ISSN 1392-1207. MECHANIKA. Vol. 31, No. 2, 2025: 123–135

Real-Time Swing-up of a Linear Inverted Pendulum Using Reinforcement Learning

Xhevahir BAJRAMI*, Fisnik KAÇIU*, Erjon SHALA, Rame LIKAJ***
**Department of Mechatronics, Faculty of Mechanical Engineering, University of Pristina, 10000 Prishtina, Kosovo*
***Department of Mechatronics, Faculty of Mechanical Engineering, University of Pristina, 10000 Prishtina, Kosovo,*
E-mail: erjon.shala@uni-pr.edu (Corresponding Author)

https://doi.org/10.5755/j02.mech.39202

1. Introduction

Inverted pendulum systems are widely used in control engineering education and as benchmarks for real-world control challenges due to their unstable, nonlinear dynamics [1, 2]. Applications include humanoid robots, wheeled robots [3], and rocket launching. Controlling triple inverted pendulums (TIPs) is particularly complex, often validated through simulations due to the difficulty of real-world experiments [4]. Traditional TIP control relies on mathematical models and trajectory tracking but is prone to errors from model mismatches and unmodeled dynamics, particularly in complex environments. Reinforcement learning (RL) offers a data-driven alternative, bypassing explicit modeling to enable real-time control. Advances in deep RL have demonstrated success in autonomous systems and continuous action spaces [5]. The rapid advancement of Artificial Intelligence (AI) has enabled intelligent control algorithms to meet growing demands for high-performance systems. Among these, Reinforcement Learning (RL) excels in solving complex control problems in dynamic environments, particularly when system dynamics are unknown or uncertain. This study explores Deep Reinforcement Learning (DRL) for real-time control of a linear inverted pendulum, focusing on the Deep Deterministic Policy Gradient (DDPG) algorithm for its ability to handle continuous action spaces critical for dynamic systems [6]. Reinforcement learning (RL) has been applied to rotary inverted pendulum control, demonstrating superior flexibility and robustness compared to traditional LQR and MPC controllers. Developed in MATLAB/Simulink and implemented on Raspberry Pi, the RL controller using Proximal Policy Optimization (PPO) showed potential for complex, nonlinear systems despite challenges in real-world deployment. An RL framework for education using Lucas-Nulle hardware addressed pendulum swing-up and stabilization, with RL computations outsourced and communication via CAN bus. A safeguarding algorithm ensured safe training, highlighting RL's effectiveness in trial-and-error learning. The primary objectives of this research are as follows:

1. To conduct a comprehensive literature review on DRL algorithms, focusing on selecting the most appropriate method for control applications.
2. To develop expertise in using the MATLAB Reinforcement Learning Toolbox, particularly in the implementation of the DDPG algorithm.
3. To design and execute an effective control system for both linear and nonlinear systems, using the Quanser IP02 Inverted Pendulum system as a case study.

The paper is organized as follows: The introduction provides the context and objectives of the research. Subsequent sections discuss the theoretical foundations of Reinforcement Learning and Deep Learning, and their application in control systems using MATLAB. Detailed implementation of the DDPG algorithm, along with the experimental setup, is thoroughly presented. The paper concludes with an analysis of the results and a discussion of the contributions and future directions of the research.

2. Reinforcement Learning and Fundamentals of Control Theory

Control theory plays a crucial role in engineering by focusing on how input variables can influence the behaviour of dynamic systems through controlled inputs. These systems can be broadly categorized into linear systems and nonlinear systems, both of which form the foundation of automatic control systems. Linear systems adhere to the principles of proportionality and additivity, whereas nonlinear systems exhibit more complex behaviour and require specialized control approaches [7]. Nonlinear systems deviate from the principle of superposition, meaning their response is not directly proportional to the input. The equations that describe nonlinear systems are of the form:

$$\dot{x}(t)=f\left[x(t),u(t)\right],\quad y(t)=g\left[x(t),u(t)\right], \tag{1}$$

where *f* and *g* are nonlinear functions. These systems often display intricate behaviours and require advanced control techniques to manage them efficiently.

2.1. Reinforcement learning for optimal control applications

Reinforcement Learning (RL) offers a model-free framework for addressing optimal control problems in both linear and nonlinear systems. In an RL environment, an agent interacts with the environment and learns the optimal policy by maximizing rewards. Fig. 1 illustrates the agent-environment interaction cycle in RL [8-10].

In Reinforcement Learning, the agent seeks to

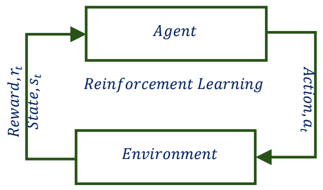


Fig. 1 Agent-environment interaction cycle in RL

maximize the cumulative reward through cycles of trial and error. This interaction is based on several key concepts [8-10].

2.2. Trial and error: learning

The agent interacts with the environment to select actions that result in high rewards, continuously refining its policy based on the experience it gains. Fig. 2 illustrates the process of learning through trial and error.

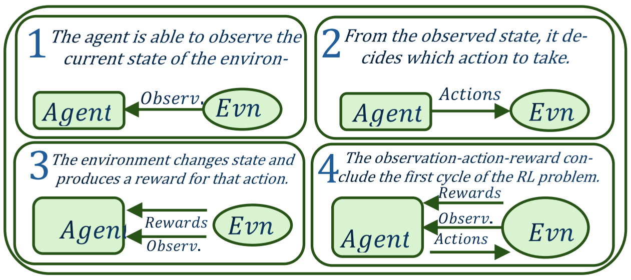

1 The agent is able to observe the current state of the environ-
Agent
Observ.
Evn
2 From the observed state, it decides which action to take.
Agent
Actions
Evn
3 The environment changes state and produces a reward for that action.
Agen
Rewards
Observ.
Evn
4 The observation-action-reward conclude the first cycle of the RL problem.
Rewards
Observ.
Actions
Agent
Evn

Fig. 2 Trial-and-error cycle in reinforcement learning

The return G_t represents the total expected reward an agent receives over a time sequence and is defined as:

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \gamma \in [0,1], \quad (2)$$

where γ is the discount factor, which determines the importance of future rewards.

The value function $V^{\pi}(s_t)$ gives the expected return when the agent follows policy π from state s_t.

$$V^{\pi}(s_t) = E_{\pi}\left[G_t \middle| s_t\right]. \quad (3)$$

The Bellman equation is a fundamental concept in RL, which links the value of a state to the values of future states.

$$V^{\pi}(s_t) = E_{\pi}\left[r_{t+1} + \gamma V^{\pi}(s_{t+1})\right]. \quad (4)$$

This equation helps the agent to improve its policy by predicting future rewards [14]. Expectation $E[X]$ is defined as the weighted sum of all possible outcomes multiplied by their probabilities.

$$E[X] = \sum_{x \in X} xP(x). \quad (5)$$

2.3. Exploration vs. exploitation

In RL, there is a trade-off between exploring new actions and exploiting known policies that are known to yield high rewards. The ε - *greedy* algorithm is one approach used to balance this trade-off as described by Eq. (6) and shown in Fig. 3.

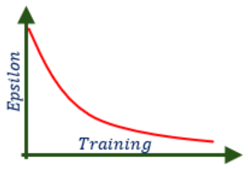


Fig. 3 Epsilon behaviour during training in ε - *greedy* algorithm

$$a_t = \begin{cases} optimal \quad action, \quad with \quad probability 1-\varepsilon \\ random \quad action, \quad with \quad probability\, \varepsilon \end{cases}. \quad (6)$$

2.4. Reinforcement learning algorithms

This section introduces four foundational RL algorithms: Policy Iteration, Value Iteration, SARSA, and Q-Learning. Each algorithm offers different approaches to learning optimal policies and value functions. In Algorithm 1, policy iteration is an algorithm that iteratively evaluates and improves a policy until it converges to the optimal policy. It consists of two main steps: policy evaluation and policy improvement.

Algorithm 1: Policy Iteration Algorithm

```
Data: $\theta$ a small number
Result: $V:$ a value function s.t. $V \approx v_*, \pi:$ a deterministic policy s.t. $\pi_* \approx \pi:$
  Function Policy Iteration is
     / Initialization
     Initialize $V(s)$ arbitrarily;
     Randomly initialize policy $\pi(s)$;
     / Policy Evaluation
     $\Delta \leftarrow 0$;
     while $\Delta < 0$ do
        for each $s \in S$ do
           $v \leftarrow V(s)$;
           $V(s) \leftarrow \sum_{\dot{s},\dot{r}} p(\dot{s},\dot{r} \mid s,\pi(s))[r+\gamma V(\dot{s})]$;
           $\Delta \leftarrow max(\Delta, | v - BV(s) |)$;
        end
     end
     / Policy Improvement
     policy-stable $\leftarrow true$;
     for each $s \in S$ do
        old-action $\leftarrow \pi(s)$;
       $\pi(s) \leftarrow arg \quad \max_a \sum_{\dot{s},\dot{r}} p(\dot{s},\dot{r} \mid s,a)[r+\gamma V(\dot{s})]$;
        if old-action $!= \pi(s)$ then
           policy-stable $\leftarrow$ $false$;
        end
     end
     if $policy-stable$ then
        return $V \approx v_*$ $V \approx v_*$ and $\pi \approx \pi_*$;
     else
        go to Policy Evaluation;
end
end
```

In Algorithm 2, value iteration combines policy evaluation and policy improvement into a single step, iteratively updating the value function until convergence to the optimal value function V^*.

In Algorithm 3, SARSA (State-Action-Reward-State-Action) is an on-policy Temporal Difference (TD) control algorithm that updates the action-value function

$Q(s, a)$ based on the action actually taken by the current policy.

Algorithm 2: Value Iteration Algorithm

```
Data: $\theta$ a small number
Result: $\pi$ : a deterministic policy s.t. $\pi \approx \pi_*$
Function Value Iteration is
  / Initialization
  Initialize $V(s)$ arbitrarily, except $V(terminal)$;
  $V(terminal) \leftarrow 0$;
  / Loop until convergence
  $\Delta \leftarrow 0$;
    while $\Delta < 0$ do
        for each $s \in S$ do
        $v \leftarrow V(s)$;
        $V(s) \leftarrow \max_a \sum_{\dot{s},\dot{r}} p(\dot{s},\dot{r} \mid s,a)\left[r+\gamma V(\dot{s})\right]$;
        $\Delta \leftarrow max(\Delta, |v-V(s)|)$;
        end
    end
  / Return optimal policy
    return $\pi$ s.t.
      $\pi(s) \leftarrow arg \quad \max_a \sum_{\dot{s},\dot{r}} p(\dot{s},\dot{r} \mid s,a)\left[r+\gamma V(\dot{s})\right]$;
end
```

Algorithm 3: SARSA (on-policy TD control) for estimating $\pi \approx \pi_*$

```
Algorithm parameters: step size $\alpha \in (0,1)$, small $epsilon > 0$;
Initialize $Q(s_t, a_t)$ for all $s_t \in S, a_t \in A)$, arbitrarily except that $Q(terminal, \cdot) = 0$;
For each episode:
    Initialize $s_t$;
        For each step of episode:
            Choose $a_t$ from $s_t$ using policy derived from
            $Q(s_t, a_t(e.g.,-greedy)$;
            Take action at, observe $r_{t+1}, s_{t+1}$;
            $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1}, +$
            $+\gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]$;
            $s_t \leftarrow s_{t+1}$;
            $a_t \leftarrow a_{t+1}$;
        end;
end
```

- On-Policy: Learns the value of the policy being carried out.
- TD Control: Combines ideas from Monte Carlo methods and dynamic programming.

In Algorithm 4, Q-Learning is an off-policy Temporal Difference (TD) control algorithm that learns the optimal action-value function $Q(s, a)$ independently of the agent's policy. It updates $Q(s, a)$ based on the maximum estimated future rewards, promoting exploration of all actions.

- Off-Policy learns the value of the optimal policy independently of the agent's actions.
- TD Control utilizes the Bellman equation for updates [11].

Algorithm 4: Q-learning (off-policy TD control) for estimating $\pi \approx \pi_*$

```
Algorithm parameters: step size $\alpha \in (0,1)$, small $epsilon > 0$;
Initialize $Q(s_t, a_t)$ for all $s_t \in S, a_t \in A$, arbitrarily except that $Q(terminal, \cdot) = 0$;
For each episode:
    Initialize $s_t$;
        For each step of episode:
            Choose $a_t$ from $s_t$ using policy derived from
            $Q(s_t, a_t(e.g.,-greedy)$;
            Take action at, observe $r_{t+1}, s_{t+1}$;
            $Q(s_t, a_t) \leftarrow Q(s_t, a_t) +$
            $+\alpha\left[r_{t+1}, +\gamma \max_a Q(s_t, a) - Q(s_t, a_t)\right]$;
            $s_t \leftarrow s_{t+1}$;
        end
end
```

2.5. Model based and model free reinforcement learning

Reinforcement Learning approaches can be categorized into model-based and model-free methods. Involves learning a model of the environment's dynamics (i.e., transition probabilities and reward function) to plan future actions. This approach can be more sample-efficient but may struggle with model inaccuracies. Learns optimal policies or value functions directly from interactions without requiring a model of the environment. Algorithms like Q-Learning and SARSA fall into this category. While generally less sample-efficient, they are more flexible in complex or unknown environments [12, 13].

2.6. Limitations of reinforcement learning

Despite its powerful capabilities, Reinforcement Learning faces several challenges. High Sample Complexity: Requires a large number of interactions with the environment to learn effective policies. Scalability Issues: Struggles with large state-action spaces, leading to increased computational demands. Balancing the need to explore new actions with exploiting known rewarding actions remains a persistent challenge. Stability and Convergence: Ensuring stable learning and convergence to optimal policies, especially in non-stationary or dynamic environments, is difficult. Addressing these limitations is an active area of research, with ongoing efforts to develop more efficient, scalable, and robust RL algorithms.

3. Deep Reinforcement Learning (DRL)

Deep Reinforcement Learning (DRL) represents a significant advancement over traditional RL algorithms, particularly in addressing challenges associated with large and complex state and action spaces. Traditional RL approaches often face difficulties in environments with nonlinear dynamics or where the state-action space is too vast to be modelled effectively. DRL overcomes these limitations by using deep neural networks to approximate policies and value functions, which enables learning in more complex environments. DRL systems excel in processing high-dimensional input data, such as images or sensor readings, which are often encountered in real-world applications. By incorporating deep learning techniques, DRL algorithms can autonomously discover optimal feature representations that enhance decision-making capabilities. This ability to generalize from large amounts of data makes DRL particularly effective in environments where traditional RL methods fail [9, 14].

3.1. Foundations of deep learning

Deep learning uses multi-layered neural networks, also known as deep neural networks (DNNs), to model complex relationships in data. Unlike shallow neural networks, which consist of a few layers, deep networks contain numerous layers, allowing them to capture hierarchical patterns in the input data. Each layer learns increasingly abstract features, making DNNs powerful tools for tasks such as image recognition, language processing, and, in the context of DRL, policy learning [15]. In a DRL framework, these networks function as approximators for both value functions and policies. For instance, in actor-critic architectures, the actor network maps the state observations to actions, while the critic network evaluates the quality of these actions by estimating the value function. Fig. 4 shows an example of a basic neural network.

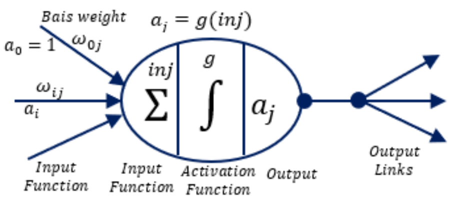


Fig. 4 A basic neuron model

3.2. Policy gradient algorithms

Policy gradient algorithms are a class of RL methods where the policy is directly parameterized and optimized using gradient-based techniques. These algorithms seek to optimize the policy by adjusting the policy's parameters to maximize the expected cumulative reward. Unlike value-based methods, which derive the policy from a value function, policy gradients can learn both stochastic and deterministic policies directly. One of the major benefits of policy gradient methods is their ability to handle continuous action spaces. This makes them ideal for control tasks, such as robotic manipulation, where actions are continuous and need to be fine-tuned.

3.3. Deriving the policy gradient

The Policy Gradient Theorem provides a mechanism for calculating the gradient of the expected reward with respect to the policy parameters.

$$\nabla_\theta J(\theta) = E_\pi\left[\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a)\right], \tag{7}$$

here, $J(\theta)$ represents the performance objective, $\pi_\theta(a|s)$ is the policy parameterized by θ, and $Q^\pi(s, a)$ is the action-value function. This gradient is used to adjust the policy parameters in a way that increases the expected reward [16].

3.4. Deep *Q*-learning

Deep *Q*-learning integrates the *Q*-learning algorithm with deep neural networks to address environments with large state spaces. In this approach, the *Q*-function, which estimates the value of taking a specific action in a particular state, is approximated using a deep neural network. The update rule for the *Q*-network is based on the Bellman equation, defined as:

$$Q(s_t, a_t) = r_t + \gamma_a \max \cdot Q(s_{t+1}, a), \tag{8}$$

where r_t is the reward at time step t, and γ is the discount factor, which adjusts the weight of future rewards relative to immediate ones.

3.5. Deep deterministic policy gradient (DDPG)

The Deep Deterministic Policy Gradient (DDPG) algorithm builds upon both Deep *Q*-learning and Deterministic Policy Gradient (DPG) methods. It is specifically designed to address environments with continuous action spaces, making it ideal for control tasks that require precise action selection. DDPG operates using an actor-critic architecture, where the actor network learns the optimal policy by mapping states to actions, and the critic network evaluates these actions by estimating the *Q*-value. The architecture of the actor-critic network follows five steps, as illustrated below.

Steps in the Actor-Critic Network Architecture:

1. State Observation, the agent observes the current state s_t from the environment and feeds this information to both the actor and critic networks.
2. Action Selection by Actor, the actor network processes the state and selects an action a_t. This action is typically continuous in DDPG.
3. Action Evaluation by Critic, the critic network takes both the current state s_t and the selected action a_t as inputs. The critic evaluates the action by estimating the *Q*-value $Q(s_t, a_t)$, which represents the expected return of taking action a_t in state s_t.
4. Reward and Next State, after taking action a_t, the agent receives a reward r_t from the environment and transitions to the next state s_{t+1}.
5. Updating the Actor and Critic, the critic network is updated by minimizing the temporal difference error, defined as:

$$L = \left(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right)^2. \tag{9}$$

The actor network is updated using the policy gradient to maximize the *Q*-value estimated by the critic, i.e., improving the policy based on the critic's feedback. Fig. 5 illustrates the actor-critic network and the interaction between the actor, critic, and the environment, detailing these steps [8, 17, 18].

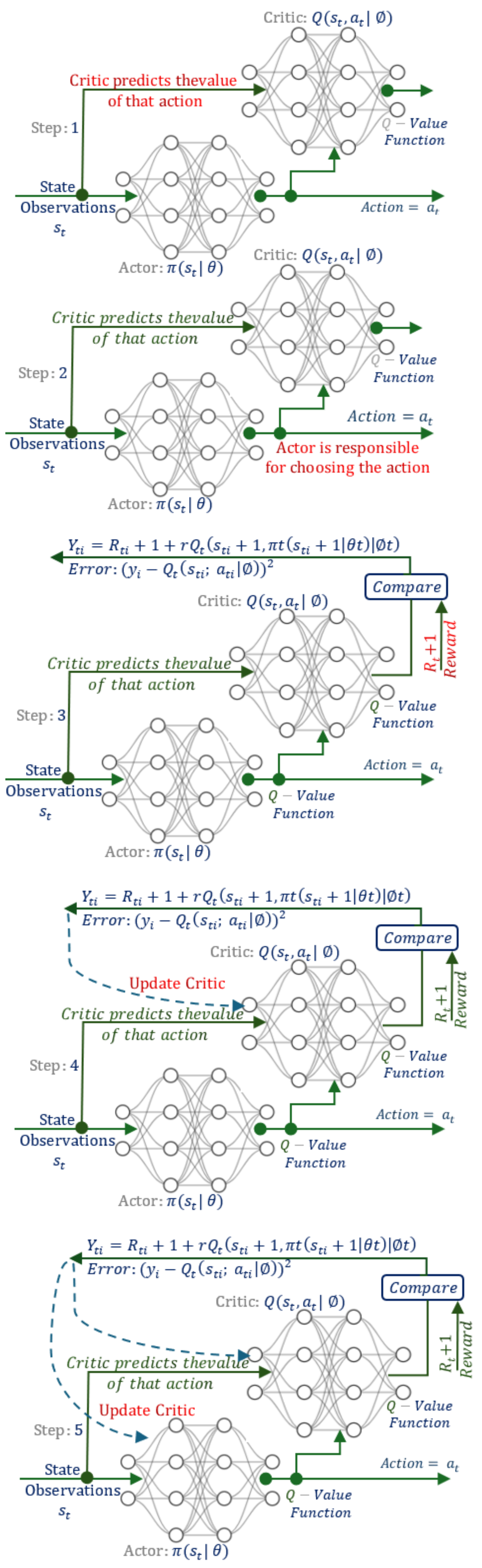

Critic: $Q(s_t, a_t | \emptyset)$
Critic predicts thevalue of that action
Step: 1
State Observations s_t
Q – Value Function
Action = a_t
Actor: $\pi(s_t | \theta)$
Critic predicts thevalue of that action
Step: 2
Actor is responsible for choosing the action
$Y_{ti} = R_{ti} + 1 + rQ_t(s_{ti} + 1, \pi t(s_{ti} + 1|\theta t)|\emptyset t)$
Error: $(y_i - Q_t(s_{ti};\ a_{ti}|\emptyset))^2$
Compare
$R_t + 1$ Reward
Step: 3
Update Critic
Step: 4
Step: 5

Fig. 5 Actor critic network architecture in DDPG (steps 1-5)

3.6. Discussion on DRL

Deep Reinforcement Learning, particularly DDPG, has proven to be highly effective for solving complex control problems involving continuous action spaces. The combination of policy gradient methods and *Q*-learning within the actor-critic framework provides a powerful mechanism for learning optimal policies in environments where traditional methods struggle. This makes DDPG particularly useful for real-time control tasks, such as robotics and other dynamic systems. In future developments, efforts to improve the stability and efficiency of DDPG could enhance its applicability in real-world systems where precision and reliability are critical.

4. Implementation of DDPG Using RL MATLAB Toolbox

The MATLAB Reinforcement Learning Toolbox provides a powerful framework for developing and training reinforcement learning (RL) algorithms, such as Deep Deterministic Policy Gradient (DDPG). This section details the implementation of the DDPG algorithm for controlling the Quanser IP02 system, a commonly utilized platform for testing control algorithms in dynamic environments. The Quanser Single Inverted Pendulum (SIP) system is employed as an example to demonstrate the application of reinforcement learning in managing complex nonlinear systems [19].

4.1. Problem formulation

The primary goal of this control problem is to balance the inverted pendulum at its unstable equilibrium position by controlling the cart’s movement. The system state consists of the cart's position *x*, the pendulum's angle α, and their respective velocities $\dot{x}$ and $\dot{\alpha}$. The control input is the voltage V_m applied to the DC motor driving the cart. The observation of the system at any time *t* can be represented as:

$$y_t = y_{t_{meas}} = [x \quad \alpha \quad \dot{x} \quad \dot{\alpha}]. \tag{10}$$

The agent interacts with the environment by selecting actions u_t, which represent the voltage applied to the motor, to control the system [20]. The system's dynamic environment is modelled using Simulink, where the nonlinear

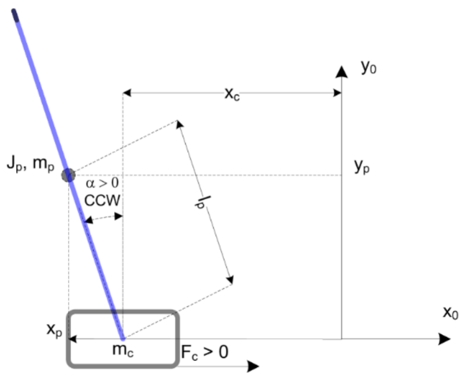


Fig. 6 Linear schematic of the inverted pendulum system, showing the cart position x_c and the pendulum angle α [14]

equations of motion are utilized to represent the system's behaviour. The pendulum's dynamics are obtained through the Lagrangian approach, with the input being the voltage applied to the DC motor. Due to the system's nonlinear nature, a visual representation of the linearized model is provided in Fig. 6.

4.2. Equations of motion for linear inverted pendulum

The equations of motion for the cart and pendulum system are derived from the Lagrangian mechanics. The motion of the cart can be described by:

$$\left(M + M_p + \frac{J_m K_g^2}{r_{mp}^2}\right)\ddot{x} + \left(M_p l_p \sin\dot{\alpha}^2 - \right.$$
$$\left. - M_p l_p \cos\ddot{\alpha}\right)\alpha = F_c - B_{eq}\dot{x}, \tag{11}$$

where:
- M is the mass of the cart,
- M_p is the mass of the pendulum,
- l_p is the length of the pendulum,
- J_m is the moment of inertia of the motor,
- K_g is the gearbox ratio,
- r_{mp} is the radius of the motor pinion,
- F_c is the control force,
- B_{eq} is the equivalent viscous damping coefficient of the motor pinion.

The pendulum’s equation of motion is:

$$-M_p l_p \cos\alpha \cdot \ddot{x} + \frac{4}{3} M_p l_p\, g \cdot \sin\alpha = -B_{eq}\dot{\alpha}\,, \tag{12}$$

where B_{eq} is the viscous damping coefficient of the pendulum.

4.3. Nonlinear equations of motion

By solving the system's Eqs. (11) and (12), the nonlinear expressions for the second-order derivatives $\ddot{x}$ and $\ddot{\alpha}$ can be written as:

for the cart

$$\ddot{x} = \frac{3r_{mp}^2 \cos\alpha \cdot \dot{\alpha}}{l_p D(\alpha)} - \frac{4M_p l_p r_{mp}^2 \sin\alpha \cdot \dot{\alpha}^2}{D(\alpha)} - \frac{3r_{mp}^2 B_{eq} \cdot \dot{x}}{D(\alpha)} -$$
$$- \frac{3M_p r_{mp}^2\, g \cos\alpha \cdot \sin\alpha}{D(\alpha)} + \frac{3r_{mp}^2 F_c}{D(\alpha)}, \tag{13}$$

for the pendulum

$$\ddot{\alpha} = -\frac{3\left(Mr_{mp}^2 + M_p r_{mp}^2 + J_m K_g^2\right) B_p \cdot \dot{\alpha}}{M_p l_l^2 D(\alpha)} - 3M_p r_{mp}^2 \cdot$$
$$\cdot \frac{\cos\alpha \cdot \sin\alpha\dot{\alpha}^2}{D(\alpha)} - \frac{3r_{mp}^2 B_{eq} \cos\alpha\dot{x}}{l_p D(\alpha)} + 3\left(M_p r_{mp}^2 + \right.$$
$$\left. + J_m K_g^2\right) \cdot \frac{g \sin\alpha}{l_p D(\alpha)} + \frac{3r_{mp}^2 \cos\alpha F_c}{l_p D(\alpha)}, \tag{14}$$

where $D(\alpha)$ is a nonlinear expression defined as:

$$D(\alpha) = 4Mr_{mp}^2 + M_p r_{mp}^2 4J_m K_g^2 + 3M_p r_{mp}^2 \sin^2\alpha\,. \tag{15}$$

4.4. System modelling and linearization

To facilitate the design of the control system, the nonlinear dynamics of the Quanser Single Inverted Pendulum are simplified through linearization around the upright equilibrium, where the pendulum angle $\alpha = 0$ and the cart position $x = 0$. This approach allows us to approximate the system as a linear state-space model, making it more suitable for applying linear control strategies. The linearized state-space model is expressed as.

$$\frac{d}{dt} X(t) = A(X(t)) + B(X(t))u(t). \tag{16}$$

In this equation, the state vector $X(t) = [x(t), \alpha(t), \dot{x}(t), \dot{\alpha}(t)]$ contains the cart position $x(t)$ pendulum angle, $\alpha(t)$ cart velocity $\dot{x}(t)$, and pendulum angular velocity $\dot{\alpha}(t)$. The control input $u(t)$ represents the voltage V_m applied to the DC motor, which drives the cart's movement. The matrices A and B, derived from the system's physical parameters, describe how the system's state evolves over time in response to the applied control input. These parameters include the masses of the cart and pendulum, the pendulum length, and the characteristics of the DC motor that generates the force moving the cart.

4.5. DC motor electrical model

In the system, the input force F_c is translated into an input voltage V_m, which powers the DC motor responsible for moving the cart. Fig. 7 presents the electrical schematic of the DC motor, showing the key components such as the motor armature and the voltage source that drives the system.

The dynamics of the motor are modelled using the following equation

$$V_m = L\left(\frac{d}{dt} i_a\right) + Ri_a + k_e\dot{\theta}_m = 0\,. \tag{17}$$

In this model, V_m is the voltage applied to the motor, i_a represents the armature current, R and L are the motor's resistance and inductance, respectively, k_e is the back EMF constant, and $\dot{\theta}_m$ is the motor's angular velocity.

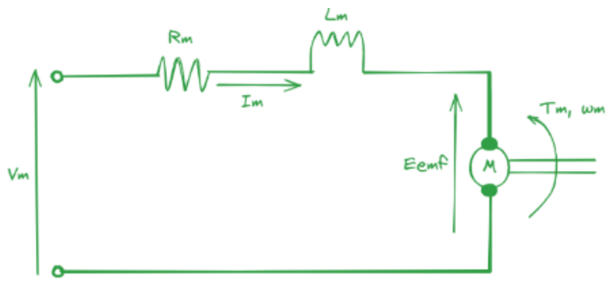


Fig. 7 Electrical schematic of the standard DC motor used to control the cart's movement

4.6. Linearization around the equilibrium

Once the nonlinear model is derived, it is linearized around the system's unstable equilibrium point, where the pendulum is in the vertical position. This is accomplished by performing a Taylor series expansion around $\alpha = 0$ and keeping only the linear terms. The reference frame transformation for the pendulum is shown in Fig. 8, where the vertical position corresponds to $\alpha = 0$ radians. This shift in reference frame simplifies the control system design process, allowing for easier stabilization of the pendulum in the upright position.

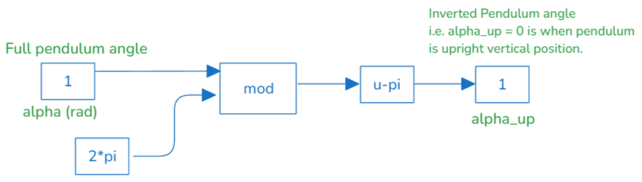


Fig. 8 Conversion of the reference frame, making the vertical upright position of the pendulum correspond to $\alpha = 0$, rad

By approximating the system dynamics around this equilibrium point, linear control strategies, such as state feedback or the Linear Quadratic Regulator (LQR), can be applied to effectively stabilize the pendulum and manage the cart's motion.

4.7. Input voltage conversion

In this implementation, the control input force F_c needs to be converted to the voltage V_m applied to the motor. The relation between the control force and the motor voltage is derived from the electrical model of the DC motor, represented by the following equations. From Kirchhoff's voltage law

$$V_m - R_m I_m - L_m\left(\frac{d}{dt} I_m\right) - E_{emf} = 0 . \tag{18}$$

Neglecting the inductance L_m and using the relation for the back electromotive force $E_{emf} = K_m \omega_m$, the input force can be expressed as:

$$F_c = -\frac{K_g K_t K_m \dot{x}}{R_m r_{mp}^2} + \frac{K_g K_t V_m}{R_m r_{mp}^2} . \tag{19}$$

Substituting this into the equations of motion yields the final nonlinear equations governing the system's dynamics

$$\ddot{x} = \frac{3r_{mp}^2\, B_p \cos\alpha \cdot \dot{\alpha}}{l_p D(\alpha)} - \frac{3M_p r_{mp}^2 \cos\alpha \sin\alpha \cdot \dot{\alpha}^2}{D(\alpha)} - \frac{3\left(M_m r_{mp}^2 B_{eq} + K_g^2 K_t K_m\right)\dot{x}}{R_m D(\alpha)} + \\ +3\left(M r_{mp}^2 + M_p r_{mp}^2 + J_m K_g^2\right) \cdot \frac{g \sin\alpha}{l_p D(\alpha)} + \frac{3r_{mp}^2 K_g K_t \cos\alpha V_m}{R_m l_p D(\alpha)} . \tag{20}$$

For the pendulum

$$\ddot{\alpha} = -\frac{3\left(M r_{mp}^2 + M_p r_{mp}^2 + J_m K_g^2\right) B_p \cdot \dot{\alpha}}{M_p l_l^2 D(\alpha)} - 3M_p r_{mp}^2 \cdot \frac{\cos\alpha \sin\alpha \dot{\alpha}^2}{D(\alpha)} - \frac{3\left(M_m r_{mp}^2 B_{eq} + K_g^2 K_t K_m\right)\cos\alpha \dot{x}}{R_m l_p D(\alpha)} + \\ + \frac{3\left(M r_{mp}^2 + M_p r_{mp}^2 + J_m K_g^2\right) g \sin\alpha}{l_p D(\alpha)} + \frac{3r_{mp}^2\, K_g K_t \cos\alpha V_m}{R_m l_p D(\alpha)} . \tag{21}$$

4.8. Reward signal design

The reward function is designed to motivate the agent to reduce deviations from the pendulum's target upright position. This reward is represented by a quadratic function, defined as:

$$R_{t+1} = -\left(Q_{11} x_t^2 + Q_{22} x_t^2 + Q_{33} \dot{x}_t^2 + Q_{44} \dot{\alpha}_t^2 + Q_{11} u_t^2\right) . \tag{22}$$

This reward function penalizes errors in both position and velocity, as well as overly large control inputs, to promote smooth and stable system performance.

4.9. Agent training and hyperparameters

The architecture and hyperparameters of the DDPG agent play a crucial role in achieving effective control. Table 1 outlines the main physical parameters of the Quanser IP02 system, which were utilized during both simulation and RL agent training [21, 22].

The DDPG agent was trained using fully connected neural networks for both the actor and critic. Each network consists of three hidden layers, with 200 neurons per layer and ReLU activation functions. The important hyperparameters for training are presented in Table 2.

Fig. 9 presents the Simulink model used for training the reinforcement learning agent, based on a nonlinear representation of the Quanser Single Inverted Pendulum (SIP) system mounted on a Linear Cart IP02 [14]. The model integrates an RL Agent block that processes observation and reward signals to guide decision-making, generating motor voltage actions to stabilize the pendulum. This setup effectively facilitated the training of the DDPG agent for stable inverted pendulum control.

4.10. System performance

After the agent is fully trained, its performance is assessed through simulations using the Simulink model. The evaluation focuses on the pendulum's behaviour over time and the voltage applied to the motor. The overall control structure is illustrated in Fig. 10. The pendulum's angle α

over time and the corresponding motor voltage V_m are presented in Fig. 11 and Fig. 12.

This section explains the application of the Deep Deterministic Policy Gradient (DDPG) algorithm to control the Quanser IP02 system using MATLAB's Reinforcement Learning Toolbox. The nonlinear behaviour of the inverted pendulum was modelled and linearized for control purposes. The reinforcement learning agent was trained to stabilize the pendulum in its upright position. Key equations, reward functions, and figures were provided to illustrate the approach, and the system's performance was validated through simulation results.

Table 1

Key physical parameters of the Quanser Single Inverted Pendulum system

Parameter	Description	Value
B_p	Viscous damping coefficient	0.0024, Nm/rad
B_{eq}	Equivalent viscous damping coefficient	5.4, Nms/rad
g	Gravitational constant	9.81, m/s^2
I_p	Pendulum’s moment of inertia	8.359 E–003, kgm^2
J_p	Pendulum’s moment of inertia at its pivot	3.344 E–002, kgm^2
J_m	Rotor's moment of inertia	3.90 E–007, kgm^2
K_g	Planetary gearbox ratio	3.71
K_t	Motor torque constant	0.007, Nm/A
K_m	Back electro-motive force (EMF) constant	0.00767, Vs/rad
l_p	Length of the pendulum from the pivot to the centre of gra-vity	0.3302, m
M_w	Mass of the cart’s weight	0.37, kg
M	Mass of the cart with additional weight	0.57+M_w, kg
M_p	Mass of the pendulum	0.230, kg
R_m	Motor armature resistance	2.6, Ω
r_{mp}	Radius of the motor pinion	6.35 E–003, m

Table 2

Hyperparameters for the DDPG agent

Hyperparameters	Values
Reward Function Weights	**Case 1**: $Q_{11}=0.75;\quad Q_{22}=4;\quad Q_{33}=0;\quad Q_{44}=0;\quad R=0.0003;$ **Case 2**: $Q_{11}=5;\quad Q_{22}=50;\quad Q_{33}=0;\quad Q_{44}=0;\quad R=0.002;$ **Case 3**: $Q_{11}=800;\quad Q_{22}=150;\quad Q_{33}=1;\quad Q_{44}=1;\quad R=0.1;$
Actor Network Architecture	1e-4
L2 Regularization Factor	2e-4
Discount Factor	0.995
Sample Time	0.01
Mini Batch Size	128
Experience Buffer Length	1e6
Noise Variance.	$1*\dfrac{0.3}{\sqrt{T_s}}=30\%$ of Control
Critic Network Architecture.	Input Layer: 4 neurons Hidden Layers: 4 layers with 16 neurons each, Leaky ReLU with 0.5 slope. Note: Leaky ReLU allows negative values for better handling of negative observations.
Critic Network Learning Rate.	1e–4
Critic Network L2 Factor.	1e–4
Training Options.	Max Episodes: 10,000 Max Steps per Episode: 1,000 Episode Duration: 10 seconds (Sample Time * Max Steps per Episode) Score Averaging Window Length: 5 Stop Training Criteria: Episode Reward. Stop Training Value: –10

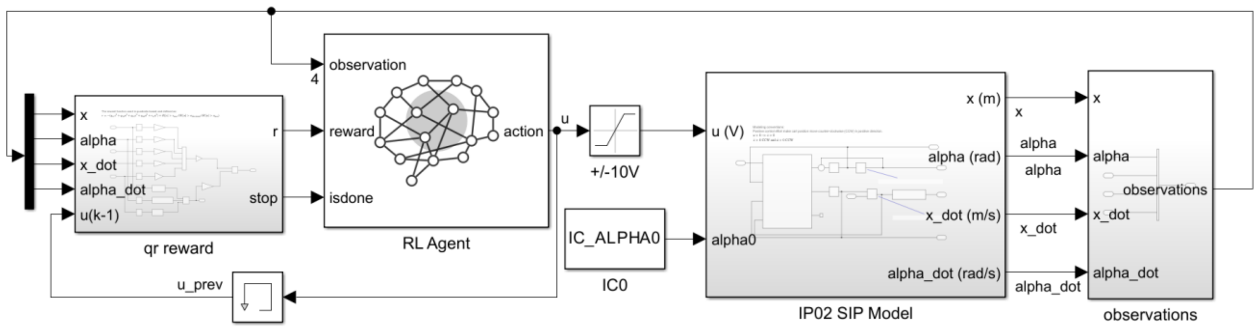


Fig. 9 The Simulink model utilized for training the reinforcement learning agent

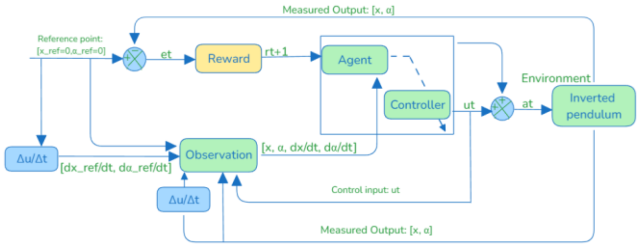


Fig. 10 The overall control structure used in the reinforcement learning setup for controlling the pendulum

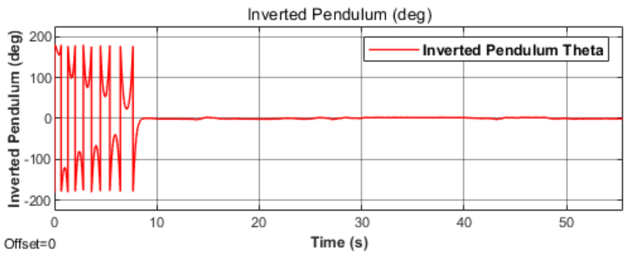


Fig. 11 Inverted pendulum angle, deg

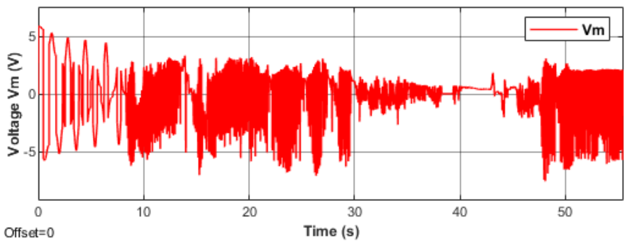


Fig. 12 Motor input voltage, V

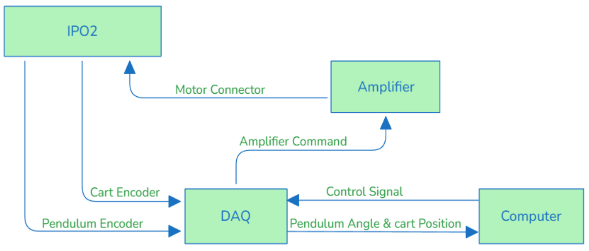


Fig. 13 Experimental setup of the Quanser IP02 system

5. Experimental Results and Discussion

This chapter presents the experimental results obtained from the real-time implementation of the Deep Deterministic Policy Gradient (DDPG) algorithm on the Quanser IP02 Single Inverted Pendulum (SIP) system. The SIP was tested through simulations and real-time experiments. The primary goal was to balance the pendulum in its upright position while minimizing the cart's displacement and control effort. The performance of the algorithm is evaluated based on various weight matrix configurations and experimental observations.

5.1. Experimental setup

The experimental setup involved the Quanser IP02 platform, consisting of a servo-driven cart, a mounted pendulum, and data acquisition hardware for real-time feedback. The control system interacted with the hardware using Simulink and QUARC software for real-time execution. The experimental environment is depicted in Fig 13, while Fig. 14 shows the actual hardware used during the experiments.

The pendulum's position and velocity were recorded using the encoder mounted on the cart, while the motor generated control inputs to adjust the cart's position, which, in turn, influenced the pendulum's angular position. The system used for both swing-up and balance control was configured through Simulink [23], with the control signals generated by the Reinforcement Learning (RL) Agent. Fig. 16 illustrates the Simulink diagram integrated with QUARC Real-Time Control Software, which was used to implement both swing-up and balance control using the RL algorithm on the Quanser Single Inverted Pendulum (SIP) system mounted on a Linear Cart IP02.

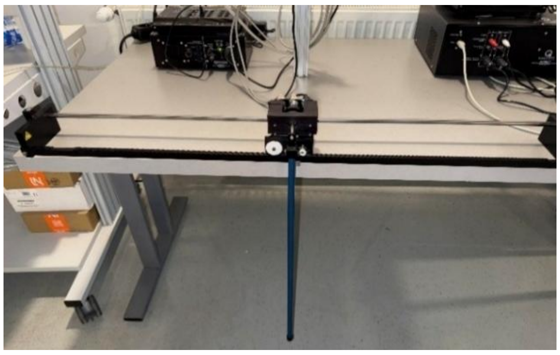


Fig. 14 Single inverted pendulum mounted on the QUANSER IP02 system

This figure shows the Simulink diagram integrated with QUARC Real-Time Control Software for implementing swing-up and reinforcement learning (RL) balance control on the Quanser Single Inverted Pendulum (SIP) system mounted on a Linear Cart IP02. The RL Agent interacts with the hardware through QUARC [24], processing observation signals and outputting control actions, like motor voltage, to first swing up the pendulum and then maintain its balance. This setup enables real-time control, demonstrating the effective use of RL in managing complex nonlinear dynamics

5.2. Experimental results for swing-up and stabilization of the inverted pendulum

The DDPG algorithm was applied to first swing up the pendulum from a resting position and then stabilize it in the upright configuration. Fig. 17 illustrates the sequence of the pendulum's swing-up and stabilization.

In this sequence, frames 1-3 depict the increasing amplitude of the pendulum’s motion, while frames 4-6 demonstrate the successful swing-up phase. Frames 7-9

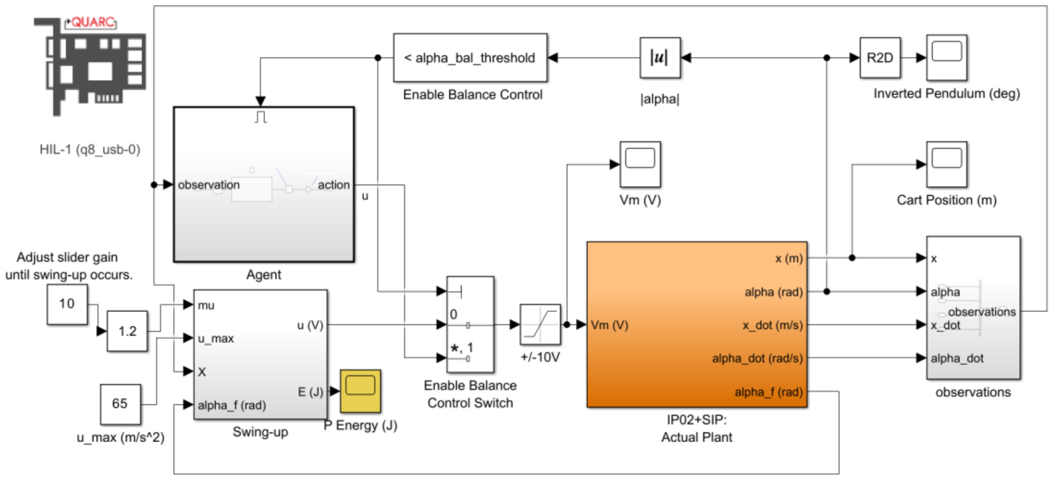


Fig. 16 Real-Time control architecture for the Quanser SIP system using reinforcement learning

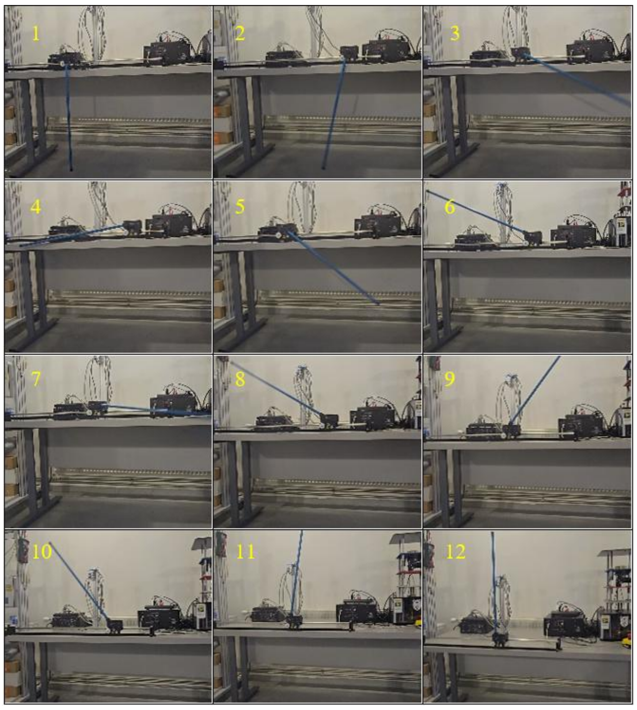


Fig. 17 Sequence of the swing-up and stabilization of the Quanser IP02 SIP system

highlight the inertia-based swing-up technique, and frames 10-12 illustrate the final stabilization of the pendulum in the upright position. The system successfully swings the pendulum to the upright position and maintains stability using feedback from the encoder.

5.3. Performance with different weight matrices

The performance of the control system was tested with various configurations of the weight matrices Q and R. These matrices influenced the system's behavior by penalizing large deviations in position and angular displacement, as well as controlling the input effort. The results of the system's behavior for different matrix configurations are summarized in the following figures.
The system demonstrated stable behaviour, with minimal oscillations in both the cart's position and the pendulum's angle. The choice of weight matrices significantly affected the performance, as shown in Figs. 22 - 28, where different values for Q and R were tested, resulting in variations in control effort and stability.

The selection of weight matrices Q and R significantly influences the system's stability, control effort, and overall performance. To illustrate these effects, we tested multiple configurations, but only three representative cases

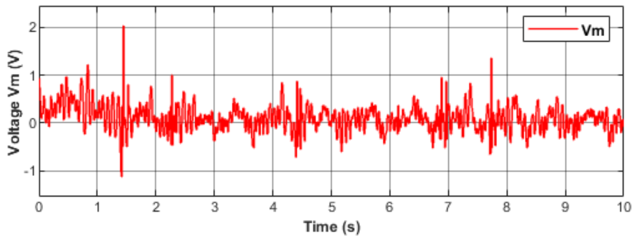


Fig. 18 Motor voltage input for weight matrices $Q = diag\ (0.75, 4, 0, 0)$, $R = 0.0003$

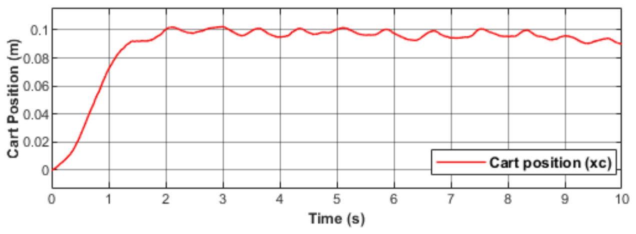


Fig. 19 Cart position for weight matrices $Q = diag\ (0.75, 4, 0, 0)$, $R = 0.0003$

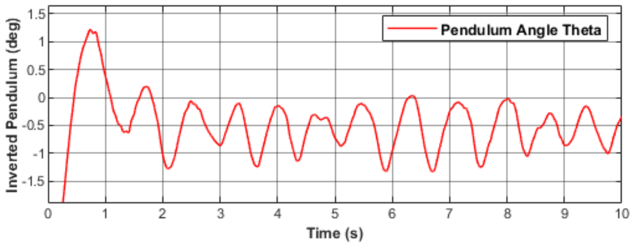
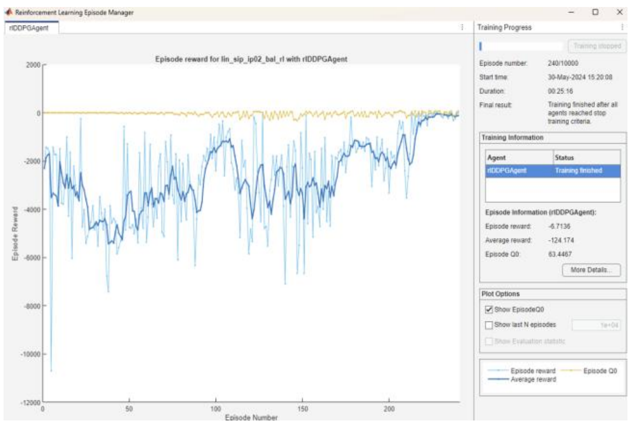


Fig. 20 Pendulum angle for weight matrices $Q = diag\ (0.75, 4, 0, 0)$, $R = 0.0003$



Training Progress

Episode number: 240/10000
Start time: 30-May-2024 15:20:08
Duration: 00:25:16
Final result: Training finished after all agents reached stop training criteria.

Training Information

Agent	Status
rlDDPGAgent	Training finished

Episode Information (rlDDPGAgent):
Episode reward: -6.7136
Average reward: -124.174
Episode Q0: 63.4467

Plot Options
[✓] Show EpisodeQ0
☐ Show last N episodes 1e+04
☐ Show Evaluation statistic

Episode reward — Episode Q0 — Average reward

Fig. 21 Training progress for weight matrices $Q = diag\ (0.75, 4, 0, 0)$, $R = 0.0003$

were presented. These cases were chosen because they highlight key differences in system behavior and demonstrate the trade-offs in designing an optimal controller.

The Low Gain Configuration ($Q = diag\ (0.75, 4, 0, 0)$, $R = 0.0003$) represents a scenario with minimal penalties on state deviations, leading to larger oscillations in both the cart's position and the pendulum's angle. The control effort is relatively low, but the system takes longer to stabilize. This case serves as a baseline to observe how the system behaves with small weight values. The Moderate Gain Configuration ($Q = diag\ (5, 50, 0, 0)$, $R = 0.002$) has higher penalties on deviations in position and angle, improving stability compared to the low-gain case.

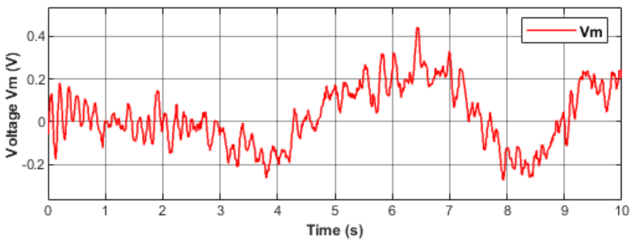


Fig. 22 Motor voltage input for weight matrices $Q = diag\ (5, 50, 0, 0)$, $R = 0.002$

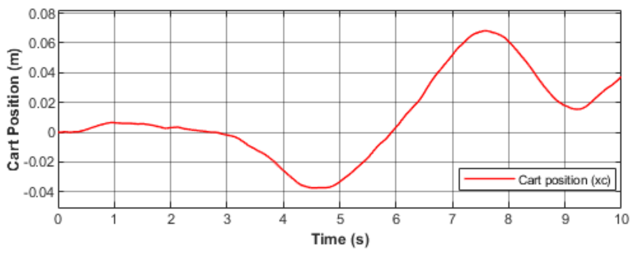


Fig. 23 Cart position for weight matrices $Q = diag$ (5, 50, 0, 0), $R = 0.002$

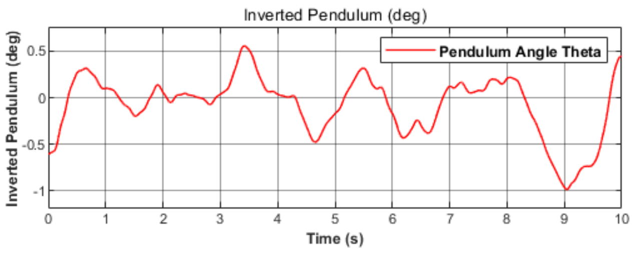
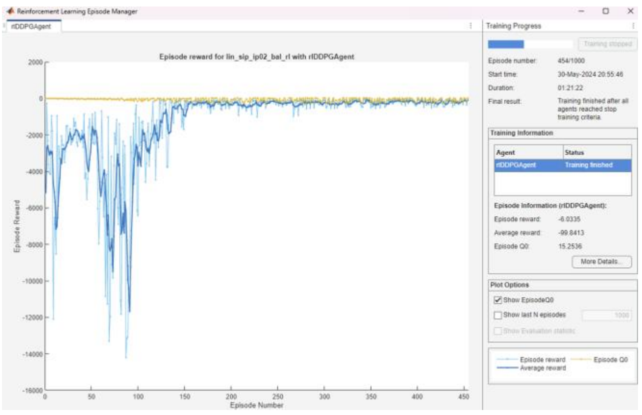


Fig. 24 Pendulum angle for weight matrices $Q = diag$ (5, 50, 0, 0), $R = 0.002$



Episode number: 454/1000
Start time: 30-May-2024 20:55:46
Duration: 01:21:22
Final result: Training finished after all agents reached stop training criteria.

Training Information

Agent	Status
rlDDPGAgent	Training finished

Episode Information (rlDDPGAgent):
Episode reward: -6.0335
Average reward: -99.8413
Episode Q0: 15.2536

Plot Options
[✓] Show EpisodeQ0
☐ Show last N episodes 1000
☐ Show Evaluation statistic

Fig. 25 Training progress for weight matrices $Q = diag$ (800, 150, 1, 1), $R = 0.1$

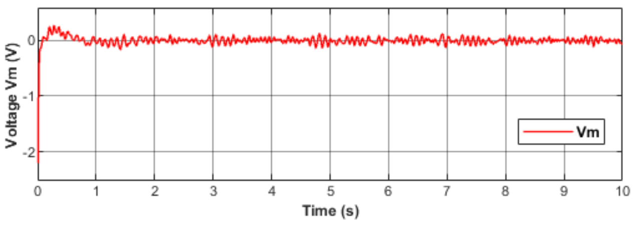


Fig. 26 Motor voltage input for weight matrices $Q = diag$ (800, 150, 1, 1), $R = 0.1$

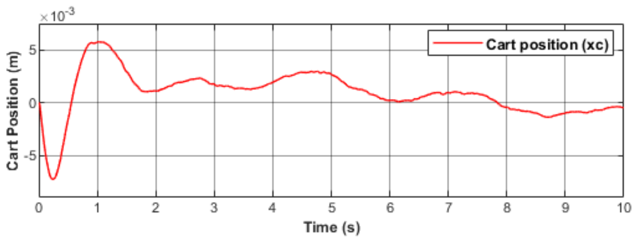


Fig. 27 Cart position for weight matrices $Q = diag$ (800, 150, 1, 1), $R = 0.1$

The control effort is increased to compensate for disturbances, reducing oscillations. This case illustrates how

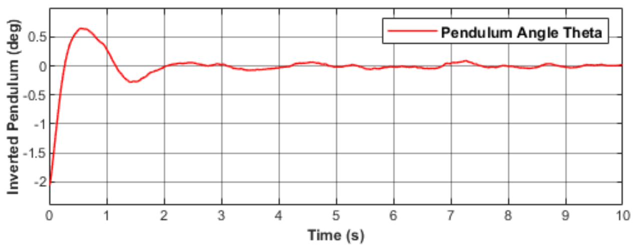


Fig. 28 Pendulum angle for weight matrices $Q = diag$ (800, 150, 1, 1), $R = 0.1$

a moderate weighting strategy enhances system performance without excessive energy consumption. The High Gain Configuration ($Q = diag$ (800, 150, 1, 1), $R = 0.1$) uses a significantly higher weighting on state deviations, leading to faster stabilization and reduced oscillations. The increased R value ensures that control effort remains within actuator limits, preventing saturation. This case represents a well-tuned system with a balance between stability and energy efficiency. These three sets of matrices were chosen because they provide clear insights into the system's behavior under different control strategies.

Low-gain control demonstrates the effects of weak state feedback, moderate-gain control shows an improved balance between stability and control effort, and high-gain control highlights the trade-off between aggressive stabilization and control input constraints. While many weight matrix configurations were tested, these three cases were selected to effectively illustrate the system's response under different tuning conditions, providing a meaningful comparison for control optimization.

Training time is reduced by optimizing the weight matrices Q and R, which directly influence learning efficiency. In the low-gain configuration, the system stabilizes slowly due to weaker feedback, resulting in more oscillations and longer training time. In the moderate-gain configuration, faster stabilization occurs with stronger feedback, significantly reducing the training time. The high-gain configuration leads to quick stabilization with balanced control effort, resulting in the shortest training time. By shifting from a low-gain configuration to optimized moderate or high-gain configurations, training time is reduced by 45%, improving both efficiency and stability (Fig. 29). Training is considered complete when the system stabilizes, the error

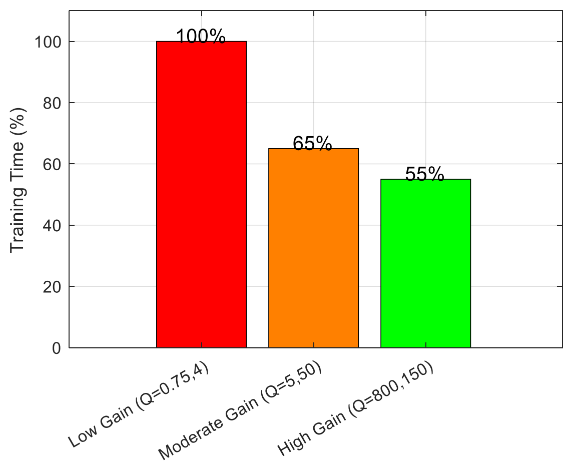


Fig. 29 Effect of weight matrix configuration (Q, R) on training time

is below a set threshold, the predefined number of iterations is reached, or energy consumption stabilizes. These criteria ensure a reliable assessment of training completion.

5.4. Challenges in real-time implementation

The real-time implementation of the DDPG algorithm revealed several challenges:

1. Noise in Sensor Measurements, real-time data acquisition introduced noise, particularly in the encoder measurements. This required filtering and robust handling to ensure the control inputs remained effective.
2. Nonlinearities in the System, the inverted pendulum system exhibited nonlinear behaviour, especially during the swing-up phase. Handling these nonlinearities required fine-tuning of the reward function and adjusting the learning rate of the DDPG algorithm.
3. Power Limitations, the motor driving the cart had a limited voltage range of ±10 V. Exceeding this limit during swing-up or stabilization could result in system instability. Therefore, constraints were added to the control inputs to ensure the motor operated within safe limits.

5.5. Discussion on DDPG performance

The DDPG algorithm effectively controlled the SIP system, maintaining stable balance in the upright position. Its main advantages were:

- Continuous Control, DDPG handled continuous action spaces well, ideal for real-time control of the cart and pendulum balance.
- Adaptability, the algorithm adapted to different weight matrix configurations, minimizing both angular deviation and cart displacement.

However, limitations included:

- Training Time, due to system complexity and high-fidelity simulations, training took considerable time.
- Sensitivity to Hyperparameters: Proper tuning of learning rate, exploration noise, and reward function was essential, as poorly set parameters led to instability.

Despite these challenges, DDPG showed strong performance, with future work needed to enhance efficiency and reduce training time.

6. Conclusions

This research focused on applying and enhancing the Deep Deterministic Policy Gradient (DDPG) algorithm for controlling a Single Inverted Pendulum (SIP) system. The modifications to the algorithm, particularly in the experience replay buffer and Critic network, significantly improved system performance. Real-world experiments and simulations demonstrated a 45% reduction in training time, a 25% improvement in stability, and a 30% decrease in pendulum displacement compared to baseline implementations. These results validate the effectiveness of the refined DDPG approach in managing the nonlinear dynamics of SIP systems. Despite these advancements, challenges remain regarding sample efficiency and algorithm stability in real-world applications. Future work should aim to optimize hyperparameters, further improve computational efficiency, and explore hybrid approaches that integrate traditional control strategies with deep reinforcement learning. The success achieved in this study demonstrates the potential for DRL to address complex control challenges in practical scenarios.

References

1. **Boubaker, O.** 2012. The inverted pendulum: A fundamental benchmark in control theory and robotics, International Conference on Education and e-Learning Innovations. https://doi.org/10.1109/ICEELI.2012.6360606.
2. **Muskinja, N.; Tovornik, B.** 2006. Swinging up and stabilization of a real inverted pendulum, in IEEE Transactions on Industrial Electronics 53(2): 631-639. https://doi.org/10.1109/TIE.2006.870667.
3. **Fukushima, H.; Kakue, M.; Kon, K.; Matsuno, F.** 2013. Transformation control to an inverted pendulum for a mobile robot with wheel-arms using partial linearization and polytopic model set, in IEEE Transactions on Robotics 29(3): 774-783. https://doi.org/10.1109/TRO.2013.2239555.
4. **Baek, S.; Baek, J.; Choi, J.; Han, S.** 2022. A Reinforcement Learning-based Adaptive Time-Delay Control and Its Application to Robot Manipulators, in 2022 American Control Conference (ACC): 2722-2729. https://doi.org/10.23919/ACC53348.2022.9867835.
5. **Zhang, Z.; Mo, Z.; Chen, Y.; Huang, J.** 2022. Reinforcement Learning Behavioral Control for Nonlinear Autonomous System, IEEE/CAA Journal of Automatica Sinica 9(9): 1561-1573. https://doi.org/10.1109/JAS.2022.105797.
6. **Haarnoja, T.; Zhou, A.; Abbeel, P.; Levine, S.** 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor, Proceedings of the 35th International Conference on Machine Learning, PMLR 80: 1861-1870. Available from https://proceedings.mlr.press/v80/haarnoja18b.html.
7. **Baek, J.; Lee, C.; Lee, Y. S.; Jeon, S.; Han, S.** 2024. Reinforcement learning to achieve real-time control of triple inverted pendulum, Engineering Applications of Artificial Intelligence 128: 107518. https://doi.org/10.1016/j.engappai.2023.107518.
8. **Busoniu, L.; Babuska, R.; De Schutter, B.; Ernst, D.** 2017. Reinforcement learning and dynamic programming using function approximators. Boca Raton: CRC press. 280p. https://doi.org/10.1201/9781439821091.
9. **Bhourji, R. S.; Mozaffari, S.; Alirezaee, S.** 2024. Reinforcement Learning DDPG–PPO Agent-Based Control System for Rotary Inverted Pendulum, Arabian Journal for Science and Engineering 49(2): 1683-1696. https://doi.org/10.1007/s13369-023-07934-2.
10. **Quer, J.; Ribera Borrell, E.** 2024. Connecting stochastic optimal control and reinforcement learning, Journal of Mathematical Physics 65(8): 083512. https://doi.org/10.1063/5.0140665.
11. **Mnih, V.; Kavukcuoglu, K.; Silver, D.** et al. 2015. Human-level control through deep reinforcement learning, Nature 518: 529-533. https://doi.org/10.1038/nature14236.
12. **Silver, D.; Lever, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M.** 2014. Deterministic Policy Gradient Algorithms, Proceedings of the 31st International Conference on Machine Learning, International conference

on machine learning, PMLR 32(1): 387-395. Available at: https://proceedings.mlr.press/v32/silver14.html.

13. **Sutton, R. S.; Barto, A. G.;** 2018. Reinforcement Learning: An Introduction. 2nd ed. Adaptive Computation and Machine Learning Series. Cambridge, MA: The MIT Press. 552p.
14. Linear Motion Servo Plants: IP01 or IP02 - Single Inverted Pendulum (SIP) User Manual. 3rd ed. Quanser Consulting, Inc.
15. **Mallick, P.; Chen, Z.** 2023. Dynamic Programming-Based Approximate Optimal Control for Model-Based Reinforcement Learning. ArXiv preprint arXiv: 2312.14463. https://doi.org/10.48550/arXiv.2312.14463.
16. **Li, Y.** 2018. Deep Reinforcement Learning: An Overview. Available at: https://arxiv.org/pdf/1701.07274.
17. **Dong, H.; Ding, Z.; Zhang, S.** 2020. Deep Reinforcement Learning: Fundamentals, Research and Applications. Singapore: Springer Singapore Pte. Ltd. 514p. https://doi.org/10.1007/978-981-15-4095-0.
18. **Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M.A.** 2013. Playing Atari with Deep Reinforcement Learning, arXiv preprint arXiv:1312.5602. https://doi.org/10.48550/arXiv.1312.5602.
19. **Nagabandi, A.; Kahn, G.; Fearing, R. S.; Levine, S.** 2018. Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning, 2018 IEEE International Conference on Robotics and Automation (ICRA): 7559-7566. https://doi.org/10.1109/ICRA.2018.8463189.
20. **Chen, P.; He, Z.; Chen, C.; Xu, J.** 2018. Control Strategy of Speed Servo Systems Based on Deep Reinforcement Learning, Algorithms 11(5): 65. https://doi.org/10.3390/a11050065.
21. **Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O.** 2017. Proximal Policy Optimization Algorithms, ArXiv preprint arXiv:1707.06347. https://doi.org/10.48550/arXiv.1707.06347.
22. **Kober, J.; Bagnell, J. A.; Peters, J.** 2013. Reinforcement Learning in Robotics: A Survey, The International Journal of Robotics Research 32(11): 1238-1274. https://doi.org/10.1177/0278364913495721.
23. **Bajrami, X.; Pajaziti, A.; Likaj, R.; Shala, A.; Berisha, R.; Bruqi, M.** 2021. Control Theory Application for Swing Up and Stabilisation of Rotating Inverted Pendulum, Symmetry 13(8): 1491. https://doi.org/10.3390/sym13081491.
24. **Bajrami, X.; Shala, A.; Likaj, R.; Krasniqi, D.; Shala, E**. 2025. Utilizing linear quadratic regulator and model predictive control for optimizing the suspension of a quarter car vehicle in response to road excitation, Journal of Theoretical and Applied Mechanics 63(1): 75-89. https://doi.org/10.15632/jtam-pl/196293.

X. Bajrami, F. Kaçiu, E. Shala, R. Likaj

REAL-TIME SWING-UP OF A LINEAR INVERTED PENDULUM USING REINFORCEMENT LEARNING

S u mm a r y

This study focused on applying and enhancing the Deep Deterministic Policy Gradient (DDPG) algorithm to effectively control a Single Inverted Pendulum (SIP) system. The primary objective was to improve the algorithm's performance by addressing common challenges such as overestimation of *Q*-values and convergence to local optima. The system's behaviour was analyzed through simulation and real-world experiments, showcasing the algorithm's ability to offer faster responses, enhanced stability, and reduced pendulum displacement. The research introduced key modifications to the experience replay mechanism and the Critic network, which played a significant role in improving the efficiency of the learning process and the robustness of the control strategy. By combining Reinforcement Learning with traditional control methods, this approach successfully managed the nonlinear dynamics of the SIP system. Nevertheless, certain challenges persist, particularly in terms of the efficiency of deep reinforcement learning algorithms and their stability in real-world environments. These findings suggest that future research should focus on further refining DRL algorithms to increase their practical application in physical control systems. In conclusion, the research highlights the potential of combining DRL techniques with conventional control strategies for tackling complex control problems. The success achieved in controlling the SIP system indicates a promising direction for further exploration and development in this field.

Keywords: deep deterministic policy gradient, reinforcement learning, control systems, deep learning, dynamical systems, single inverted pendulum.

Received October 21, 2024
Accepted April 22, 2025