Model-Free Swing-Up and Balance Control of a Rotary Inverted Pendulum using the TD3 Algorithm: Simulation and Experiments

Trong-Nguyen Ho
Faculty of Electrical and Electronics Engineering, Ho Chi Minh City University of Technology and Education, Ho Chi Minh City, Vietnam
2391103@student.hcmute.edu.vn

Van-Dong-Hai Nguyen
Department of Automation and Control, Faculty of Electrical and Electronics Engineering, Ho Chi Minh City University of Technology and Education, Ho Chi Minh City, Vietnam
hainvd@hcmute.edu.vn (corresponding author)

Received: 20 October 2024 | Revised: 20 November 2024 | Accepted: 23 November 2024

 DOI: https://doi.org/10.48084/etasr.9335

ABSTRACT

The Rotary Inverted Pendulum (RIP) system is a highly nonlinear and under-actuated mechanical system, which presents significant challenges for traditional control techniques. In recent years, Reinforcement Learning (RL) has emerged as a prominent nonlinear control technique, demonstrating efficacy in regulating systems exhibiting intricate dynamics and pronounced nonlinearity. This research presents a novel approach to the swing-up and balance control of the RIP system, employing a RL algorithm, Twin Delayed (TD3) Deep Deterministic Policy Gradient (DDPG), obviating the necessity for a predefined mathematical model. The physical model of the RIP was designed in SolidWorks and subsequently transferred to MATLAB Simscape and Simulink for the purpose of training the RL agent. The system was successfully trained to perform both swing-up and balance control using a single algorithm for both tasks, representing a significant innovation that eliminates the need for two or more separate algorithms. Additionally, the trained agent was successfully deployed onto an experimental model, with the results demonstrating the feasibility and effectiveness of the model-free TD3 approach in controlling under-actuated mechanical systems with complex dynamics, such as the RIP. Furthermore, the results highlight the sim-to-real transfer capability of this method.

Keywords-Rotary Inverted Pendulum (RIP); Reinforcement Learning (RL); twin delayed deep deterministic policy gradient; model-free control; swing-up control; balance control; solidworks; matlab; simscape; simulink

I. INTRODUCTION

The control of under-actuated mechanical systems has emerged as a vibrant field of research, driven by their extensive applications across a range of domains, including robotics, aerospace, and marine technology. This category includes a diverse range of devices, such as flexible-link robots, mobile robots, walking robots, vehicles on mobile platforms, cars, locomotives, snake-like and aquatic robots, acrobatic robots, aircraft, spacecraft, helicopters, satellites, surface vessels, and underwater vehicles. It is noteworthy that the pendulum is regarded as one of the most emblematic under-actuated mechanical systems, frequently serving as a benchmark for addressing a multitude of control issues [1]. The control of RIPs presents considerable challenges due to their intrinsic complexity and dynamic characteristics [2]. The development of control strategies for these systems has historically progressed from simple Proportional-Integral-Derivative (PID) control, state feedback control, and Linear Quadratic Regulator (LQR) to more advanced methods like H-infinity control and sliding [4-6], and finally to intelligent techniques [7]. Nevertheless, although these approaches are effective for systems that are well-modeled, their use is constrained in situations where obtaining precise system models is challenging or when the system dynamics exhibit high levels of nonlinearity and under actuation. To address these challenges, this research proposes a novel model-free approach for achieving swing-up and balance control in RIP using RL.

In recent years, the application of RL techniques, especially Deep RL (DRL), has become increasingly important as a promising alternative for control tasks in complex dynamic systems without the need for explicit system models. This is evidenced by a growing body of literature on the subject [8-

15]. In the context of RIP systems, several pioneering studies have examined the potential of DRL algorithms for swing-up and balance control [16-20]. To demonstrate the adaptability and learning capabilities of these algorithms, researchers have trained agents for swing-up control using DRL techniques within simulation environments. However, the majority of these studies required the use of at least two distinct algorithms within the system, one for the swing-up task and the other for balance control. Moreover, there have been few studies that have sought to examine the transferability of simulation-trained DRL agents into real-world experimental platforms for practical validation and implementation. This work employs the DRL technique, particularly the TD3 algorithm, to leverage the large system nonlinearity without requiring an a priori mathematical model of the underlying processes. The agent, which was trained through simulations, is tested in a real-time application of RIP to demonstrate its robustness and applicability. A further advantage of using a single algorithm for both swing-up and balance control is the simplicity this affords. By employing a single optimization process, several features can be disabled, thus avoiding the potential for internal conflict between different algorithms. A unified approach has the additional benefit of reducing the complexity of implementation, the computational burden, and the overall efficiency of the control system. This study contributes to the advancement of control engineering by demonstrating a scalable and effective methodology for swing-up and balance control. The feasibility and effectiveness of the model-free DRL approach are demonstrated, thereby proving its potential for real-world implementation and cost-effective system optimization.

II. BACKGROUND

A. Reinforcement Learning

RL is a machine learning approach whereby an agent (controller) learns to make decisions by interacting with an environment, with the objective of maximizing cumulative rewards (feedback). It operates on the principles of Markov Decision Processes (MDPs), which entail the consideration of the following elements: the space of possible states *X*, the space of possible actions *U*, transition probability *P*, reward function *R*, and a discount factor γ [21]. As shown in Figure 1, the MDP framework demonstrates the interaction between the agent and the environment during the decision-making process.

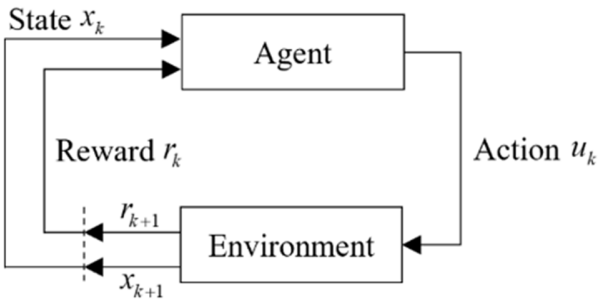

Fig. 1. The MDP framework in RL.

At each time step k, the agent observes the current state of the environment, represented as x_k. Based on this observation, the agent chooses an action $u_k \in U$, forming the state-action pair (x_k, u_k). In the subsequent time step, $k+1$, the environment transitions to a new state $x_{k+1} \in X$, and the agent receives a reward $r_{k+1} \in R$ for the action u_k taken from the state x_k. The agent's goal is to find an optimal policy π^* that maximizes the expected return from any initial state x_0, defined as:

$$J(\pi) = E[\sum_{k=0}^{\infty} \gamma^k r_k | \pi] \quad (1)$$

with a discount factor $\gamma \in [0,1)$ representing the difference between future rewards and immediate rewards and the expectation E is taken over the stochastic process induced by the policy π and the environment dynamics.

Two fundamental concepts in RL are the state-value function $V^{\pi}(x)$ and the action-value function $Q^{\pi}(x,u)$. These functions represent the expected return starting from state x and action u, respectively, under a given policy π. The state-value function is defined as:

$$V^{\pi}(x) = E_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_k | x_0 = x] \quad (2)$$

The action-value function is defined as:

$$Q^{\pi}(x,u) = E_{\pi}[\sum_{k=0}^{\infty} \gamma^k r_k | x_0 = x, u_0 = u] \quad (3)$$

The optimal value functions, $V^*(x)$ and $Q^*(x,u)$, represent the maximum expected cumulative reward that can be achieved by following the optimal policy π^* and satisfy the Bellman optimality equations:

$$V^*(x) = \max_u E[r(x,u) + \gamma V^*(x') | x,u] Q^*(x,u) = E\left[r(x,u) + \gamma \max_{u'} Q^*(x',u') | x,u\right] \quad (4)$$

In policy gradient methods, the policy $\pi(u,x;\theta)$ is parameterized by a set of parameters θ. The goal is to maximize the expected return $J(\pi_\theta)$. The gradient of the objective function with respect to θ is given by:

$$\nabla_\theta J(\pi_\theta) = E_{x\sim\rho^{\pi_\theta}, u\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(u|x) Q^{\pi_\theta}(x,u)] \quad (5)$$

where ρ^{π_θ} represents the state visitation distribution under policy π_θ. Actor-Critic methods combine value-based and policy-based approaches. The actor updates the policy parameters θ based on feedback from the critic, which estimates the value function. The policy is updated using the gradient:

$$\nabla_\theta J(\theta) = E\left[\nabla_\theta \log \pi_\theta(u|x) Q_\phi(x,u)\right] \quad (6)$$

where $Q_\phi(x,u)$ is the action-value function parameterized by ϕ. The critic is updated by minimizing the Bellman error:

$$L(\phi) = E\left[\left(Q_\phi(x,u) - \left(r + \gamma Q_\phi(x',u')\right)\right)^2\right] \quad (7)$$

B. Twin Delayed Deep Deterministic Policy Gradient

TD3 is an advanced algorithm within the actor-critic family, specifically designed to handle environments with continuous action spaces [22]. It builds upon its predecessor, the DDPG algorithm [23], addressing overestimation bias and enhancing stability in the learning process. The DDPG extends

RL to continuous action spaces, using an actor-critic architecture where the actor $\mu_\theta(x)$ outputs deterministic actions given the state x, and the critic $Q_\phi(x,u)$ evaluates these actions. The actor is updated using:

$$\nabla_\theta J(\theta) = E_{x\sim D}\left[\nabla_u Q_\phi(x,u)|_{u=\mu_\theta(x)} \nabla_\theta \mu_\theta(x)\right] \quad (8)$$

where D is the reply buffer. The critic is updated by minimizing:

$$L(\phi) = E_{(x,u,r,x')\sim D}\left[\left(Q_\phi(x,u) - y\right)^2\right] \quad (9)$$

with the target value:

$$y = r + \gamma Q_{\phi'}\left(x', \mu_{\theta'}(x')\right) \quad (10)$$

TD3 algorithm builds upon DDPG, introducing enhancements such as clipped double Q-learning, delayed policy updates, and target policy smoothing to address issues like overestimation bias and instability. TD3 employs two critic networks $Q_{\phi_1}(x,u)$, $Q_{\phi_2}(x,u)$ and uses the minimum Q-value to form the target:

$$y = r + \gamma \min_{i=1,2} Q_{\phi'_i}\left(x', \mu_{\theta'}(x')\right) \quad (11)$$

TD3 introduces target policy smoothing to address the issue of deterministic policies that can lead to high variance in Q-value estimates. To smooth the target policy, TD3 adds noise to the action produced by the target policy network:

$$\tilde{\mu}(x') = \mu_{\theta'}(x') + \varepsilon \quad (12)$$

where $\mu_{\theta'}(x')$ is the action suggested by the target policy network, and ε is noise sampled from a clipped Gaussian distribution:

$$\varepsilon \sim \text{clip}(N(0,\sigma), -c, c) \quad (13)$$

where σ is the standard deviation of the noise and c is a clipping constant. In TD3, the target networks for both the policy and the critics are updated using a soft update mechanism. This gradual update approach ensures that the target networks change slowly, which contributes to the stability of the training process:

$$\phi'_i \leftarrow \tau\phi_i + (1-\tau)\phi'_i \theta' \leftarrow \tau\theta + (1-\tau)\theta' \quad (14)$$

where τ is a small constant that controls the rate of the update.

These enhancements make the TD3 algorithm more robust and effective for continuous action spaces, addressing key limitations of the DDPG algorithm and improving its performance in complex control tasks, such as the swing-up and balancing control of a RIP.

III. SYSTEM MODELLING AND PROPOSED METHODOLOGY

A. System Modelling

Figures 2 and 3 present the Computer-Aided Design (CAD) model of the RIP system, which was designed using the SolidWorks software. The system comprises a rotary arm and a pendulum. The rotary servo unit, driven by a Direct Current (DC) motor, enables rotational movement at one end. An encoder is affixed to the DC motor, serving to quantify the angular displacement of the rotary arm. Another encoder is positioned to monitor the angular position of the pendulum. The lengths of the rotary arm and pendulum are denoted as L_1 and L_2. The pendulum angle φ ranges from $[-\pi, \pi]$ radians, with $\varphi = 0$ radians indicating the upright position of the pendulum. The arm angle θ is constrained within $\left[-\frac{\pi}{3}, \frac{\pi}{3}\right]$ radians, with $\theta = 0$ radians representing the arm in the central position. Angular velocities are denoted by $\dot{\alpha}$ and $\dot{\theta}$ radians per second. Table I provides a concise summary of the system components, along with their respective materials and mass.

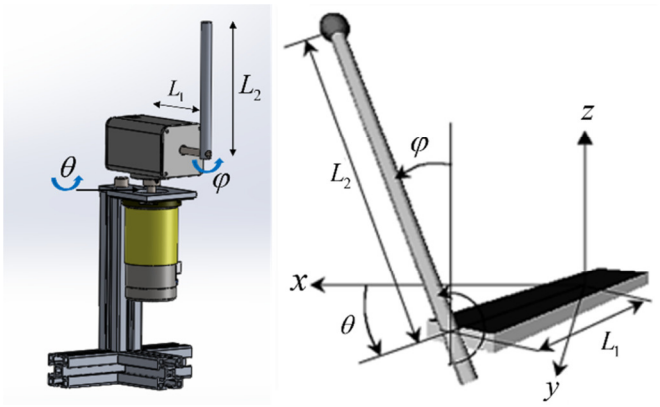

Fig. 2. Modelling of RIP: CAD model on Solidworks (left), RIP scheme (right).

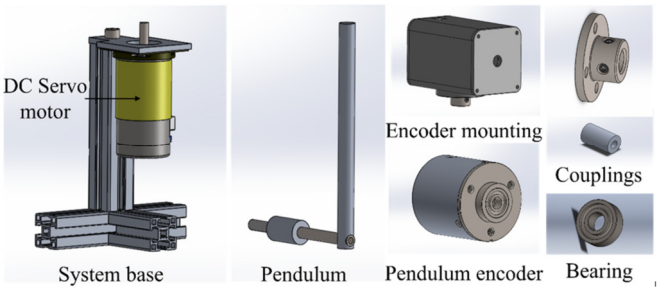

Fig. 3. RIP components on Solidworks.

TABLE I. SYSTEM COMPONENT DESCRIPTION

Nr.	RIP Components		
	Part	**Material / Parameter**	**Mass [g] / Value**
1	Base	Shaped aluminum	352.77
2	Rid flange coupling	Stainless steel	27.60
3	Encoder	Library part	79.57
4	Bearing	Stainless steel	4.61
5	Coupling	Aluminum	8.41
6	Encoder mounting	PLA 3D printing	90.53
7	Pendulum	Aluminum	51.38
8	Motor	Motor resistance	6.8357 Ω
9	Motor	Torque constant	0.0649 Nm/A
10	Motor	Motor Inductance	0.2509 H
11	Motor	Back EMF constant	0.06494 V/rad/s
12	Motor	Shaft inertia	0.00013 kgm^2
13	Bearing	Viscous friction torque of arm	0.001 Nm/rad/s
14	Bearing	Viscous damping of pendulum	0.001 Nm/rad/s

As shown in Figure 4, the RIP model is then transferred to MATLAB Simscape. This modeling process employs the Simulink Simscape toolbox, incorporating essential components such as the DC motor, mechanical joints, and transformation blocks to capture the dynamic behavior of the system. The simulation commences with the control voltage source block, which is directly connected to the DC motor block. The motor is operated within a voltage range of -24 V to 24 V. The motor output is expressed in terms of angular velocity and angular position, which serve as the input to a rotational multibody interface block. This block serves as an interface between the Simscape Multibody joint and the Simscape mechanical rotational system, facilitating the assembly of the mechanical components. The primary outputs from this interface are mechanical rotational torque and angular velocity, which drive revolute joint 1. This joint connects the DC motor to the rotary arm, enabling motion. The rotary arm is coupled to the pendulum through a reversible connection at the frame ports, which negates the transformation. During simulation, these frames remain attached to each other, functioning as a unified system. The model tracks two key outputs: the position θ for the rotary arm, φ for the pendulum and their respective angular velocities ($\dot{\theta}$, $\dot{\varphi}$). These measurements are vital for analyzing the dynamic response of the system during operation. A configuration block manages uniform gravity $[x = 0, y = 0, z = -9.8665]$ and includes solver and mechanism settings. The simulation is driven by *'ode23t'* solver. This approach captures the complex interactions of the system while maintaining computational efficiency, providing a realistic representation of the RIP's dynamics.

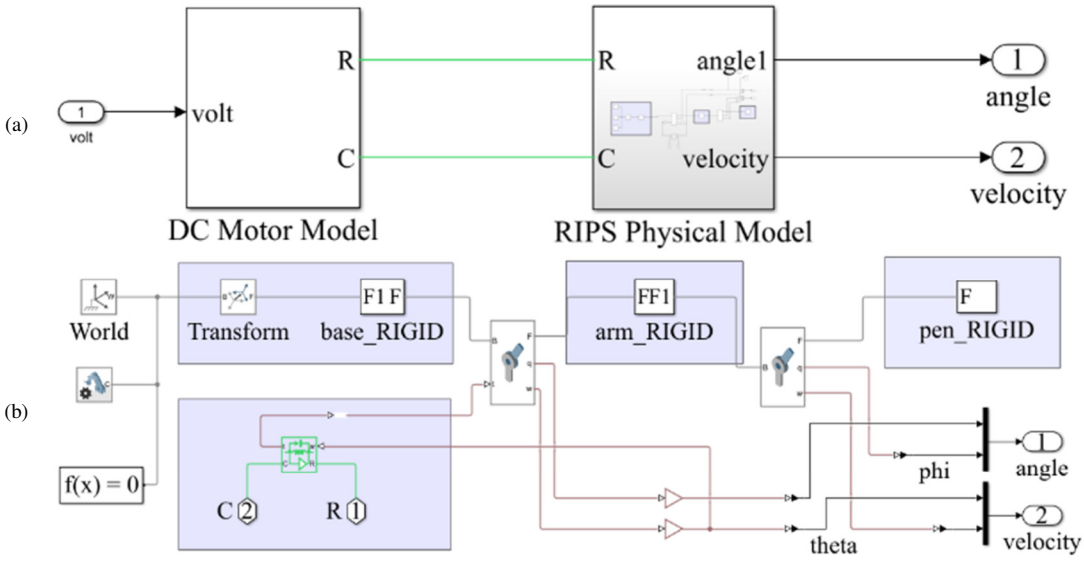

Fig. 4. RIP physical modelling in MATLAB Simscape: (a) overview, (b) inside view.

B. TD3 Algorithm Training Methodology

1) Observation

The RIP state is the vector $x \in \mathbb{R}^4$ is composed of 4 states variables:

$$x = [\theta \quad \dot{\theta} \quad \varphi \quad \dot{\varphi}]^T \tag{15}$$

To compute the observation, we use sine and cosine functions to capture variations in the deflection angle. This method simplifies the analysis of oscillatory and rotational mechanisms, helping to determine the system's vibrational characteristics. Additionally, the control action from the previous time step is added, resulting in $o(x) \in \mathbb{R}^7$, (15) becoming:

$$o(x_k) = [\sin\theta_k \quad \cos\theta_k \quad \dot{\theta}_k \quad \sin\varphi_k \quad \cos\varphi_k \quad \dot{\varphi}_k \quad u_{k-1}] \tag{16}$$

2) Agent Action

The agent's action $|u_k| \leq 1$ is scaled to the motor voltage (in direct current – DC) within the environment. It is then multiplied by $V_{control} = 24$ to provide the control voltage to the DC motor, which generates force to move the arm in either direction.

3) Reward Function

Referring to the LQR cost function, the reward function at each time step k is given by:

$$J(r_k) = \sum_{k=0}^{\infty}(x_k^T Q x_k + u_m R u_m) \tag{17}$$

where the control vector $u_m = [u_{k-1} \quad u_{k-1} - u_{k-2}]^T$ and the weight matrices Q and R as follows:

$$Q = \begin{bmatrix} \alpha_1 & 0 & 0 & 0 \\ 0 & \alpha_2 & 0 & 0 \\ 0 & 0 & \alpha_3 & 0 \\ 0 & 0 & 0 & \alpha_4 \end{bmatrix}, R = \begin{bmatrix} \alpha_5 & 0 \\ 0 & \alpha_6 \end{bmatrix} \tag{18}$$

By additional the P_k value, acting as the baseline reward:

$$P_k = \begin{cases} 1 \text{ if } \theta_k \in \pm\theta_{\max} \text{ and } \dot{\theta}_k \in \pm\dot{\theta}_{\max} \\ 0 \text{ otherwise} \end{cases} \quad (19)$$

The reward function can be presented in detail:

$$r_k = P_k - \left(\alpha_1\theta_k^2 + \alpha_2\varphi_k^2 + \alpha_3\dot{\theta}_k^2 + \alpha_4\dot{\varphi}_k^2 + \alpha_5 u_{k-1}^2 + \alpha_6(u_{k-1} - u_{k-2})^2\right) \quad (20)$$

The matrices (18) indicate the relative importance and priority of each state component in influencing the system's performance and learning behavior. The ability to adjust these weights allows for a targeted focus on specific states, thereby influencing the optimization process in RL or autonomous control systems. The reward function represents a negative cost with a baseline reward and is constituted of five component terms:

- Deviations from the forward position of the motor arm θ_k^2.
- Deviations from the inverted position of the pendulum φ_k^2.
- The angular speed of the motor arm $\dot{\theta}_k^2$.
- The angular speed of the pendulum $\dot{\varphi}_k^2$.
- Changes in the control action u_{k-1}^2 and $(u_{k-1} - u_{k-2})^2$.

4) Critic Network

In order to model the parameterized Q-value function within the critic, it is necessary to conduct the neural network with the following structure: The input layers comprise a single unit for the observation vector and a separate unit for the action vector. The hidden layers comprise two layers with 64 neurons each, with Rectified Linear Unit (ReLU) activation functions. The output layer generates a scalar value for the state-action pair. The structure of the two critic networks is presented in Figure 5.

5) Actor Network

A neural network is employed with a single input layer for the observation vector and a single output layer that generates actions for the environment's action channel. The network comprises two hidden layers, each comprising 64 neurons and employing the ReLU activation function. The structure of the actor network is shown in Figure 6.

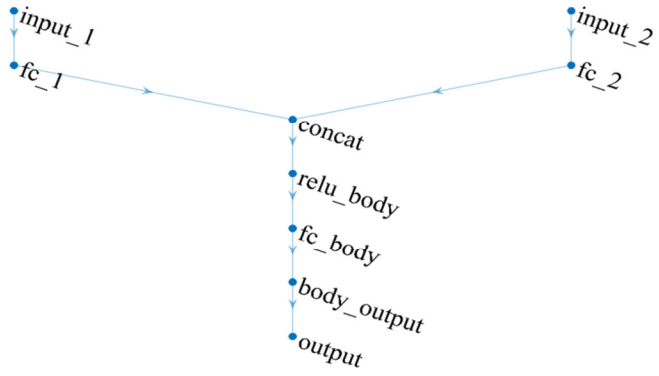

Fig. 5. The structure of the two critic networks.

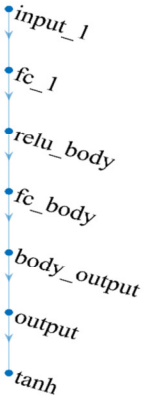

Fig. 6. The structure of the actor network.

6) Training Environment

A training environment is created with a Simulink model, shown in Figure 7, allowing for interaction between the agent and physical model. The environmental interface is defined as:

- Agent Block: Specifies the agent object created in the MATLAB workspace using RL toolbox functions.
- Generate RL Signals Subsystem: Produces the observation signal, reward function, and *'isDone'* signal for the agent.

RIP Subsystem: Contains the system in the Simscape model.

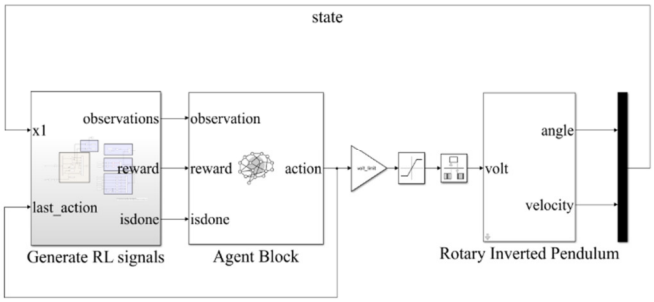

Fig. 7. Simulink environment for training RIPS.

7) Training Parameters

The configuration of the TD3 agent specifies various parameters crucial for the training process shown in Table II.

8) Reset Function

A reset function is designed to reset the environment to its initial state before starting a new episode while training:

```
function localResetFcn(input)
theta_init = -π/4 + random() * π/2
phi_int = π - π/4 + random() * π/2
setVariable(input, "theta_int", theta_int)
setVariable(input, "phi_int", phi_int)
setVariable(input, "dtheta_int", 0)
setVariable(input, "dphi_int", 0)
return input
end function
```

TABLE II. TRAINING PARAMETERS

Agent Parameters		
Parameter	Description	Value
T_s	Sample time	0.005 seconds
T_f	Length of each episode	5 seconds
Buffer Length	Size of the experience replay buffer	10^6
Mini Batch Size	Number of samples in each mini batch	128
Learning Freq	Policy update frequency	1
Mini Batch	Number of mini batches processed per training epoch	1
Actor LearnRate	Learning rate for the actor network optimizer	10^{-4}
Actor GradThresh	Gradient threshold for the actor network optimizer	1
Critic1 LearnRate	Learning rate for the first critic network optimizer	10^{-3}
Critic1 GradThresh	Gradient threshold for the first critic network optimizer	1
Critic2 LearnRate	Learning rate for the second critic network optimizer	10^{-3}
Critic2 GradThresh	Gradient threshold for the second critic network optimizer	1
Max Episodes	Maximum number of episodes for training	1,000
Window Length	Window size for averaging the score to evaluate progress	10
Stopping Criteria	Training stops when the average reward surpasses this threshold	860

IV. SIMULATION AND EXPERIMENTAL RESULTS

A. *Training Reward Value*

As presented in Figure 8, the TD3 algorithm exhibited convergence of the reward value after approximately 200 episodes, reaching the desired performance level by the 400th episode. This outcome was achieved through the usage of the MATLAB RL Toolbox. At this juncture, the agent successfully completed the swing-up task and maintained the pendulum in an upright position with minimal control effort. The training was terminated at an average reward value of 860.

B. *Simulation Results*

The output responses for the RIP are presented in Figures 9 and 10, with the input control shown in Figure 11.

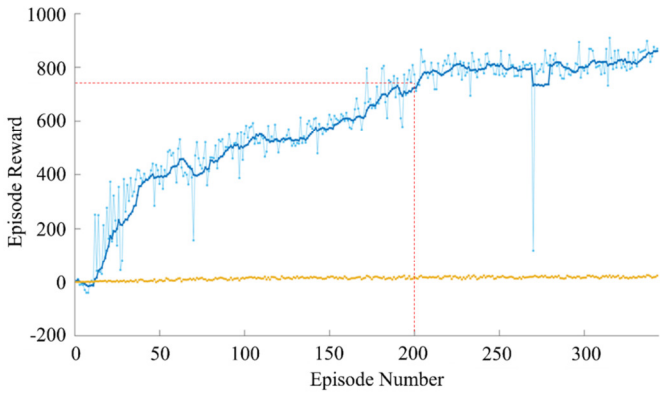

Fig. 8. Reward value curve.

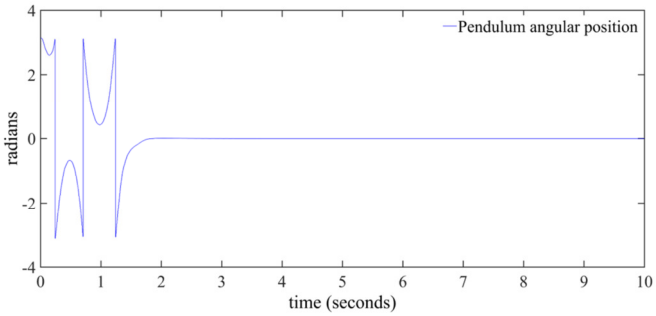

Fig. 9. Pendulum angular position while swing-up and balancing.

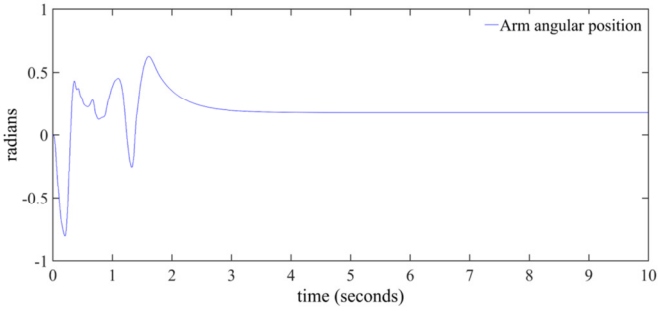

Fig. 10. Arm angular position while swing-up and balancing.

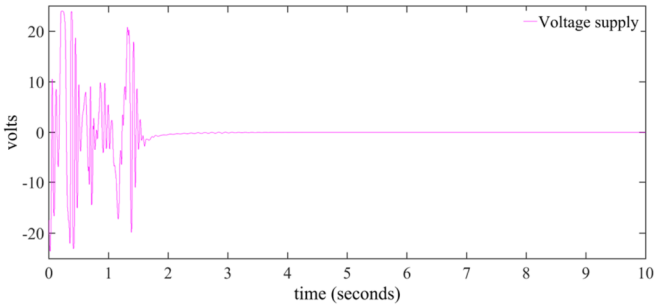

Fig. 11. Control input while swing-up and balancing.

he performance of the pendulum control system, shown in Table III, is characterized by a high degree of precision, as evidenced by a Root-Mean-Square Error (RMSE) of 9×10^{-4}. This value reflects a minimal average deviation from the desired angle, demonstrating the system's ability to maintain the pendulum's position with remarkable accuracy. This also indicates excellent control accuracy, given the very low percentage overshoot of 0.50%. The arm exhibits suboptimal performance with regard to its position, as evidenced by an RMSE of 0.1828, which reflects a larger-than-desirable average error in maintaining its position. The arm exhibits a considerable overshoot of 27.56%, which underscores the necessity for more precise control adjustments in the weight of the θ variable in the Q matrix (18). The time required for the pendulum to complete one full swing is 2.24 seconds, which is an acceptable duration. In conclusion, despite the robust nature of the pendulum control system, the arm's performance indicates the necessity for enhancements aimed at minimizing overshoot and attaining a more stable and precise control system.

TABLE III. CONTROL PERFORMANCE

Parameter	Evaluating Components		
	Arm section	Pendulum section	Description
RMSE	0.1823	9×10^{-4}	Root means squared error after swung-up
Swing-Up Time	-	2.24 seconds	Time when the pendulum angle remains below a threshold of 0.2 radians for at least 1 second
Overshoot	27.56 %	0.50 %	Percentage overshoot of the pendulum after swung-up

C. Experimental Results

An experimental model was constructed for the purpose of evaluating the performance of the agent, as shown in Figure 12. The experimental response of the RIP is presented in Figures 13 and 14, respectively.

Fig. 12. RIP model in experimental: front (left) and side views (right).

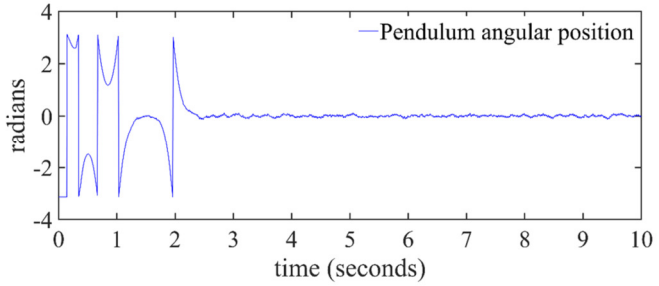

Fig. 13. Pendulum angular position while swing-up and balancing in experimental.

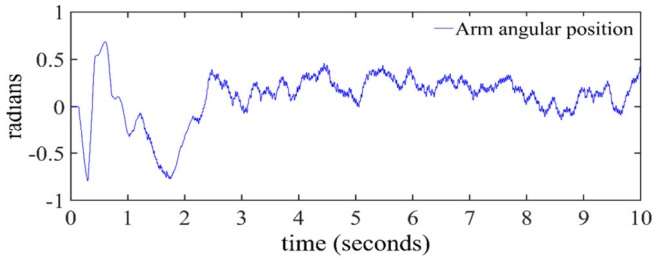

Fig. 14. Arm angular position while swing-up and balancing.

The system exhibits successful swing-up and balancing, thereby confirming the effectiveness of the control strategy. The controller effectively maneuvers the pendulum from its initial state to a balanced position, thereby further validating the robustness of the TD3-based approach. The response plots of the experiments demonstrate that the TD3 agent consistently meets the performance goals, and that the agent is stable once the pendulum attains an upright position. The minor oscillations observed around the equilibrium point, the extended time taken by the swing-up (2.5 seconds compared to the 2.24 seconds of the pendulum), and the larger variations of the arm angle with respect to the physical model from the simulation indicate that practical variables like environmental perturbations, such as the evenness of the table surface on which the robot operates, power dissipation within the power supply and motor, encoder signal noises, limits created due to the manual fabrication process, and differences in 3D printing of the model introduce errors that result in a deterioration of performance.

V. CONCLUSIONS

The implementation of Twin Delayed (TD3) Deep Deterministic Policy Gradient (DDPG) for the regulation of the Rotary Inverted Pendulum (RIP) has demonstrated considerable promise. The simulation results demonstrate the efficacy of the TD3 algorithm in facilitating the efficient completion of the swing-up maneuver and stabilization of the RIP. Following a brief oscillatory phase, the pendulum attains and maintains an upright position throughout the remainder of the simulation. Similarly, the angular position of the arm demonstrates a rapid stabilization following initial perturbations, reflecting the high degree of precision in control. Nevertheless, the occurrence of oscillations around the equilibrium during experimentation may necessitate the implementation of online retraining of the agent, with the objective of enhancing its resilience against experimental uncertainties. These practical complications underscore the vital importance of model-free methodologies. While most theoretical mathematical models assume optimal conditions and frameworks, model-free approaches like TD3 offer significant advantages in accommodating the imperfections and uncertainties that arise in real-world scenarios, which are often unavoidable. This adaptation is crucial for attaining consistent performance in practical circumstances, where deviations from idealized models are pervasive and often unavoidable. The results confirm that TD3 is one of the most promising model-free approaches and single-agent solution capable of effectively handling both swing-up and stabilization tasks. Consequently, there is no need to develop a complicated mathematical model of the system. The TD3 algorithm learns through direct interaction with the environment, enabling it to perform the swing-up and stabilization tasks in the absence of a well-defined system model.

Concurrently, this study presents a novel application of TD3 to a highly complex nonlinear system such as RIP, eliminating the necessity for prerequisite mathematical modeling. In contrast with conventional methodologies reliant on system identification, TD3 employs a reinforcement-based learning approach, whereby the system interacts with its

environment to refine its internal representation. Furthermore, this work extends TD3 toward the solution of both swing-up and stabilization tasks simultaneously, a domain that has not been widely explored within the field of model-free control methods. This study highlights the superiority of TD3 in reducing training time in comparison to other algorithms, such as Deep Deterministic Policy Gradient (DDPG) and Policy Proximal Optimization (PPO), where only a single agent is capable of executing both tasks. This represents a significant advantage over other methods, which often necessitate the use of multiple agents or a combined strategy to effectively address complex tasks. This approach, therefore, demonstrates the flexibility and sensitivity of TD3 when dealing with such complex nonlinear systems as the RIP, extending its application for model-free reinforcement learning to a broader class of dynamic control frameworks.

REFERENCES

[1] B. S. Cazzolato and Z. Prime, "On the Dynamics of the Furuta Pendulum," *Journal of Control Science and Engineering*, vol. 2011, no. 1, 2011, Art. no. 528341, https://doi.org/10.1155/2011/528341.

[2] Z.-P. Jiang, "Controlling Underactuated Mechanical Systems: A Review and Open Problems," in *Advances in the Theory of Control, Signals and Systems with Physical Modeling*, J. Lévine and P. Müllhaupt, Eds. Berlin, Heidelberg: Springer, 2011, pp. 77–88.

[3] C. Sravan Bharadwaj, T. Sudhakar Babu, and N. Rajasekar, "Tuning PID Controller for Inverted Pendulum Using Genetic Algorithm," in *Advances in Systems, Control and Automation: ETAEERE-2016*, A. Konkani, R. Bera, and S. Paul, Eds. Singapore: Springer, 2018, pp. 395–404.

[4] K. Chhabra and Mohd. Rihan, "Design of linear quadratic regulator for rotary inverted pendulum using LabVIEW," in *2016 International Conference on Advances in Computing, Communication, & Automation (ICACCA) (Spring)*, Dehradun, India, Apr. 2016, pp. 1–5, https://doi.org/10.1109/ICACCA.2016.7578892.

[5] G. Rigatos, P. Siano, M. Abbaszadeh, and S. Ademi, "Nonlinear H-infinity control for the rotary pendulum," in *2017 11th International Workshop on Robot Motion and Control (RoMoCo)*, Wasowo Palace, Poland, Jul. 2017, pp. 217–222, https://doi.org/10.1109/RoMoCo.2017.8003916.

[6] I. S. Trenev and D. D. Devyatkin, "Feedback Linearization Control of Nonlinear System," *Engineering Proceedings*, vol. 33, no. 1, 2023, Art. no. 36, https://doi.org/10.3390/engproc2023033036.

[7] T. V. A. Nguyen and N. H. Tran, "A T-S Fuzzy Approach with Extended LMI Conditions for Inverted Pendulum on a Cart," *Engineering, Technology & Applied Science Research*, vol. 14, no. 1, pp. 12670–12675, Feb. 2024, https://doi.org/10.48084/etasr.6577.

[8] P. Dev, S. Jain, P. Kumar Arora, and H. Kumar, "Machine learning and its impact on control systems: A review," *Materials Today: Proceedings*, vol. 47, pp. 3744–3749, Jan. 2021, https://doi.org/10.1016/j.matpr.2021.02.281.

[9] M. Jin and J. Lavaei, "Stability-certified reinforcement learning: A control-theoretic perspective." arXiv, Oct. 26, 2018, https://doi.org/10.48550/arXiv.1810.11505.

[10] F. L. Lewis and D. Vrabie, "Reinforcement learning and adaptive dynamic programming for feedback control," *IEEE Circuits and Systems Magazine*, vol. 9, no. 3, pp. 32–50, 2009, https://doi.org/10.1109/MCAS.2009.933854.

[11] J. Moos, K. Hansel, H. Abdulsamad, S. Stark, D. Clever, and J. Peters, "Robust Reinforcement Learning: A Review of Foundations and Recent Advances," *Machine Learning and Knowledge Extraction*, vol. 4, no. 1, pp. 276–315, Mar. 2022, https://doi.org/10.3390/make4010013.

[12] S. Mosharafian, S. Afzali, Y. Bao, and J. M. Velni, "A Deep Reinforcement Learning-based Sliding Mode Control Design for Partially-known Nonlinear Systems," in *2022 European Control Conference (ECC)*, London, UK, Jul. 2022, pp. 2241–2246, https://doi.org/10.23919/ECC55457.2022.9838169.

[13] M. Ran, J. Li, and L. Xie, "Reinforcement-Learning-Based Disturbance Rejection Control for Uncertain Nonlinear Systems," *IEEE Transactions on Cybernetics*, vol. 52, no. 9, pp. 9621–9633, Sep. 2022, https://doi.org/10.1109/TCYB.2021.3060736.

[14] A. F. ud Din *et al.*, "Deep Reinforcement Learning for Integrated Non-Linear Control of Autonomous UAVs," *Processes*, vol. 10, no. 7, Jul. 2022, Art. no. 1307, https://doi.org/10.3390/pr10071307.

[15] Z. Zhang, Z. Mo, Y. Chen, and J. Huang, "Reinforcement Learning Behavioral Control for Nonlinear Autonomous System," *IEEE/CAA Journal of Automatica Sinica*, vol. 9, no. 9, pp. 1561–1573, Sep. 2022, https://doi.org/10.1109/JAS.2022.105797.

[16] R. Özalp, N. K. Varol, B. Taşci, and A. Uçar, "A Review of Deep Reinforcement Learning Algorithms and Comparative Results on Inverted Pendulum System," in *Machine Learning Paradigms: Advances in Deep Learning-based Technological Applications*, G. A. Tsihrintzis and L. C. Jain, Eds. Cham, Switzerland: Springer International Publishing, 2020, pp. 237–256.

[17] N. Mellatshahi, S. Mozaffari, M. Saif, and S. Alirezaee, "Inverted Pendulum Control with a Robotic Arm using Deep Reinforcement Learning," in *2021 International Symposium on Signals, Circuits and Systems (ISSCS)*, Iasi, Romania, Jul. 2021, pp. 1–6, https://doi.org/10.1109/ISSCS52333.2021.9497411.

[18] Z. Ben Hazem, "Study of Q-learning and deep Q-network learning control for a rotary inverted pendulum system," *Discover Applied Sciences*, vol. 6, no. 2, Feb. 2024, Art. no. 49, https://doi.org/10.1007/s42452-024-05690-y.

[19] R. S. Bhourji, S. Mozaffari, and S. Alirezaee, "Reinforcement Learning DDPG–PPO Agent-Based Control System for Rotary Inverted Pendulum,"*Arabian Journal for Science and Engineering*, vol. 49, no. 2, pp. 1683–1696, Feb. 2024, https://doi.org/10.1007/s13369-023-07934-2.

[20] M. Safeea and P. Neto, "A Q-learning approach to the continuous control problem of robot inverted pendulum balancing, " *Intelligent Systems with Applications*, vol. 21, Mar. 2024, Art. no. 200313, https://doi.org/10.1016/j.iswa.2023.200313.

[21] R. S. Sutton and A. G. Barto, "Reinforcement Learning: An Introduction," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 1054–1054, Sep. 1998, https://doi.org/10.1109/TNN.1998.712192.

[22] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods." arXiv, Oct. 22, 2018, https://doi.org/10.48550/arXiv.1802.09477.

[23] T. P. Lillicrap *et al.*, "Continuous control with deep reinforcement learning." arXiv, Jul. 05, 2019, https://doi.org/10.48550/arXiv.1509.02971.