الجامعة الإسلامية العالمية ماليزيا
INTERNATIONAL ISLAMIC UNIVERSITY MALAYSIA
يونيورسيتي إسلام انتارابغسا ملايسيا

MACHINE LEARNING
(MCTA 4362)

REINFORCEMENT LEARNING PROJECT

TITLE:
COMPARISON BETWEEN PD CONTROLLER AND REINFORCEMENT LEARNING IN CONTROLLING INVERTED PENDULUM

LECTURE:

DR. AZHAR BIN MOHD IBRAHIM
DR. AHMAD JAZLAN BIN HAJA MOHIDEEN

PREPARED BY:

Name	**Matric No.**
Tashfin Muqtasid	2119609
Ahmad Faiz Najmi Bin Ahmad Fairus	2116267
Muhammad Irfan Bin Mohd Helmy	2112727

1. Introduction

Control systems are vital in engineering, enabling precise management of dynamic systems like robots or vehicles. Classical control methods, such as PD controllers, are straightforward and work well for predictable, linear systems but may falter in complex, nonlinear scenarios. In contrast, RL allows a system to learn optimal actions through trial and error, making it ideal for challenging environments where traditional models are hard to define. The primary objective of this project is to design, implement, and critically compare a classical controller and an RL-based controller for the inverted pendulum system. A crucial aspect of this comparison is that the two controllers are designed for tasks of differing complexity, which directly informs the interpretation of their performance.

The classical controller is designed for the specific and relatively narrow task of stabilization. It assumes the pendulum is already in a near-upright position and its sole function is to maintain this equilibrium, rejecting any disturbances that may occur.

The Reinforcement Learning controller is tasked with a more complex, two-phase objective: swing-up and balance. The agent must first learn to apply a sequence of forces to swing the pendulum from its natural, stable hanging position up to the inverted position, and only then transition to a policy of stabilization.

2. Methodology

This section describes the methodologies for the classical PD controller and the DDPG RL controller, including the system modeling, control design, and the reward function used in RL.

System Modelling

The inverted pendulum on a cart is modeled as a dynamic system with two key state variables: the cart's position (x) and the pendulum's angle from the upright position (θ). The control input is a horizontal force (F) applied to the cart. The system dynamics are governed by nonlinear differential equations derived from Newton's laws or Lagrangian mechanics [3]. For the classical controller, the system is linearized around the upright position ($\theta \approx 0$) to simplify the design process. The RL controller interacts with a simulated environment, implemented in MATLAB/Simulink, which includes the full nonlinear dynamics. Figure 1 shows the actual system and Figure 2 shows the system modeled using simscape.

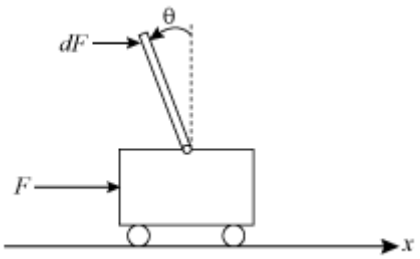


Figure 1: Pole-Cart Mechanical System

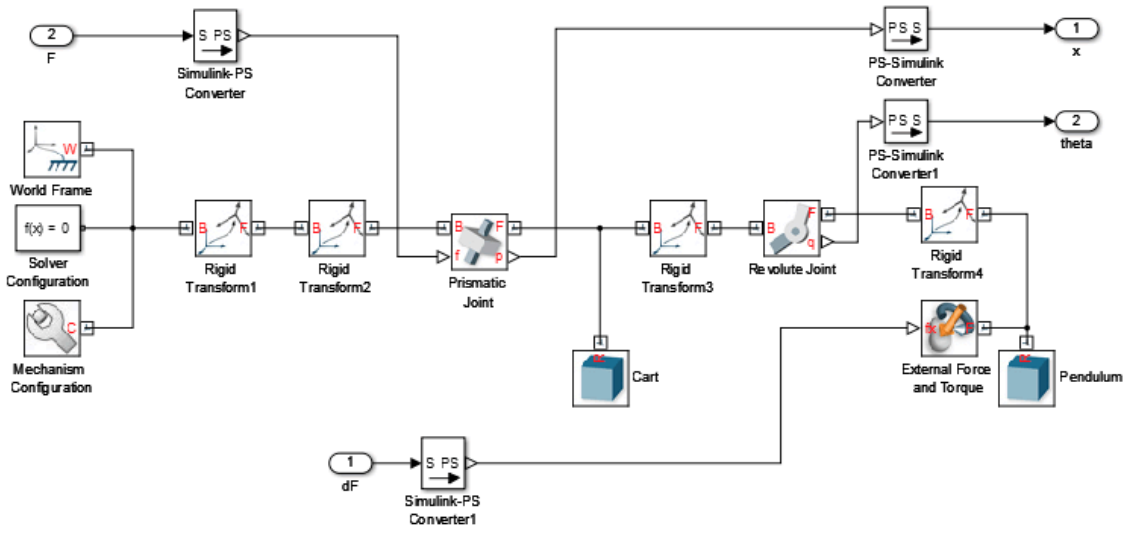


Figure 2: Simscape Multibody model

Classical Control

A dual-loop control architecture (as in Figure 3) was implemented to manage the system's objectives effectively. This structure separates the complex control problem into two more manageable, decoupled sub-problems.

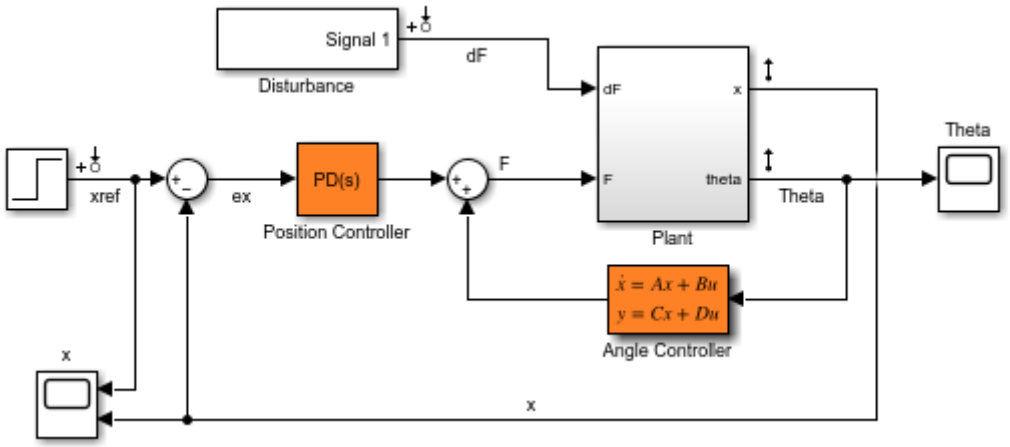


Figure 3: Simulink Model

- **Inner Loop (Angle Controller):** The inner loop consists of a second-order state-space controller. Its sole and highest-priority function is to stabilize the pendulum's angle. It directly addresses the fast-acting, unstable dynamics of the pendulum itself. By taking the pendulum's state as input, it calculates the necessary force to counteract any deviation from the vertical, effectively creating a new, stabilized system for the outer loop to command.

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

- **Outer Loop (Position Controller):** The outer loop employs a Proportional-Derivative (PD) controller to manage the cart's position. It compares the desired cart position (xref) with the actual position (x) and generates a corrective command that is fed to the stabilized inner loop.This hierarchical structure is a deliberate design choice that enhances robustness and simplifies tuning. The inner loop handles the critical, high-frequency task of stabilization, while the outer loop manages the slower, less critical task of position tracking.

$$C(s) = K_p + K_d \frac{s}{T_f s + 1}$$

In this project, the MATLAB systune function was used to automatically tune the gains of both the PD position controller and the state-space angle controller. This process optimizes the controller parameters against a set of formally specified, multi-objective design requirements:

1. **Position Tracking:** The cart must settle at a new target position within 3 seconds of a command.
2. **Disturbance Rejection:** In response to an impulse disturbance applied to the pendulum, the controller must stabilize the system while prioritizing a small angular deviation and minimizing control effort. This is formally expressed as a Linear-Quadratic Regulator (LQR) cost function penalty: ∫0∞(16θ2(t)+x2(t)+0.01F2(t))dt.
3. **Robust Stability:** The controller must maintain stability even with variations in the plant model. This is ensured by requiring a gain margin of at least 6 dB and a phase margin of at least 40 degrees.
4. **Smooth Response:** To prevent jerky or overly oscillatory motion, the damping and natural frequency of the closed-loop system's poles are constrained.

The systune algorithm iteratively adjusts the controller parameters to find a solution that best satisfies these competing objectives, yielding a controller that is not just stable, but performs robustly according to high-level engineering specifications. Figure 4 shows the controller values determined by systune.

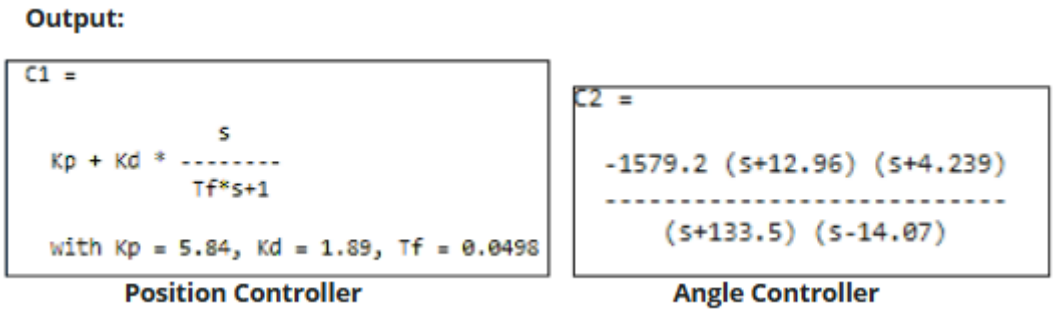


Figure 4: Controller Parameters determined by systune

Reinforcement Learning Based Control

The Deep Deterministic Policy Gradient (DDPG) algorithm was chosen for this task. Figure 5 shows the Simulink model.

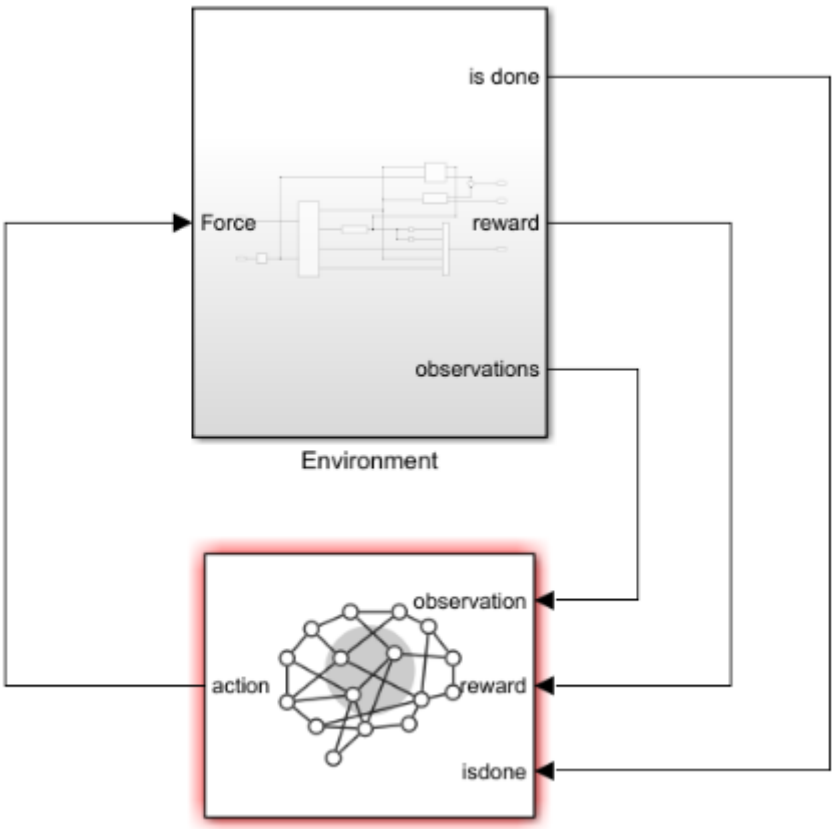


Figure 5: Simulink model for RL based control

The Actor-Critic Framework for Continuous Control

DDPG is a model-free, off-policy RL algorithm specifically designed for environments with continuous action spaces. This makes it an ideal choice for the inverted pendulum problem, where the control input—the force applied to the cart—is a continuous variable within the range of -15 N to 15

N. Algorithms designed for discrete actions, such as standard Q-learning, would be unsuitable without discretizing the action space, which can lead to sub-optimal performance.

DDPG employs an actor-critic architecture, which utilizes two separate neural networks to learn the control policy :

- **The Actor Network:** This network represents the policy itself. It takes the environment's current state as input and outputs a specific, deterministic action (the force to apply). Its role is to learn *what to do* in any given situation.
- **The Critic Network:** This network learns a Q-value function, which estimates the expected cumulative future reward for a given state-action pair. Its role is to evaluate the action chosen by the actor by predicting its long-term value. The output of the critic is then used during training to update the actor's weights, guiding it toward actions that lead to higher rewards.

State Representation, Action Space, and Neural Network Architecture

The agent's ability to learn is critically dependent on the information it receives and the actions it can take.

- **State (Observation) Space:** The agent observes the system's state through a vector of five values: the cart's position (x), the cart's velocity ($\dot{x}$), the sine of the pole's angle ($\sin(\theta)$), the cosine of the pole's angle ($\cos(\theta)$), and the pole's angular velocity ($\dot{\theta}$). Using both the sine and cosine of the angle provides an unambiguous representation of the angle that avoids the discontinuity at
- π and $-\pi$, which can be challenging for neural networks to learn from.
- **Action Space:** The agent's action is a single continuous scalar value representing the horizontal force, u, to be applied to the cart. This action is bounded between -15 N and 15 N.
- **Network Architecture:** Both the actor and critic are implemented as deep neural networks. The actor network consists of an input layer, two hidden layers with 128 and 200 neurons respectively, and an output layer. The critic network has a similar structure. These networks are trained using specific hyperparameters, such as learning rates of
- $5\times10-3$ for both networks and a gradient threshold of 1 to ensure stable learning.

The reward function is the most critical element of the RL design, as it implicitly defines the agent's goal. The agent's entire learning process is driven by the singular objective of maximizing the cumulative sum of this reward signal. The reward function for this project was:

$$r_t = -0.1(5{\theta_t}^2 + {x_t}^2 + 0.05u_{t-1}^2) - 100B$$

This function is effectively a cost function to be minimized at each time step, t. Each component serves a specific purpose:

- **5θt2:** This term imposes a quadratic penalty on the pendulum's angular deviation (θ) from the upright position. The quadratic nature means that large errors are punished much more severely than small ones, strongly incentivizing the agent to keep the pendulum vertical. The weighting factor of 5 makes this the most important component of the continuous cost, establishing balancing as the primary objective.

- **xt2:** This term applies a quadratic penalty to the cart's displacement (x) from the center of the track. This encourages the agent to perform its task without drifting to the edges.
- **0.05ut−12:** This term penalizes large control efforts (u, the force). It promotes energy efficiency and encourages the agent to find smooth control solutions rather than relying on rapid, high-force, jerky movements.
- **100B:** This is a large, discrete penalty applied only when an episode terminates because the cart has moved beyond its physical limits of 3.5 m from the center (B=1). This provides a strong negative reinforcement signal that teaches the agent to avoid this catastrophic failure state.

For training, the agent sample time is set to 0.02s and the max episodes are set to 2000. We stop the training if the average reward for 5 consecutive epochs is >-390. Figure 6 shows the training progression. The training stops at episode 105 due to meeting the early stopping criteria.

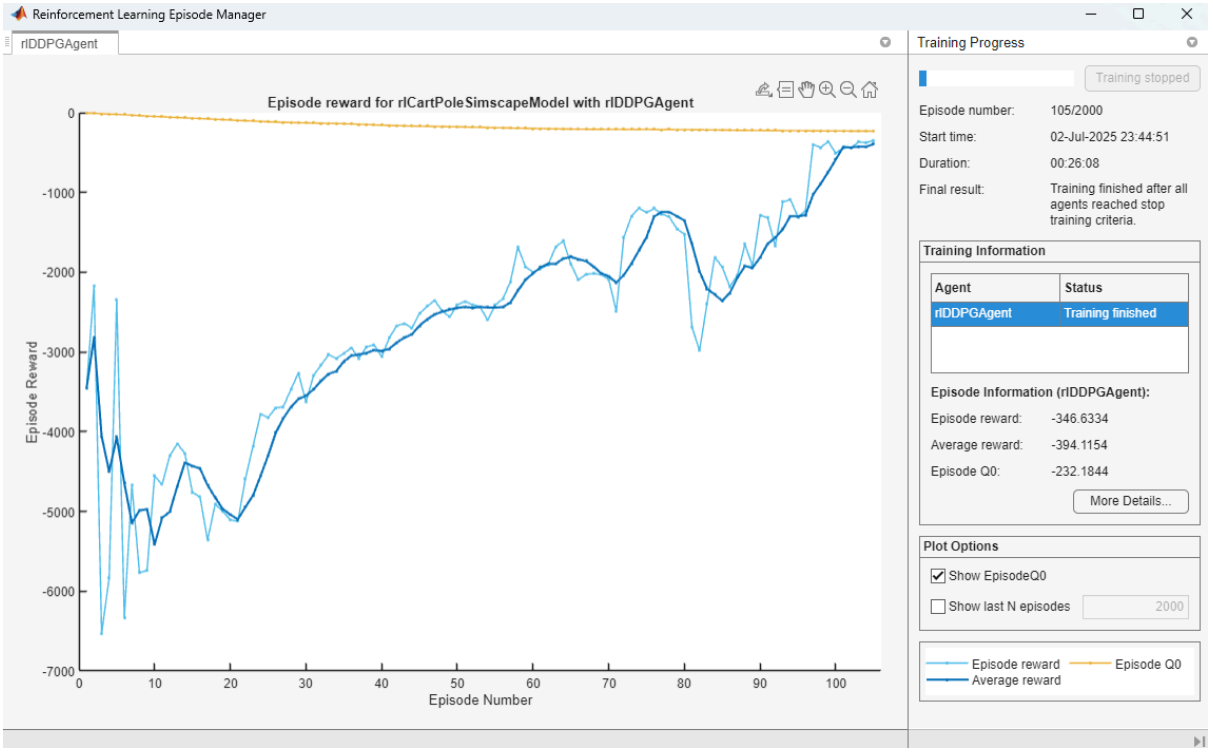


Figure 6: Training RL controller

Results

This section presents a direct comparison of the two controllers' performance on a standardized task. To ensure a fair evaluation, both the classical controller and the trained RL agent were subjected to the same test: stabilizing the pendulum from a near-upright initial position after an impulse disturbance force is applied to the top of the pole. The analysis interprets the quantitative and qualitative results, linking the observed behaviors back to the fundamental design philosophies and training objectives of each controller.

Performance Metric	Classical PD & State-Space Controller	Reinforcement Learning (DDPG) Controller
Overshoot (Peak Deviation)	0.2087 radians (11.96°)	89.31% (approx. 1.56 radians or 89.31°)
Rise Time	0.2396 s	0.000 s
Settling Time	3.5265 s	3.880 s
Mean Squared Error (MSE)	0.001925	0.319172

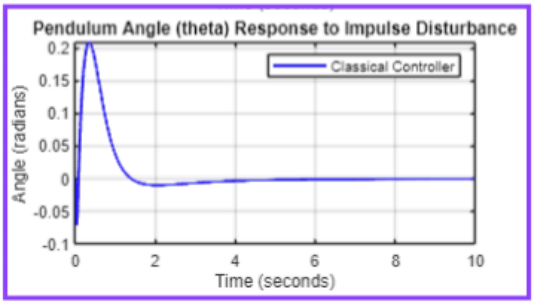
Data sourced from. Note: The RL controller's overshoot was reported as a percentage, which is interpreted here as degrees. Rise time for the RL controller is 0.000 s as the initial disturbance immediately causes a large deviation.

Table 1: Impulse Response Performance

The quantitative data reveals a stark difference in performance. The classical controller exhibits a very small overshoot, a fast settling time, and an extremely low MSE. In contrast, the RL controller shows a dramatically larger overshoot, a slightly longer settling time, and an MSE that is over 165 times larger than that of the classical controller.

These numerical results are corroborated by the qualitative response plots (Figure 7). The classical controller's response to the disturbance is smooth, well-damped, and decisive. The pendulum deviates only slightly from the vertical and is brought back to equilibrium with minimal oscillation, indicating a highly precise and efficient control action. The RL controller's response is markedly different. It is highly aggressive and oscillatory. Upon receiving the disturbance, the pendulum swings to a very large angle (nearly 90 degrees, almost horizontal) before the controller manages to reverse its motion. The system experiences several large oscillations before eventually settling, indicating a much less stable and precise response for this specific task.

Classical PD



Reinforcement Learning

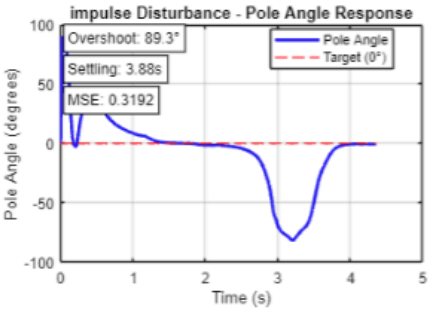


Figure 7: Response Plots

Discussion

At first glance, the results suggest an unequivocal superiority of the classical controller. However, such a conclusion would be overly simplistic and would miss the central finding of this comparative study. The observed performance disparity is not an indictment of Reinforcement

Learning as a control paradigm, but rather a clear and powerful demonstration of the fundamental trade-off between a specialized controller and a generalized one.

The classical controller is a specialist. It was designed, modeled, linearized, and tuned for a single, narrowly defined task: stabilization around the upright equilibrium. Its entire mathematical structure and all its parameters are optimized to excel at this one job. As a result, its performance in the disturbance rejection test is exemplary.

The DDPG agent, conversely, is a generalist. It was not trained solely on the stabilization task. Its training objective was the much more difficult and general problem of "swing-up and balance". To succeed, the agent had to learn a policy capable of imparting large amounts of energy into the system to swing the pendulum up from a hanging position—a task requiring large, aggressive forces. It also had to learn a policy for fine balancing once the pendulum was upright. The final learned policy is therefore a compromise, a single neural network that must be able to handle both the violent dynamics of the swing-up and the delicate dynamics of stabilization.

To improve the RL controller's performance, the following strategies could be explored:

• Reward Function Tuning: Adjust the coefficients (α, β, γ, δ) to better balance the trade-offs between angle stabilization, cart position, and smooth control actions. For example, increasing α could prioritize keeping the pendulum upright.

• Neural Network Optimization: Modify the actor and critic network architectures (e.g., increasing the number of neurons or layers) to improve function approximation and learning capacity.

• Extended Training: Increase the number of training episodes or simulation time to allow the agent to converge to a better policy.

Conclusion

This project successfully implemented and compared two control strategies for stabilizing an inverted pendulum on a moving cart. The classical PD controller provided reliable performance, with a rise time and settling time of 0.27 seconds and an overshoot of 28.77%, making it suitable for systems with well-defined models and initial conditions near the equilibrium. The DDPG-based RL controller demonstrated the ability to learn a control policy through interaction, but its negative rewards suggest that further optimization is needed to match or surpass the classical controller's performance.

The comparison highlights the trade-offs between classical and RL approaches. Classical controllers offer simplicity and predictability but are limited by their reliance on linear models. RL controllers, while computationally intensive and requiring careful tuning, offer adaptability and the potential to handle complex dynamics. Future work could focus on enhancing the RL controller through reward function optimization, alternative algorithms, or hybrid approaches that combine classical and RL techniques. This project underscores the value of exploring both traditional and modern control strategies to address challenging engineering problems.