


Article

# Modeling, Simulation, and Control of a Rotary Inverted Pendulum: A Reinforcement Learning-Based Control Approach

Ruben Hernandez , Ramon Garcia-Hernandez \*  and Francisco Jurado 

Instituto Tecnológico de La Laguna, Tecnológico Nacional de México, Blvd. Revolución y Av. Instituto Tecnológico de La Laguna S/N, Torreon 27000, Mexico; d.rhernandezr@correo.itlalaguna.edu.mx (R.H.); fjuradoz@lalaguna.tecnm.mx (F.J.)

\* Correspondence: rgarciah@lalaguna.tecnm.mx

**Abstract:** In this paper, we address the modeling, simulation, and control of a rotary inverted pendulum (RIP). The RIP model assembled via the MATLAB (Matlab 2021a)<sup>®</sup>/Simulink (Simulink 10.3) Simscape (Simscape 7.3)<sup>™</sup> environment demonstrates a high degree of fidelity in its capacity to capture the dynamic characteristics of an actual system, including nonlinear friction. The mathematical model of the RIP is obtained via the Euler–Lagrange approach, and a parameter identification procedure is carried out over the Simscape model for the purpose of validating the mathematical model. The usefulness of the proposed Simscape model is demonstrated by the implementation of a variety of control strategies, including linear controllers as the linear quadratic regulator (LQR), proportional–integral–derivative (PID) and model predictive control (MPC), nonlinear controllers such as feedback linearization (FL) and sliding mode control (SMC), and artificial intelligence (AI)-based controllers such as FL with adaptive neural network compensation (FL-ANC) and reinforcement learning (RL). A design methodology that integrates RL with other control techniques is proposed. Following the proposed methodology, a FL-RL and a proportional–derivative control with RL (PD-RL) are implemented as strategies to achieve stabilization of the RIP. The swing-up control is incorporated into all controllers. The visual environment provided by Simscape facilitates a better comprehension and understanding of the RIP behavior. A comprehensive analysis of the performance of each control strategy is conducted, revealing that AI-based controllers demonstrate superior performance compared to linear and nonlinear controllers. In addition, the FL-RL and PD-RL controllers exhibit improved performance with respect to the FL-ANC and RL controllers when subjected to external disturbance.

**Keywords:** underactuated mechanical systems; modeling; simulation; rotary inverted pendulum; simulation visualization; machine learning methods; reinforcement learning



**Citation:** Hernandez, R.; Garcia-Hernandez, R.; Jurado, F. Modeling, Simulation, and Control of a Rotary Inverted Pendulum: A Reinforcement Learning-Based Control Approach. *Modelling* **2024**, *5*, 1824–1852. <https://doi.org/10.3390/modelling5040095>

Academic Editors: He Cai and Maobin Lv

Received: 28 September 2024  
Revised: 13 November 2024  
Accepted: 22 November 2024  
Published: 27 November 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Underactuated mechanical systems (UMSs) are often considered an ideal benchmark for designing and testing new algorithms because of their nonlinear and complex nature, as well as the fact that their number of generalized coordinates is greater than the number of control inputs [1]. Compared to fully actuated systems, UMSs present several advantages, including decreased energy consumption, low weight, and a reduced chance of failure. UMSs are present in many robotic settings [2–4], including spacecraft and space vehicles [5–7], underwater and marine vehicles [8,9], and pendular systems [10,11]. Many researchers have identified certain UMSs, including the inertia wheel pendulum (IWP), acrobot, pendubot, rotary inverted pendulum (RIP), inverted pendulum on a cart, translational oscillator with rotational actuator (TORA), and a ball and beam system, which are valuable test bed platforms for developing and evaluating novel control algorithms.

The RIP, also known as Furuta Pendulum, is an UMS with two degrees of freedom (DOFs) and a single control input [12]. In consequence, the RIP may be defined as an UMS, exhibiting a single degree of underactuation. It is characterized by nonlinear and

underactuated dynamics and is classified as an inverted pendulum. This system comprises an underactuated pendulum that is capable of rotating freely around the vertical axis and attached to the end of a horizontally rotating arm, which is driven by an actuator. There are four common control objectives reported in the literature when controlling the RIP [13]: (1) swing-up control, which causes a transition from the downward pendulum position to an unstable, upward one; (2) stabilization control, which involves maintaining the pendulum vertically upward via regulation; (3) switching control, which consists of shifting between swing-up control and stabilization control; and (4) tracking control, entailing the upkeep of the upright pendulum position while the horizontal arm follows a desired trajectory. Linear, nonlinear, and artificial intelligence (AI) control methods have been used to achieve these latter control tasks. Linear control schemes offer simple controller implementations. In [14], a combination of a linear quadratic regulator (LQR) controller with an adaptive neuro-fuzzy inference system (ANFIS) was proposed to add robustness to the control scheme. In [15], a robust controller was designed using the  $H_\infty$ -linear matrix inequality technique to stabilize the pendulum from its unstable position under magnetic external disturbance commands. The experimental results validate the proposal. Advanced control methods such as feedback linearization (FL) are well-known nonlinear control methods. In [16], a controller for the RIP was developed using the input–output FL technique. Experimental findings showed that the arm position effectively tracked the command signal, whereas the pendulum remained in an unstable position. A sliding mode control (SMC) scheme that offers a systematic approach for handling underactuation was proposed in [17]. The proposal suggests to construct the underactuated system into cascaded subsystems comprising a linear subsystem as well as a reduced-order nonlinear subsystem. In [18], the interconnection and damping assignment passivity-based control (IDA-PBC) approach was applied to control an RIP by considering dynamic friction compensation. To demonstrate the efficacy of this approach, experimental tests were conducted. Mofid et al. [19] proposed a backstepping SMC to ensure finite-time convergence while adaptive control approximates the unknown upper bound of disturbances and uncertainties for the RIP system. The proposed method was validated through both simulations and experiments, thereby demonstrating its efficacy. AI has emerged as a promising alternative for controlling UMS. This field has witnessed a surge of interest in recent years, with a myriad of studies exploring its potential applications. In [20], a parameter self-tuning fuzzy controller for balance control of an RIP was proposed. The controller parameters were adjusted using particle swarm optimization (PSO) algorithm. The performance of this controller is superior to that of a conventional fuzzy controller. In [21], an adaptive neural network controller was designed to solve the tracking control problem of the arm and pendulum regulation from the RIP. The results from the experiments demonstrated that this latter controller outperformed other neural network controllers structures. Machine learning (ML) is a rapidly evolving subarea of AI that has recently garnered considerable attention. Reinforcement learning (RL) represents a subfield of ML in which an autonomous agent interacts with its environment and learns to select the optimal action given a specific state. The aforementioned is formulated within the context of a Markov decision process for discrete states and actions spaces. However, in practical applications, discrete state and action spaces incur a considerable computational burden. One potential solution to this last inconvenience is to integrate deep neural networks (DNNs) with RL algorithms in order to address tasks in continuous time. In [22], the soft actor critic (SAC) algorithm was used with a novel reward function based on the cosine function to stabilize the RIP. The agent was trained in a virtual environment, showing faster learning than polynomial and step-type reward functions. This agent was transferred to a real system to validate the policy learned. In [23], the proximal policy optimization (PPO) algorithm was used to swing up and stabilize the RIP. The learning process was divided into three stages. Each stage was designed using a specific reward function. Simulation results showed that the pendulum reached its unstable upward position from its stable downward position in a shorter time than that from a conventional controller. Fault-tolerant control

is another issue tackled by RL methods. In [24], a hybrid control approach that uses RL to improve fault tolerance in unmanned aerial vehicles (UAVs) focusing on mitigating actuator faults was proposed. By integrating RL, specifically the PPO algorithm, with a base controller, compensatory control signals help maintain stability and mission continuity despite operational faults. Tests in both simulations and experiments demonstrated the effectiveness of this approach in significantly improving the safety and stability of UAV operations during actuator failures. In [25], a data-driven 2D Q-learning algorithm within a two-dimensional framework was proposed, allowing for optimal control of batch processes experiencing actuator and sensor faults. This method was applied to the injection molding process. Unlike traditional model-based methods, this approach achieved enhanced control performance and tracking accuracy by learning solely from measurable data, even under model uncertainties and detection noise. In [26], an RL-based approach for optimal fault-tolerant tracking control that enables improved control performance in systems with actuator faults was proposed. The proposed RL algorithm learns the optimal control law, expands the system's fault tolerance, and enhances performance prior to fault resolution. Its effectiveness was validated through a case study in a three-capacity water tank, showing superior control effects over traditional model-based approaches. In [27], a novel RL-based, model-free min–max fault-tolerant control approach for handling fault-tolerant tracking in the presence of external disturbances and actuator failures was proposed. This approach solves a Game Algebraic Riccati Equation (GARE) directly from measured data, eliminating the need for knowledge of the system dynamics. The simulation results of the injection molding process validated the proposed method.

Although control strategies may be diverse and varied, it is not possible to evaluate their performance in an experimental platform for all cases. In a considerable number of proposals, the control algorithms are only verified in the corresponding mathematical model for the system of interest, and the results may show inconsistencies between this model and the actual system. With the constant improvement of simulation software in which three-dimensional (3D) modeling tools are integrated to allow the modeling of mechanical elements with real material properties, it is possible to design and to build complete mechanical systems similar to those from real systems. The MATLAB®/Simulink Simscape™Multibody™ toolbox, previously known as SimMechanics, is a simulation environment specifically focused on the modeling and simulation of 3D mechanical systems, such as vehicle suspensions, robots, construction machinery, and aircrafts vehicles, among other mechatronic systems. In previous studies, the Simulink/Simscape environment was used to simulate the RIP in order to implement several control strategies. In [28], the RIP was built using ADAMS software and then imported into the Simulink/Simscape environment to implement an adaptive neural network controller to stabilize the orientation angle of the arm and pendulum position. The controller was able to handle both parametric uncertainties and external disturbances. In [29], QL (Q-learning) and DQNL (Deep-Q network learning) algorithms were implemented to swing up and stabilize the RIP. The simulation results in the Simulink/SimMechanics environment showed that both RL algorithms outperformed the LQR and proportional integral derivative (PID) control methods. Guida et al. [30] implemented the deep deterministic policy gradient (DDPG) algorithm to swing up and stabilize the RIP in presence of dry friction. Simulation results in a Simulink multibody environment demonstrated the effectiveness of the approach. In [31], a combination of DDPG and PPO algorithms was applied to the Quanser Qube-Servo RIP platform built in Simulink/Simscape/Multibody environment. The simulations results demonstrated the superiority of DDPG-PPO over both PID control and the combination of SAC-PPO algorithms.

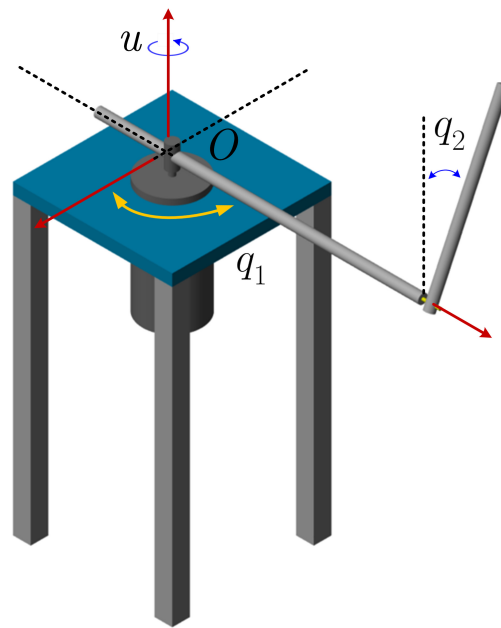
In addition, merging RL with other control approaches aims to enhance the robustness of the learned control policies because its performance may deteriorate in the presence of non-modeled dynamics, parametric uncertainties, and external disturbances. In [32], Q-learning was used to balance a double-inverted pendulum, and the learned policy was combined with proportional–derivative (PD) control. Simulation results showed that this hybrid control offers better results than individual controllers. Cheng et al. [33] proposed an add-on method to enhance the robustness of pre-trained RL policies by integrating a  $\mathcal{L}_1$  adaptive controller ( $\mathcal{L}_1$ -AC).  $\mathcal{L}_1$ -AC swiftly estimates and compensates for dynamic variations, thereby reinforcing the RL policy against diverse system changes. The feasibility of the proposed method was proven through numerical simulations and real-time experiments in a pendubot robotic system. In [34], authors proposed a compound controller by merging a traditional feedback states controller with RL applied to a ballbot robotic system. Numerical simulations illustrated that the proposed controller enabled the ballbot robot to maintain balance across a wide range of initial tilting angles than conventional model-based controllers. Kim et al. [35] suggested bolstering the robustness of RL-based controllers through the application of a disturbance observer. This method compensates for the mismatch between the real plant and the simulation model and rejects disturbance to maintain the nominal performance while guaranteeing robust stability. The proposed method was verified through simulations on an inverted pendulum.

While adaptive neural network compensation (ANNC) has demonstrated favorable outcomes in controlling the RIP, RL offers an alternative avenue for addressing this challenge. Therefore, the novelty and contribution of this work is twofold: (a) a virtual prototype of the RIP is proposed in a Simscape/Multibody environment with the aim of capturing the entire nonlinear dynamics from a real system, including nonlinear Coulomb friction and viscous friction, which have not been considered in previous virtual models of the RIP; and (b) a methodology that combines RL with other control strategies is proposed. In particular, RL-based compensation provides better results than a conventional ANNC in an FL controller [21] in order to stabilize the RIP, highlighting that the control using RL-based compensation shows better results when facing external disturbances.

This manuscript is organized as follows. Section 2 provides a detailed description of the design of the RIP in a Simscape multibody environment. Section 3 describes the mathematical model of the RIP, its parameter identification, and validation. The proposed methodology is explained in Section 4. The control algorithms used for comparison are described in Section 5. The simulation results are presented in Section 6. Finally, the concluding remarks are established in Section 7.

## 2. Modeling of the RIP in the Simscape Environment

In this study, the RIP is modeled and simulated as a multibody system using the MATLAB<sup>®</sup>/Simulink simulation platform. A representation of the Simscape model of the RIP is shown in Figure 1. The RIP comprises a horizontal arm that is affixed at one end to an actuator, which is capable of rotating around its own axis. The opposite end of the horizontal arm supports a vertical pendulum that swings freely. The control input, denoted by  $u$ , responsible for the movement of the horizontal arm is expressed in [Nm]. The angular position  $q_1$  from the horizontal arm is restricted to a range of  $[-\pi, \pi]$  [rad]. The position of the pendulum  $q_2$  spans from  $-\pi$  to  $\pi$  [rad], and its vertical position is set at zero. The angular position of the horizontal arm  $q_1$  is zero when the arm is on its origin. The angular velocity of the horizontal arm is denoted by  $\dot{q}_1$ , whereas for the vertical pendulum it is represented by  $\dot{q}_2$ . The usefulness of our model is demonstrated by implementing various control strategies. Also, a swing-up controller is included for each control strategy.



**Figure 1.** Simscape model of the RIP. The control input is denoted by  $u$ ; the angular displacement of the horizontal arm is denoted by  $q_1$ , and the angular position of the pendulum is denoted by  $q_2$ .

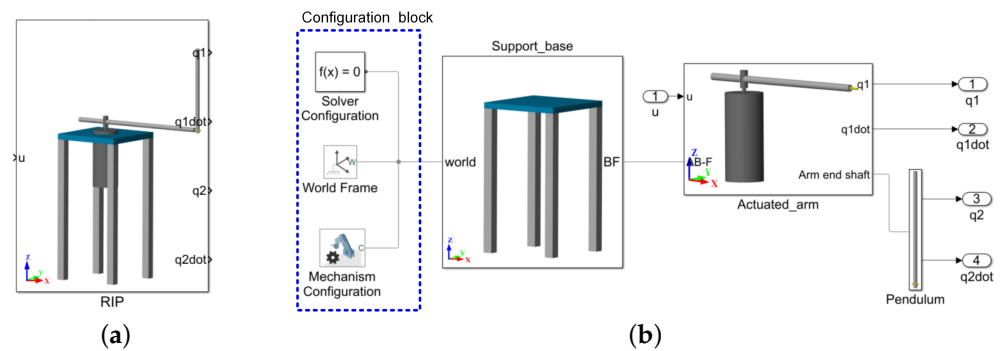
### 2.1. Modeling Software

Simscape (Simscape 7.3)<sup>TM</sup> Multibody<sup>TM</sup> provides a 3D simulation environment for multibody systems, wherein it constructs and resolves the equations of motion that govern a multibody mechanical system. In recent years, the improvement of various tools in the Simscape library has led to improvements in the modeling of robotic structures [36].

The Simscape model of the RIP proposed in this work is divided into one *Configuration block* and three subsystems, namely *Support\_base*, *Actuated\_arm*, and *Pendulum*, as it is shown in Figure 2b, which includes all the elements that together make up the entire model (Figure 2a). The *Configuration block* serves as the source of uniform gravity for the entire mechanism. This block consists of three blocks, namely *Solver Configuration*, *World Frame*, and *Mechanism Configuration*. The *Solver configuration* block is utilized to define the solver parameters that are required by the 3D model. In our model, a Backward Euler solver is used. The *World Frame* block is utilized to represent the global reference frame within the model. This frame provides access to either the world frame or the ground frame and serves as a fixed coordinate system that is predefined for each mechanical model. The world frame serves as the reference point for defining all other frames, either directly or indirectly. The *Mechanism Configuration* block establishes mechanical and simulation parameters applicable to the whole target system to which it is connected. The parameters included are the gravity constant and a linearization delta parameter, which are used for the calculation of numerical partial derivatives in the linearization process. The value of the gravity parameter is set along the  $z$ -axis as  $[x \ y \ z] = [0 \ 0 \ -9.81] \text{ [m/s}^2\text{]}$  and the linearization delta parameter is set to 0.001. The *Support\_base* subsystem consists of four legs arranged to support a rectangular base on top to form a solid structure attached to the world frame that supports the UMS. The *Actuated\_arm* subsystem, as its name suggests, contains the body of the horizontal arm that is attached to the actuator body and provides the position and velocity from the arm as it is translated by the applied torque between the bodies. In this subsystem, the dynamics also include the rotational friction force, characterized by the Coulomb friction and viscous friction, in order to better replicate the real behavior. The *Pendulum* subsystem represents the pendulum that is hanging, attained to the horizontal arm which freely rotates around the tip of the shaft of the arm. This subsystem provides the angular position and velocity from the pendulum and its rotational motion includes both Coulomb and viscous



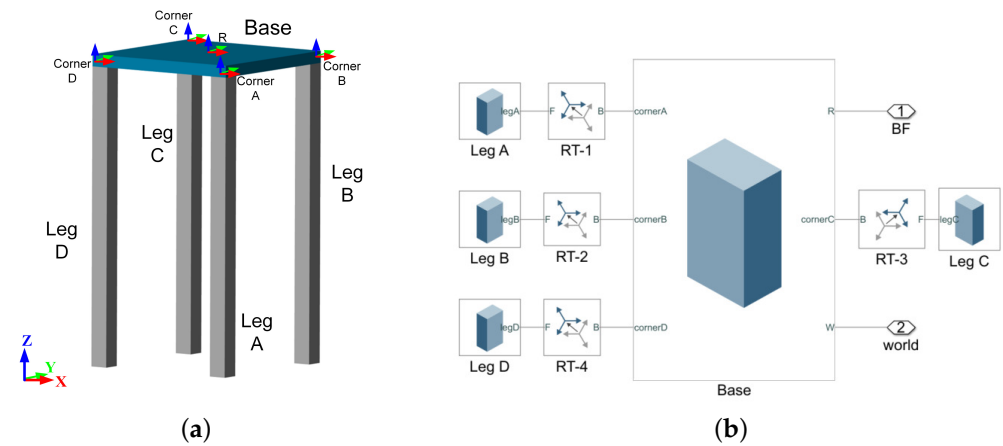
friction components. The density of all elements that constitute the prototype is chosen as  $2700 \text{ kg/m}^3$ , which is the average value from the density of aluminum alloys.



**Figure 2.** Simulink blocks of the RIP system: (a) main subsystem, (b) elements of the main subsystem: Configuration block components (box-dashed lines), Support\_base, Actuated\_arm and Pendulum subsystems.

2.2. Modeling of the Support Base

The base structure is an element that supports all components of the RIP. This structure consists of four legs with a rectangular base at the top as is shown in Figure 3a. For this purpose, five brick solid blocks are used to build this structure with respect to the  $x$ - $y$ - $z$  axes of the coordinate system. The assembly of these components and their orientation relative to the coordinate framework is performed via the rigid transform blocks RT-1 to RT-4, as is shown in Figure 3b. Port R of the brick solid block Base is a local frame that represents a reference associated with its geometry, and the support body of the arm is placed over this frame. The choice of design parameters for the support-base components is shown in Table 1.



**Figure 3.** (a) Simulink block of the support base and (b) components of the support base.

**Table 1.** Design parameters of the Support\_base subsystem components.

Component	Dimensions [x y z]m	Inertia Properties		
		Type	Based On	Density [Kg/m <sup>3</sup> ]
Leg A	[0.03 0.03 0.50]	From geometry	Density	2700
Leg B	[0.03 0.03 0.50]	From geometry	Density	2700
Leg C	[0.03 0.03 0.50]	From geometry	Density	2700
Leg D	[0.03 0.03 0.50]	From geometry	Density	2700
Base	[0.27 0.27 0.02]	From geometry	Density	2700

### 2.3. Modeling of Horizontal Arm

Once the support structure is built, the next step is to design the elements that make up the arm-support body and the horizontal arm body within the subsystem block labeled as *Actuated\_arm* as it is shown in Figure 4a. In this subsystem, five cylindrical solid blocks are used to model the geometric bodies that emulate the arm support and the horizontal arm. Three of these components are utilized to build the arm support and are featured in a subsystem labeled as *Actuator*; see Figure 4b. The remaining two components are the *Horizontal\_arm* and *Arm\_end\_shaft* blocks, connected to build the structure that constitutes the horizontal arm and their orientation relative to the coordinate framework made via the rigid transform blocks RT-6 and RT-7.

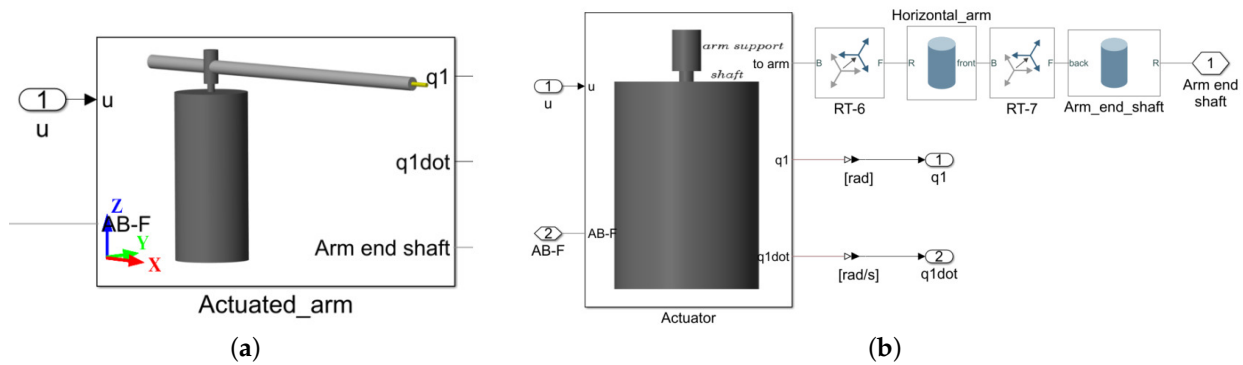


Figure 4. (a) Simulink block of the *Actuated\_arm* subsystem and (b) elements of the subsystem.

The elements that constitute the body of the actuated arm are the first to be considered since the horizontal arm and the vertical pendulum are attached to them. The subsystem labeled as *Actuator* is designed to resemble the body of an actuator conformed by *actuator\_body*, *actuator\_shaft*, and *arm\_support*. These components are illustrated in Figure 5. The orientation of the *Actuator\_body* with respect to the coordinate framework is achieved via the rigid transform block RT-5. The *Revolute Joint 1* block allows the *arm\_support* to rotate about its own axis. The description of the subsystem that represents friction from each joint is discussed in Section 2.5. The selection of parameters which define each of the components of the *Actuated\_arm* block are shown in Table 2.

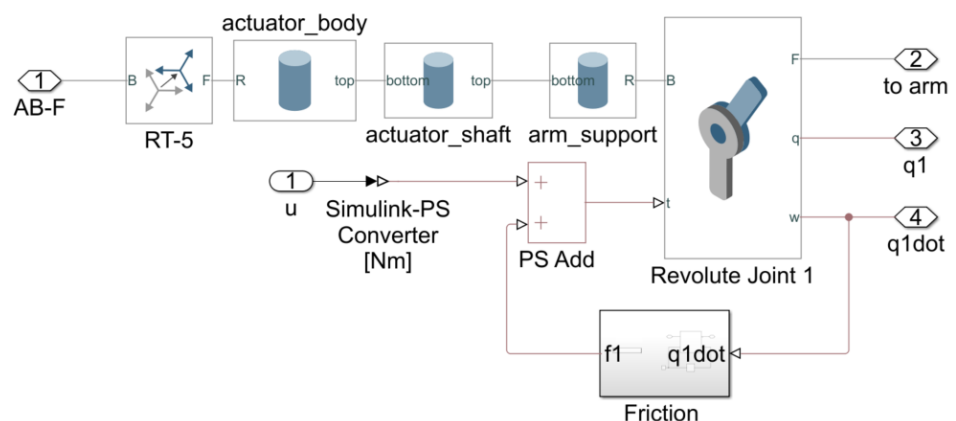


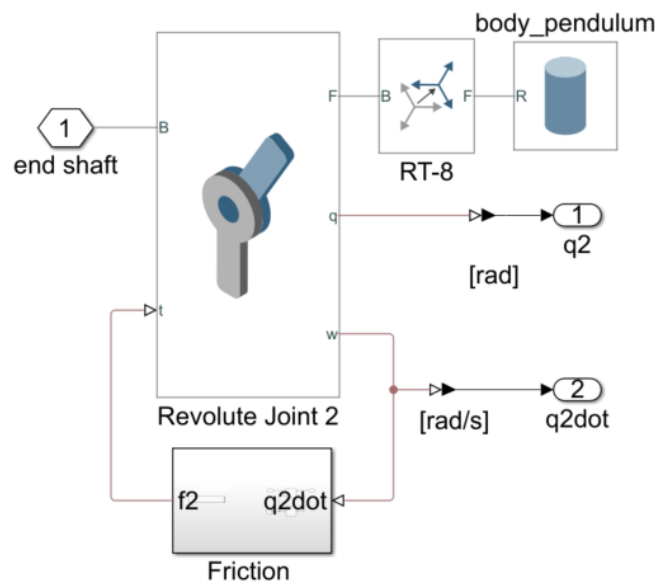
Figure 5. Elements of the *Actuator* subsystem Simulink block.

**Table 2.** Geometric dimensions of the *Actuated\_arm* subsystem elements.

Component	Radius [m]	Length [m]	Inertia Properties		
			Type	Based On	Density [Kg/m <sup>3</sup> ]
Actuator_body	0.0500	0.20	From geometry	Density	2700
Actuator_shaft	0.0500	0.01	From geometry	Density	2700
Arm_support	0.0100	0.04	From geometry	Density	2700
Horizontal_arm	0.0081	0.55	From geometry	Density	2700
Arm_end_shaft	0.0025	0.03	From geometry	Density	2700

2.4. Modeling of the Vertical Pendulum

The last component to be described is the vertical pendulum, which rotates freely at the tip of the end shaft of the horizontal arm. The geometric bodies of these elements are shown in Figure 6. A cylindrical block labeled as *Body\_pendulum* is used to represent the body of the pendulum, and its orientation relative to the coordinate framework is defined by the rigid transform block RT-8. The *Revolute Joint 2* block allows the pendulum to rotate freely at the tip of the end shaft of the horizontal arm. The *Friction in Joint q2* subsystem block describes the friction for the joint connecting the pendulum to the tip of the end shaft of the horizontal arm, which is discussed in more detail in Section 2.5. The parameters which define each of the components of the *Pendulum* subsystem block are given in Table 3.



**Figure 6.** Elements of the *Pendulum* subsystem Simulink block.

**Table 3.** Dimensioning parameters of the vertical pendulum.

Component	Radius [m]	Length [m]	Inertia Properties		
			Type	Based On	Density [Kg/m <sup>3</sup> ]
Body_pendulum	0.0081	0.3	From geometry	Density	2700

2.5. Rotational Friction Modeling

The *Friction* subsystem incorporates the rotational friction present at each joint. In real mechanical systems, the friction between their surfaces exhibits nonlinear behavior, characterized by a decrease in its value with increasing sliding speed until a specific condition is attained. At this point, the friction may remain constant, increase, or decrease with increasing sliding speed due to viscous and temperature effects [37]. These phenomena



between contacting bodies is modeled by (1), depicted in Figure 7a, and is incorporated through a Simscape *Rotational Friction* block, Figure 7b. The torque caused by friction in each joint is referred to as  $f_i$  and is comprised of three components, namely Stribeck, Coulomb, and viscous friction, and it is modeled as a function of relative angular velocity as follows:

$$f_i = \sqrt{2e}(f_{si} - f_{ci})\exp\left(-\left(\frac{\dot{q}_i}{\omega_{si}}\right)^2\right)\frac{\dot{q}_i}{\omega_{si}} + f_{ci}\tanh(\beta_c\dot{q}_i) + f_{vi}\dot{q}_i, \quad (1)$$

where  $f_{si}$  is the Stribeck friction coefficient in proximity to zero velocity,  $f_{ci}$  is the Coulomb friction coefficient,  $f_{vi}$  is the viscous friction coefficient,  $\omega_{si} = \omega_{brki}\sqrt{2}$  is the Stribeck velocity threshold,  $\omega_{brki}$  is the breakaway friction velocity,  $\omega_c = \omega_{brki}/10$  is the Coulomb velocity threshold, and  $\beta_c = 1/\omega_c$ . The proper choice of the friction parameters is given in Table 4.

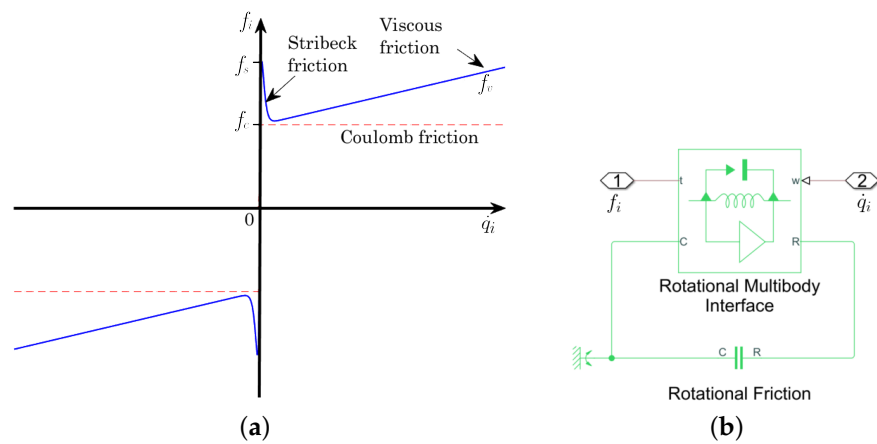


Figure 7. (a) Rotational friction torque and (b) components of the friction subsystem.

Table 4. Friction parameters of the Simscape RIP model.

$i$	$f_{si}$ [Nm]	$\omega_{brki}$ [rad/s]	$f_{ci}$ [Nm]	$f_{vi}$ [Nm·s/rad]
1	0.0188	0.1	0.0188	0.0083
2	0.0087	0.1	0.0087	0.0007

These selected parameters represent a continuous friction model, described by the Coulomb friction plus a viscous friction component.

### 3. Mathematical Model, Parameter Identification, and Validation

#### 3.1. Mathematical Model

The mathematical model of the RIP is represented using the Euler–Lagrange method [38] and expressed as follows:

$$M(q)\ddot{q} + C(q, \dot{q})\dot{q} + g(q) + f(\dot{q}) = \tau \quad (2)$$

where  $q = [q_1 \ q_2]^T$  is the vector of joint positions;  $\tau = [u \ 0]^T$  is the vector of control inputs, where  $u \in \mathbb{R}$  is the control input in the actuated joint;  $M(q) \in \mathbb{R}^{2 \times 2}$  represents the inertia matrix. The centripetal and Coriolis forces are represented by vector  $C(q, \dot{q})\dot{q} \in \mathbb{R}^2$ ,  $g(q) \in \mathbb{R}^2$  is the vector of gravitational forces, and  $f(\dot{q}) \in \mathbb{R}^2$  represents the friction components. The elements of the dynamic model are the following:

$$M(q) = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix} \quad C(q, \dot{q}) = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \quad g(q) = \begin{bmatrix} g_{11} \\ g_{21} \end{bmatrix} \quad f(\dot{q}) = \begin{bmatrix} f_{11} \\ f_{21} \end{bmatrix}$$

where

$$\begin{aligned}
 m_{11} &= p_1 + p_2 \sin^2(q_2) & c_{11} &= \frac{1}{2} p_2 \sin(2q_2) \dot{q}_2 & g_{11} &= 0 \\
 m_{12} &= p_3 \cos(q_2) & c_{12} &= -p_3 \sin(q_2) \dot{q}_2 + \frac{1}{2} p_2 \sin(2q_2) \dot{q}_1 & g_{21} &= -p_5 \sin(q_2) \\
 m_{21} &= m_{12} & c_{21} &= -\frac{1}{2} p_2 \sin(2q_2) \dot{q}_1 & f_{11} &= p_8 \tanh(\beta_c \dot{q}_1) + p_6 \dot{q}_1 \\
 m_{22} &= p_4 & c_{22} &= 0 & f_{21} &= p_9 \tanh(\beta_c \dot{q}_2) + p_7 \dot{q}_2
 \end{aligned}$$

where positive constants  $p_i$  are determined by the physical parameters of the RIP. Table 5 describes these parameters and their estimated values; they are composed of the inertia  $J_1$  of the horizontal arm, the mass  $m_2$  of the pendulum, the total length  $L_1$  of the horizontal arm, the distance  $l_2$  to the center of the pendulum, the inertia  $J_2$  of the pendulum, the viscous friction coefficients  $f_{v_1}$  and  $f_{v_2}$  from the arm and pendulum joints, respectively, the Coulomb friction coefficients  $f_{c_1}$  and  $f_{c_2}$  from the arm and pendulum joints, respectively, and the constant  $g$  of the gravitational acceleration of  $9.81 \text{ [m/s}^2\text{]}$ .

The dynamic Equation (2) exhibits linearity in its parameters [39]. This feature allows the system to be represented as a linear-regression model, and the least-squares algorithm can be used to identify unknown parameters, as described in [40]. The time evolution of the estimated parameters  $\hat{p}_i$  is shown in Figure 8. It is worthwhile to note that the estimated friction parameters are in close approximation to those defined in the Simscape model.

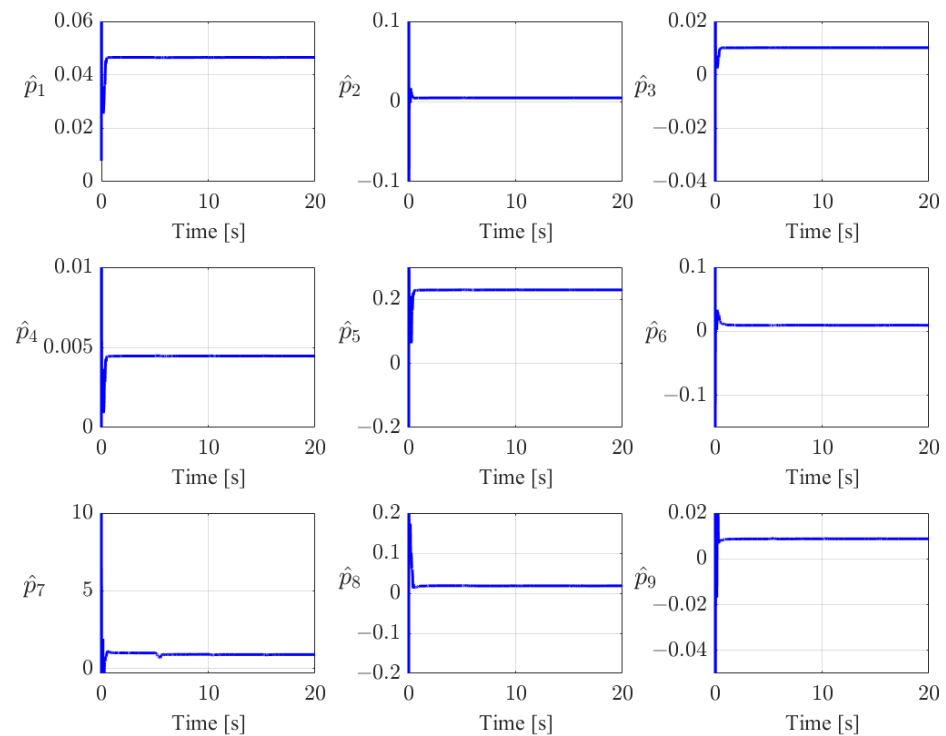


Figure 8. Evolution of the estimated parameters  $\hat{p}_i(t)$ .

**Table 5.** Parameters for the Simscape RIP model obtained via least-square identification procedure.

Parameter	Description Parameter	Value	Units
$p_1$	$J_1 + m_2L_1^2$	0.04651	$\text{kg}\cdot\text{m}^2\cdot\text{rad}$
$p_2$	$m_2l_2^2$	0.00448	$\text{kg}\cdot\text{m}^2\cdot\text{rad}$
$p_3$	$L_1l_2m_2$	0.01014	$\text{kg}\cdot\text{m}^2\cdot\text{rad}$
$p_4$	$J_2 + m_2l_2^2$	0.00446	$\text{kg}\cdot\text{m}^2\cdot\text{rad}$
$p_5$	$l_2m_2g$	0.22936	$\text{kg}\cdot\text{m}^2\cdot\text{rad}$
$p_6$	$f_{v_1}$	0.00953	$\text{N}\cdot\text{m}\cdot\text{s}/\text{rad}$
$p_7$	$f_{v_2}$	0.00090	$\text{N}\cdot\text{m}\cdot\text{s}/\text{rad}$
$p_8$	$f_{c_1}$	0.01836	$\text{N}\cdot\text{m}$
$p_9$	$f_{c_2}$	0.00877	$\text{N}\cdot\text{m}$

### 3.2. Validation of the Mathematical Model

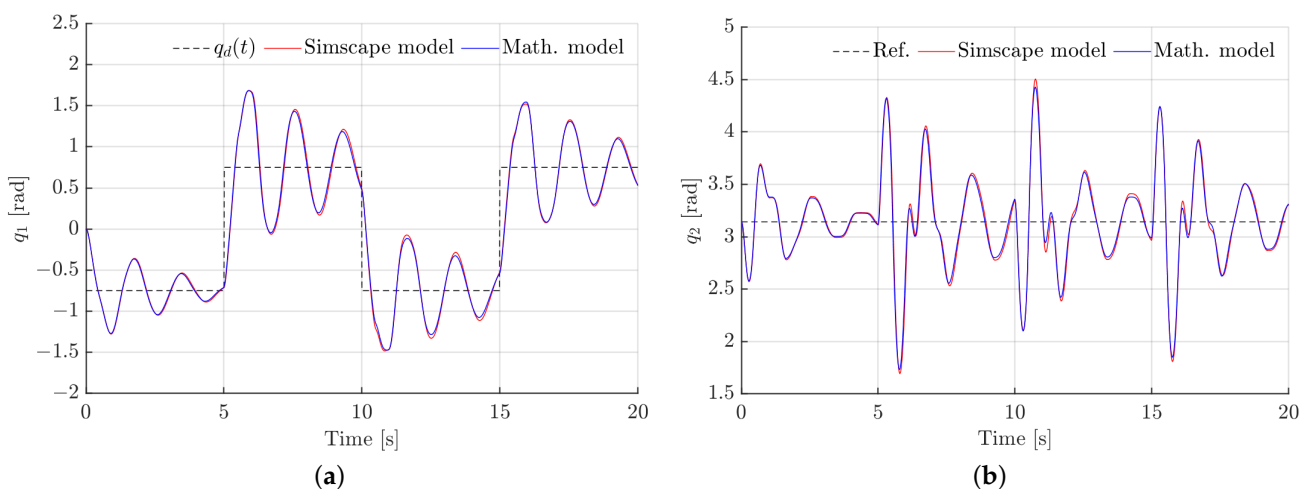
In order to validate the mathematical model using the estimated parameters and to compare it with the Simscape model, a proportional-type control signal is applied to regulate arm position in accordance with the following specifications:

$$u(t) = K_p(q_d(t) - q_1(t)). \tag{3}$$

The proportional gain  $K_p$  is set to 0.7271. The desired position  $q_d(t)$  is a periodic signal with a period of 10 [s] and is defined as follows:

$$q_d(t) = \begin{cases} -0.75 \text{ [rad]} & \text{para } 0 \leq t \leq 5, \\ 0.75 \text{ [rad]} & \text{para } 5 < t \leq 10. \end{cases} \tag{4}$$

This signal is applied for 20 [s] with a sampling time of 0.01 [s]. The initial conditions are set to  $q_1(0) = 0$ ,  $q_2(0) = \pi$ ,  $\dot{q}_1(0) = 0$ , and  $\dot{q}_2(0) = 0$ . The time evolutions of the angular positions for both the Simscape and mathematical models are shown in Figure 9.



**Figure 9.** Time evolution of angular positions of the Simscape and mathematical model for (a) arm position and (b) pendulum position.

### 3.3. Linear Mathematical Model

When a linear control approach is designed, nonlinear model (2) cannot be used directly. In such instances, it is essential to identify an approximate linear model that accurately represents model (2) to a satisfactory degree for controller design purposes. A linear controller is effective for maintaining the pendulum at the position  $q_2 = 0$  only when the pendulum is in close proximity to this configuration and remains within a similar range. To this end, a linearization technique must be applied to approximate the nonlinear

dynamics of the system at a linear operating point. The first step is to express nonlinear model (2) in the state variable form, defining the state vector as  $x = [x_1 \ x_2 \ x_3 \ x_4]^T = [q_1 \ \dot{q}_1 \ q_2 \ \dot{q}_2]^T$ . Neglecting the frictional forces, the state equation can be written as

$$\dot{x} = \begin{bmatrix} f_1(x, u) \\ f_2(x, u) \\ f_3(x, u) \\ f_4(x, u) \end{bmatrix} = \begin{bmatrix} x_2 \\ \frac{m_{22}(-c_{11}x_2 - c_{12}x_4 - g_{11}) - m_{12}(-c_{21}x_2 - c_{22}x_4 - g_{21})}{m_{11}m_{22} - m_{21}m_{12}} + \frac{m_{22}}{m_{11}m_{22} - m_{21}m_{12}}u \\ x_4 \\ \frac{-m_{21}(-c_{11}x_2 - c_{12}x_4 - g_{11}) + m_{11}(-c_{21}x_2 - c_{22}x_4 - g_{22})}{m_{11}m_{22} - m_{21}m_{12}} + \frac{-m_{21}}{m_{11}m_{22} - m_{21}m_{12}}u \end{bmatrix} \quad (5)$$

A desirable operating point is for the pendulum to remain balanced in a vertically upward position. Accordingly, the following operating point is selected:

$$x^* = \begin{bmatrix} x_1^* \\ x_2^* \\ x_3^* \\ x_4^* \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad u^* = 0. \quad (6)$$

It is possible to approximate nonlinear system (5) at the operation point  $(x^*, u^*)$  using the following linear model:

$$\dot{x} = Ax + Bu \quad (7)$$

Using the approximated linearization method [41] around the operating point, the linear approximated model obtained is

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{p_3 p_5}{p_1 p_4 - p_3^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & -\frac{p_1 p_5}{p_1 p_4 - p_3^2} & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & -22.2125 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 101.8734 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ \frac{p_4}{p_1 p_4 - p_3^2} \\ 0 \\ \frac{p_3}{p_1 p_4 - p_3^2} \end{bmatrix} = \begin{bmatrix} 0 \\ 42.6128 \\ 0 \\ -96.8455 \end{bmatrix} \quad (8)$$

This linear approximation model is only applicable when states  $x$  and  $u$  remain near the operating point  $(x^*, u^*)$ .

#### 4. Proposed Methodology Based on RL

##### 4.1. RL Framework

RL involves discovering the optimal behavior that yields the highest reward in an unknown environment. This optimal behavior is acquired through interactions with the environment and observations of its responses. The main elements of RL are the agent (controller), environment (unknown system), policy (control signal), and reward (utility function), as shown in Figure 10. In a RL framework, the agent observes the states, represented by vector  $s$ , which are generated by the system. Utilizing this information, action  $u$  that affects the system is generated by the agent and leads it into another state  $s'$  [42]. The agent receives an award or punish stimulus by the action taken; this stimulus is known as reward  $r$ , determined by a predefined reward function. This reward signal provides information on whether the chosen action moves the system toward the intended goal state. The agent searches for an optimal policy that provides maximum cumulative long-term reward. Figure 11 shows the flowchart of the proposed methodology. First, for the design of the RL agent, it is important to consider the following aspects: definition of an adequate reward function as well as the tuning of its parameters, which allows the agent to perform a specific task, which in this case is to stabilize the RIP; definition of the vector of observations coming from the environment; and adjustment of the hyperparameters of the chosen RL algorithm. Next, the architecture of the neural networks is defined, as well as the activation functions for the chosen algorithm, establishing the criteria under which the learning process ends. These criteria include the total number of trials to be executed and the value of the expected cumulative long-term reward. The latter indicates that the agent has learned to control the system in an acceptable manner. Once all of the above criteria are

defined, the second stage is to execute the RL algorithm. If, at the end of training, the agent does not perform in an acceptable manner, it is necessary to adjust one of the previous criteria and run the algorithm again until an acceptable performance is obtained. Once an agent learns to perform the designed task, it is incorporated into a control scheme and its performance is evaluated.

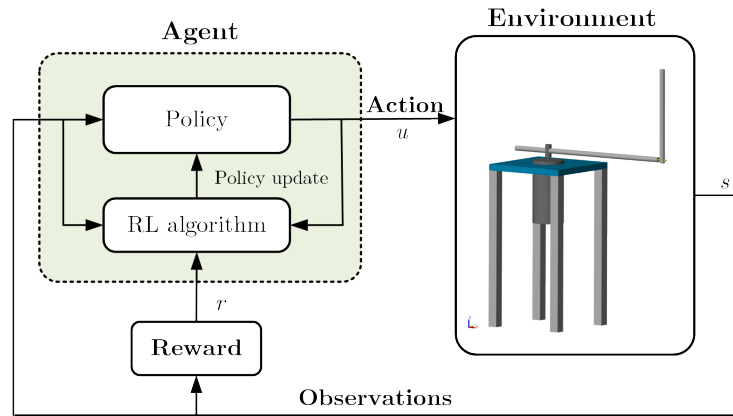


Figure 10. Block diagram of the elements of an RL framework.

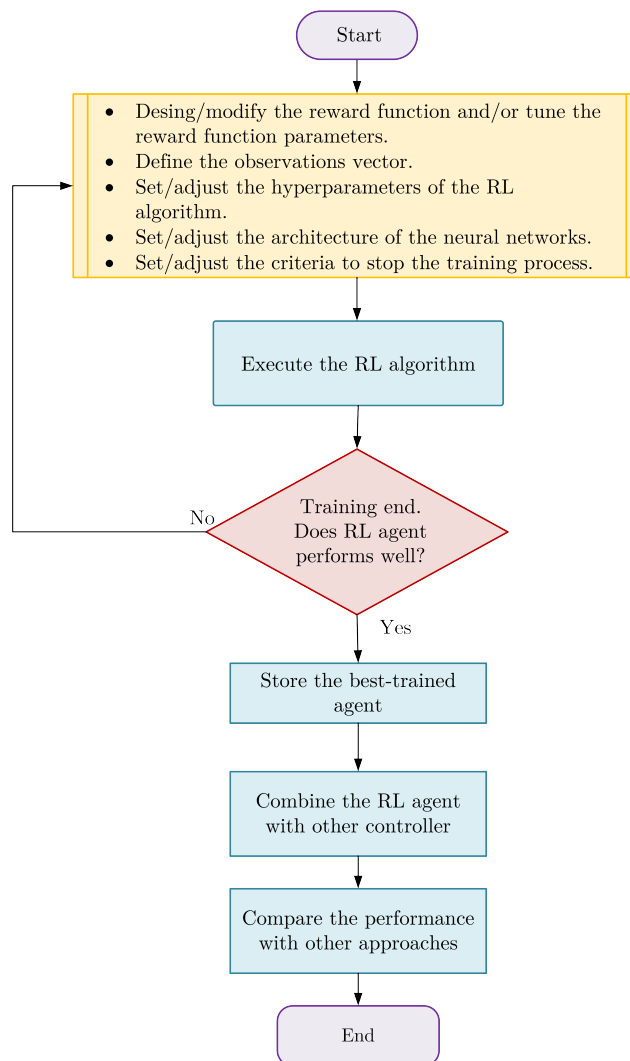


Figure 11. Flowchart of the proposed methodology.

#### 4.2. Deep Deterministic Policy Gradient Algorithm

With the aim of achieving equilibrium in the pendulum’s vertical upward position and preserving the angular position of the arm as close to its point of origin as possible, we used the DDPG algorithm [43]. The block diagram of the DDPG algorithm is shown in Figure 12. This algorithm provides a solution based on an actor–critic structure, which entrusts the actor  $\mu^\phi(s)$  to formulate the policy. In contrast, a critic  $Q^\theta(s, u)$  assesses the excellence of the actions produced by approximating the anticipated total reward according to the current policy. DDPG utilizes target neural networks for both the actor  $\mu^{\phi'}(s)$  and the critic  $Q^{\theta'}(s, u)$  with the objective of enhancing the stability of the learning process. These target neural networks are copies of the original actor and critic, and they undergo gradual updates through a soft update mechanism. To promote exploration within the action space, a Gaussian white noise-based stochastic signal, known as Ornstein–Uhlenbeck  $\mathcal{N}$ , is incorporated into the actor’s output. The agent selects an action  $u = \mu^\phi(s) + \mathcal{N}$ , executes the action  $u$ , observes the reward  $r$  and the new state  $s'$ . While exploiting the learned policy to some extent, the agent has the ability to explore alternative actions. As part of the learning process, the agent interacts with its environment, acquiring experiences in the form of tuples comprising four elements  $(s, u, r, s')$  at each time step. These tuples are stored in the replay buffer, designated as  $\mathcal{R}$ . Every  $m$  episodes, the agent selects a small set of experiences  $D \in \mathbb{N}$  from its memory bank and utilizes them to modify the parameters of both the actor and critic neural networks. The training of the actor aims to optimize the anticipated total reward, which is evaluated by the critic for the given state. The target value function (Q-value) is computed as

$$y_i = r_i + \gamma Q^{\theta'}(s'_i, \mu^{\phi'}(s'_i)) \tag{9}$$

where  $\gamma \in (0, 1)$  represents a discount factor. The training of the critic utilizes the mean squared error loss between the anticipated Q-value (expected total reward) and the target Q-value by means of the mean-square Bellman difference equation:

$$L = \frac{1}{D} \sum_{i=1}^D (y_i - Q^\theta(s_i, u_i))^2 \tag{10}$$

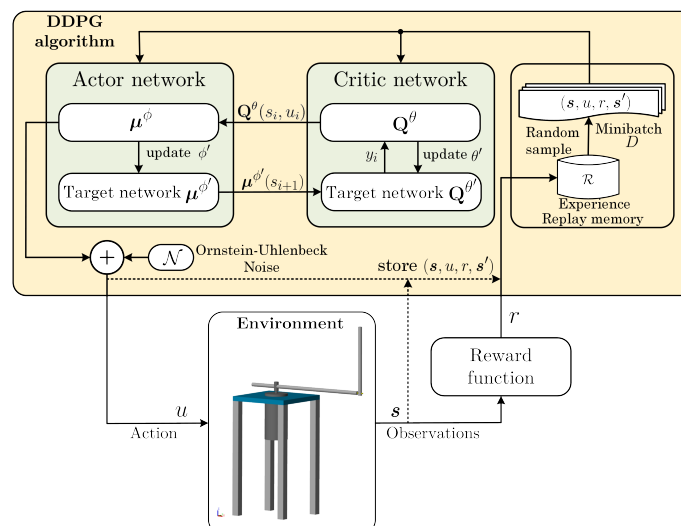


Figure 12. Block diagram of the DDPG algorithm.

The parameters of the critic  $Q^\theta$  are updated using this difference in a gradient form. The policy of the critic is updated using this gradient. Subsequently, both critic and actor targets update their parameters using a smoothing factor  $\eta$ . To update their parameters, the critic-target applies the formula  $\theta' \leftarrow \eta\theta + (1 - \eta)\theta'$ , while the actor-target network



uses  $\phi' \leftarrow \eta\phi + (1 - \eta)\phi'$ . The agent’s learning process continues in a repetitive manner until it converges a locally optimal or nearly optimal policy for the specified task.

#### 4.3. Design of the Reward Function

To stabilize the RIP, a reward function is designed with the following considerations: (i) the upward position of the pendulum is 0 [rad] and the downward position is  $\pi$  [rad]; (ii) the action  $u$  is bounded by  $u < |u_{\max}|$  with  $u_{\max} = \pm 10$  [Nm]; (iii) the observation signal, denoted as  $s$ , is represented by a vector defined as  $s = [q_1 \ \dot{q}_1 \ q_2 \ \dot{q}_2]^\top$ ; and (iv) a training episode terminates under the following conditions: (a)  $|q_1| > q_{1\max}$ , with  $q_{1\max} = \frac{\pi}{3}$  [rad]; (b)  $|q_2| > q_{2\max}$ , with  $q_{2\max} = \frac{\pi}{6}$  [rad], and (c)  $|u| > u_{\max}$ .

The reward function designed for this purpose is as follows:

$$r = -\alpha_1 \left[ \alpha_2 q_1^2 + \alpha_3 \dot{q}_1^2 + \alpha_4 q_2^2 + \alpha_5 \dot{q}_2^2 + \alpha_6 u_{t-1}^2 \right] - F \tag{11}$$

$$F = \begin{cases} B & \text{if } |q_1| > q_{1\max} \ \parallel \ |q_2| > q_{2\max} \ \parallel \ |u| > u_{\max} \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

where  $\alpha_i$  are tuning parameters,  $F$  is a numeric indicator (set to value  $B$ ) to interrupt the training process if any of the conditions set in (12) are true, and restart the training. The use of quadratic terms in the reward function enables training of a successful policy and simplifies the tuning process.

#### 4.4. Learning of the RL Agent

The RL Toolbox of MATLAB® is used to train the RL agent. The RL agent is limited to a maximum of 1500 training episodes. The elapsed time for each episode set to 10 [s]. The simulation is conducted on a PC with the following features: CPU (11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz 2.42 GHz) and RAM (12 GB). The Simulink diagram of the RL controller implementation is shown in Figure 13. The actor and critic neural networks are described in Table 6.

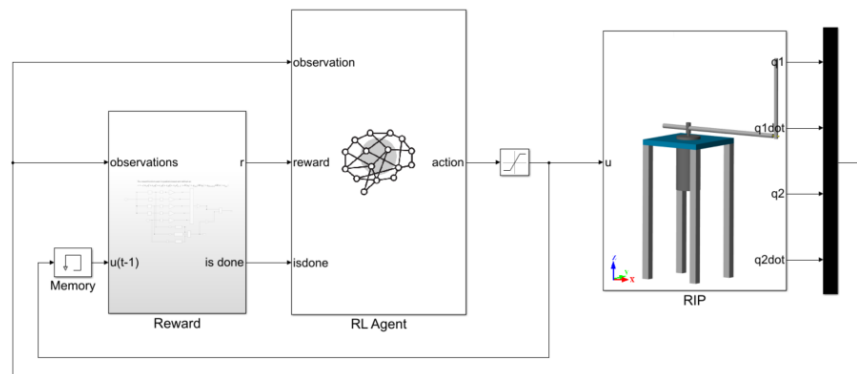


Figure 13. Simulink diagram of the RL controller.

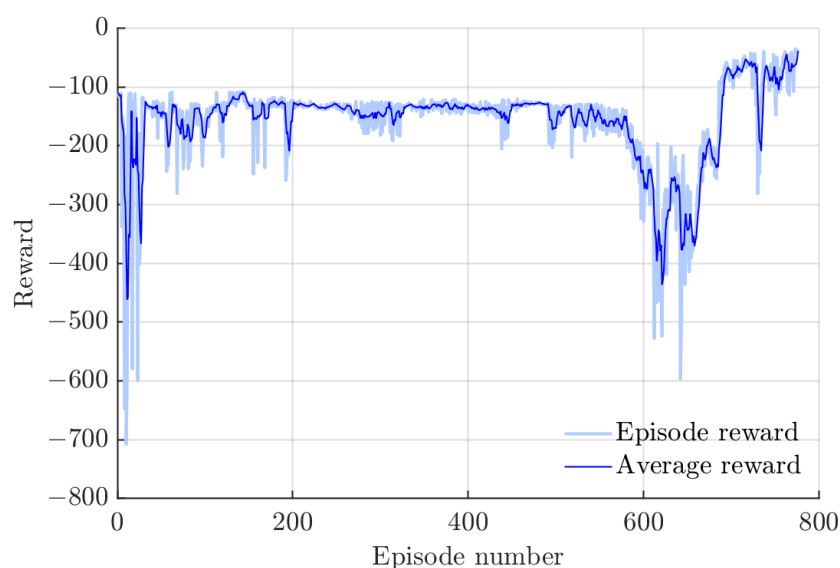
Table 6. Actor and critic neural network architectures.

Actor			Critic		
Layer	Neurons	Activation Function	Layer	Neurons	Activation Function
input	4	-	input	4	-
1	128	ReLU	1	128	ReLU
2	200	ReLU	2	400	ReLU
3	1	Tanh	3	1	Lineal

The initial conditions for each episode of the learning process are as follows:

$$q(0) = [q_1(0) \ \dot{q}_1(0) \ q_2(0) \ \dot{q}_2(0)]^\top = [0 \ 0 \ q_2(0) \ 0]^\top.$$

In each episode, the initial position  $q_2(0)$  assume random values of  $\pm \frac{\pi}{9}$ . The reward function parameters and hyperparameters of the RL algorithm are described in Table 7. The minibatch size determines the number of experiences utilized during each gradient update of the neural networks. The use of large mini-batches has the effect of reducing the variance when computing gradients; however, this is accompanied by an increase in the computational effort. The experience buffer defines the number of past experiences (rewards, states, and actions observed) that are stored and mixed with current experiences. In general, a relatively large value for this parameter is utilized in the majority of cases. The smoothing factor serves as a relaxation factor for the copied network parameters. This implies that the target values are constrained to evolve gradually, enhancing the stability of the learning process. A common value is to select  $\eta \ll 1$ . The learning rates of the actor and critic define the step size of the gradient update when optimizing network parameters. A low learning rate results in prolonged training times. Conversely, a high learning rate may yield suboptimal results or divergence. With respect to the exploration noise variance parameter, it has generally been observed that low values of this parameter are conducive to enhanced learning performance, although this is influenced by the specific environment. A typical value is to have  $\sigma_n \sqrt{T_s}$  between 1 and 10 percent of the action range for the Ornstein–Uhlenbeck noise. The rate of noise variance decay also exerts influence on the convergence of the algorithm. A low value of this parameter increases the extent of exploration. Parameter  $\alpha_1$  is a scaling factor that helps to achieve a stable learning process. Parameters  $\alpha_2, \dots, \alpha_5$  penalize or award the deviations of the states from the desired value and parameter  $\alpha_6$  defines the importance of past actions. The design of network architecture is a crucial aspect that requires careful consideration. While expanding the number of nodes and hidden layers can improve learning capabilities, it may also introduce problems such as overfitting and increased computational cost. The proper selection of the network architecture depends on the specific problem and environment, and it is often an empirical matter to determine the proper architecture. In this study, the selected architecture allows acceptable learning of the agent. All the configuration parameters are set in an “.m file” in MATLAB. As part of the learning process, the agent interacts with the RIP. The optimal policy is reached in 776 episodes over a five-episode window, considering an average reward criterion of  $r_{\text{avg}} < -40$ . Figure 14 shows the learning curve of the RL agent. This curve shows that in the initial phase of learning, the reward is bad owing to exploration. As the learning progresses, the reward increases and approaches the expected cumulative reward.



**Figure 14.** Curve of the RL agent learning process.

**Table 7.** Hyperparameters for the DDPG algorithm and reward function.

Symbol	Meaning	Value
$D$	Minibatch size	128
$\mathcal{R}$	Experience buffer length	$1 \times 10^6$
$\eta$	Smoothing factor	$1 \times 10^{-3}$
$\gamma$	Discount factor	0.99
	Actor learning rate	$5 \times 10^{-4}$
	Critic Learning rate	$1 \times 10^{-3}$
$\sigma_n$	Exploration noise variance	0.1
	Noise variance decay rate	$1 \times 10^{-5}$
$T_s$	Simulation time step	0.01 s
	Maximum steps in an episode	1000
$[\alpha_1, \dots, \alpha_6]$	Tuning parameters	[0.1, 10, 0.1, 20, 1, 0.5]
$B$	Numeric indicator	100

#### 4.5. FL Controller with RL Compensation

An FL controller with adaptive neural compensation (FL-ANC) is proposed in [21]. The control law is given by

$$u = -\hat{W}^T \mathbf{w}(\hat{V}^T \chi) - k_p y - \delta \text{sign}(y) \tag{13}$$

with  $k_p > 0$ ,  $\delta > 0$ , the term  $\hat{W}^T \mathbf{w}(\hat{V}^T \chi)$  represents the adaptive neural network that compensates the unknown dynamics. The output function is defined as

$$y = \dot{e}_1 + \Delta_1 e_1 + \dot{e}_2 + \Delta_2 e_2 \tag{14}$$

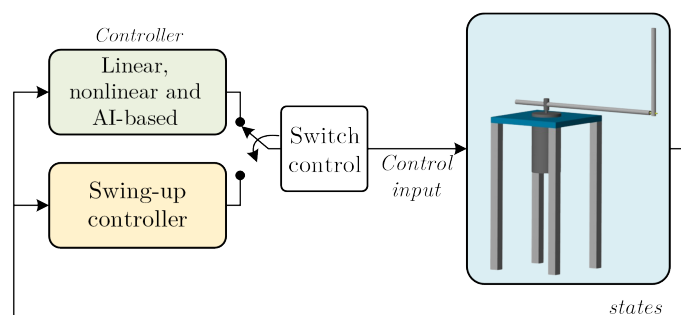
with  $e_1 = q_{1d} - q_1$ , and  $e_2 = q_{2d} - q_2$  and  $\Delta_1, \Delta_2 > 0$  being the adjust parameters. In this work, our proposal suggests substituting the term with ANNC given in (13) by the previously trained RL agent. Consequently, the FL controller incorporating RL-based compensation is referred to as FL-RL and is established as follows:

$$\begin{aligned} u_{FLRL} &= u_{FL} + u_{RL} \\ u_{FL} &= -k_p y - \delta \text{sign}(y) \end{aligned} \tag{15}$$

where  $u_{RL}$  is the action control provided by the RL agent.

### 5. Control Algorithms for Comparison

In this section, linear, nonlinear, and AI-based control schemes are proposed as comparison controllers. LQR, PID and model predictive control (MPC) are described as linear approaches, whereas SMC and FL are nonlinear control approaches. Finally, the RL agent and controllers (13) and (15) are considered as AI-based control approaches. For all controllers, a swing-up controller is considered, which brings the pendulum from its hanging position near the desired upright position and then switches it to a previously designed controller. A block diagram of the implemented controllers is shown in Figure 15.



**Figure 15.** Block diagram of implemented controllers.

### 5.1. LQR Controller

For the linear time invariant System (7), the LQR control law is subject to the optimal control problem designed to minimize the quadratic cost function [44]:

$$J(x, t) = \int_0^{\infty} (x^T Q x + u^T R u) d\tau. \quad (16)$$

In this case,  $Q$  penalizes the state variables and  $R$  penalizes the control signal. The unique minimum of the cost function can be obtained by solving the Algebraic Riccati Equation (ARE),

$$A^T S + SA - SBR^{-1}B^T S + Q = 0 \quad (17)$$

where  $S$  is known as the *kernel* matrix and is the only solution positive definite to the ARE such that the control gains  $K_{lqr}$  are obtained by  $K_{lqr} = R^{-1}B^T S$  and the control signal is

$$u = -K_{lqr}x(t). \quad (18)$$

Matrix  $Q$  must be  $Q \geq 0$  and  $R > 0$ . This implies that the scalar quantity  $x^T Q x$  is invariably positive or zero at each time instant for all state  $x(t)$  values, and the scalar quantity  $u^T R u$  is always positive for each time  $t$  for all values of  $u(t)$ . This ensures that  $J$  remains a relatively small quantity. Selecting  $Q$  to be a large value results in a smaller state  $x(t)$ . Conversely, selecting large values of  $R$  implies that the control input  $u(t)$  must be smaller in order to maintain a small value of  $J$ . This implies that larger values of  $Q$  typically result in the poles of the closed-loop system matrix  $A_c = (A - BK_{lqr})$  being situated further to the left in the  $s$ -plane, thereby facilitating a faster decay of the state to zero. Conversely, larger values of  $R$  imply a reduction in control effort, resulting in slower poles and larger values of the state  $x(t)$ . Therefore, selecting appropriate values of  $Q$  and  $R$  requires a trial-and-error process until a satisfactory response is obtained. The selected values for the weighted matrices are  $Q = \text{diag}\{1, 1, 1, 1\}$  and  $R = 1$ , which produce a relatively fast state response and low control input. MATLAB<sup>®</sup> command `lqr(A, B, Q, R)` is used to compute the control gains, where  $A$  and  $B$  are the coefficient matrices of Linear System (8). The gains computed are

$$K_{lqr} = [-1.0000 \quad -1.3805 \quad -11.2936 \quad -1.7981]. \quad (19)$$

### 5.2. PID Controller

To stabilize the angular positions of the RIP, the following PID controller is considered:

$$u = -k_{p1}e_1 - k_{i1} \int_0^t e_1 dt - k_{d1}\dot{e}_1 - k_{p2}e_2 - k_{i2} \int_0^t e_2 dt - k_{d2}\dot{e}_2 \quad (20)$$

where  $k_{p1}$  and  $k_{p2}$  are the proportional gains;  $k_{d1}$  and  $k_{d2}$  are the derivative gains; and  $k_{i1}$  and  $k_{i2}$  are the integral gains. The error signals are defined as  $e_1 = q_{d1} - q_1$  and  $e_2 = q_{d2} - q_2$  with  $q_{d1} = q_{d2} = 0$  and their corresponding derivatives are  $\dot{e}_1 = -\dot{q}_1$  and  $\dot{e}_2 = -\dot{q}_2$ . The corresponding selected gains are  $k_{p1} = 4.7306$ ,  $k_{p2} = 8.1237$ ,  $k_{d1} = 1.2306$ ,  $k_{d2} = 1.1306$ ,  $k_{i1} = 0.017$  and  $k_{i2} = 1.4570$ .

### 5.3. Model Predictive Control

Model predictive control (MPC) represents a set of control methods that utilize a model to predict the behavior of a system. Considering this prediction, MPC formulates an optimal control signal through the resolution of a constrained optimization problem. The MPC Toolbox of MATLAB<sup>®</sup> is used to design an MPC in order to stabilize the RIP. The design parameters for the MPC are described in Table 8.

**Table 8.** Parameter design for the MPC.

Parameter Description	Value
Sample time	0.01 [s]
Prediction horizon	50
Horizon	5
Scale factor for the control signal	100
Scale factor for $q_1$	1
Scale factor for $q_2$	0.7
Output weight for $q_1$	1.5
Output weight for $q_2$	1
Constraints on control signal	$-10 \leq u \leq 10$ [Nm]
Constraints on arm position	$-\pi \leq q_1 \leq \pi$ [rad]
Constraints on pendulum position	$-\pi/4 \leq q_2 \leq \pi/4$ [rad]

#### 5.4. FL Controller

The FL is a well-known nonlinear control methodology. The primary objective of this method is to transform a complex system into a simple equivalent model [41]. For this purpose, the nonlinear model of the RIP (2) can be expressed in the space-state form as

$$\frac{d}{dt}q_1 = \dot{q}_1 \quad (21)$$

$$\frac{d}{dt}\dot{q}_1 = f_{n1} + g_{n1}u \quad (22)$$

$$\frac{d}{dt}q_2 = \dot{q}_2 \quad (23)$$

$$\frac{d}{dt}\dot{q}_2 = f_{n2} + g_{n2}u \quad (24)$$

where

$$f_{n1} = \frac{1}{\det\{M(q)\}} [m_{22}z_1 - m_{12}z_2] \quad (25)$$

$$g_{n1} = \frac{1}{\det\{M(q)\}} [m_{22}] \quad (26)$$

$$f_{n2} = \frac{1}{\det\{M(q)\}} [-m_{22}z_1 + m_{11}z_2] \quad (27)$$

$$g_{n2} = \frac{1}{\det\{M(q)\}} [-m_{21}] \quad (28)$$

with

$$z_1 = -c_{11}\dot{q}_1 - c_{12}\dot{q}_2 - g_{12} - f_{11} \quad (29)$$

$$z_2 = -c_{21}\dot{q}_1 - c_{22}\dot{q}_2 - g_{21} - f_{21} \quad (30)$$

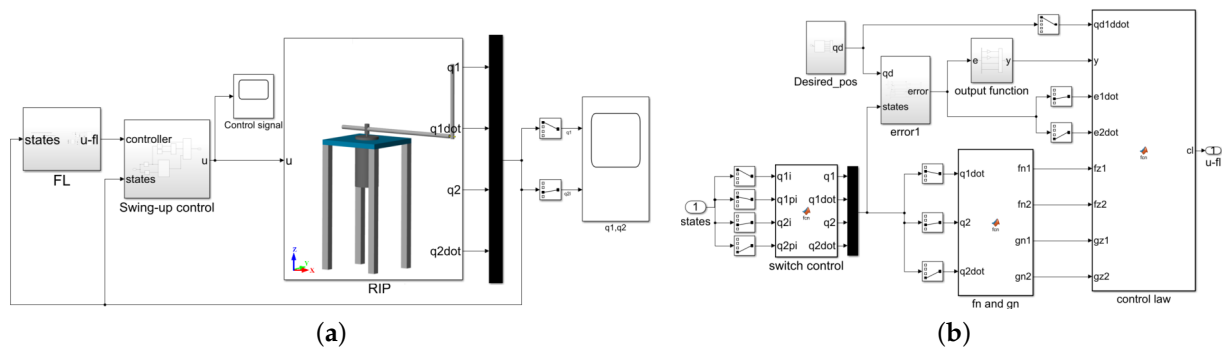
The following FL controller [16] was implemented to stabilize the angular positions of the RIP:

$$u = \frac{-\Delta_1\dot{e}_1 - \Delta_2\dot{e}_2 - \ddot{q}_{d1} + f_{n1} + f_{n2} - k_p y}{-(g_{n1} + g_{n2})} \quad (31)$$

The output function is

$$y = \Delta_1 e_1 + \Delta_2 e_2 + \dot{e}_1 + \dot{e}_2 \quad (32)$$

where the error signals are defined as  $e_1 = q_{d1} - q_1$  and  $e_2 = q_{d2} - q_2$  with  $q_{d1} = q_{d2} = 0$ . The Simulink model for this controller is shown in Figure 16. The selected gains for this controller are the same as those in [16] with  $\Delta_1 = 4$ ,  $\Delta_2 = 6$  and  $k_p = 2$ .



**Figure 16.** Simulink diagram of the FL controller: (a) control scheme implementation, (b) control law implementation.

5.5. Sliding Mode Controller

SMC does not require knowledge of the dynamics in (2). We consider the following SMC law,

$$u = K_s \text{sign}(v) \tag{33}$$

with  $v = K_m x$  and  $x \in \mathbb{R}^4$  being the state vector. The switching control is sufficient to stabilize the RIP when the states are in proximity to the zero position, and the gain of the sliding mode is greater than the upper limit of all unknown dynamics. This implies that the states are on the sliding surface [45,46]. To avoid chattering in (33), instead of the  $\text{sign}(\cdot)$ , the following saturation function can be used:

$$\text{sat}(v) = \begin{cases} 1 & \text{if } v > \Delta \\ \lambda v & \text{if } |v| \leq \Delta \\ -1 & \text{if } v < -\Delta \end{cases} \quad \lambda = \frac{1}{\Delta}, \tag{34}$$

Parameter  $\Delta$  is known as the boundary layer. The controller gains are selected as follows:  $K_m = [-1.0000 \quad -1.3805 \quad -11.2936 \quad -1.7981]$ , which are the same gains used in the LQR controller, while  $K_s = 1.2$  and  $\Delta = 0.1$ .

5.6. Feedback Linearization Controller with ANNC

We consider an FL-ANC (13). The estimated input weights are contained in the matrix  $\hat{V} \in \mathbb{R}^{8 \times L}$ ,  $\hat{W} \in \mathbb{R}^L$  contains the estimated output weights, the activation functions are contained in the vector represented by  $w(\cdot) \in \mathbb{R}^L$ , which uses the  $\tanh(\cdot)$  function in the hidden layer as the activation function,  $L$  is the number of neurons, and  $\chi \in \mathbb{R}^8$  contains the input signals to the neural network and is defined as

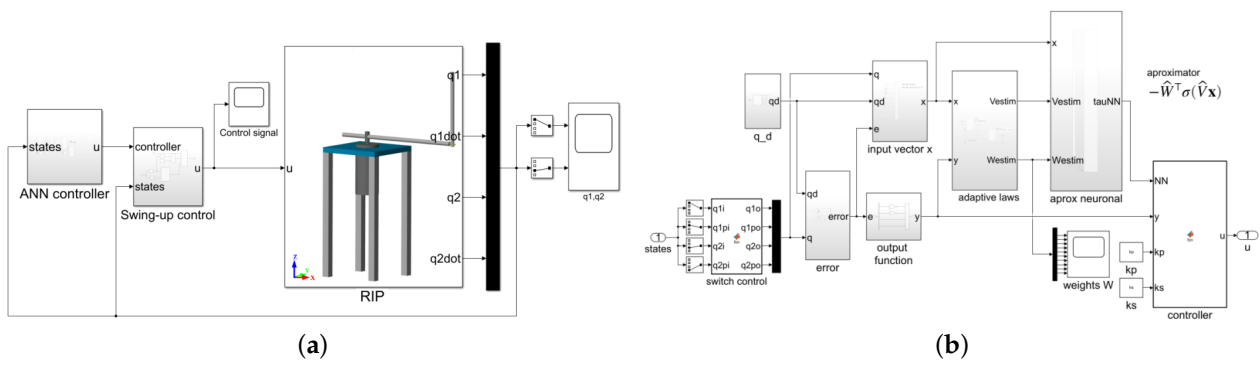
$$\chi = [q_1 \quad q_2 \quad \dot{q}_1 \quad \dot{q}_2 \quad \ddot{q}_1 \quad \ddot{q}_2 \quad \ddot{q}_{d1} \quad 1]^T. \tag{35}$$

The adaptive laws for the weights of the neural network are given by

$$\begin{aligned} \dot{\hat{W}} &= \alpha y N w(\hat{V}^T \chi) - \alpha y N w'(\hat{V}^T \chi) \hat{V}^T \chi \\ \dot{\hat{V}} &= \alpha y \Lambda \chi \hat{W}^T w'(\hat{V}^T \chi) \end{aligned}$$

where  $\alpha \in \mathbb{R}_+$ ,  $N \in \mathbb{R}^{L \times L}$ ,  $\Lambda \in \mathbb{R}^{8 \times 8}$ , and  $w'(\cdot) \in \mathbb{R}^{L \times L}$  is the Jacobian matrix containing the derivatives of the activation function [47]. The Simulink model for this control scheme is depicted in Figure 17. The parameters of the controller are the same as those in [21] with  $k_p = 1.05$ ,  $\delta = 0.035$ ,  $\Delta_1 = 3$ , and  $\Delta_2 = 8$ . The hidden layer of neurons is configured to have  $L = 10$  elements. The parameters of the adaptive law are established as  $\alpha = 1$ ,  $N = 1.05 I_{10 \times 10}$ , and  $\Lambda = 8.53 I_{8 \times 8}$ , and the initial weights are set randomly in the range of  $[-1, 1]$ .





**Figure 17.** Simulink diagram of the FL-ANC controller: (a) control scheme implementation, (b) adaptive neural network controller subsystem block.

5.7. Swing-Up Controller and Switch Control

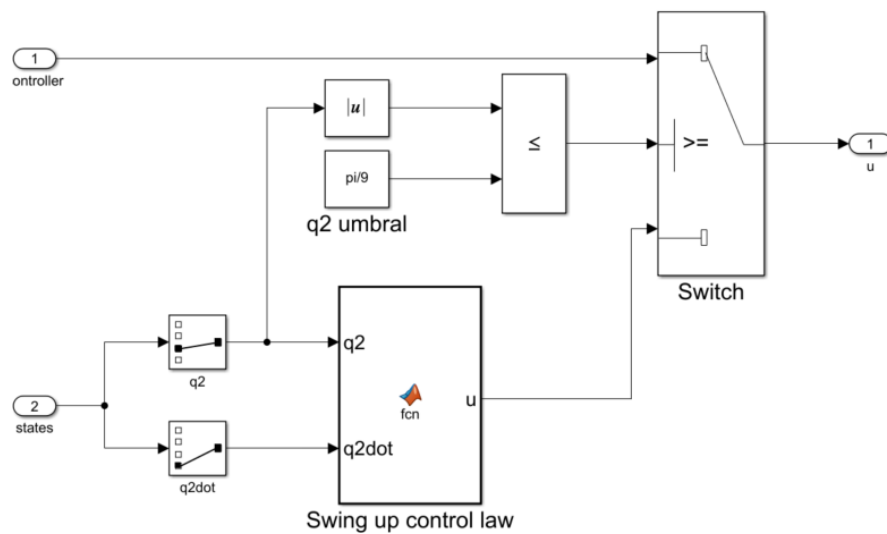
The swing up is a well-known problem in pendular systems. In this study, we slightly modified the energy controller law [48]. The swing-up control law is expressed as follows:

$$u_{sw} = ng \tanh(k(E - E_o)\dot{q}_2 \cos(q_2)) \tag{36}$$

where  $n$  and  $k$  are design parameters,  $g$  is the gravitational constant, and  $E_o$  is the energy of the pendulum in the upward vertical position. Therefore, it is reasonable to assume that  $E_o = 0$  when the pendulum is vertically positioned. The energy function of the pendulum is

$$E = \frac{1}{2}J_2\dot{q}^2 + p_2(\cos(q_2) - 1) \tag{37}$$

The moment of inertia of the pendulum is computed as  $J_2 = \frac{1}{12}p_2 = 0.000313$  [kg·m<sup>2</sup>]. After several trials, the selected gains for the swing-up controller are set to  $k = 0.5$  and  $n = 0.05$ . The choice of the initial pendulum position from its downward position depends on the choice value for parameter  $n$ . With these values, it is possible to start the initial position of the pendulum from  $q_2(0) = 17\pi/18$  [rad]. The switch control block guarantees that while swing up occurs, the state signals remain at zero, and once the switching threshold is reached, they take on their current values. The threshold for switching is selected as  $|q_2| \leq \frac{\pi}{9}$  [rad]. Figure 18 shows the Simulink diagram of the swing-up controller implemented for all control schemes.



**Figure 18.** Simulink block diagram of swing-up controller.


## 6. Simulation Results and Discussion

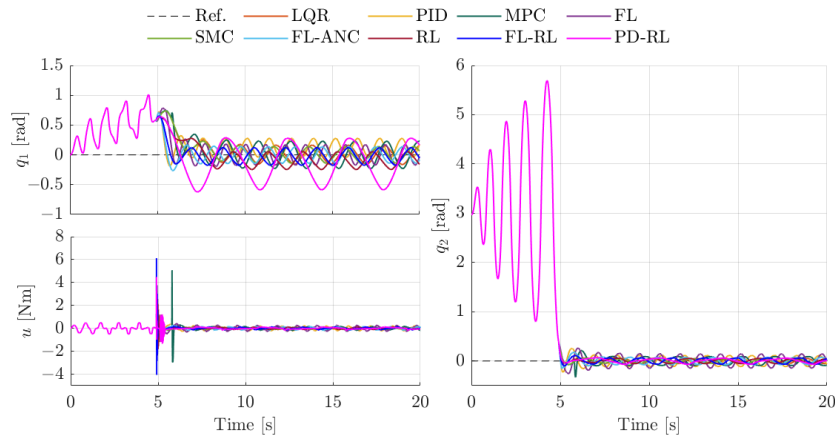
### 6.1. Simulation Specifications

To assess the efficacy of the suggested control method and highlight the usefulness of the proposed Simscape RIP model, simulation results in two scenarios are studied. Additionally, a proportional derivative controller combined with the RL-agent (PD-RL) is implemented to validate the proposed method. This controller is defined as  $u = -8.4e_2 - 1.05\dot{e}_2 + u_{RL}$  and its gains are equivalent to the proportional and derivative terms of the pendulum position error from the FL-ANC controller. First, the swing-up controller is considered to bring the pendulum from its bottom position to its vertical position, and then it switches to one of the proposed controllers. In the second scenario, the pendulum position starts close to the vertical position, and an external disturbance is included in the control input to test the ability of each controller to maintain pendulum stability under external forces. Simulations are conducted in the Simscape environment in MATLAB<sup>®</sup>/Simulink 2021a. For the first scenario, the initial conditions are set at  $\mathbf{q}(0) = [q_1(0) \ \dot{q}_1(0) \ q_2(0) \ \dot{q}_2(0)]^T = [0 \ 0 \ 17\pi/18 \ 0]^T$ . In the second scenario, the initial conditions are set at  $\mathbf{q}(0) = [q_1(0) \ \dot{q}_1(0) \ q_2(0) \ \dot{q}_2(0)]^T = [0 \ 0 \ 11\pi/90 \ 0]^T$ . A fixed-step backward Euler solver with a sampling interval of 0.01 s is used. The performance is evaluated in terms of the absolute value of the maximum error, the root-mean-square (RMS) value of the error positions, the RMS value of the control signal  $u(t)$ , maximum overshoot, and settling time.

### 6.2. Simulation Results: Non-Perturbed Case

The results of the controllers in stabilizing the pendulum to its upright position once the threshold value is reached are shown in Figure 19. It can be observed that the swing-up control brings the pendulum position to the switching threshold at approximately 5 s. All the proposed controllers effectively stabilize the angular positions of both the arm and the pendulum. The effect of nonlinear friction on the angular positions of each joint is noteworthy in the steady-state oscillations. Table 9 presents a comparative analysis of pendulum position errors. To quantify the maximum error, the first interval of time  $5 \leq t \leq 10$  evidences the performance of each controller in attaining a stable position within a shorter time. The PD-RL controller exhibits the lowest absolute value of maximum error in the angular displacement of the horizontal arm. Moreover, the maximum error of the FL-RL controller in the angular displacement of the horizontal arm is reduced by 8.56% compared to the FL-ANC controller. The FL-RL controller shows the lowest maximum absolute error of the angular position of the pendulum with respect to the rest of the controllers, reducing its value by 37.52% with respect to the FL-ANC controller. The second interval of time,  $5 \leq t \leq 20$ , indicates the effectiveness of each controller after the swing-up controller, during which the RIP is stabilized. In this interval, the RMS index of the error positions indicates their performance relative to the desired value. A small value of this index indicates a better controller performance. FL-ANC exhibits the best performance in terms of the  $\text{RMS}\{e_1\}$ , demonstrating an improvement of 2.47% and 26.46% relative to the RL and FL-RL controllers, respectively. The SMC controller achieves the best result in terms of  $\text{RMS}\{e_2\}$ , with improvements of 42.52%, 27.09%, and 2.19% relative to the FL-ANC, FL-RL, and RL controllers, respectively. On the other hand, FL-RL improves by 21.19% in terms of  $\text{RMS}\{e_2\}$  relative to the FL-ANC controller. The complete performance of the controllers is evaluated in the interval  $5 \leq t \leq 20$ ; the RMS value of the error  $e = [e_1 \ e_2]^T$  reveals that the AI-based controllers exhibit the best performance. Accordingly, the best performance is obtained using the FL-RL controller and demonstrates an improvement of 0.90% and 24.51% over the FL-ANC and RL controllers, respectively. In terms of lower power consumption, the RMS value of the control signal  $u(t)$  of the LQR controller provides the best performance, and the FL-RL controller improves the  $\text{RMS}\{u\}$  performance index by 31.13% and 33.81% relative to the FL-ANC and RL controllers, respectively. The RL controller exhibits the lowest peak value of overshoot. Moreover, the FL-RL controller improves its overshoot by 38.16% relative to the FL-ANC controller and the PD-RL controller improves its overshoot

by 4.19% relative to the FL-ANC controller. The settling time is defined as the time at which the angular position  $q_2$  remains within the limits defined by its upper and lower values in the steady state. The FL-RL controller shows the best settling time with 6.09 s and improves by 32.33% and 27.59% over the FL-ANC and RL controllers, respectively. A virtual simulation for the non-perturbed case was carried out and is available by clicking on the next icon  <https://www.youtube.com/watch?v=Ul-iqBunxal&t=54s> (accessed on 21 November 2024).



**Figure 19.** Time evolution of controller signals. Left-hand side upper plot: arm position  $q_1$ . Left-hand side bottom plot: control signal  $u$ . Right-hand side plot: pendulum position  $q_2$ .

**Table 9.** Quantitative index performance and quantitative metric: non-perturbed case.

Index	Linear			Nonlinear		AI			
	LQR	PID	MPC	FL	SMC	FL-ANC	RL	FL-RL*	PD-RL*
$\max \{ e_1 \}[\text{rad}]$ $5 < t \leq 10$	0.7422	0.7239	0.7421	0.7849	0.7363	0.7135	0.6420	0.6524	<b>0.6389</b>
$\text{RMS} \{e_1\}[\text{rad}]$ $5 \leq t \leq 20$	0.2099	0.1879	0.2351	0.1931	0.2038	<b>0.1420</b>	0.1931	0.1456	0.3516
$\max \{ e_2 \}[\text{rad}]$ $5 < t \leq 10$	0.2205	0.2549	0.3316	0.2633	0.2107	0.1735	0.1299	<b>0.1084</b>	0.1406
$\text{RMS} \{e_2\}[\text{rad}]$ $5 \leq t \leq 20$	0.0401	0.0919	0.0770	0.1142	<b>0.0358</b>	0.0623	0.0366	0.0491	0.0396
$\text{RMS} \{e\}[\text{rad}]$ $5 \leq t \leq 20$	0.2099	0.2092	0.2474	0.2244	0.2070	0.1551	0.2036	<b>0.1537</b>	0.3538
$\text{RMS} \{u\}[\text{Nm}]$ $5 \leq t \leq 20$	<b>0.0573</b>	0.1381	0.2364	0.1677	0.1512	0.1012	0.1056	0.0699	0.1320
$\text{M.O.}\{q_2\}[\text{rad}]$ $5 \leq t \leq 20$	0.0609	0.2549	0.2123	0.2633	0.0567	0.1753	<b>0.0566</b>	0.1084	0.0681
$\text{S.T.}\{q_2\}[\text{s}]$ $5 \leq t \leq 20$	8.63	7.24	6.45	10.63	8.55	9.00	8.41	<b>6.09</b>	7.75


\* Proposed method; M.O.(Max. overshoot); S.T. (Settling time). The best results are denoted in bold format.

### 6.3. Simulation Results: Perturbed Case

External force  $u_d$  is introduced to evaluate the robustness of the proposed controllers. In this case, the swing-up controller is not considered, because the aim is to evaluate the robustness in the steady state. This force is added to the control signal as  $u = u_c + u_d$ .

In this context,  $u_c$  corresponds to the control signal generated by the linear, nonlinear, or AI-based controller, while  $u_d$  is the disturbance defined as

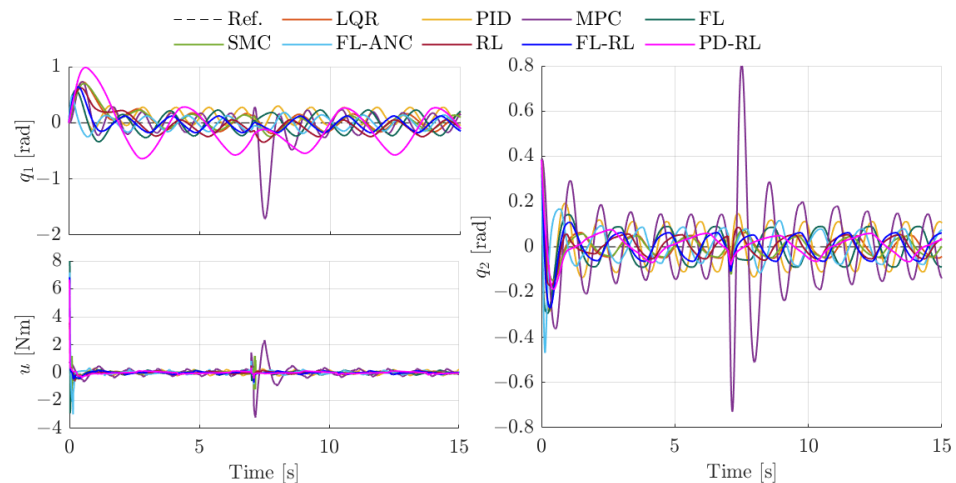
$$u_d = \begin{cases} 1.5 \text{ [Nm]} & 7 \leq t \leq 7.1, \\ 0 \text{ [Nm]} & \text{otherwise.} \end{cases}$$

Figure 20 illustrates the temporal progression of the angular positions and control signal in the presence of an external force. All controllers remain stable under this condition. According to Table 10, the FL-RL controller shows the best result in terms of  $\max\{|e_1|\}$ , with improvements of 7.33%, 7.28%, 25.58%, and 28.08% over the LQR, FL-ANC, MPC, and SMC controllers, respectively. The best performance in relation to the  $\text{RMS}\{e_1\}$  index is achieved by the FL-RL controller, outstanding of the remaining controllers, and performance is improved by 1.83%, 4.86%, 33.81%, and 33.68% relative to the FL-ANC, SMC, MPC, and RL controllers, respectively. For the angular position of the pendulum, the PD-RL controller shows the best result in terms of  $\max\{|e_2|\}$  improving by 39.07%, 22.12%, and 35.07% compared to the MPC, RL, and FL-RL controllers, respectively. The best performance for the pendulum position in terms of the  $\text{RMS}\{e_2\}$  index is achieved by the SMC and PD-RL controllers. In addition, the MPC controller exhibits the best result in terms of the maximum overshoot after the external disturbance, followed by the FL-RL controller. The best performance in terms of  $\text{RMS}\{e\}$  index is obtained by the FL-RL controller. Finally, in terms of  $\text{RMS}\{u(t)\}$  index, the PD-RL controller provides the best performance. A virtual simulation for the perturbed case is available by clicking on the next icon  <https://www.youtube.com/watch?v=CMeSbsyRoEI&t=28s> (accessed on 21 November 2024).

**Table 10.** Quantitative index performance and quantitative metric: perturbed case.

Index	Linear			Nonlinear		AI			
	LQR	PID	MPC	FL	SMC	FL-ANC	RL	FL-RL*	PD-RL*
$\max\{ e_1 \}$ [rad] $7 < t \leq 15$	0.1896	0.2990	0.2361	1.7100	0.2443	0.1895	0.3469	<b>0.1757</b>	0.5733
$\text{RMS}\{e_1\}$ [rad] $7 \leq t \leq 15$	0.1107	0.1601	0.1538	0.3748	0.1070	0.1037	0.1535	<b>0.1018</b>	0.3045
$\max\{ e_2 \}$ [rad] $7 < t \leq 15$	0.0724	0.1464	0.1167	0.8072	0.1223	0.1158	0.0913	0.1095	<b>0.0711</b>
$\text{RMS}\{e_2\}$ [rad] $7 \leq t \leq 15$	0.0383	0.0847	0.0657	0.2252	<b>0.0366</b>	0.0606	0.0403	0.0462	0.0368
M.O. $\{q_2\}$ [rad] $7 \leq t \leq 15$	0.0625	0.1464	<b>0.0165</b>	0.8072	0.0742	0.0788	0.0851	0.0522	0.0755
$\text{RMS}\{e\}$ [rad] $7 \leq t \leq 15$	0.1171	0.1811	0.1672	0.4372	0.1131	0.1201	0.1587	<b>0.1118</b>	0.3067
$\text{RMS}\{u\}$ [Nm] $7 \leq t \leq 15$	0.0684	0.1362	0.1053	0.5569	0.1331	0.0995	0.0685	0.0697	<b>0.0633</b>

\* Proposed method; M.O.(Max. overshoot); The best results are denoted in bold format.



**Figure 20.** Time evolution of controller signals under an external force. Left-hand side upper plot: arm position  $q_1$ . Left-hand side bottom plot: control signal  $u$ . Right-hand side plot: pendulum position  $q_2$ .

#### 6.4. Discussion

In summary, modeling and control of the RIP is not a simple task. The creation of a comprehensive model that accurately represents the dynamic and mechanical properties of a real system is a highly advantageous undertaking. On the other hand, although control strategies can be diverse and varied, precise and adaptive control is required to maintain stability and balance. The linear and nonlinear approaches implemented in this study deteriorate their performance when external forces affect the system. AI controllers, such as those based on RL, are not required to have knowledge of the dynamical model of the system. Through interaction with the system and observation of its response, they can find a control policy that satisfies the control requirements. Nonetheless, the primary challenges associated with the implementation of RL methods in the RIP lie in the adequate design of the reward function and the architecture of the actor and critic, as well as the precise tuning of their associated hyperparameters. Consequently, the experience and expertise of the designer play an important role for the success of this approach. In this work, the proposed RL controller was able to stabilize the RIP without prior knowledge of the system. Moreover, this controller was merged with both FL and PD controllers to compensate for unknown dynamics, and its performance was improved. Simulation outcomes obtained in a virtual environment provide the basis for designing and validating in an experimental platform. Furthermore, the results obtained from the SMC approach suggest the development of an RL control strategy combined with SMC, as reported in [49]. Another important issue is the reward function. Different shapes of reward functions, such as continuous and sparse rewards or those that include exponential and Gaussian functions, can be designed to enhance the learning process and improve the performance of the controller [30,50]; however, this aim is beyond the scope of this work.

#### 7. Conclusions

This work details step-by-step procedure for modeling and designing a RIP within the Simscape™ Multibody™ environment. The parameters that define the mathematical model of the designed system are identified to confirm the accuracy of the Simscape model. A control framework that integrates RL with other controllers such FL and PD is proposed. To demonstrate the usefulness of the Simscape model and the proposed control framework, a comparison coming from the performance of some linear, nonlinear, and AI-based controllers is carried out. The results obtained confirm that AI-based controllers offer a superior alternative for controlling the RIP. Furthermore, the proposed methodology shows promising results for controlling the RIP. The implementation of AI-based controllers such as RL in virtual environments diminishes the risk of damaging the experimental

prototype during the learning stages. The latter is advantageous since it allows a safe training of the agent, which can be then transferred to the real plant for its evaluation in real time. In future work, this methodology will be applied to the trajectory-tracking problem in UMS, which is still an outstanding problem in RL.

**Author Contributions:** Conceptualization, R.H., R.G.-H. and F.J.; methodology, R.H., R.G.-H. and F.J.; software, R.H.; validation, R.H., R.G.-H. and F.J.; formal analysis, R.H., R.G.-H. and F.J.; investigation, R.H., R.G.-H. and F.J.; resources, R.H., R.G.-H. and F.J.; writing—original draft preparation, R.H. and R.G.-H.; writing—review and editing, R.H., R.G.-H. and F.J.; visualization, R.H., R.G.-H. and F.J.; supervision, R.G.-H. and F.J.; project administration, R.G.-H.; funding acquisition, R.G.-H. and F.J. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was supported by TecNM projects and CONAHCYT under CVU 102856.

**Data Availability Statement:** The Simulink file for the Simscape prototype is available upon reasonable request from the first author.

**Acknowledgments:** The authors thank all the staff in Division de Estudios de Posgrado e Investigacion del Tecnológico Nacional de México/I.T. La Laguna and the first author thanks the TecNM/I.T.S. de San Pedro de las Colonias for their support.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

UMS	Underactuated Mechanical System
RIP	Rotary Inverted Pendulum
LQR	Linear Quadratic Regulator
PID	Proportional–Integral–Derivative
FL	Feedback Linearization
SMC	Sliding Mode Control
AI	Artificial Intelligence
ANNC	Adaptive Neural Network Compensation
FL-ANC	Feedback Linearization Controller with Adaptive Neural Compensation
ML	Machine Learning
RL	Reinforcement Learning
FL-RL	Feedback Linearization Controller with Reinforcement Learning Compensation
PD-RL	Proportional Derivative Controller with Reinforcement Learning Compensation
MPC	Model Predictive Control
IWP	Inertia Wheel Pendulum
TORA	Translational Oscillator with Rotational Actuator
ANFIS	Adaptive Neuro-Fuzzy Inference System
PSO	Particle Swarm Optimization
UAV	Unmanned Aerial Vehicle
IDA-PBC	Interconnection and Damping Assignment Passivity-Based Control
DNN	Deep Neural Network
DDPG	Deep Deterministic Policy Gradient
PPO	Proximal Policy Optimization
SAC	Soft Actor Critic
3D	Three-Dimensional
CAD	Computer-Aided Design
DOF	Degrees of Freedom
RMS	Root Mean Square



### Nomenclature

$q_i$	Angular position at joint $i$	$Q^\theta(u, s)$	Critic neural network
$\dot{q}_i$	Angular velocity at joint $i$	$\mu^\phi(s)$	Target actor network
$u$	Control input	$Q^{\theta'}(u, s)$	Target critic network
$f_i$	Friction force at joint $i$	$\mathcal{R}$	Experience buffer length
$f_{si}$	Stribeck friction coefficient at joint $i$	$D$	Minibatch size
$f_{ci}$	Coulomb friction coefficient at joint $i$	$r$	Reward function
$f_{vi}$	Viscous friction coefficient at joint $i$	$\gamma$	Discount factor
$\omega_{si}$	Stribeck velocity threshold at joint $i$	$e_i$	Position error in joint $i$
$\omega_{brki}$	Breakaway friction velocity at joint $i$	$\dot{e}_i$	Velocity error in joint $i$
$\omega_c$	Coulomb velocity threshold	$k_p, \delta, \alpha, \Delta_i$	Control gains
$p_i$	Lumped parameter $i$	$N, \Lambda$	Gain matrices
$x$	State vector	$A, B$	Matrices of the linear system
$s$	Observations vector	$Q, R$	Weight matrices
$\mu^\phi(s)$	Actor neural network	$K_{lqr}, K_m, K_s$	Gain vectors

### References

- Spong, M.W.; Hutchinson, S.; Vidyasagar, M. *Robot Modeling and Control*, 2nd ed.; John Wiley & Sons: Hoboken, NJ, USA, 2020; p. 608. Available online : <https://www.wiley.com/en-us/Robot+Modeling+and+Control%2C+2nd+Edition-p-9781119524045> (accessed on 25 September 2024).
- Walsh, C.J.; Pasch, K.; Herr, H. An autonomous, underactuated exoskeleton for load-carrying augmentation. In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 9–15 October 2006; pp. 1410–1415. [[CrossRef](#)]
- Arimoto, S. Intelligent control of multi-fingered hands. *Annu. Rev. Control* **2004**, *28*, 75–85. 2003.12.001. [[CrossRef](#)]
- Gupta, S.; Kumar, A. A brief review of dynamics and control of underactuated biped robots. *Adv. Robot.* **2017**, *31*, 607–623. [[CrossRef](#)]
- Tsiotras, P.; Luo, J. Control of underactuated spacecraft with bounded inputs. *Automatica* **2000**, *36*, 1153–1169. [[CrossRef](#)]
- Yao, Q.; Li, Q.; Jahanshahi, H. Convergence guaranteed attitude control of underactuated spacecraft using two control torques. *Adv. Space Res.* **2024**, *73*, 2663–2673. [[CrossRef](#)]
- Lu, J.; Meng, Z. Underactuated attitude-orbit coupling control for micro-satellite based on a single orbital thruster. *IEEE Trans. Aerosp. Electron. Syst.* **2024**, *60*, 2082–2092. [[CrossRef](#)]
- Duan, Y.; Xiang, X.; Liu, C.; Yang, L. Double-loop LQR depth tracking control of underactuated AUV: Methodology and comparative experiments. *Ocean. Eng.* **2024**, *300*, 117410. [[CrossRef](#)]
- Wang, Z.; Xiang, X.; Duan, Y.; Yang, S. Adversarial deep reinforcement learning based robust depth tracking control for underactuated autonomous underwater vehicle. *Eng. Appl. Artif. Intell.* **2024**, *30*, 107728. [[CrossRef](#)]
- Kinoshita, D.; Uchida, M.; Matsumoto, I. Swing up Control of the Pendubot with Elbow Joint Extended Using Energy-Based Methods. *Adv. Sci. Technol.* **2024**, *139*, 77–85. [[CrossRef](#)]
- Lei, C.; Li, R.; Zhu, Q. Design and stability analysis of semi-implicit cascaded proportional-derivative controller for underactuated cart-pole inverted pendulum system. *Robotica* **2024**, *42*, 87–117. [[CrossRef](#)]
- Furuta, K.; Yamakita, M.; Kobayashi, S.; Nishimura, M. A new inverted pendulum apparatus for education. In *Advances in Control Education*; Kheir, N.A., Franklin, J.F., Rabins, M.J., Eds.; IFAC Symposia Series: Boston, MA, USA, 1992; pp. 133–138. [[CrossRef](#)]
- Hamza, M.F.; Yap, H.J.; Choudhury, I.A.; Isa, A.I.; Zimit, A.Y.; Kumbasar, T. Current development on using Rotary Inverted Pendulum as a benchmark for testing linear and nonlinear control algorithms. *Mech. Syst. Signal Process.* **2019**, *116*, 347–369. [[CrossRef](#)]
- Chawla, I.; Singla, A. Real-time control of a rotary inverted pendulum using robust LQR-based ANFIS controller. *Int. J. Nonlinear Sci. Numer. Simul.* **2018**, *19*, 379–389. [[CrossRef](#)]
- Pujol-Vazquez, G.; Acho, L.; Mobayen, S.; Nápoles, A.; Pérez, V. Rotary inverted pendulum with magnetically external perturbations as a source of the pendulum's base navigation commands. *J. Frankl. Inst.* **2018**, *355*, 4077–4096. [[CrossRef](#)]
- Moreno-Valenzuela, J.; Aguilar-Avelar, C. *Motion Control of Underactuated Mechanical Systems*; Springer: Cham, Switzerland, 2018; Volume XI, p. 223. Available online: <https://link.springer.com/book/10.1007/978-3-319-58319-8> (accessed on 25 September 2024).
- Shah, I.; Abbasi, W.; Alhussein, M.; Khan, I.; Ur Rehman, F.; Anwar, M.S.; Aurangzeb, K. Robust Approach for Global Stabilization of a Class of Underactuated Mechanical Systems in Presence of Uncertainties. *Complexity* **2023**, *2023*, 8207980. [[CrossRef](#)]
- Sandoval, J.; Kelly, R.; Santibáñez, V. Interconnection and damping assignment passivity-based control of a class of underactuated mechanical systems with dynamic friction. *Int. J. Robust Nonlinear Control* **2011**, *21*, 738–751. [[CrossRef](#)]

19. Mofid, O.; Alattas, K.A.; Mobayen, S.; Vu, M.T.; Bouteraa, Y. Adaptive finite-time command-filtered backstepping sliding mode control for stabilization of a disturbed rotary-inverted-pendulum with experimental validation. *J. Vib. Control* **2023**, *29*, 1431–1446. [[CrossRef](#)]
20. Wang, Y.; Mao, W.; Wang, Q.; Xin, B. Fuzzy Cooperative Control for the Stabilization of the Rotating Inverted Pendulum System. *J. Adv. Comput. Intell. Inform.* **2023**, *27*, 360–371. [[CrossRef](#)]
21. Moreno-Valenzuela, J.; Aguilar-Avelar, C.; Puga-Guzmán, S.A.; Santibáñez, V. Adaptive neural network control for the trajectory tracking of the Furuta pendulum. *IEEE Trans. Cybern.* **2016**, *46*, 3439–3452. [[CrossRef](#)]
22. Hong, M.R.; Kang, S.; Lee, J.G.; Seo, S.; Han, S.; Koh, J.S.; Kang, D. Optimizing Reinforcement Learning Control Model in Furuta Pendulum and Transferring it to Real-World. *IEEE Access* **2023**, *11*, 95195–95200. [[CrossRef](#)]
23. Brown, D.; Strube, M. Design of a Neural Controller Using Reinforcement Learning to Control a Rotational Inverted Pendulum. In Proceedings of the 21st International Conference on Research and Education in Mechatronics (REM), Cracow, Poland, 9–11 December 2020; IEEE: New York, NY, USA, 2020. [[CrossRef](#)]
24. Liu, X.; Yuan, Z.; Gao, Z.; Zhang, W. Reinforcement Learning-Based Fault-Tolerant Control for Quadrotor UAVs Under Actuator Fault. *IEEE Trans. Ind. Inform.* **2024**, 1–10. [[CrossRef](#)]
25. Wang, L.; Jia, L.; Zou, T.; Zhang, R.; Gao, F. Two-dimensional reinforcement learning model-free fault-tolerant control for batch processes against multi-faults. *Comput. Chem. Eng.* **2025**, *192*, 108883. [[CrossRef](#)]
26. Wang, L.; Li, X.; Zhang, R.; Gao, F. Reinforcement Learning-Based Optimal Fault-Tolerant Tracking Control of Industrial Processes. *Ind. Eng. Chem. Res.* **2023**, *62*, 16014–16024. [[CrossRef](#)]
27. Li, X.; Luo, Q.; Wang, L.; Zhang, R.; Gao, F. Off-policy reinforcement learning-based novel model-free minmax fault-tolerant tracking control for industrial processes. *J. Process. Control* **2022**, *115*, 145–156. [[CrossRef](#)]
28. Zabihifar, S.H.; Yushchenko, A.S.; Navvabi, H. Robust control based on adaptive neural network for Rotary inverted pendulum with oscillation compensation. *Neural Comput. Appl.* **2020**, *32*, 14667–14679. [[CrossRef](#)]
29. Ben Hazem, Z. Study of Q-learning and deep Q-network learning control for a rotary inverted pendulum system. *Discov. Appl. Sci.* **2024**, *6*, 49. [[CrossRef](#)]
30. Guida, D.; Manrique Escobar, C.A.; Pappalardo, C.M. A Reinforcement Learning Controller for the Swing-Up of the Furuta Pendulum. In *New Technologies, Development and Application III*; Springer International Publishing: Berlin/Heidelberg, Germany, 2020; pp. 31–38.3. [[CrossRef](#)]
31. Bhourji, R.S.; Mozaffari, S.; Alirezaee, S. Reinforcement Learning DDPG–PPO Agent-Based Control System for Rotary Inverted Pendulum. *Arab. J. Sci. Eng.* **2024**, *49*, 1683–1696. [[CrossRef](#)]
32. Puriel Gil, G.; Yu, W.; Sossa, H. Reinforcement Learning Compensation based PD Control for a Double Inverted Pendulum. *IEEE Lat. Am. Trans.* **2019**, *17*, 323–329. [[CrossRef](#)]
33. Cheng, Y.; Zhao, P.; Wang, F.; Block, D.J.; Hovakimyan, N. Improving the Robustness of Reinforcement Learning Policies With  $\mathcal{L}_1$  Adaptive Control. *IEEE Robot. Autom. Lett.* **2022**, *7*, 6574–6581. [[CrossRef](#)]
34. Zhou, Y.; Lin, J.; Wang, S.; Zhang, C. Learning Ball-Balancing Robot through Deep Reinforcement Learning. In Proceedings of the 2021 International Conference on Computer, Control and Robotics (ICCCR), Shanghai, China, 8–10 January 2021. [[CrossRef](#)]
35. Kim, J.W.; Shim, H.; Yang, I. On Improving the Robustness of Reinforcement Learning-based Controllers using Disturbance Observer. In Proceedings of the 2019 IEEE 58th Conference on Decision and Control (CDC), Nice, France, 11–13 December 2019. [[CrossRef](#)]
36. The MathWorks, Inc. *Simscape™ Multibody User's Guide, 2021*; The MathWorks, Inc.: Natick, MA, USA, 2021. Available online: [https://la.mathworks.com/help/releases/R2021b/pdf\\_doc/physmod/sm/index.html](https://la.mathworks.com/help/releases/R2021b/pdf_doc/physmod/sm/index.html) (accessed on 25 September 2024).
37. Armstrong-Hélouvry, B.; Canudas de Wit, C. Friction Modeling and Compensation. In *The Control Handbook*; CRC Press: Boca Raton, FL, USA, 1996; pp. 1369–1382.
38. Fantoni, I.; Lozano, R. *Non-Linear Control for Underactuated Mechanical Systems*, 1st ed.; Springer London: London, UK, 2002; Volume XI, p. 295. [[CrossRef](#)]
39. Khalil, W.; Dombre, E. *Modeling, Identification and Control of Robots*, 1st ed.; Butterworth-Heinemann: Jordan Hill, UK, 2004; p. 483. [[CrossRef](#)]
40. García-Alarcón, O.; Puga-Guzmán, S.; Moreno-Valenzuela, J. On parameter identification of the Furuta pendulum. *Procedia Eng.* **2012**, *35*, 77–84. [[CrossRef](#)]
41. Slotine, J.J.E.; Weiping, L. *Applied Nonlinear Control*; Prentice-Hall: Englewood Cliffs, NJ, USA, 2004; p. 459.
42. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; The MIT Press: Cambridge, MA, USA, 2018; p. 519. Available online: <https://mitpress.mit.edu/9780262039246/reinforcement-learning/> (accessed on 25 September 2024).
43. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2017**, *31*, 607–623. [[CrossRef](#)]
44. Lewis, F.L.; Vrabie, D.; Syrmos, V.L. *Optimal Control*, 3rd ed.; John Wiley & Sons: Hoboken, NJ, USA, 2012; p. 552. Available online: <https://onlinelibrary.wiley.com/doi/book/10.1002/9781118122631> (accessed on 25 September 2024).
45. Shtessel, Y.; Edwards, C.; Fridman, L.; Levant, A. *Sliding Mode Control and Observation*, 1st ed.; Birkhäuser: New York, NY, USA, 2014; Volume XVII, p. 356. [[CrossRef](#)]
46. Perrusquia, A.; Yu, W. Robust control under worst-case uncertainty for unknown nonlinear systems using modified reinforcement learning. *Int. J. Robust Nonlinear Control* **2020**, *30*, 2920–2936. [[CrossRef](#)]

47. Lewis, F.W.; Jagannathan, S.; Yesildirak, A. *Neural Network Control of Robot Manipulators and Non-Linear Systems*, 1st ed.; Taylor & Francis: Philadelphia, PA, USA, 1998; p. 468. [[CrossRef](#)]
48. Åström, K.J.; Furuta, K. Swinging up a pendulum by energy control. *Automatica* **2017**, *36*, 287–295. [[CrossRef](#)]
49. Hu, W.; Yang, Y.; Liu, Z. Deep Deterministic Policy Gradient (DDPG) Agent-Based Sliding Mode Control for Quadrotor Attitudes. *Drones* **2024**, *8*, 95. [[CrossRef](#)]
50. Lu, P.; Huang, W.; Xiao, J.; Zhou, F.; Hu, W. Adaptive proportional integral robust control of an uncertain robotic manipulator based on deep deterministic policy gradient. *Mathematics* **2021**, *9*, 2055. [[CrossRef](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.