parameters to achieve optimal performance. These hyperparameters can include learning rates, discount factors, and exploration strategies.

3.3.1 RL Controller Design

When designing an environment for training an agent to follow control signal trajectories, various factors must be considered. These factors can be divided into those related to the agent and those related to the environment. Agent-related factors include the structure of the observation vector and the reward strategy, which are essential for providing the agent with the necessary information and evaluating its actions. Environment-related factors involve the training strategy, the nature of the training signals, the initial conditions of the environment, and the criteria for ending an episode.

The training strategy includes the methods and algorithms used to train the agent, ensuring efficient learning. Training signals are the inputs given to the agent during training, which can vary in complexity. Initial conditions set the starting state for each training episode, affecting the learning process. Finally, the criteria for ending an episode define when a training session concludes, based on time, performance, or other conditions which shown in stop simulation block in Figure 3.7 and the execution of training options indicated by the Maximum episode at 5000 episode and stopping the training process when the average reward reach about 4800 point.

```
trainOpts = rlTrainingOptions(...
    MaxEpisodes=5000, ...
    MaxStepsPerEpisode=ceil(Tf/Ts), ...
    ScoreAveragingWindowLength=20, ...
    Verbose=false, ...
    Plots="training-progress",...
    StopTrainingCriteria="AverageReward",...
    SaveAgentCriteria="EpisodeCount",...
    SaveAgentValue = 100,...
    StopTrainingValue = 4800);
```

By carefully considering and optimizing above factors, a robust environment can be created to facilitate the efficient training of agents to follow control signal trajectories.

Single-Tank System

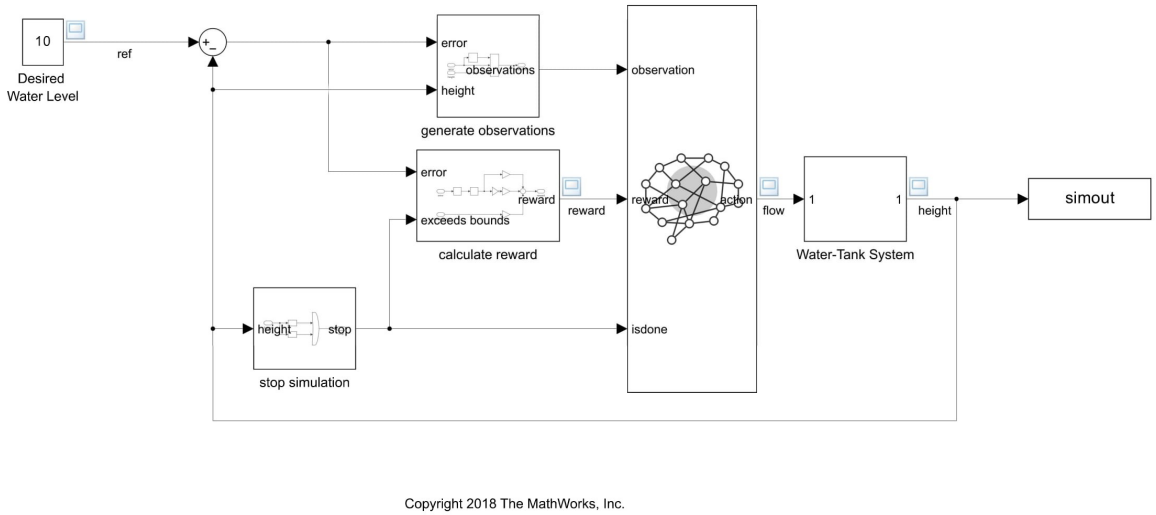

Figure 3.7: *Single-Tank RL Controller*

Figure 3.7 illustrates the Single-tank Agent block, which receives feedback from the environment through the Observations vector. It also includes the block responsible for calculating Rewards and the Stop-simulation block that manages the end of an episode.

Quadruple-Tank System

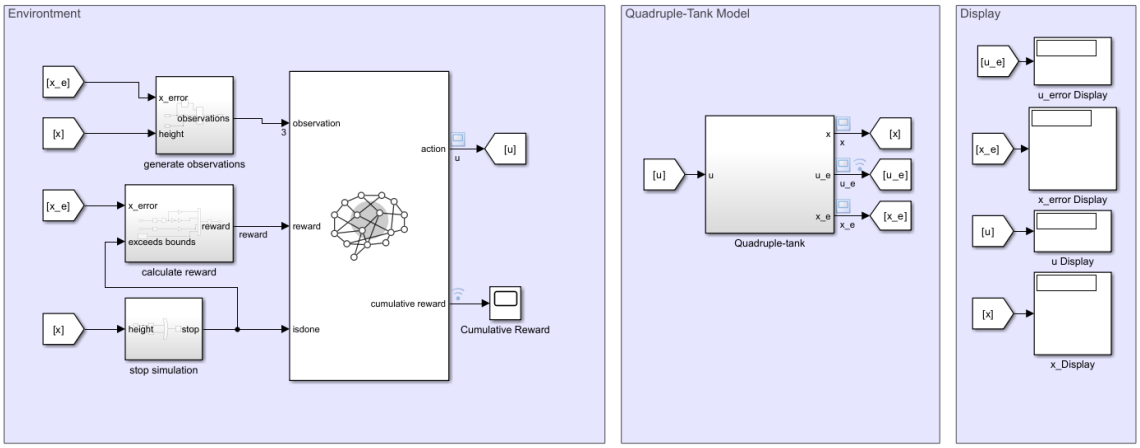

Figure 3.8: *Quadruple-Tank RL Controller*

Figure 3.8 illustrates the Quadruple-tank Agent block, which provided consists of two main blocks: the Environment and the Quadratic Tank Model. The Environment block includes components for defining Observations vector, calculating rewards, and stopping the simulation based on certain conditions. The Quadratic Tank Model represents the

system being controlled, featuring various display blocks that show real-time data such as the height of the liquid in the tank and control signals.

3.3.2 Training Strategy

Rather than training the RL agent to adhere strictly to the benchmark trajectory shown, which is quite limiting, the agent was instead trained to follow various levels of constant signals. Additionally, the agent was tasked with starting from a randomly set flow value. This approach creates an effective training strategy, enabling the agent to follow any control signal trajectory made up of straight lines. The RL Toolbox supports this by allowing customization of the default reset function to implement the strategy.

The reset function alters the reference signal and the starting water level randomly, adjusting the relevant block parameters accordingly.

Single-Tank System

The reset function alters the reference signal and the starting water level randomly, adjusting the relevant block parameters accordingly.

```
function in = localResetFcn(in)
% Randomize reference signal
blk = sprintf("rlwatertank/Desired \nWater Level");
h = 3*randn + randi([1 10]);
while h <= 0 || h >= 20
    h = 3*randn + randi([1 10]);
end
in = setBlockParameter(in,blk,Value=num2str(h));

% Randomize initial height
h = 3*randn + randi([1 10]);
while h <= 0 || h >= 20
    h = 3*randn + randi([1 10]);
end
blk = "rlwatertank/Water-Tank System/H";
in = setBlockParameter(in,blk,InitialCondition=num2str(h));

end
```

In this code, the reference signal would vary randomly between a constraint $0 - 20dm^3$ for each episodical training and also randomize the starting point of initial height of the tank.

Quadruple-Tank System

The reset function randomly changes the reference signal and initial condition of water level, updating the corresponding block parameters accordingly.

```
function in = localResetFcn(in)
% % randomize reference signal
in = setVariable(in,'x0',[25;16;20;21]);
end
```

3.3.3 Observation Vector

The observation vector, depicted in Figure 17, includes the actual flow achieved (y), the error relative to the reference (e), and the integral of the error: $[y; e; \int e.dtx^2\, dx]$. The integral of the error, representing the cumulative error over time, is crucial for training RL controllers as it helps reduce the total error, unlike instantaneous error which lacks memory. The code definition can be seen below:

```
% Observation info
obsInfo = rlNumericSpec([3 1],...
    LowerLimit=[-inf -inf 0  ]',...
    UpperLimit=[ inf  inf inf]');

% Name and description are optional and not used by the
   software
obsInfo.Name = "observations";
obsInfo.Description = "integrated error, error, and measured
   height";
```

3.3.4 Reward Strategy

The reward strategy is crucial as it guides the agent's learning process. The reward function provides feedback to the agent about the success of its actions in achieving the desired outcome.

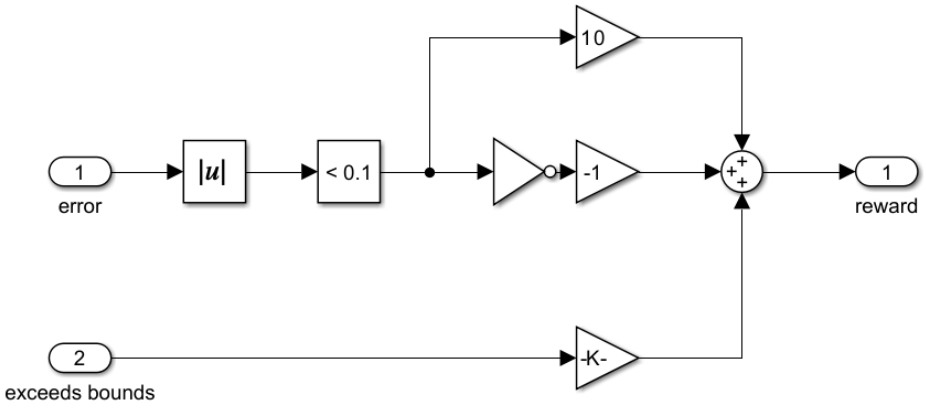

Figure 3.9: *Reward Strategy for Single-Tank System*

Equation assigns a reward of -100 if the absolute error is within a threshold, otherwise, it assigns -1. The threshold is set at 100, applicable when the flow variable is below its maximum limit.

$$\text{Reward} = \begin{cases} -100 & \text{if } |e| \leq \Delta \\ -1 & \text{otherwise} \end{cases} \tag{3.18}$$

where $\Delta = 100$, if $y(s) < y_{\text{Max.Flow}}$.

Figure 3.9 illustrates the practical application of these rules, showing how they help reduce errors while assigning discrete rewards within flow limits. If the discrepancy between the reference signal and the current height falls below 0.1, a reward of 10 points per step will be awarded.

$$\text{reward} = 10(|e| < 0.1) - 1(|e| > 0.1) - 100(|h| > 20) \tag{3.19}$$

3.3.5 Hyperparameter

Hyperparameters in reinforcement learning are adjustable parameters that govern the learning process, rather than the parameters learned by the agent. They are external to the agent and are specified prior to training process. These parameters influence factors such as the rate of learning, the balance between exploration and exploitation, and the neural network's architecture. Optimizing hyperparameters is essential for enhancing agent performance, ensuring stability, and improve generalization.

Hyperparameter	PPO	DDPG	SAC
Network Architecture	[64, 64]	[256, 256]	[256, 256]
Activation	ReLU	ReLU	ReLU
Optimizer	Adam	Adam	Adam
Learning Rate	0.0003	0.001	0.0003
Target Update Rate	2048 Steps	1 Episode	1 Episode
Batch Size	64	100	256
Epochs	10	-	-
Discount Factor (γ)	0.99	0.99	0.99
Replay Buffer Size	-	10^6	10^6
Clip Range (ϵ)	0.2	-	-
GAE (λ)	0.95	-	-
Soft Update Coefficient (τ)	-	0.005	0.005
Target Entropy (α)	-	-	Auto
Action Noise	-	$\mathcal{N}(0, 0.1)$	-
Policy Delay	-	2	-

3.4 Summary

The methodology seeks to develop a reinforcement learning (RL) control architecture with two distinct levels of modeling complexity: a single-tank system and a quadruple-tank system. For each model, three different algorithms are employed to identify the optimal control configuration. The unique backbone and mathematical framework of each algorithm result in distinct processes for achieving the optimal policy. These processes vary in several aspects, including the duration of training, the stability of control, the performance in continuous cases, and the balance between exploration and exploitation. The single-tank system serves as a simpler model to test the basic principles of RL control, while the quadruple-tank system introduces additional complexity and challenges, providing a more rigorous test of the algorithms' capabilities. By comparing the outcomes across these two levels of complexity, the study aims to gain insights into the scalability and robustness of the RL control architecture. Furthermore, the differences in algorithmic approaches highlight the importance of selecting appropriate methods based on the specific requirements and constraints of the control system being modeled.

Chapter 4

Results

This chapter provides a comprehensive analysis of control systems, starting with the single tank system where PID (Proportional-Integral-Derivative) and DDPG (Deep Deterministic Policy Gradient) are benchmarked. It then compares the performance of SAC (Soft Actor-Critic), DDPG, and PPO (Proximal Policy Optimization) in this context, highlighting SAC's, DDPG's, and PPO's control performance. The discussion then shifts to the more complex quadruple-tank system, where the same algorithms are evaluated. In this chapter, the concept of episodical traininig to improve the RL performance will also be provided.

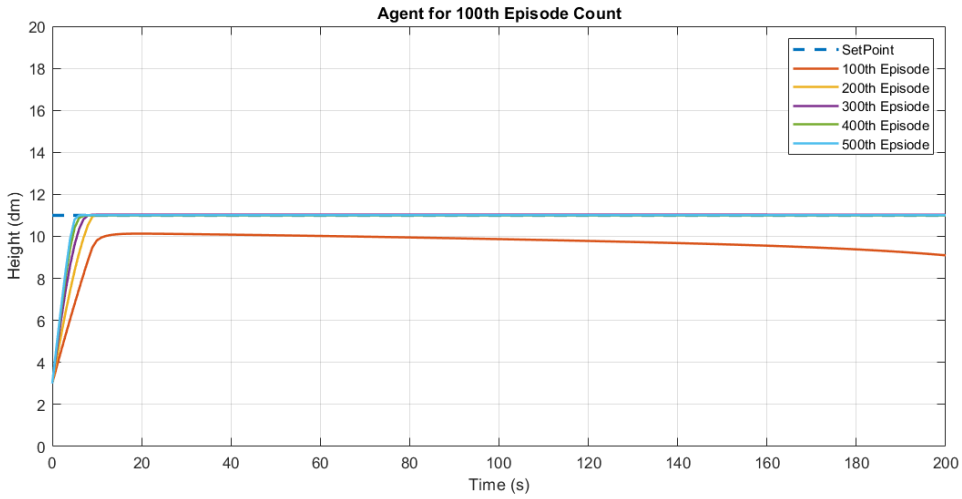

Figure 4.1: *Agent Performance every* 100^{th} *episode*

In Figure 4.1, the performance of Adaptive Dynamic Programming (ADP) in establishing control law over an extended period is depicted. The figure illustrates the progressive enhancement of control capabilities from the 100th episode to the 500th episode. Each line in the graph represents a distinct phase of improvement, highlighting the incremental advancements in both stability and robustness. This progression underscores the efficacy of ADP in refining control mechanisms, as evidenced by the continuous and measurable

improvements observed throughout the episodes.

4.1 Single-tank System

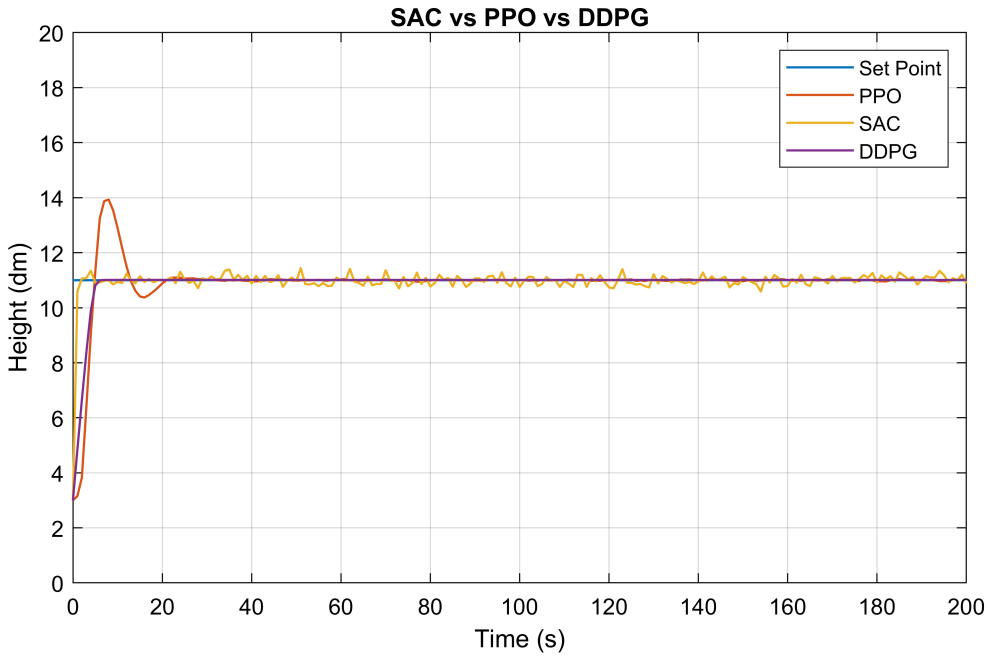

Figure 4.2: *Algorithm Comparison for Single-Tank System between SAC, PPO, and DDPG*

For each controller, we adjusted the training times, starting at 08:00:00 AM, to evaluate the training duration for each algorithm. The Soft Actor-Critic (SAC) algorithm achieved the target average reward the fastest, taking only 19 minutes and 29 seconds (08:00:00 - 08:19:29 AM). The Deep Deterministic Policy Gradient (DDPG) algorithm followed, requiring 51 minutes and 24 seconds (08:00:00 - 08:51:24 AM). Lastly, the Proximal Policy Optimization (PPO) algorithm took approximately 1 hour, 46 minutes, and 2 seconds (08:00:00 - 09:46:02 AM), nearly reaching the maximum training episode duration. The process of the agent training session can be viewed in the Figure A.1 A.2 A.3.

In this study, as illustrated in Figure 4.2, we evaluate the performance of various Reinforcement Learning (RL) controllers. The Deep Deterministic Policy Gradient (DDPG) algorithm serves as the benchmark controller, as depicted by the purple line in the figure.

The DDPG algorithm, which is based on a deterministic policy, demonstrates a more stable response compared to other algorithms. This stability is a key advantage of DDPG, making it a reliable choice for deterministic tasks.

Conversely, the Soft Actor-Critic (SAC) algorithm exhibits the most robust response when applied to the single-tank model, although it shows slightly less stability than DDPG.

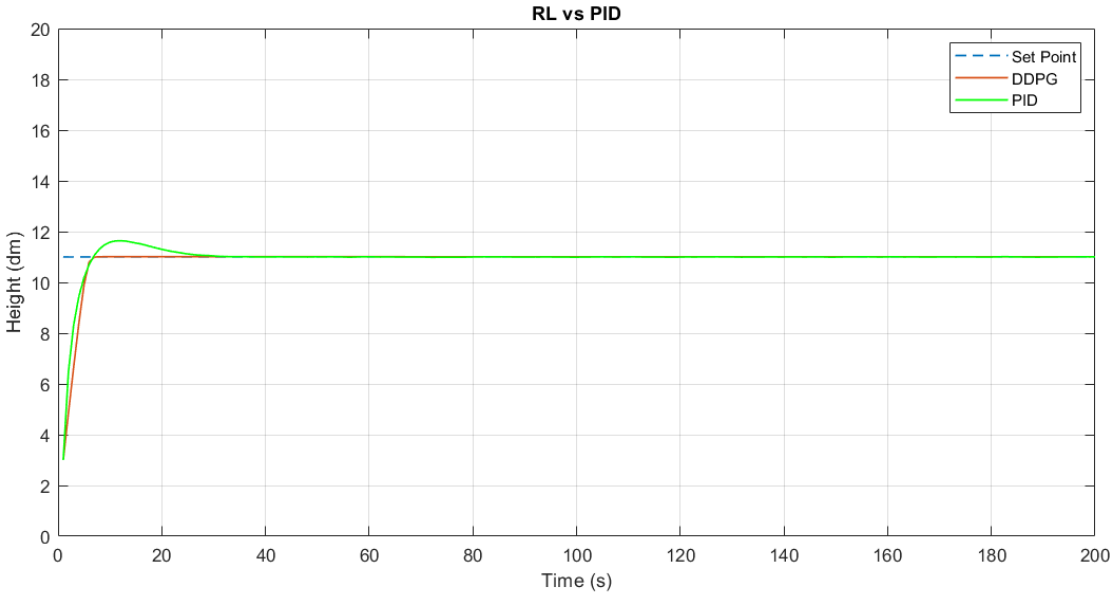

Figure 4.3: *Algorithm Comparison for Single-Tank System between RL and PID*

The robustness of SAC highlights its effectiveness in handling variability and disturbances within the system.

The Proximal Policy Optimization (PPO) algorithm, while generally effective, displays a slight overshoot that is larger compared to the other algorithms. This overshoot indicates a tendency for PPO to exceed the desired control parameters momentarily before stabilizing.

Since the PID is the common controller which used broadly, this research also use PID as the benchmark of the comparsion studies. Using the autotuning feature of Matlab, PID controller is established for the single-tank system using the framework in research of Rajesh in 2021 [54]. Based on the result of the simulation, DDPG has better response indicated by the period to reach settling time and reach the minimum value of overshoot.

The detailed numerical results of these evaluations are presented in Table 4.1, providing a comprehensive comparison of the performance metrics for each algorithm and underscores the trade-offs between stability and robustness among the different RL controllers.

	DDPG	**SAC**	**PPO**	**PID**
Rise Time	4.8 s	3.6 s	4.3 s	3.6 s
Settling Time	6.4 s	20.7 s	19.4 s	20.72 s
Overshoot	0.021	5.7	26.4	5.7
Undershoot	0	0	0	0
Peak	10.03	11.6	13.9	11.64
Peak Time	11 s	11 s	8 s	11 s

Table 4.1: *Performance Comparison between SAC, PPO, DDPG, and PID*

4.2 Quadruple-tank system

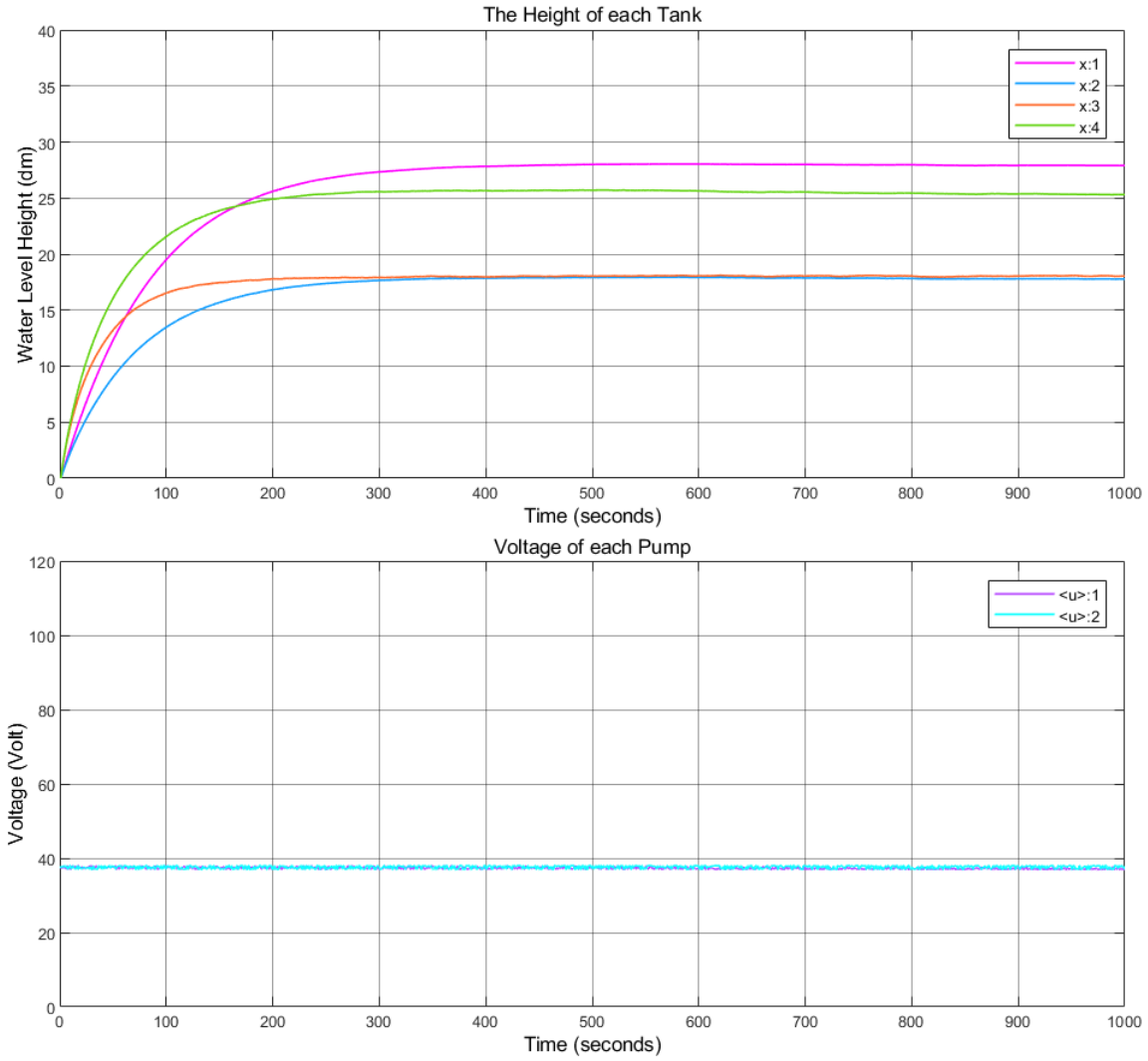

Figure 4.4: *SAC for Quadruple-tank system*

In the case with higher complexity, which could be defined as Multi Input and Multi output (MIMO) to control four different tank in Quadruple-Tank System, to assess the effectiveness of the proposed control approach for level control of a quadruple tank process, we conducted numerical simulations. The parameters of the process are detailed in Table 3.2. As shown in Table 1, the system is minimum phase since $\gamma_1 + \gamma_2 < 2$. The desired steady-state water tank levels were set to $h_{d1} = 25$, $h_{d2} = 16$, $h_{d3} = 20$, and $h_{d2} = 21$ respectively. Initial conditions for the water tank levels were $h_1(0) = h_2(0) = h_3(0) = h_4(0) = 0$.

The Soft Actor-Critic (SAC) agent demonstrated exceptional performance in controlling the water tank levels, as illustrated in Figure 4.4. The control process commenced at 8:00 AM and concluded at 1:41 PM. The corresponding control signals, which reflect the agent's

actions to maintain desired water levels, are visualized in Figure 4.4 shown in Figure A.4 and Figure A.5 also need to comprehend

A comprehensive analysis of Figure 4.4 and the performance metrics presented in Table 4.2 reveals the remarkable stability achieved by the SAC agent. Notably, the settling time, a critical measure of control system performance, was a mere 20.7 seconds. This rapid convergence to the desired steady-state underscores the agent's exceptional ability to effectively regulate the water tank levels, thereby validating the theory that SAC is a promising choice for handling continuous action spaces [54].

	h_1	h_2	h_3	h_4
Rise Time	3.6 s	3.6 s	3.6 s	3.6 s
Settling Time	20.7 s	20.7 s	20.7 s	20.7 s
Overshoot	5.7	5	6.3	6.9
Undershoot	0	0	0	0
Peak	11.6	11.6	11.6	11.6
Peak Time	11 s	11 s	11 s	11 s

Table 4.2: *Performance Comparison of each Tank for SAC Agent*

As depicted in Figure A.5, PPO algorithms often exhibit a tendency to diverge over time. This phenomenon can be attributed to several factors, as explored by Timothy et al. in their 2015 study [62]. The research findings suggest that PPO's strengths lie primarily in handling discrete action spaces with simple model architecture such as SISO. This preference can be attributed to the prevalence of PPO implementations in domains involving discrete game manipulation, such as Atari and board games [44]. The discrete nature of these environments aligns well with PPO's underlying mechanisms, facilitating effective learning and control

4.3 Summary

By conducting a comprehensive comparison of Single-Tank and Quadruple Tank systems against traditional PID controllers, we can obtain valuable insights into the strengths and limitations of each approach. This comparative analysis will enable us to identify the most suitable control strategy for various process dynamics and operational requirements.

1. Single-Tank System: In the context of a single-tank system, it has been demonstrated that the Soft Actor-Critic (SAC) algorithm achieves superior performance in terms of settling time, overshoot, and rise time. This is attributed to SAC's utilization of double Q-learning and entropy weighting, which significantly accelerate the learning process. In contrast, the Proximal Policy Optimization (PPO) algorithm relies solely on the value function, resulting in a slower training process and occasional underfitting. Overall, SAC, PPO, and Deep Deterministic Policy Gradient (DDPG) can

surpass traditional Single-Input Single-Output (SISO) modeling alike PID controller, with the primary difference being the duration of the training period.

2. Quadruple-Tank System: When examining more complex systems, such as the quadruple tank system, the comparative study between SAC and PPO reveals a notable disparity. Previous research has established control systems using PPO, concluding that PPO is more suitable for SISO and discrete action spaces. While PPO has demonstrated some capability in continuous action spaces, it often requires further reparameterization to perform effectively.

Chapter 5

Conclusions and Future Work

This research introduces an approach to comparing the performance of various control strategies, including traditional PID controllers and three reinforcement learning (RL) algorithms, across Single-Tank and Quadruple-Tank systems of varying complexity. Previous studies have explored the application of single RL agents or hybrid systems in complex domains such as propulsion and power systems, exemplified by gas turbine engines [30]. A comprehensive analysis of the fundamental differences between these agents has not been conducted.

The key contributions of this report are as follows:

- **Development of a comparative analytical framework**: This research introduces a comprehensive framework capable of evaluating the performance of various control strategies across different system complexities and action spaces, including discrete and continuous domains.
- **Application of RL controllers to Quadruple-Tank systems**: This study pioneers the application of reinforcement learning controllers to Quadruple-Tank systems, addressing a significant gap in existing research.

The current design of the reinforcement learning (RL) controller requires enhancements to address oscillations induced by high-frequency disturbances with significant amplitudes. One potential solution is to implement a low-pass filter, which can effectively reduce high-variance noise at both the input and output stages of the controller. However, this approach might slow down the system's response time. Therefore, further research is essential to develop objective and reward functions that can inherently prevent noisy RL trajectory behavior. If successful, this method would be more advantageous than using a filter, as it would maintain the system's responsiveness.

Additionally, the increasing popularity of non-linear model predictive control (NMPC) warrants a thorough evaluation. NMPC has been successfully applied in various fields, including industrial measurement and aerial or flight measurement. This control method is

renowned for its robustness and stability, particularly in scenarios involving infinite horizons or unlimited operational time in machinery. Evaluating NMPC's performance in these contexts could provide valuable insights into its potential advantages over traditional control methods.

Bibliography

[1] M. Abu-Khalaf and F. L. Lewis. Nearly optimal control laws for nonlinear systems with saturating actuators using a neural network hjb approach. *Automatica (Oxford)*, 41:779–791, 2005. ISSN 0005-1098. doi: 10.1016/j.automatica.2004.11.034.

[2] H.-S. Ahn, Y. Chen, and K. L. Moore. Iterative learning control: Brief survey and categorization. *IEEE transactions on systems, man and cybernetics. Part C, Applications and reviews*, 37:1099–1121, 2007. ISSN 1094-6977. doi: 10.1109/TSMCC.2007.905759. ObjectType-Article-2 SourceType-Scholarly Journals-1 ObjectType-Feature-1 content type line 23.

[3] B. D. O. Anderson and A. Dehghani. Challenges of adaptive control–past, permanent and future. *Annual reviews in control*, 32:123–135, 2008. ISSN 1367-5788. doi: 10.1016/j.arcontrol.2008.06.001.

[4] M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning dexterous in-hand manipulation, 2019. URL `https://arxiv.org/abs/1808.00177`.

[5] L. C. Baird. Reinforcement learning in continuous time: advantage updating. In *ICNN*, volume 4, pages 2448–2453 vol.4. IEEE, 1994. ISBN 078031901X. doi: 10.1109/ICNN.1994.374604.

[6] R. Bellman. A markovian decision process. *Journal of mathematics and mechanics*, 6: 679–684, 1957. ISSN 0095-9057. doi: 10.1512/iumj.1957.6.56038.

[7] M. Benosman. Model-based vs data-driven adaptive control: An overview. *International journal of adaptive control and signal processing*, 32:753–776, 2018. ISSN 0890-6327. doi: 10.1002/acs.2862.

[8] D. P. Bertsekas. *Dynamic programming and optimal control*. Belmont, 3rd edition. edition, 2005. ISBN 1886529264.

[9] S. Bhasin. Reinforcement learning and optimal control methods for uncertain nonlinear systems. 2011.

[10] M. Biemann, F. Scheller, X. Liu, and L. Huang. Experimental evaluation of model-free reinforcement learning algorithms for continuous hvac control. *Applied Energy*, 298:117164, 2021. ISSN 0306-2619. doi: https://doi.org/10.1016/j.apenergy.2021.117164. URL `https://www.sciencedirect.com/science/article/pii/S0306261921005961`.

[11] I. Carlucho, M. D. Paula, and G. G. Acosta. An adaptive deep reinforcement learning approach for mimo pid control of mobile robots. *ISA transactions*, 102:280–294, 2020. ISSN 0019-0578. doi: 10.1016/j.isatra.2020.02.017. ObjectType-Article-1 SourceType-Scholarly Journals-1 ObjectType-Feature-2 content type line 23.

[12] David. Deterministic policy gradient algorithms. Technical report, Deepmind, 2013.

[13] L. Desborough and R. Miller. Increasing customer value of industrial control performance monitoring—honeywell's experience. Technical report, Honeywell Hi-Spec Solutions, 2002.

[14] Z. Dong, X. Huang, Y. Dong, and Z. Zhang. Multilayer perception based reinforcement learning supervisory control of energy systems with application to a nuclear steam supply system. *Applied energy*, 259:114193, 2020. ISSN 0306-2619. doi: 10.1016/j.apenergy.2019.114193.

[15] E. Eker, M. Kayri, S. Ekinci, and D. Izci. A new fusion of aso with sa algorithm and its applications to mlp training and dc motor speed control. *Arabian Journal for Science and Engineering*, 46:3889–3911, 2021. ISSN 2191-4281. doi: 10.1007/s13369-020-05228-5. URL `https://doi.org/10.1007/s13369-020-05228-5`.

[16] M. Elham and T. Mohammad. Control of quadruple tank process using an adaptive fractional-order sliding mode controller. *Journal of Control, Automation and Electrical Systems*, 32(1):605–614, June 2021.

[17] S. Ferrari and R. F. Stengel. An adaptive critic global controller. In *ACC*, volume 4, pages 2665–2670 vol.4. Piscataway NJ: IEEE, 2002. ISBN 0743-1619. doi: 10.1109/ACC.2002.1025189. ObjectType-Conference-1 SourceType-Conference Papers Proceedings-1 content type line 25.

[18] K. G. V. Frank L. Lewis, Draguna Vrabie. Reinforcement learning and feedback control: Using natural decision methods to design optimal adaptive controllers. *IEEE Control Systems*, 8:76–105, 2012.

[19] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[20] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018. URL `http://arxiv.org/abs/1812.05905`.

[21] R. Hafner and M. Riedmiller. Reinforcement learning in feedback control. *Machine learning*, 84:137, 2011. ISSN 0885-6125. doi: 10.1007/s10994-011-5235-x.

[22] Z. Han and K. S. Narendra. New concepts in adaptive control using multiple models. *IEEE transactions on automatic control*, 57:78–89, 2012. ISSN 0018-9286. doi: 10.1109/TAC.2011.2152470. ObjectType-Article-2 SourceType-Scholarly Journals-1 ObjectType-Feature-1 content type line 23.

[23] Z.-S. Hou and Z. Wang. From model-based control to data-driven control: Survey, classification and perspective. *Information sciences*, 235:3–35, 2013. ISSN 0020-0255. doi: 10.1016/j.ins.2012.07.014.

[24] R. A. Howard. *Dynamic programming and Markov processes.* Cambridge, Mass. : M.I.T. Press, c1960, 1960.

[25] S. S. M. James W. Mock. Sim-to-real: A performance comparison of ppo, td3, and sac reinforcement learning algorithms for quadruped walking gait generation. *Journal of Intelligent Learning Systems and Applications*, 16(2):23–43, May 2024. doi: 10.4236/jilsa.2024.162003.

[26] W. Jian and C. Wenjian. Development of an adaptive neuro-fuzzy method for supply air pressure control in hvac system. In *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics. 'cybernetics evolving to systems, humans, organizations, and their complex interactions' (cat. no.0)*, volume 5, pages 3806–3809 vol.5, 2000. doi: 10.1109/ICSMC.2000.886603.

[27] K. H. Johansson. The quadruple-tank process: A multivariable laboratory process with an adjustable zero. *IEEE Transactions On Control Systems Technology*, 8(3):456–464, May 2000.

[28] J. Jäschke, Y. Cao, and V. Kariwala. Self-optimizing control – a survey. *Annual reviews in control*, 43:199–223, 2017. ISSN 1367-5788. doi: 10.1016/j.arcontrol.2017.03.001.

[29] H. Kaufman, I. Bar-Kana, and K. Sobel. *Direct adaptive control algorithms : theory and applications.* New York : Springer, c1998, 2nd ed. edition, 1998. ISBN 0387948848. Includes bibliographical references (p. [409]-419) and index.

[30] H. Kazmi, F. Mehmood, S. Lodeweyckx, and J. Driesen. Gigawatt-hour scale savings on a budget of zero: Deep reinforcement learning based optimal control of hot water systems. *Energy (Oxford)*, 144:159–168, 2018. ISSN 0360-5442. doi: 10.1016/j.energy.2017.12.019.

[31] S. G. Khan, G. Herrmann, F. L. Lewis, T. Pipe, and C. Melhuish. Corrigendum to 'reinforcement learning and optimal adaptive control: An overview and implementation examples' [annual reviews in control 36 (1) (2012) 42–59]. *Annual reviews in control*, 36:359, 2012. ISSN 1367-5788. doi: 10.1016/j.arcontrol.2012.07.001.

[32] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. A. Sallab, S. Yogamani, and P. Perez. Deep reinforcement learning for autonomous driving: A survey. *IEEE transactions on intelligent transportation systems*, 23:4909–4926, 2022. ISSN 1524-9050. doi: 10.1109/TITS.2021.3054625.

[33] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, and F. L. Lewis. Optimal and autonomous control using reinforcement learning: A survey. *IEEE transaction on neural networks and learning systems*, 29:2042–2062, 2018. ISSN 2162-237X. doi: 10.1109/TNNLS.2017.2773458. ObjectType-Article-1 SourceType-Scholarly Journals-1 ObjectType-Feature-2 content type line 23.

[34] B. Kiumarsi-Khomartash, F. L. Lewis, M.-B. Naghibi-Sistani, and A. Karimpour. Optimal tracking control for linear discrete-time systems using reinforcement learning. In *CDC*, pages 3845–3850. IEEE, 2013. ISBN 0191-2216. doi: 10.1109/CDC.2013.6760476.

[35] T. Landelius. Reinforcement learning and distributed local model synthesis. 1997.

[36] F. L. Lewis, D. Vrabie, and V. L. Syrmos. *Optimal Control*. John Wiley Sons, Incorporated, 2012. ISBN 9781118122709. URL `http://ebookcentral.proquest.com/lib/sheffield/detail.action?docID=817332`.

[37] L. Ljung. Perspectives on system identification. *Annual reviews in control*, 34:1–12, 2010. ISSN 1367-5788. doi: 10.1016/j.arcontrol.2009.12.001.

[38] Y. Lv, J. Na, Q. Yang, X. Wu, and Y. Guo. Online adaptive optimal control for continuous-time nonlinear systems with completely unknown dynamics. *International journal of control*, 89:99–112, 2016. ISSN 0020-7179. doi: 10.1080/00207179.2015.1060362. ObjectType-Article-1 SourceType-Scholarly Journals-1 ObjectType-Feature-2 content type line 23.

[39] Y. Lv, J. Na, Q. Yang, X. Wu, and Y. Guo. Online adaptive optimal control for continuous-time nonlinear systems with completely unknown dynamics. *International journal of control*, 89:99–112, 2016. ISSN 0020-7179. doi: 10.1080/00207179.2015.1060362. ObjectType-Article-1 SourceType-Scholarly Journals-1 ObjectType-Feature-2 content type line 23.

[40] G. M. Malwatkar, S. H. Sonawane, and L. M. Waghmare. Tuning pid controllers for higher-order oscillatory systems with improved performance. *ISA Transactions*, 48:347–353, 2009. ISSN

0019-0578. doi: https://doi.org/10.1016/j.isatra.2009.04.005. URL https://www.sciencedirect.com/science/article/pii/S0019057809000305.

[41] MathWorks. *Reinforcement Learning Toolbox™ User's Guide*. MathWorks. Inc, Natick, Massachusetts, 2019.

[42] W. T. Miller, R. S. Sutton, and P. J. Werbos. *A Menu of Designs for Reinforcement Learning Over Time*, pages 67–95. 1995.

[43] H. Modares, F. L. Lewis, and M.-B. Naghibi-Sistani. Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems. *Automatica (Oxford)*, 50:193–202, 2014. ISSN 0005-1098. doi: 10.1016/j.automatica.2013.09.043. ObjectType-Article-2 SourceType-Scholarly Journals-1 ObjectType-Feature-1 content type line 23.

[44] R. Nian, J. Liu, and B. Huang. A review on reinforcement learning: Introduction and applications in industrial process control. *Computers chemical engineering*, 139: 106886, 2020. ISSN 0098-1354. doi: 10.1016/j.compchemeng.2020.106886.

[45] M. Olivecrona, T. Blaschke, O. Engkvist, and H. Chen. Molecular de novo design through deep reinforcement learning, 2017. URL https://arxiv.org/abs/1704.07555.

[46] P. Palanisamy. Multi-agent connected autonomous driving using deep reinforcement learning. In *IJCNN*, pages 1–7. IEEE, 2020. doi: 10.1109/IJCNN48605.2020.9207663.

[47] W. B. Powell. *Approximate dynamic programming : solving the curses of dimensionality*. Hoboken, N.J. : J. Wiley Sons, c2011, 2nd ed. edition, 2011. ISBN 1-283-27370-5. Includes bibliographical references and index.

[48] D. V. Prokhorov and D. C. Wunsch. Adaptive critic designs. *IEEE transactions on neural networks*, 8:997–1007, 1997. ISSN 1045-9227. doi: 10.1109/72.623201. ObjectType-Article-2 SourceType-Scholarly Journals-1 ObjectType-Feature-1 content type line 23 ObjectType-Article-1 ObjectType-Feature-2.

[49] W. J. Rugh and J. S. Shamma. Research on gain scheduling. *Automatica (Oxford)*, 36:1401–1425, 2000. ISSN 0005-1098. doi: 10.1016/S0005-1098(00)00058-3.

[50] A. Saenz-Aguirre, E. Zulueta, U. Fernandez-Gamiz, J. Lozano, and J. M. Lopez-Guede. Artificial neural network based reinforcement learning for wind turbine yaw control. *Energies (Basel)*, 12:436, 2019. ISSN 1996-1073. doi: 10.3390/en12030436.

[51] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015. ISSN 0893-6080. doi: 10.1016/j.neunet.2014.09.003. ObjectType-Article-2 SourceType-Scholarly Journals-1 ObjectType-Feature-3 content type line 23 ObjectType-Review-1.

[52] K. Shingate, K. Jagdale, and Y. Dias. Adaptive traffic control system using reinforcement learning. *International Journal of Engineering Research and*, V9, 2 2020. doi: 10.17577/IJERTV9IS020159.

[53] D. T. W. D. R. M. Silver D., Heess N. Deterministic policy gradient algorithms. *ICLR 2016*, 8(3):456–464, September 2015.

[54] R. Siraskar. Reinforcement learning for control of valves. *ScienceDirect*, 4:30, 2021. doi: https://doi.org/10.1016/j.mlwa.2021.100030.

[55] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, Massachusetts, 2015.

[56] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In S. Solla, T. Leen, and K. Müller, editors, *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999. URL `https://proceedings.neurips.cc/paper_files/paper/1999/file/464d828b85b0bed98e80ade0a5c43b0f-Paper.pdf`.

[57] L. F. Tack and M. L. K. M. I. Solihin. Tuning of pid controller using particle swarm optimization (pso). In *Proceeding of the International Conference on Advanced Science, Engineering and Information Technology 2011*. ISC 2011, 2011.

[58] Y. Takahashi. Discussion: "theoretical consideration of retarded control" (cohen, g. h., and coon, g. a., 1953, trans. asme, 75, pp. 827–834). *Transactions of the American Society of Mechanical Engineers*, 75:834, 7 2022. ISSN 0097-6822. doi: 10.1115/1.4015452. URL `https://doi.org/10.1115/1.4015452`.

[59] G. Tao. Multivariable adaptive control: A survey. *Automatica (Oxford)*, 50:2737–2764, 2014. ISSN 0005-1098. doi: 10.1016/j.automatica.2014.10.015.

[60] S. ten Hagen and B. J. A. Kröse. Linear quadratic regulation using reinforcement learning. 1998. URL `https://api.semanticscholar.org/CorpusID:18820549`.

[61] R. S. S. Thomas Degris, Martha White. Off-policy actor-critic. Technical report, INRIA, 2012.

[62] A. P. N. H. T. E. Y. T. D. S. . D. W. Timothy P. Lillicrap, Jonathan J. Hunt. Continuous control with deep reinforcement learning. *ICLR 2016*, 8(3):456–464, September 2015.

[63] K. G. Vamvoudakis, D. Vrabie, and F. L. Lewis. Online adaptive algorithm for optimal control with integral reinforcement learning: Online adaptive algorithm for optimal control. *International journal of robust and nonlinear control*, 24:2686–2710, 2014. ISSN 1049-8923. doi: 10.1002/rnc.3018.

[64] M. Vecerik, O. Sushkov, D. Barker, T. Rothorl, T. Hester, and J. Scholz. A practical approach to insertion with variable socket position using deep reinforcement learning. In *ICRA*, volume 2019-, pages 754–760. IEEE, 2019. ISBN 1050-4729. doi: 10.1109/ICRA.2019.8794074.

[65] M. Vitelli and A. Nayebi. Carma : A deep reinforcement learning approach to autonomous driving. 2016. URL `https://api.semanticscholar.org/CorpusID:26168074`.

[66] D. Vrabie and F. Lewis. Neural network approach to continuous-time direct adaptive optimal control for partially unknown nonlinear systems. *Neural networks*, 22:237–246, 2009. ISSN 0893-6080. doi: 10.1016/j.neunet.2009.03.008. ObjectType-Article-1 SourceType-Scholarly Journals-1 ObjectType-Feature-2 content type line 23.

[67] D. Vrabie, O. Pastravanu, M. Abu-Khalaf, and F. L. Lewis. Adaptive optimal control for continuous-time linear systems based on policy iteration. *Automatica (Oxford)*, 45: 477–484, 2009. ISSN 0005-1098. doi: 10.1016/j.automatica.2008.08.017.

[68] T. Wei, Y. Wang, and Q. Zhu. Deep reinforcement learning for building hvac control. volume 128280, 2017. ISBN 0738-100X. doi: 10.1145/3061639.3062224.

[69] P. J. Werbos. Approximate dynamic programming for real-time control and neural modeling. *handbook of intelligent control neural fuzzy adaptive approaches*, 1992.

[70] Z. Yang, F. Zhu, and F. Lin. Deep-reinforcement-learning-based energy management strategy for supercapacitor energy storage systems in urban rail transit. *IEEE transactions on intelligent transportation systems*, 22:1150–1160, 2021. ISSN 1524-9050. doi: 10.1109/TITS.2019.2963785.

[71] C. Yu, X. Wang, X. Xu, M. Zhang, H. Ge, J. Ren, L. Sun, B. Chen, and G. Tan. Distributed multiagent coordinated learning for autonomous driving in highways based on dynamic coordination graphs, 2020.

[72] Q. Zhao, H. Xu, and J. Sarangapani. Finite-horizon near optimal adaptive control of uncertain linear discrete-time systems. *Optimal control applications methods*, 36: 853–872, 2015. ISSN 0143-2087. doi: 10.1002/oca.2143. ark:/67375/WNG-95SQT4J7-X ArticleID:OCA2143 istex:AAE8FF106FE332FF3FA17462FAD05B2FC2829CEA ObjectType-Article-1 SourceType-Scholarly Journals-1 ObjectType-Feature-2 content type line 23.

[73] J. G. Ziegler and N. B. Nichols. Optimum settings for automatic controllers. *Journal of Dynamic Systems, Measurement, and Control*, 115:220–222, 6 1993. ISSN 0022-0434. doi: 10.1115/1.2899060. URL `https://doi.org/10.1115/1.2899060`.

[74] K. J. Åström and T. Hägglund. Automatic tuning of simple regulators with specifications on phase and amplitude margins. *Automatica*, 20:645–651, 1984. ISSN 0005-1098. doi: https://doi.org/10.1016/0005-1098(84)90014-1. URL `https://www.sciencedirect.com/science/article/pii/0005109884900141`.

[75] K. J. K. J. Åström and B. Wittenmark. *Adaptive control*. Mineola, N.Y. : Dover Publications, 2008, 2nd ed., dover ed. edition, 2008. ISBN 9780486462783. Includes bibliographical references and index.

Appendices

Appendix A

Appendix

A.1 Single-Tank

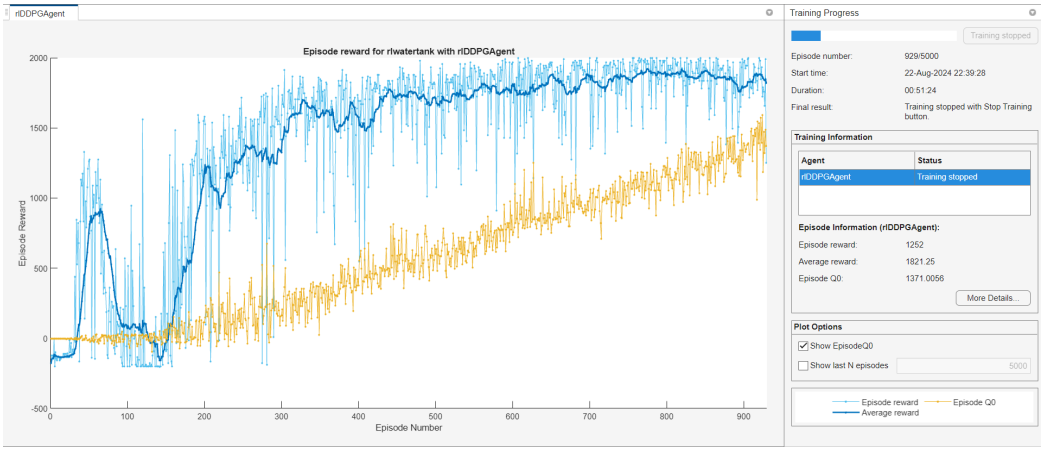

Figure A.1: *Training Monitor of DDPG Algorithm in Single-tank system*

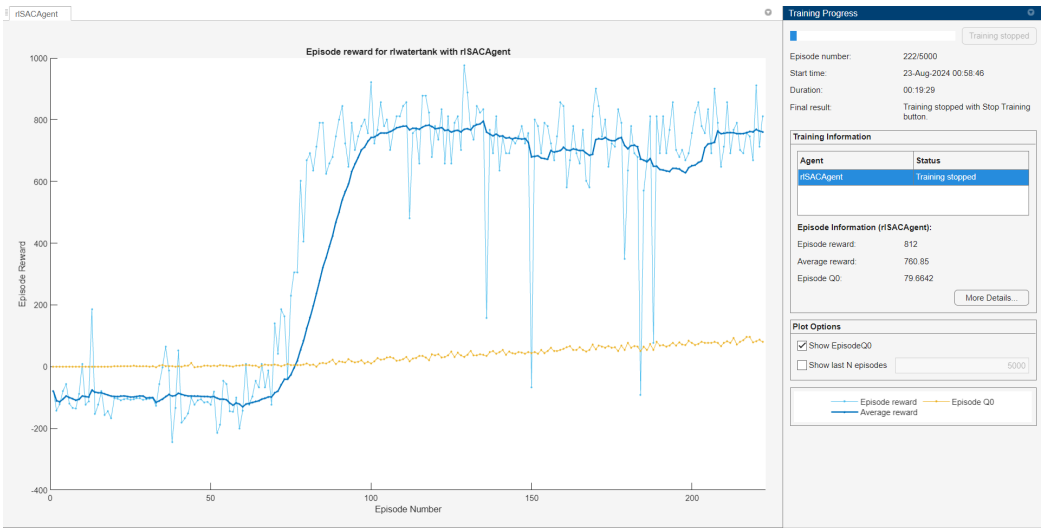

Figure A.2: *Training Monitor of SAC Algorithm in Single-tank system*

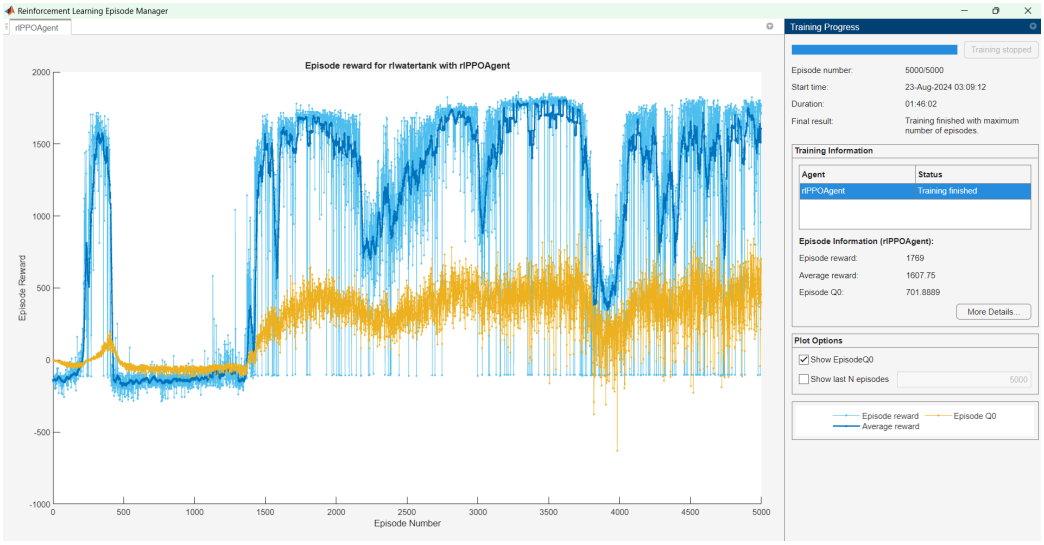

Figure A.3: *Training Monitor of PPO Algorithm in Single-tank system*

A.2 Quadruple-Tank

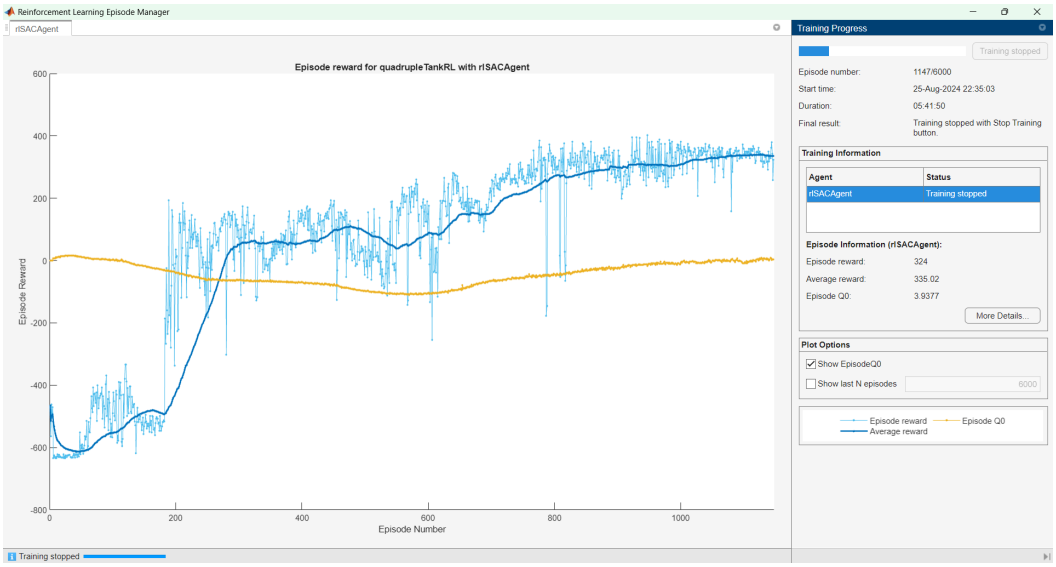

Figure A.4: *Training Monitor of SAC Algorithm in Quadruple-tank system*

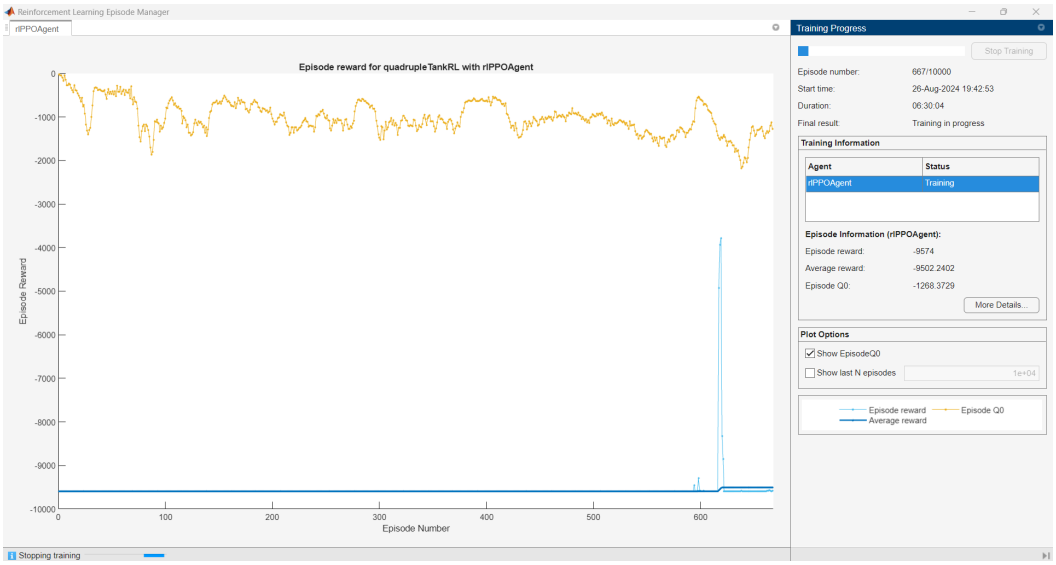

Figure A.5: *Training Monitor of PPO Algorithm in Quadruple-tank system*