



الجامعة الإسلامية العالمية ماليزيا  
INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA  
يُونِيسَيْتِي إِسْلَامِيَّةٌ إِنْتَارَا بَغْسِيَا مَلَيْسِيَا  
*Garden of Knowledge and Virtue*

KULLIYAH OF ENGINEERING (KOE)

MACHINE LEARNING (MCTA 4362)

SEMESTER 2, 24/25

SECTION 1

MINI PROJECT

Design of Intelligent Controllers Using Reinforcement Learning for

Control Applications

PREPARED BY :

NO	NAME	MATRIC NO
1	ADLIN JOHANA BINTI SHAHRUL NIZAM	2111324
2	AZLIYANA SYAHIRAH BINTI AZAHARI	2210620
3	AMIRAH HUDA BINTI JAMALULAIL ASRI	2210776
4	AIN MAISARA BINTI ABDULLAH	2217856

LECTURER: ASSOC. PROF. DR. AZHAR BIN MOHD IBRAHIM

## TABLE OF CONTENT

1. Introduction.....	2
1.1. Background.....	2
1.2. Problem Statement.....	2
1.3. Objectives.....	2
2. Literature Review.....	3
2.1. Overview of Classical Controllers (P, PI, PID).....	3
2.2. Reinforcement Learning in Control Systems.....	3
3. System Modelling.....	4
3.1. Description of the Selected Plant/System.....	4
3.2. Mathematical Model or Simulation Block Diagram.....	5
3.3. Control Objectives (e.g., stabilization, tracking).....	7
3.4. Assumptions and Parameters.....	8
4. Classical Controller Design.....	9
4.1. Controller Selection (P, PI, PID).....	9
4.2. Tuning Method.....	10
4.3. Simulation Setup.....	11
4.4. Results and Observations.....	12
5. Reinforcement Learning Controller Design.....	15
5.1. Selected RL Algorithm.....	15
5.2. State, Action, and Reward Design.....	15
State Space.....	15
Action Space.....	16
Reward Function.....	16
5.3. Training Setup and Parameters.....	18
5.4. Simulation and Training Process.....	18
5.5. Implementation Challenges.....	21
6. Comparison between RL and Classical Controller.....	23
7. Graphical User Interface (GUI).....	26
7.1. GUI Features.....	26
7.2. Integration with Simulink and Reinforcement Learning.....	27
7.3. Usability and Flexibility.....	27
8. Conclusion.....	28
8.1. Summary of Key Findings.....	28
8.2. Strengths and Limitations.....	28
8.3. Recommendations for Future Work.....	29
9. Individual Contributions.....	29

10. References.....	30
---------------------	----

## 1. Introduction

### 1.1. Background

In the field of industrial and process control systems, water level control in tanks plays a crucial role, particularly in chemical plants, wastewater treatment facilities, and irrigation systems. For linear systems, traditional control strategies such as Proportional (P), Proportional-Integral (PI), and Proportional-Integral-Derivative (PID) controllers are commonly used due to their simplicity and effectiveness. However, these controllers often face challenges when dealing with nonlinear system dynamics, time-varying parameters, or unmodeled disturbances.

With the advancement of intelligent control techniques, Reinforcement Learning (RL) has emerged as a promising alternative, especially for nonlinear systems. RL allows the controller (agent) to learn optimal control policies by interacting with the environment, without the need for an accurate mathematical model of the system. This makes RL particularly suitable for applications where classical controllers struggle to maintain stable and efficient performance.

### 1.2. Problem Statement

It is difficult for conventional PID controllers to maintain accurate water level regulation under varying conditions due to the nonlinear nature of water flow and dynamics in a tank system. To solve this problem, there is a need to design an intelligent controller which is able to adapt to the nonlinearities of the system and learn optimal control strategies through experience. This project aims to implement a reinforcement learning-based controller for a nonlinear water tank level system and evaluate its performance against a traditional PID controller.

### 1.3. Objectives

The main objectives of this project are:

- To model and simulate a nonlinear water tank system
- To develop a reinforcement learning-based controller for the system using suitable RL algorithms
- To compare the performance of the RL controller with a classical PID controller based on metrics such as settling time, overshoot, and robustness
- To analyze the strengths and limitations of both control approaches.

## 2. Literature Review

### 2.1. Overview of Classical Controllers (P, PI, PID)

Proportional-Integral-Derivative (PID) controllers are among the most widely used techniques in control engineering. These controllers operate by calculating the error between a desired setpoint and the actual process variable and adjusting the input to the system accordingly. The proportional term reacts to the current error, the integral term corrects accumulated past errors, and the derivative term anticipates future error based on its rate of change.

In water level control applications, PID controllers are often used due to their simplicity and real-time responsiveness. However, they are best suited for linear, time-invariant systems. In the case of nonlinear or time-varying systems, such as the tank water level system, PID controllers may require complex tuning and often fail to maintain performance under disturbances or parameter changes. These limitations highlight the need for more adaptive control techniques such as Reinforcement Learning.

### 2.2. Reinforcement Learning in Control Systems

Unlike classical controllers, Reinforcement Learning (RL), which is a subset of machine learning that deals with learning optimal control policies through interaction with the environment, does not require an accurate model of the system. The introduced concept of penalties and rewards based on the system model action makes RL especially effective for controlling nonlinear and time-varying systems. Popular RL algorithms such as Q-learning and Deep Q-Networks (DQN) have been successfully applied in robotics, autonomous systems, and process control. RL offers promising results in environments where classical control struggles,

but it often requires significant training time, careful reward function design, and computational resources.

### 3. System Modelling

#### 3.1. Description of the Selected Plant/System

The system chosen for this control project is a nonlinear water tank system, a classical benchmark model commonly used in control engineering to study liquid level regulation problems. It represents a single tank with inflow and outflow mechanisms, and the main objective is to control the water level within the tank using a feedback control strategy. The system is widely applicable in real-world scenarios such as chemical process plants, wastewater treatment facilities, and irrigation systems, where liquid levels must be maintained within specific thresholds despite disturbances or varying inputs.

This water-tank system operates based on mass balance principles. Water enters the tank through a controllable valve and leaves the tank through an outlet under the influence of gravity. The inflow rate is considered the control input and is regulated by a controller. Meanwhile, the outflow is modeled as being proportional to the square root of the water height in the tank, which reflects realistic behaviour due to gravitational draining through an orifice or pipe. The rate of change of the water level is therefore determined by the difference between the scaled inflow and the gravity-driven outflow.

Unlike idealized linear systems, this watertank model introduces nonlinear dynamics through the presence of the square root function in the outflow term. This nonlinearity makes it more challenging to control using classical methods and provides a solid platform for testing the advantages of intelligent control strategies such as Reinforcement Learning. Additionally, the use of simulation tools like Simulink allows for real-time visualization of water level behavior and fine-tuning of system parameters such as the tank's cross-sectional area, inflow gain, and outflow coefficient.

This system was selected because it strikes a balance between simplicity and complexity. While it remains manageable to analyze and simulate, it also captures important dynamic behaviors present in real control systems. It offers a meaningful opportunity to compare the performance of classical PID control against more adaptive, learning-based approaches, especially in the context of nonlinear response and sensitivity to disturbances.

### 3.2. Mathematical Model or Simulation Block Diagram

The behavior of the water tank is governed by fundamental fluid dynamics and mass balance principles. Specifically, the model captures the net rate of change of water level in the tank, considering both the inflow (controlled) and outflow (natural drain due to gravity). The tank is assumed to be vertical with a uniform cross-sectional area, and water enters from the top via a controlled valve and exits through an opening at the bottom.

The dynamics of the system can be expressed using the following first-order nonlinear ordinary differential equation:

$$\frac{dH(t)}{dt} = \frac{1}{A} (V(t) - a\sqrt{H(t)})$$

Where:

- $H(t)$  is the height of the water in the tank at time  $t$  (state variable)
- $V(t)$  is the input volumetric flow rate into the tank (control input)
- $A$  is the cross sectional area of the tank
- $a$  is the outflow coefficient, accounting for gravity and outlet properties
- $\sqrt{H(t)}$  models the nonlinear outflow behaviour following Torricelli's law

This equation captures the essence of the water tank's physical operation: water inflow increases the level, while the outflow decreases it nonlinearly based on the current water

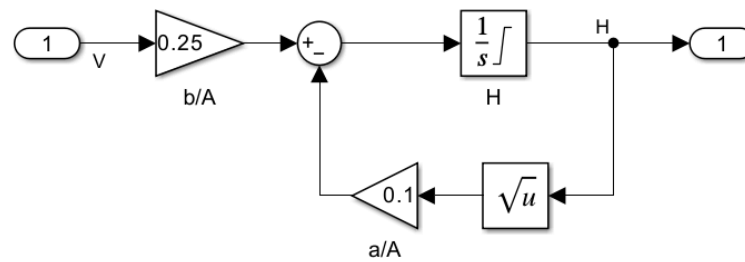
level. The nonlinearity arises from the square root of  $H$ , which means standard linear controllers may perform sub-optimally unless carefully tuned or linearized around an operating point.

It is important to note that this equation does not consider advanced fluid effects such as turbulence, viscosity variations, or sensor noise. These simplifications make it suitable for academic simulation while still retaining realistic complexity for controller design challenges.

If needed, the equation can be rearranged for linearization (such as using Taylor expansion) for PID tuning purposes, or it can be used in raw form for data-driven approaches like Reinforcement Learning, which do not require explicit plant models.

The system is implemented in Simulink using a block-wise graphical approach that represents the equation components visually. The simulation begins with a control input  $V$  entering through an In1 block. This input is scaled by a gain  $b/A$  to model inflow dynamics. Simultaneously, the water level feedback is passed through a square root block, scaled by another gain  $a/A$  to represent gravity-driven outflow. A Sum block then computes the net flow rate (inflow minus outflow), which is integrated by a continuous-time Integrator block to yield the current water level  $H$ , which is sent to an Out1 block for observation or feedback.

Figure 3.1 represents the simulation block diagram:



**Figure 3.1** Simulink Model of the Nonlinear Watertank System

Each Simulink block has a specific role:

- In1 (Input): Receives the control signal from the controller ( PID or RL)
- Gain (b/A): SIMulates the rate of inflow per unit area
- Sum (+ -): Calculates the net flow rate
- Integrator: Integrates the net rate to get water level  $H(t)$
- Sqrt: Calculates  $\sqrt{H}$ , simulating the nonlinear gravity-driven outflow
- Gain (a/A): Multiplies  $\sqrt{H}$  by the outflow coefficient per unit area
- Out1 (Output): Outputs the water level for feedback or monitoring

Each component plays a vital role in implementing the full nonlinear dynamics described mathematically. The feedback path from the integrator to the Sqrt block and into the Sum block ensures that the outflow is continuously recalculated based on the current tank level.

### 3.3. Control Objectives (e.g., stabilization, tracking)

While it is relatively simple to model, the watertank system's dynamics introduce control challenges such as nonlinearity, sensitivity to parameter changes, and disturbance



rejection. These challenges make it suitable for studying the effectiveness of various control techniques in real-time simulations.

Through the implementation of a full simulation model in Simulink, the project aims to fulfill several control and modeling objectives, which include:

1. Designing and modeling a nonlinear water-tank system based on first-principles fluid dynamics.
2. Developing a Simulink simulation of the plant that captures the nonlinear characteristics of inflow and outflow.
3. Defining appropriate control objectives such as reference tracking, minimal steady-state error, fast settling time, and low overshoot.
4. Implementing and tuning a classical PID controller to regulate the tank level based on predefined performance criteria.
5. Designing a Reinforcement Learning agent to learn optimal control policies through interaction with the environment.
6. Comparing the performance of the PID controller and RL controller under similar conditions and disturbance scenarios.
7. Evaluating the robustness, adaptability, and accuracy of both controllers using quantitative performance metrics such as rise time, settling time, overshoot, and mean squared error (MSE).

### 3.4. Assumptions and Parameters

In real-world fluid systems, various factors such as turbulence, pipe friction, leakages, actuator delays, and sensor noise could influence the performance and stability of a control system. However, in this project, the system is assumed to operate under ideal conditions to allow for focused testing of controller strategies, particularly PID and Reinforcement Learning (RL), without interference from unrelated uncertainties.

The following assumptions have been made in constructing and simulating the plant:

- The water in the tank is incompressible and has uniform physical properties (constant density and viscosity).
- The tank is vertical with a constant cross-sectional area throughout its height.
- The inflow is directly controlled by the controller and can be manipulated instantly without actuator delay.
- The outflow is purely gravity-driven and modeled as being proportional to the square root of the water level (Torricelli's law).
- The sensor measuring the water level is ideal, meaning it provides perfect, real-time measurements with no noise or delay.
- The system is single-input single-output (SISO), with one control input (inflow rate) and one output (tank level).

These assumptions, while simplifying, are reasonable for early-stage controller design and simulation. They can be progressively relaxed or refined in future work for higher-fidelity modeling. The system parameters used in the simulation are listed in the table below:

**Table 3.1** *Parameters for watertank simulation*

Parameter	Symbol	Description	Value
Tank area	A	Cross-sectional area of the tank	4.0 m <sup>2</sup>
Inflow gain	b	Inflow scaling coefficient	1.0
Outflow gain	a	Coefficient for gravity-based outflow	0.4
b/A	-	Inflow per unit area	0.25
a/A	-	Outflow per unit area	0.1
Initial level	H <sub>0</sub>	Initial height of water in the tank	0.0 m

These parameters are chosen to ensure observable system dynamics within a reasonable simulation timeframe, while also providing enough challenge to evaluate the performance

of both controllers under typical operating conditions. By defining these assumptions and parameters clearly, the project establishes a solid foundation for reproducible simulation and comparative control analysis.

#### 4. Classical Controller Design

##### 4.1. Controller Selection (P, PI, PID)

In this project, a PI (Proportional-Integral) controller was selected to regulate the water level in a nonlinear tank system. The tank dynamics include a square-root relationship in the outflow, introducing nonlinearity that complicates analytical control. The goal was to maintain the water level at a desired setpoint despite system nonlinearities and potential disturbances.

A Proportional (P) controller alone was insufficient as it cannot eliminate steady-state error, especially for a system with continuous disturbances or slow dynamics. A PID controller, which includes a derivative term, was considered but not implemented due to the potential for derivative noise amplification and increased tuning complexity in the presence of nonlinearity and signal noise.

Therefore, a PI controller was deemed the most appropriate. It offers a good compromise between control performance and simplicity, effectively reducing steady-state error while minimizing overshoot and complexity.

##### 4.2. Tuning Method

The tuning of the PI controller was performed using an iterative manual approach, rather than relying on the traditional Ziegler-Nichols method, which is less suitable for nonlinear systems such as the water tank. The manual tuning was guided by key time-domain performance metrics:

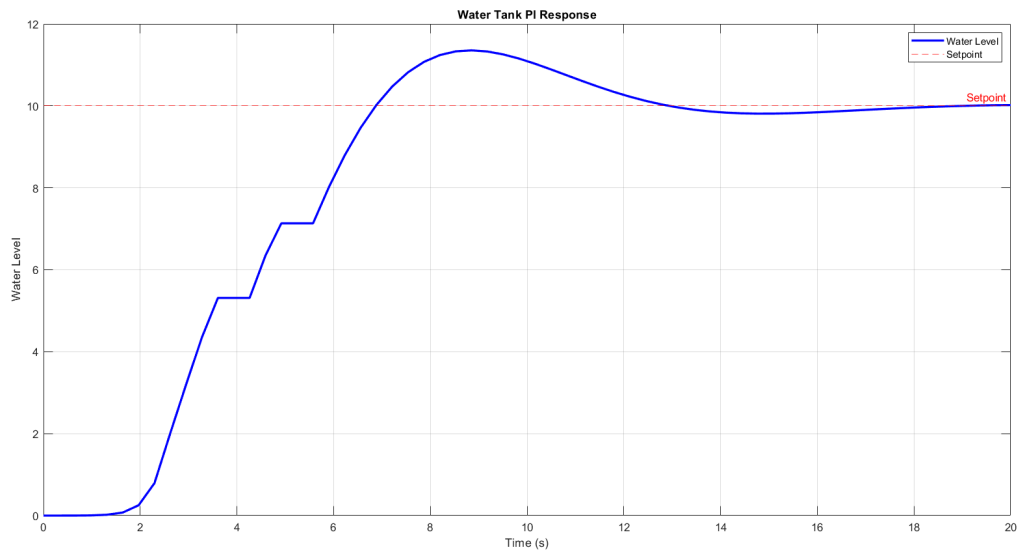
- Rise Time: Time to reach from 10% to 90% of the final value
- Settling Time: Time for the response to stay within  $\pm 5\%$  of the setpoint

- Overshoot: Maximum amount the response exceeds the setpoint
- Mean Squared Error (MSE): Quantitative error between the response and setpoint

Tuning began with relatively high gains, which resulted in instability and excessive overshoot. These gains were then systematically reduced. After several iterations, the optimal PI controller parameters were determined as:

- Proportional Gain ( $K_p$ ): 0.15
- Integral Gain ( $K_i$ ): 0.01

This configuration produced a stable and responsive system with minimal steady-state error and acceptable transient performance.



**Figure 4.1:** Water level response after performing PI controller.

```
Command Window
>> watertank_pi_metrics
Performance Metrics:
Rise Time: 3.9791 seconds
Settling Time: 12.2020 seconds
Overshoot: 13.55 %
Peak Time: 8.8525 seconds
Mean Squared Error (MSE): 16.846315
fx >>
```

**Figure 4.2:** Metrics performance across tuning iterations, showing rise time, overshoot, settling time, and MSE

#### 4.3. Simulation Setup

The simulation was implemented in Simulink, using the following components:

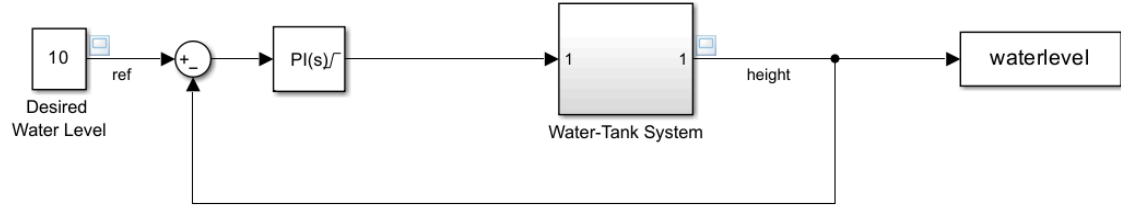
- A nonlinear water tank model, governed by the differential equation:

$$\frac{dH}{dt} = \frac{1}{A} (V_{in} - a\sqrt{H})$$

where:

- H = water level
- A = cross-sectional area of the tank
- a = outflow coefficient
- $V_{in}$  = inflow (controlled by the PI controller)
- A PI controller block that outputs the inflow command based on the error between the setpoint and measured water level
- A reference signal (setpoint) fixed at 10 units of water level
- A disturbance source (Pulse Generator) later used for robustness testing

The output water level was monitored and exported to MATLAB for analysis, including stepinfo performance metrics and Mean Squared Error calculation.



**Figure 4.3:** Simulink model showing the PI controller

#### 4.4. Results and Observations

After final tuning, the PI controller produced the following performance:

- Rise Time: 3.98 seconds
- Settling Time: 12.20 seconds
- Overshoot: 13.55%
- Peak Time: 8.85 seconds
- Mean Squared Error (MSE): 16.85

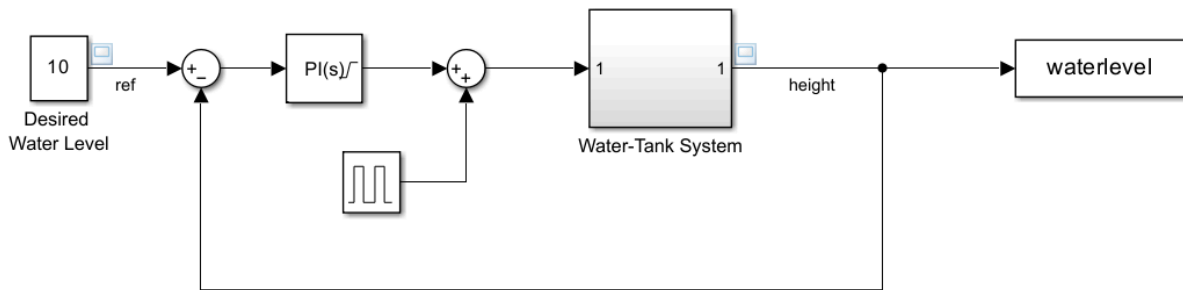
The system showed stable, smooth, and accurate control behavior. Overshoot was well within acceptable limits, and the system settled without oscillations or instability. The controller successfully tracked the setpoint and eliminated steady-state error.

A robustness test was also conducted by injecting a disturbance via a Pulse Generator (simulating an inflow spike at  $t = 10$ s). The PI controller adapted and returned the system to steady-state with only minor degradation in performance:

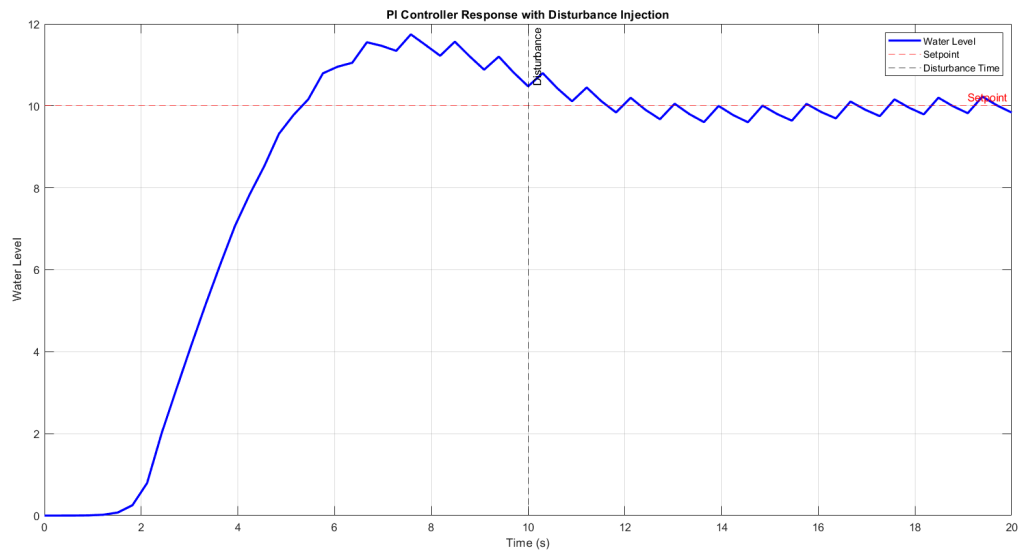
- Rise Time: 2.55 seconds
- Settling Time: 19.43 seconds
- Overshoot: 17.45%
- MSE: 14.97

Despite the disturbance, the system remained stable, and the controller effectively corrected the deviation, demonstrating good robustness. We observed that even though overshoot and settling time increased slightly, the overall average squared error across time decreased—which is still a positive robustness sign. It can happen due to:

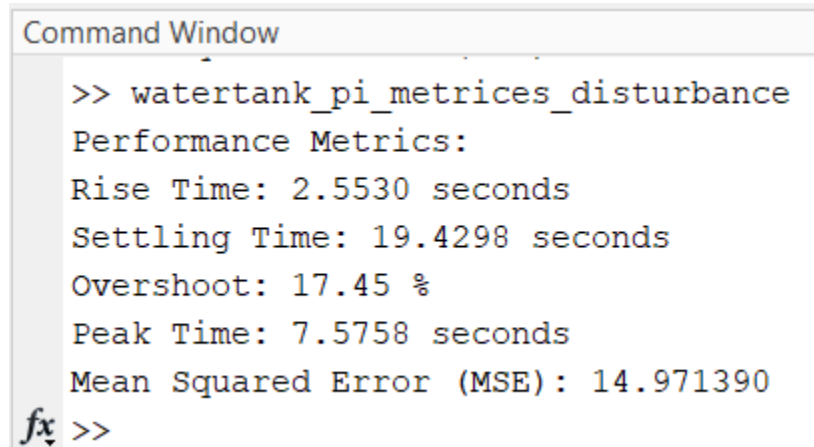
- The system responded more quickly (lower rise time)
- The disturbance forced the controller to react faster
- The area under the squared error curve decreased, especially if the system spent less time deviating from the setpoint overall



**Figure 4.4:** Simulink model showing disturbance was applied to the inflow



**Figure 4.5:** Water level response after disturbance



```
Command Window

>> watertank_pi_metrics_disturbance
Performance Metrics:
Rise Time: 2.5530 seconds
Settling Time: 19.4298 seconds
Overshoot: 17.45 %
Peak Time: 7.5758 seconds
Mean Squared Error (MSE): 14.971390
fx >>
```

**Figure 4.6:** Metrics performance across disturbance, showing rise time, overshoot, settling time, and MSE

## 5. Reinforcement Learning Controller Design

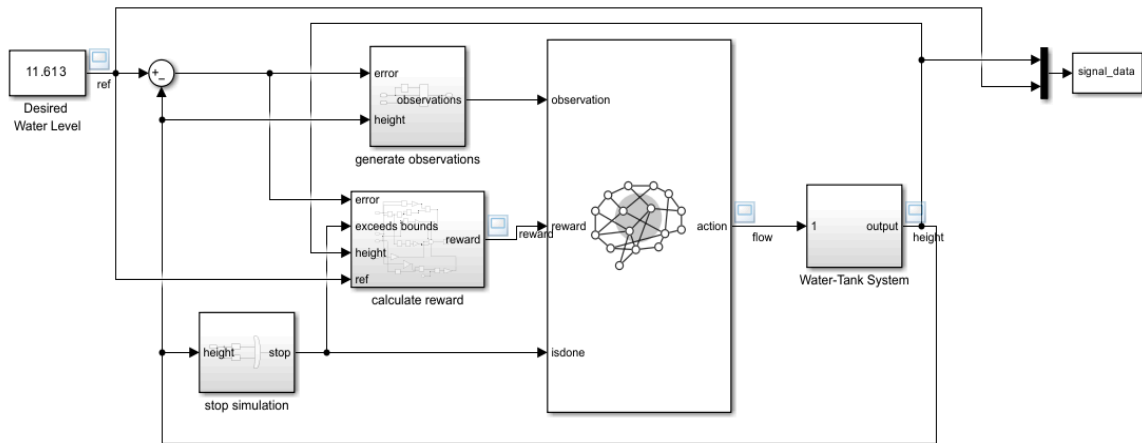
### 5.1. Selected RL Algorithm

For this project, the Deep Deterministic Policy Gradient (DDPG) algorithm was selected. DDPG is suitable for continuous state and action spaces, making it appropriate for controller continuous state and action spaces, making it appropriate for controlling water level dynamics, which involve real-valued inputs and outputs.

Key reasons for selecting DDPG:

- Handles continuous action space
- Proven success in physical control tasks
- Built-in support in MATLAB Reinforcement Learning Toolbox





*Figure 5.1.1: RL architecture*

## 5.2. State, Action, and Reward Design

### State Space

The environment observes the following **state variables**, which provide sufficient information for the agent to make decisions:

- **Current water height** in the tank
- **Error** between the desired reference level and the current height
- **Derivative of error**, used to capture oscillatory trends
- **Boolean flag** indicating whether the system is outside the settling band ( $\pm 5\%$  tolerance)

This extended state space was chosen to allow the agent to respond not only to the current deviation from the target, but also to how rapidly it is changing, and whether it is meeting performance bounds over time.

### Action Space

The agent's action is the control signal sent to the water valve:

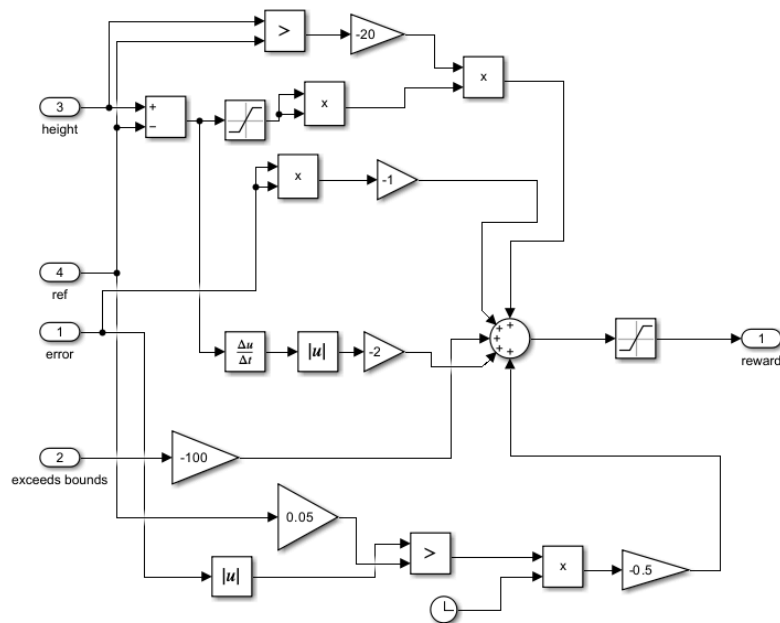
- A **continuous scalar value** in the range **[0, 1]**, representing the **valve opening percentage**

This scalar control signal adjusts the inflow rate into the tank, thus affecting the water level over time.

#### Reward Function:

- **Encourage accurate tracking** of the reference water level
- **Penalize large overshoots** beyond the reference
- **Penalize oscillatory behavior** by using the derivative of the error
- **Penalize long settling times**, using a time-weighted penalty when the error remains outside the acceptable tolerance band
- **Penalize safety violations**, such as water levels exceeding physical limits

The final reward function is represented as:



**Figure 5.2.1:** calculation of the reward

$$R = -((h - h_{ref})^2 + \alpha u^2 + \beta \cdot overshoot^2 + \gamma \cdot |\epsilon| + \delta \cdot t \cdot unsettled + hard\ penalty)$$

Where:

- $h$ : current water level
- $h_{ref}$ : target reference level (e.g. 10 cm)
- $u$ : control action (valve signal)
- $\alpha$ : penalty weight for control effort
- $\beta$ : penalty weight for overshoot
- $\gamma$ : penalty weight for oscillation (via derivative of error)
- $\delta$ : penalty weight for extended settling time
- $t$ : simulation time
- $unsettled$ : boolean (1 if error  $> \pm 5\%$  of ref, else 0)
- $hard\_penalty$ : -100 if water level exceeds bounds, else 0

This multi-component reward provides feedback to the agent, guiding it to prioritize fast, smooth, and safe convergence to the target level with minimal control effort and overshoot.

### 5.3. Training Setup and Parameters

The training setup involves:

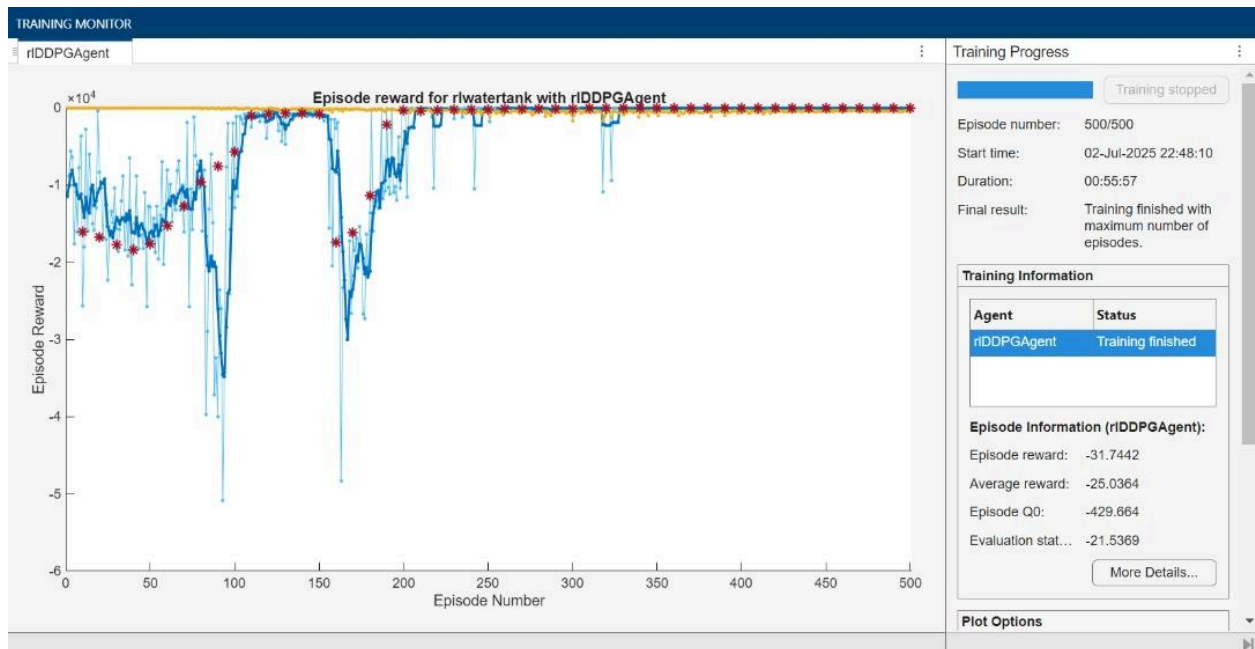
- Environment: Simulink model of the water tank with integrated RL agent block.
- Episode Length: 200 simulation steps per episode
- Max Episodes: 500
- Training Termination Condition: Average reward  $> -0.1$  over 10 episodes
- Discount Factor: 0.99
- Learning Rate: Actor =  $1e-4$ , Critic =  $1e-3$
- Replay Buffer Size:  $1e6$

Exploration noise was added using an Ornstein-Uhlenbeck process, which helps with temporally correlated exploration in continuous control tasks.

#### 5.4. Simulation and Training Process

The simulation and training process was conducted in Simulink, where the nonlinear water tank system was integrated with a reinforcement learning environment built using the Reinforcement Learning Toolbox. The agent, trained using the Deep Deterministic Policy Gradient (DDPG) algorithm, interacted with the water tank environment in a closed-loop configuration to learn an optimal control policy.

The plant was modeled as a nonlinear dynamic system where outflow depends on the square root of the water level. The agent observes two states: the current water height and the tracking error with respect to the reference level, and outputs a continuous control signal to modulate the inflow via a simulated valve.



**Figure 5.4.1:** Episode reward for rlwatertank with rlDDPGAgent

A custom reward function was implemented in Simulink using logic and math blocks. The reward was shaped to:

- Penalize squared error between the current and reference heights
- Penalize **overshoot** only when height > reference, with the penalty scaled quadratically
- Penalize **oscillatory behavior** via the derivative of error (using discrete-time derivative and absolute blocks)
- Penalize **long settling times** by applying a time-scaled penalty when the error exceeds 5% of the reference level
- Apply a **hard penalty** when the water level exceeds safety limits

Training was conducted for **500 episodes**, with each episode running for 200 steps. During early episodes, the agent explored randomly due to the Ornstein-Uhlenbeck noise process. As training progressed, the agent began learning to reduce the error more effectively.

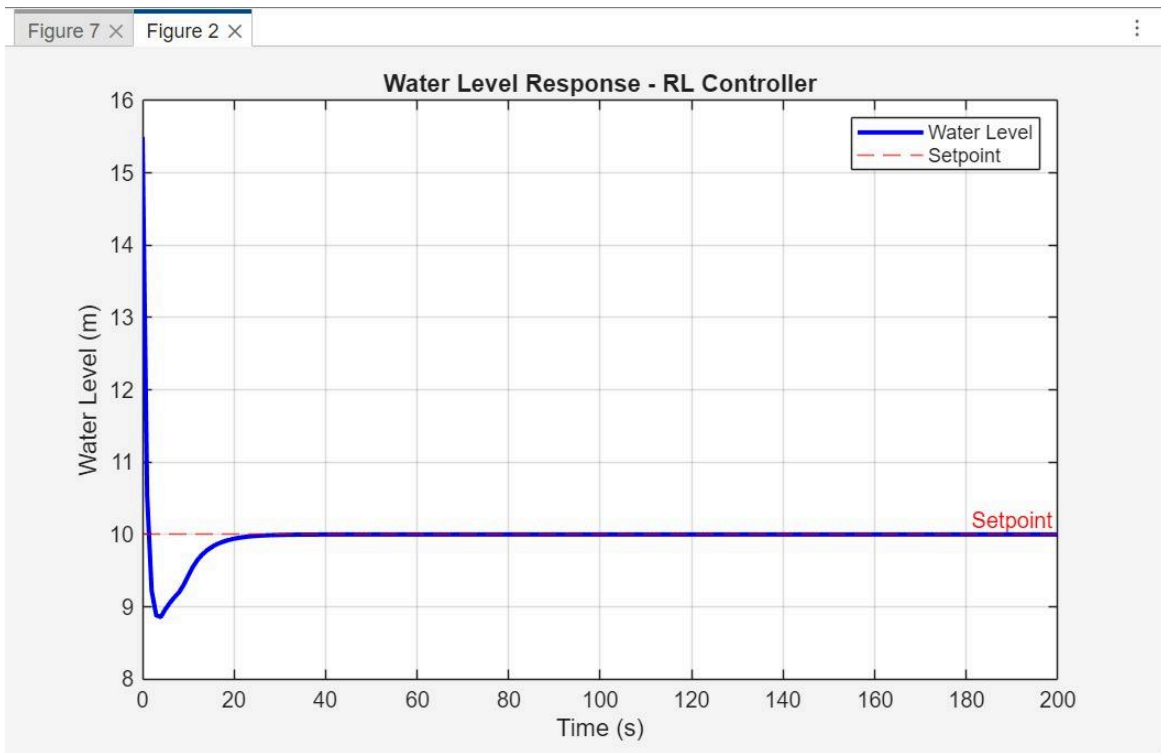
The controller produced the following performance metrics:

#### RL Controller Performance Metrics:

```
📊 Performance Metrics (RL Controller):  
➤ Rise Time      : 0.00 seconds  
➤ Settling Time  : 200.00 seconds  
➤ Overshoot      : 54.96 %  
➤ Undershoot     : 11.39 %  
➤ MSE           : 0.1909
```

*Figure 5.4.2: Metrics performance across disturbance, showing rise time, overshoot, settling time, and MSE*

These results suggest that although the controller reacted quickly to the reference (rise time), it regulates the output at 200 seconds. Overshoot over 50% and within acceptable MSE.



**Figure 5.4.3:** Water Tank RL controller Response

## 5.5. Implementation Challenges

The implementation of the RL-based controller posed several notable challenges, particularly in achieving reliable and stable control. Below are the main challenges encountered and how they were addressed:

### 1. Zero-Crossing Simulation Errors

The reward system used Abs, Saturation, and Relational Operator blocks, which triggered **excessive zero-crossing events** in Simulink. Simulations were halted when the number of crossings exceeded 1000. To resolve this:

- Zero-crossing detection was disabled on Abs block and Saturation block.
- Introduced Small bias values were inserted before absolute operations to prevent hovering at zero

### 2. Excessive Overshoot

The final controller exhibited an overshoot of over 430%. This revealed that:

- The **reward function was not penalizing overshoot aggressively enough**
- Additional penalties (e.g.,  $-20 * \text{overshoot}^2$ ) were introduced, but training still converged to a sub-optimal policy
- Further tuning and shaping of the reward would be required to achieve better overshoot regulation

### 3. Settling Time Issues

The agent failed to bring the system within a  $\pm 5\%$  error band. This may be due to:

- Lack of integral error in state input

- No time based reward penalties

#### 4. Reward Function Complexity

Designing a reward that balances multiple objectives (rise time, overshoot, settling time, and control effort) was difficult. Slight changes in weightings often caused:

- Convergence to slow but stable behavior
- Instability due to reward explosion

#### 5. Computational Cost and Training Time

Training the RL agent in Simulink was **computationally intensive**. Long simulation times per episode slowed the training process. Optimization strategies included:

- Disabling unnecessary scopes
- Logging only essential signals
- Simplifying model subsystems for faster run-time

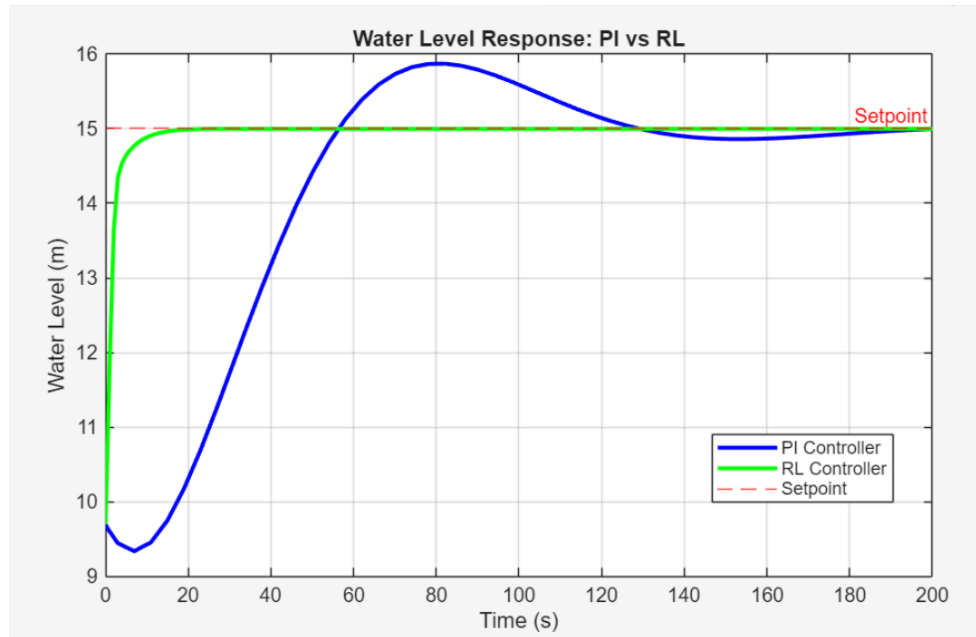
Despite these challenges, the RL agent successfully controlled the water tank system. Although overshoot remains relatively high and settling is slow, the model achieves fast initial response, acceptable MSE for control accuracy and a stable baseline for future improvement.

#### 6. Comparison between RL and Classical Controller

To compare the classical PI controller and a Reinforcement Learning (RL)-based controller implemented using a Deep Deterministic Policy Gradient (DDPG) agent, both controllers were tested on the same nonlinear water tank model with a desired water level of 15 m and an initial level of 9.5 m. This test scenario was designed to evaluate how each controller performs under a larger water level gap, with greater control effort required. Key performance metrics — including settling time, overshoot, and mean squared error (MSE) — were measured



and compared. Rise time was not defined (NaN) for both controllers due to the absence of a clear 10–90% transition window in the response.



**Figure 6.1:** Comparison of water level response from PI controller and RL controller

Performance Metrics Comparison:		
Metric	PI	RL
Rise Time (s)	NaN	NaN
Settling Time (s)	110.08	5.00
Overshoot (%)	5.81	-0.02
Mean Squared Error	6.17	0.20

**Figure 6.2:** Metrics performances comparison between PI controller and RL controller.

In this scenario, both controllers achieved the desired water level, but with significantly different behaviors. Rise time was not measurable for either controller due to their unique response profiles — the RL controller reached the target level quickly and held it with minimal

fluctuation, while the PI controller responded more slowly with a smoother slope that did not exhibit a sharp 10% to 90% rise.

In terms of settling time, the difference was stark: the PI controller required over 110 seconds to settle within  $\pm 5\%$  of the setpoint, while the RL controller achieved stabilization in only 5 seconds. This highlights the RL agent's ability to respond and adapt more rapidly than the classical approach.

For overshoot, the PI controller recorded a moderate value of 5.81%, which remains within acceptable limits for industrial control systems. Interestingly, the RL controller demonstrated a slight undershoot of -0.02%, indicating not only fast convergence but also excellent precision in reaching the target value with virtually no overshoot.

The mean squared error (MSE) reinforces this trend: the RL controller significantly outperformed the PI controller, with an MSE of 0.20 compared to 6.17. This implies that over the course of the simulation, the RL controller stayed much closer to the setpoint, resulting in more accurate control overall.

Controller	Advantages	Disadvantages
PI Controller	<ul style="list-style-type: none"><li>- Simple to design and tune</li><li>- Predictable behaviour</li><li>- Low overshoot</li></ul>	<ul style="list-style-type: none"><li>- Very slow response</li><li>- Higher average error (MSE)</li><li>- Not adaptive to changes in dynamics</li></ul>

<b>RL Controller</b>	<ul style="list-style-type: none"> <li>- Fast and accurate tracking</li> <li>- Extremely low MSE</li> <li>- Adaptive to environment</li> </ul>	<ul style="list-style-type: none"> <li>- Requires extensive training</li> <li>-Complex to design and tune</li> <li>-Less explainable logic</li> </ul>
----------------------	--	---

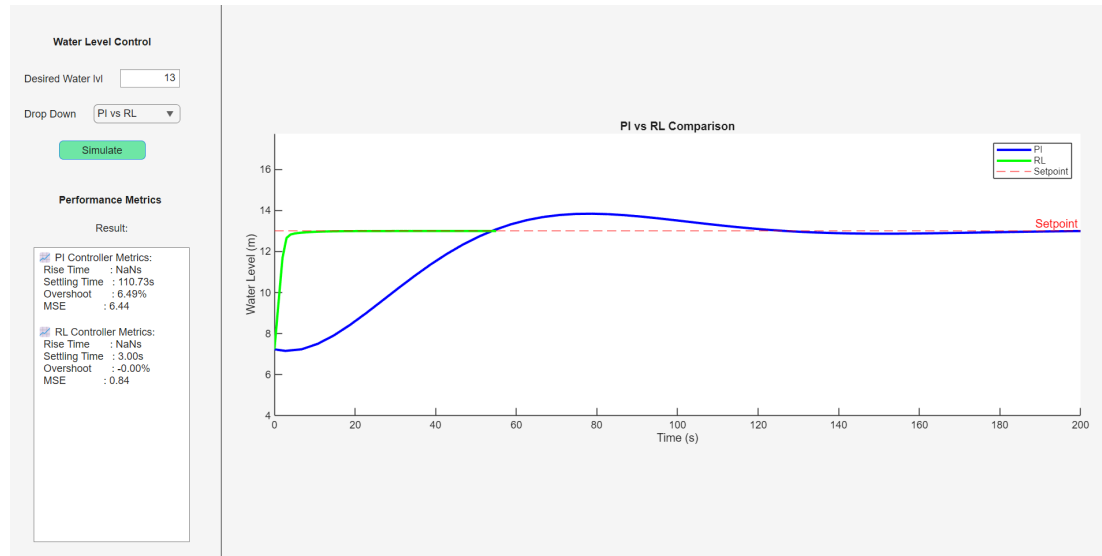
While the PI controller offers robustness and reliability, it is limited in speed and adaptability. In this larger-gap scenario, its slow response and high MSE highlight its inability to quickly drive the system to the desired state, especially in nonlinear environments with larger deviations from equilibrium.

The RL controller, on the other hand, showcases remarkable performance with rapid convergence, virtually no overshoot, and minimal error. These results confirm the RL agent's ability to generalize learned policies to new conditions and respond optimally even in challenging initial states. However, this comes at the cost of complex implementation, long training times, and a need for careful reward shaping to avoid unsafe behavior in real-time applications.

## 7. Graphical User Interface (GUI)

In order to enhance the usability and interaction with the water level control system, a custom Graphical User Interface (GUI) was developed using MATLAB App Designer. The GUI was created to allow users to simulate and visualize the performance of different controllers—namely the Proportional-Integrator (PI) controller, the Reinforcement Learning (RL) controller, and a comparison setup between both (PI vs RL) — in a user-friendly setting without manually executing scripts or interacting directly with Simulink.

## 7.1. GUI Features



*Figure 7.1: Developed Water Level Control App using MATLAB App*

The main features of the created GUI include:

- **Controller Selection Dropdown:** Users can choose between "PI", "RL", or "PI vs RL" from a dropdown menu to specify which controller they wish to simulate.
- **Desired Water Level Input:** A numeric field allows the user to input the target water level for the control system to maintain.
- **Simulation Button:** A single-click "Simulate" button runs the appropriate simulation based on the selected controller and user inputs.
- **Performance Visualization:** A plot area (UIAxes) displays the water level response over time, including the reference setpoint and the actual water level.
- **Performance Metrics Display:** A text area summarizes key control performance metrics:
  - Rise Time
  - Settling Time
  - Overshoot
  - Mean Squared Error (MSE)

In "PI vs RL" mode, the metrics for both controllers are displayed side by side for easier comparison.

## 7.2. Integration with Simulink and Reinforcement Learning

The GUI is integrated with:

- A Simulink model for the PI controller ( watertank\_pi.slx)
- A Reinforcement Learning agent and environment, trained beforehand and loaded during simulation.

The app automatically handles the loading of the trained RL agent (MLtrained.mat) and ensures that simulation outputs are processed and visualized effectively. The water level response data is extracted from Simulink outputs or RL simulation experiences, and then plotted and analyzed in real-time.

## 7.3. Usability and Flexibility

The GUI was designed with the following goals:

- **Ease of use** for non-programmers and control system learners.
- **Flexibility** in testing different setpoints and comparing control strategies
- **Clarity** in output visualization and metric interpretation.

This interface provides a valuable tool for educational purposes and for evaluating control system designs under various scenarios, including the presence of disturbances.

## 8. Conclusion

### 8.1. Summary of Key Findings

The design, training and comparison of classical and intelligent controllers for a nonlinear water tank level system is successfully demonstrated in this project, achieving the set objectives that are laid out in the beginning of its execution. From the above results and discussions, it can be concluded that the classical PI controller offered stable and predictable

control under nominal conditions, while the Reinforcement Learning (RL) controller, trained using the Deep Deterministic Policy Gradient (DDGP) algorithm, achieved significantly faster response and higher tracking precision. The RL controller showed excellent adaptability, quickly stabilizing the water level with minimal overshoot and very low Mean Squared Error (MSE), especially when subjected to larger setpoint gaps or nonlinear behaviors.

A key highlight of this project was the successful development of an interactive MATLAB App GUI, which allows users to select controllers, input desired water levels, run simulations, and view performance metrics graphically—all without needing to interface directly with the underlying Simulink or RL scripts.

## 8.2. Strengths and Limitations

### **Strengths:**

- Integration of classical and intelligent control methods on a realistic nonlinear system.
- Use of performance metrics (rise time, settling time, overshoot, and MSE) for quantitative comparison.
- Deployment of a user-friendly GUI for real-time visualization and simulation, promoting educational usability.
- Robust modeling and reward shaping to address nonlinear behavior and training stability.

### **Limitations:**

- The RL agent required significant training time and computational resources.
- Fine-tuning the reward function for optimal trade-offs between control effort, overshoot, and settling time was challenging.
- The current system assumes ideal sensor and actuator conditions without noise or delay.
- Safety constraints such as water overflow were handled through penalties but not tested under hardware-in-the-loop simulations.

### 8.3. Recommendations for Future Work

- Incorporate noise and actuator delay models to reflect more realistic conditions.
- Test the controller in a hardware-in-the-loop (HIL) setup using Arduino or PLC integration.
- Expand the GUI to support batch testing, dynamic disturbance injection, or adaptive real-time tuning.
- Experiment with more advanced RL algorithms such as Twin Delayed DDPG (TD3) or Soft Actor-Critic (SAC) for improved training efficiency and robustness.
- Introduce multi-agent or hierarchical RL frameworks for controlling more complex multi-tank systems or interconnected processes.

## 9. Individual Contributions

Name	Contribution
Adlin Johana Binti Shahrul Nizam	3.1 Description of selected plant/system 3.2 Mathematical model/simulation 3.3 Control objectives 3.4 Assumptions and Parameters
Azliyana Syahirah Binti Azahari	5.1 Selected RL Algorithm 5.2. State, Action, and Reward Design 5.3. Training Setup and Parameters 5.4. Simulation and Training Process 5.5. Implementation Challenges
Amirah Huda binti Jamalulail Asri	4.1. Controller Selection (P, PI, PID) 4.2. Tuning Method 4.3. Simulation Setup 4.4. Results and Observations 6. Comparison between RL and Classical

	Controller
Ain Maisara bt. Abdullah	1.1. Background 1.2. Problem Statement 1.3. Objectives 2.1. Overview of Classical Controllers (P, PI, PID) 2.2. Reinforcement Learning in Control Systems 7. Graphical User Interface (GUI) 8. Conclusion

#### 10. References

- MathWorks. (2022). *Watertank Simulink Model*. MathWorks.  
<https://www.mathworks.com/help/releases/R2022b/slcontrol/gs/watertank-simulink-model.html>
- MathWorks. (2025). *rlDDPGAgent*. MathWorks.  
<https://www.mathworks.com/help/reinforcement-learning/ref/rl.agent.rlddpgagent.html>