IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 18 June 2024, accepted 12 August 2024, date of publication 21 August 2024, date of current version 2 September 2024.

*Digital Object Identifier 10.1109/ACCESS.2024.3447054*

RESEARCH ARTICLE

# Implementation of DDPG-Based Reinforcement Learning Control for Self-Balancing Motorcycle

**K. VIJAYA LAKSHMI AND M. MANIMOZHI**
School of Electrical Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu 632014, India
Corresponding author: M. Manimozhi (mmanimozhi@vit.ac.in)

This work was supported by the Vellore Institute of Technology, Vellore, Tamilnadu, India.

**ABSTRACT** This paper presents the implementation of a Deep Deterministic Policy Gradient (DDPG) algorithm in Reinforcement Learning (RL) for self-balancing a motorcycle. The DDPG agent iteratively interacts with the motorcycle environment to develop an optimal control policy, utilizing states such as position and velocity, and actions like motor torque. The study evaluates the performance through simulations and real-time experimentation, demonstrating the algorithm's effectiveness in balancing the motorcycle across various leaning angles and in handling external disturbances and model uncertainties. Comparative analysis with a traditional PD controller highlights DDPG's faster response times, improved disturbance rejection, and enhanced adaptability to uncertainties. The results underscore the potential of RL algorithms in enhancing motorcycle control systems for safer and more efficient operation.

**INDEX TERMS** Arduino nano 33 IOT, DDPG algorithm, deep RL, PD control, self-balancing motorcycle.

## I. INTRODUCTION

Motorcycles, especially two-wheeled ones, are inherently less stable than four-wheeled vehicles. In India, 70-85% of motorist deaths are due to accidents, often resulting from losing balance and control over the motorcycle. According to the National Crime Records Bureau (NCRB), two-wheelers claimed the highest number of lives, with nearly 70,000 people losing their lives in road accidents across the country in 2021. Although accidents cannot be entirely prevented, taking precautions can significantly reduce life-threatening injuries. Self-balancing technology promises to enhance stability even in unbalanced conditions, providing a potential solution to improve motorcycle safety.

The development of self-balancing motorcycles has garnered significant attention in recent years due to their potential applications in enhancing rider safety, improving urban mobility, and advancing autonomous vehicle technologies. These motorcycles rely on advanced control systems to maintain balance and stability, particularly when stationary or moving at low speeds. Research in motorcycle balancing and control has predominantly focused on methods to enhance stability, with significant emphasis on optimizing the balance of two-wheeled vehicles due to their potential for increased flexibility and improved energy efficiency [1]. Research has emphasized the use of steering control mechanisms [2], [3], [4] and auxiliary balancing mechanisms like inertia wheels and mass balancers [5], [6], [7], [8], [9]. However, at low forward speeds, steering control is often insufficient to resist disturbances, making inertia wheel mechanisms more effective. Therefore, this paper considers the self-balancing motorcycle with inertia wheel mechanism.

Existing solutions for motorcycle stability include various control mechanisms such as linear controllers like PID (Proportional-Integral-Derivative) and LQR (Linear Quadratic Regulator) [10], [11], [12], as well as nonlinear [9], [13], [14] and intelligent control methods [15], [16], [17]. However, these traditional methods have limitations. PID and its variants are widely employed in balance control applications [18], [19]. Balancing errors can be minimized with LQR compared to PID with longer settling time [18], [20]. PID and LQR controllers require precise system modeling and accurate tuning, and they are often limited to small leaning angles due to their reliance on linearization around operating points. Nonlinear control techniques like Fuzzy Logic Controllers (FLC) offer better stability and adaptability but involve

The associate editor coordinating the review of this manuscript and approving it for publication was Fabrizio Messina.

complexities in rule formulation [21], [22]. Furthermore, the integration of FLC with neural networks enhances the stability and adaptability of the system [23], [24]. Therefore, these traditional approaches often struggle with handling the nonlinear and under-actuated nature of motorcycle dynamics, particularly in the presence of external disturbances and model uncertainties.

In recent years, the DDPG algorithm has gained popularity for its ability to solve continuous control problems in reinforcement learning. The DDPG algorithm is particularly relevant for self-balancing motorcycles due to its ability to handle complex, nonlinear dynamics without requiring a detailed mathematical model. By continuously interacting with the environment, the DDPG algorithm autonomously develops control policies that enhance stability and adaptability [25].

The primary challenge in developing a self-balancing motorcycle lies in the design of a control system that can effectively maintain stability under various operating conditions, including disturbances and varying rider inputs. Traditional control methods often lack the adaptability and learning capabilities required to handle these dynamic scenarios. This study aims to address these limitations by leveraging the DDPG-based RL approach to develop a robust control strategy for a self-balancing motorcycle. It seeks to overcome the limitations of fixed-gain controllers by leveraging the adaptive capabilities of reinforcement learning. The self-balancing motorcycle problem is modeled as an inverted pendulum, a classical control problem known for its instability and complexity [26]. Achieving stable control in such systems requires advanced techniques capable of handling dynamic and uncertain environments.

In the context of our study, the term ''leaning angle'' refers to the angle at which a motorcycle tilts relative to the vertical axis, whether in motion or stationary. This parameter is crucial for understanding motorcycle dynamics and control, as it directly influences the vehicle's stability and maneuverability. When the motorcycle is perfectly upright, the leaning angle is zero degrees.

Despite advancements in reinforcement learning (RL) and control systems, there is limited application of DDPG in self-balancing motorcycles, insufficient exploration of RL algorithms integrated with real-time control systems for motorcycles, and a lack of comprehensive comparative studies with traditional methods. This study aims to address these gaps by implementing a DDPG-based control system that utilizes continuous action spaces for precise motorcycle balance, incorporates real-time learning for dynamic adaptation, and performs a detailed comparative analysis with traditional control methods. This approach aims to advance autonomous motorcycle technologies, enhancing safety and urban mobility efficiency.

The primary objective of this study is to design and implement a self-balancing motorcycle using the DDPG algorithm augmented with a PD controller. The DDPG agent iteratively improves its control strategy by leveraging states such as position and velocity and actions like motor torque. This study aims to evaluate the performance of this approach through simulations and real-time experimentation, highlighting the algorithm's effectiveness in maintaining balance across various leaning angles and handling external disturbances and model uncertainties. The findings have implications for the development of autonomous motorcycles and other robotic systems requiring continuous control, potentially leading to safer and more efficient transportation solutions.

## II. SELF-BALANCING MOTORCYCLE

The self-balancing motorcycle is a two-wheeled robot designed to autonomously maintain its balance without tipping over. It incorporates advanced sensing and control technologies to achieve stability during operation. Key components and controls of this motorcycle are illustrated in Figure 1 [27].

The robot is equipped with an Inertial Measurement Unit (IMU) consisting of three essential sensors: an accelerometer for measuring acceleration, a gyroscope for tracking angular velocity, and a magnetometer for detecting magnetic field orientation. These sensors collectively provide real-time data crucial for dynamic stabilization.

A pivotal component of the motorcycle is the BNO055 IMU, a sophisticated 9-axis absolute orientation sensor. This IMU integrates an accelerometer, gyroscope, and magnetometer, enabling precise measurement of acceleration, angular velocity, and magnetic field orientation. This sensor's capabilities are instrumental in maintaining accurate orientation and stability of the motorcycle.

The control system of the motorcycle utilizes an Arduino Nano motor carrier, an add-on board designed for controlling two DC motors and one servo motor. One DC motor, equipped with an encoder, rotates the inertia wheel, which plays a critical role in maintaining the motorcycle's balance. Another DC motor drives the rear wheel for forward and backward motion, while the servo motor enables precise control of the motorcycles steering.

The motorcycle, developed from the Arduino Engineering Kit Rev 2, leverages a rotating disc known as the inertia wheel to counteract tilting movements and maintain balance. This system is centrally controlled by an Arduino Nano 33 IoT microcontroller, which coordinates the functions of the IMU, motor carrier, and other components to ensure stable and reliable operation position.

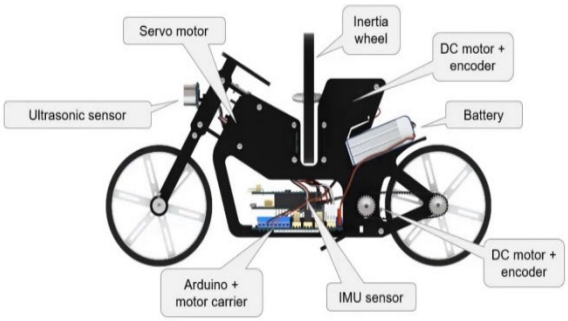


**FIGURE 1.** Self-balancing motorcycle.

The motorcycle will need a power source to run the motors and microcontroller. It includes a 3.7 V, 3000mAh lithium-polymer battery that can power the motorcycle for several hours. The purpose of an inertia wheel in a self-balancing motorcycle is to enhance stability and prevent the motorcycle from tipping over. This spinning wheel leverages gyroscopic principles to counteract tilting movements. When the motorcycle tilts, the inertia wheel generates a force that opposes the tilt, helping to maintain the motorcycle's upright position. This technology is crucial for maintaining balance, especially at low speeds and during maneuvers, making self-balancing motorcycles safer and more reliable for riders.

## III. METHODOLOGY

The methodology employed in this research investigates the efficacy of a reinforcement learning (RL) controller applied to a self-balancing motorcycle. This methodology outlines the steps taken to develop, implement, and validate the RL controller for the self-balancing motorcycle, focusing on both simulation and experimental aspects.

### A. DYNAMIC MODEL DEVELOPMENT

The self-balancing motorcycle was modelled as an inverted pendulum system using the Lagrange method. Key parameters such as mass, length, and moments of inertia were derived from specifications provided by the Arduino Engineering Kit Rev2 [28]. The dynamic model was implemented in Simulink to simulate the motorcycle's behaviour under varying conditions.

### B. REINFORCEMENT LEARNING (RL) CONTROLLER IMPLEMENTATION

A Deep Deterministic Policy Gradient (DDPG) algorithm was selected for controlling the self-balancing motorcycle. The state space included the motorcycle's angular position and velocity, while the action space consisted of the torque applied to the inertia wheel. Actor and critic networks were constructed using deep neural networks within the Simulink environment. Hyper parameters such as learning rates, batch sizes, and noise levels were fine-tuned to optimize training.

### C. SIMULATION AND TESTING

The trained DDPG agent was tested under different leaning angles and disturbances (step and pulse disturbances) to evaluate its robustness and performance. Comparative analysis was conducted against a conventional Proportional-Derivative (PD) controller. The simulation results were analysed for set point tracking, disturbance rejection, and response to model uncertainties.

### D. EXPERIMENTAL VALIDATION

To validate the simulation results, the trained DDPG policy was deployed onto a physical self-balancing motorcycle equipped with an Arduino Nano 33 IoT microcontroller and a BNO055 IMU sensor. The hardware setup was calibrated and tested under various initial conditions to observe real-time performance. The effectiveness of the RL controller was assessed based on the motorcycle's ability to maintain balance and recover from tilting angles in experimental scenarios.

### E. DATA ANALYSIS

Data from simulations and experiments were analysed to compare the performance of the DDPG-based controller with the PD controller. Metrics such as rise time, settling time, overshoot/undershoot, steady-state error, peak velocity, and control torque were used for quantitative evaluation.

The methodology outlined establishes a robust framework for implementing reinforcement learning in the control of self-balancing motorcycles. Future work will focus on refining the controller's performance through extensive experimental validation, aiming to enhance real-world applications of autonomous vehicles and robotics.

## IV. MOTORCYCLE DYNAMICS

When observing a stationary motorcycle from the rear, it resembles a pendulum rod with an attached inertia wheel. Hence, the motorcycle can be modelled as an inverted pendulum with some assumptions. The inverted pendulum is a classical control problem that has been used for many years in position control, aerospace vehicles control, and robotics [29]. Balancing the pendulum in the vertical upright position is very challenging and complex as it is highly unstable, under-actuated and nonlinear system [30]. General control techniques like LQR, MPC etc. requires a good knowledge of the system and accurate tuning to achieve desired performances. Nevertheless, describing an accurate system mathematical model is very difficult. Motorcycle can be modeled as an inverted pendulum with some assumptions as shown in Fig. 2.

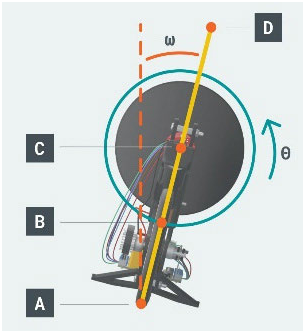


**FIGURE 2.** Motorcycle as inverted pendulum.

**A** is the rotation axis of the revolute joint between the pendulum rod and the ground. $l_{AD}$ is the length of the pendulum rod, i.e., the length of motorcycle frame. **B** is the centre of mass of the pendulum rod. **C** is the rotation axis of the revolute joint between the pendulum rod and the inertia wheel.

The dynamic model can be derived using the Lagrange method [28]. The state vector is given in (1) and time

derivative of state vector is given in (2).

$$x(t) = \begin{pmatrix} \theta \\ \dot{\theta} \\ \omega \end{pmatrix} \tag{1}$$

$$\dot{x}(t) = \begin{pmatrix} \dot{\theta} \\ \ddot{\theta} \\ \dot{\omega} \end{pmatrix} = \begin{pmatrix} \dot{\theta} \\ \frac{g \sin\theta(m_r l_{AB} + m_w l_{AC}) - \tau_m}{m_w(0.5R^2 + l_{AC}^2) + \frac{1}{12} m_r (3r^2 + l_{AD}^2) + m_r l_{AB}^2} \\ \tau_m / I_w^c - \ddot{\theta} \end{pmatrix} \tag{2}$$

where $\theta$ is angular position, $\dot{\theta}$ is angular velocity, $\ddot{\theta}$ is angular acceleration, $\omega$ *is* inertia wheel speed, $l_{AB}$ *is* length of pendulum rod between A and B, $l_{AD}$ is length of pendulum rod between A and D, $l_{AC}$ is length of pendulum rod between A and C, g is gravitational acceleration, $m_w$ is mass of inertia wheel, $m_r$ is mass of the rod, $\tau_m$ is motor torque applied to inertia wheel, R is radius of inertia wheel, r is radius of pendulum rod and $I_w^A$, $I_w^c$ are moment of inertia of the inertia wheel about points A and C respectively. Model parameters of motorcycle provided by the Arduino Engineering Kit Rev2 are shown in Table 1. $I_r^A$, $I_r^B$ are moment of inertia of the pendulum rod about points A and B respectively [28].

**TABLE 1. Model parameters.**

| Parameters | Values |
|---|---|
| *g* | 9.80655 m/sec$^2$ |
| *m_r* | 0.2948 Kg |
| *m_w* | 0.0695 Kg |
| *R* | 0.05 m |
| *r* | 0.02 m |
| *l_AD* | 0.13 m |
| *l_AC* | 0.13 m |
| *l_AB* | 0.065 m |
| *I_w_C* | 8.6875e-05 Kg-m$^2$ |
| *I_w_A* | 0.0013 Kg-m$^2$ |
| *I_r_B* | 4.4466e-04 Kg-m$^2$ |
| *I_r_A* | 0.0017 Kg-m$^2$ |

The motorcycle model that describes the dynamics of this nonlinear system in (2) is implemented in Simulink software and shown in Fig. 3.

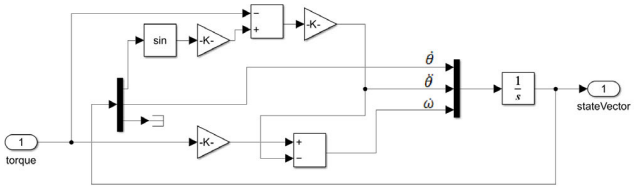


**FIGURE 3. Motorcycle model.**

## V. REINFORCEMENT LEARNING

RL makes use of a policy, a reward signal, a value function, and environment model. Reward represents the goal in RL problem. Agent interacts with the unknown environment to learn the policy on its own [31]. Policy decides the action to be taken by the agent in a particular state of the environment. When an action is performed on the environment, states of the environment changes and reward is provided to the agent. Delayed reward signals act as feedback signals to improve learning further. Reward indicates immediate benefit of being in a certain state whereas value function indicates long term signals that is expected to be collected from that state on, going into the future. Agent chooses an action based on past experiences (exploitation) and new preferences (exploration). The agent's aim is to maximize the long-term reward. The agent's goal is both to explore new states and at the same time to maximize its reward, to find an optimal policy. This is called Exploration vs Exploitation trade-off. There are several algorithms used in reinforcement learning like (Q-learning, SARSA, DDPG, PPO, etc.).

The agent contains two components, a policy and a learning algorithm as displayed in Fig. 4. The policy maps states -actions and is a function approximator, commonly a deep neural network with adjustable parameters. The learning algorithm iteratively updates the policy parameters based on the actions, observations, and reward with the aim of finding an optimal policy.

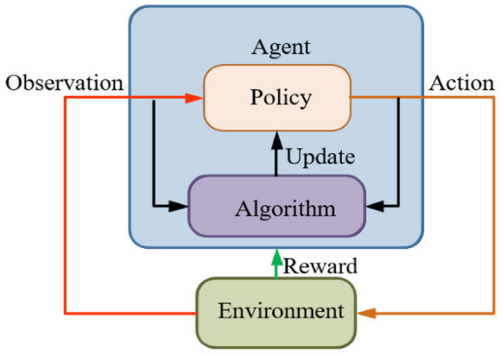


**FIGURE 4. RL workflow.**

The steps in implementing RL are as follows. Firstly, the environment needs to be defined, including the state space, action space, and dynamics. Next, a reinforcement learning algorithm needs to be chosen, such as Q-learning, SARSA, DDPG, PPO, or others. After that, deep neural networks for the agent need to be designed. This research work uses Deep Deterministic Policy Gradient (DDPG) learning algorithm for control of self-balancing motorcycle.

## VI. STEPS IN IMPLEMENTATION OF DDPG ALGORITHM FOR SELF BALANCING MOTORCYCLE

DDPG algorithm is a model-free, online, off-policy reinforcement learning (RL) technique [32], [33]. DDPG agents can be trained in environments with continuous/discrete observation and continuous action spaces. It uses the actor-critic architecture with deep neural networks. The actor network is responsible for selecting actions, while the critic network assesses the quality of those actions. DDPG uses a replay buffer to store the experiences of the agent, which are

then sampled randomly during training to reduce overfitting. Additionally, the algorithm uses a soft update mechanism to update the target networks, which helps to stabilize the learning process. The steps in implementing DDPG learning algorithm for controlling a self-balancing motorcycle are discussed here.

The first step is to model the self-balancing motorcycle's environment, including its state space, action space, and dynamics. The state space include motorcycle's angular position $\theta$ and angular velocity $\dot{\theta}$. The action space includes the torque applied to inertia wheel motor.

The second step is to select actor and critic deep neural network architecture. The actor takes the motorcycle's position and velocity as inputs and outputs the corresponding torque to be applied to the inertia wheel motor. The actor network is trained to learn a deterministic policy, which maps states to actions. The critic takes the motorcycle's position, velocity and the action selected as inputs and outputs an estimate of Q-value (expected cumulative reward). The critic provides a gradient signal that guides the actor network in adjusting its parameters.

Next step is to select the buffer size to store past experiences i.e., tuples of (state, action, reward, next state) collected during the interaction of the motorcycle with the environment. This helps to enhance training stability, due to data uncorrelation as these experiences are randomly sampled from the buffer during training.

A pseudo code outlines the basic steps of implementing the DDPG algorithm for training a self-balancing motorcycle agent [34].

1. **Initialize Networks and Parameters:**
   - Initialize actor network weights $\theta^{\mu}$ and critic network weights $\theta^{Q}$ randomly.
   - Initialize target networks weights.

$$\theta^{\mu'} \leftarrow \theta^{\mu} \text{ and } \theta^{Q'} \leftarrow \theta^{Q}$$

   - Initialize replay buffer R with maximum capacity N.
   - Set hyper parameters: discount factor $\gamma$, soft target update parameter $\tau$, exploration noise $\sigma$, mini-batch size M, learning rates $\alpha_{\mu}$ and $\alpha_{Q}$, etc.
2. **Define Actor and Critic Networks:**
   - Define the actor network $\mu(s \mid \theta^{\mu})$ to map states to actions.
   - Define the critic network $Q(s, a \mid \theta^{\mu})$ to estimate the Q-value for state-action pairs.
3. **Define Exploration Noise:**
   - Define exploration noise process N to add noise to the action outputs.
4. **Training Loop:**
   - For each episode:
     - Initialize the environment and set initial state $s_1$.
     - Initialize a random process N for action exploration.
     - For each time step:
       - Select action with noise.

$$a_t = \mu\left(s_t \mid \theta^{\mu}\right) + N_t$$

       - Execute action $a_t$ and observe reward $r_t$ and next state $s_{t+1}$.
       - Store transition $(s_t, a_t, r_t, s_{t+1})$ in replay buffer R.
       - Sample a random minibatch of M transitions $(s_i, a_i, r_i, s_{i+1})$ from R.
       - Compute targets:
         $y_i = r_i + \gamma Q'\left(s_{i+1}, \mu'\left(s_{i+1} \mid \theta^{\mu'}\right) \mid \theta^{Q'}\right)$ for critic update.
         $a'_i = \mu'\left(s_{i+1} \mid \theta^{\mu'}\right)$ for actor update.
       - Update critic by minimizing the loss:

$$L = \frac{1}{M}\sum_i \left(y_i - Q\left(s_i, a_i \mid \theta^{Q}\right)\right)^2$$

       - Update actor policy using sampled policy gradient:

$$\nabla_{\theta^{\mu}} J \approx \frac{1}{M}\sum_i \nabla_a Q\left(s, a \mid \theta^{Q}\right)\Big|_{s=si, a=\mu(s_i)} \times \nabla_{\theta^{\mu}} \mu\left(s \mid \theta^{\mu}\right)\Big|_{si}$$

       - Update target networks:

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\,\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\,\theta^{\mu'}$$

5. **Repeat Until Convergence:** Repeat the training loop until convergence or a predefined stopping criterion is met.

During training, noise is to be added to the actions to encourage the motorcycle to explore different actions. This is called exploration whereas selecting the best action is called exploitation. There should be a balance between these two. Next step is to fine-tune hyperparameters, such as learning rates, batch sizes, and noise levels, to achieve optimal performance. Once DDPG algorithm is trained and fine-tuned, the learned policy in the actor network can be used for controlling the self-balancing motorcycle in real-time.

## VII. DDPG AGENT TRAINING

Reinforcement learning environment is specified by creating a Simulink model as shown in Fig. 5 with an RL Agent block. In this model, the states, action, and reward signals are connected to the RL Agent block. Then the continuous reward function is designed by giving weights of 1, 0.1 & 0.001 for $\theta$, $\dot{\theta}$ and pervious control torque respectively. A reset function is created in MATLAB for resetting the environment at the start of each episode and starting the motorcycle at different initial states.

In DDPG algorithm, the sampling time of 0.01 sec and final time of 10 sec are considered. The critic learn rate of le-03 and actor learn rate of 1e-4 is used to train the critic and actor networks. The agent options considered for training the agent

are Target smooth factor le-3, discount factor 0.9, mini batch size 128 and experience buffer length le-6.

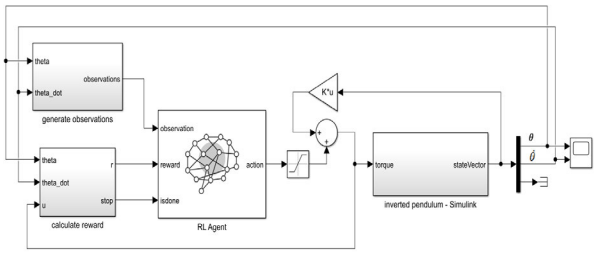


**FIGURE 5.** Simulink model of motorcycle with RL agent.

Agent is trained for 149 episodes with 1000 steps in each episode. The training is stopped manually using the stop training button, when true cumulative reward is closer to expected cumulative reward $Q_0$ estimated by the critic as shown in Fig. 6.

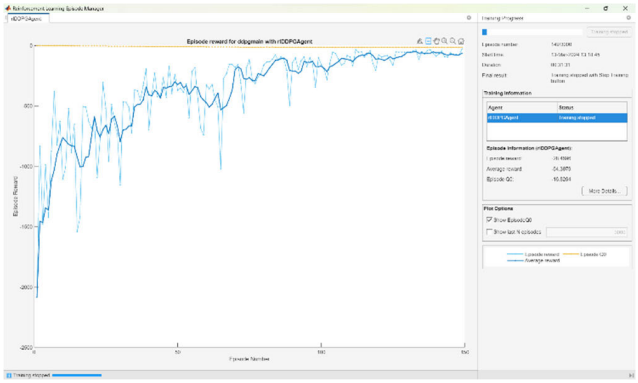

**FIGURE 6.** RL training progress.

It is observed that when the agent is trained in the correct way, the reward increases. The trained DDPG agent with motorcycle model is simulated with different positive and negative leaning angles of $10^{\circ}$ and $50^{\circ}$, pulse and step disturbances and model uncertainties.

## VIII. SIMULATION RESULTS AND DISCUSSION

The responses of motorcycle states ($\theta$, $\dot{\theta}$) and action (torque) are observed and displayed in Fig. 7 for both positive and negative leaning angles without any disturbance. From the responses, it is evident that the motorcycle is balanced very quickly i.e., angular position $\theta$ becomes 0 as the control torque provided by trained DDPG agent approaches 0.

The inertia wheel started rotating as the motorcycle position is 10°. The trained agent tries to adjust the motor torque applied on the inertia wheel to 0 N-m so that the inertia wheel stops rotating maintaining the motorcycle balance. As torque approaches 0 N-m, motorcycle position and velocity also become 0° and 0°/sec respectively. It is noticed that the motorcycle balancing is very fast even in case of higher leaning angles.

To observe disturbance rejection response, a step disturbance of amplitude 0.2 N-m is given to the torque signal for positive and negative $10^{\circ}$ and $50^{\circ}$ leaning angles at 5 sec. The regulatory response of observations ($\theta$, $\dot{\theta}$) and action (torque) are as shown in Fig. 8. It is noticed that angular velocity $\dot{\theta}$ approaches 0 very quickly in all the scenarios. Due to fast disturbance rejection at higher leaning angle of $50^{\circ}$, the motorcycle balancing is quick compared to smaller leaning angle of $10^{\circ}$ because of slow disturbance rejection.

To verify the robustness of proposed controller, a pulse disturbance of amplitude 0.05 N-m, period 5 sec and pulse width 2 % of period is also given to the torque signal for positive and negative $10^{\circ}$ and $50^{\circ}$ leaning angles.

The regulatory response of observations ($\theta$, $\dot{\theta}$) and action (torque) are as shown in Fig. 9. The motorcycle tends to respond more quickly to pulse disturbances, much like its reaction to step disturbances.

After validating the performance of the trained agent under different conditions, a policy is generated for the trained DDPG RL agent and the RL agent is replaced with the policy function, as illustrated in Fig. 10.

The states ($\theta$ and $\dot{\theta}$) and action (torque) with policy function for $50^{\circ}$ leaning angle aim for responses like those observed in the trained RL agent as shown in Fig. 11.

Finally, the policy function is working as the controller which changes torque based on states $\theta$ and $\dot{\theta}$. To evaluate the effectiveness of proposed controller developed using DDPG algorithm in RL, the responses are compared with the conventional PD controller in case of $10^{\circ}$ leaning angle and pulse disturbance of amplitude 0.2 N-m, period 5 sec and pulse width 2 % of period and shown in Fig.12 and Fig.13 respectively.

It is observed that the control torque changes quickly to bring the motorcycle to balance position with the proposed controller in case of positive $10^{\circ}$ leaning angle and no disturbance. Angular position and velocity obtained with DDPG controller settled to zero much faster than the PD controller.

In case of pulse disturbance at positive $50^{\circ}$ leaning angle, it is noticed that the control torque with the proposed controller changes similarly with PD controller to balance the motorcycle. With DDPG controller, the motorcycle is quickly balanced i.e., angular position $\theta$ approaches 0 compared to PD controller. The advantage of DDPG lies in its ability to rapidly adapt and stabilize the motorcycle.

While both PD and DDPG controllers aim to stabilize a motorcycle after a disturbance, DDPG shows superior performance in terms of speed and adaptability due to its reinforcement learning framework and continuous action policy.

The evaluation aims to demonstrate not only the effectiveness of the proposed controller under normal conditions but also its robustness in handling parameter variations that can occur in real-world applications. This approach ensures that the controller can maintain stability and performance even when the system parameters deviate from their expected values. Robustness refers to the ability of the controller to

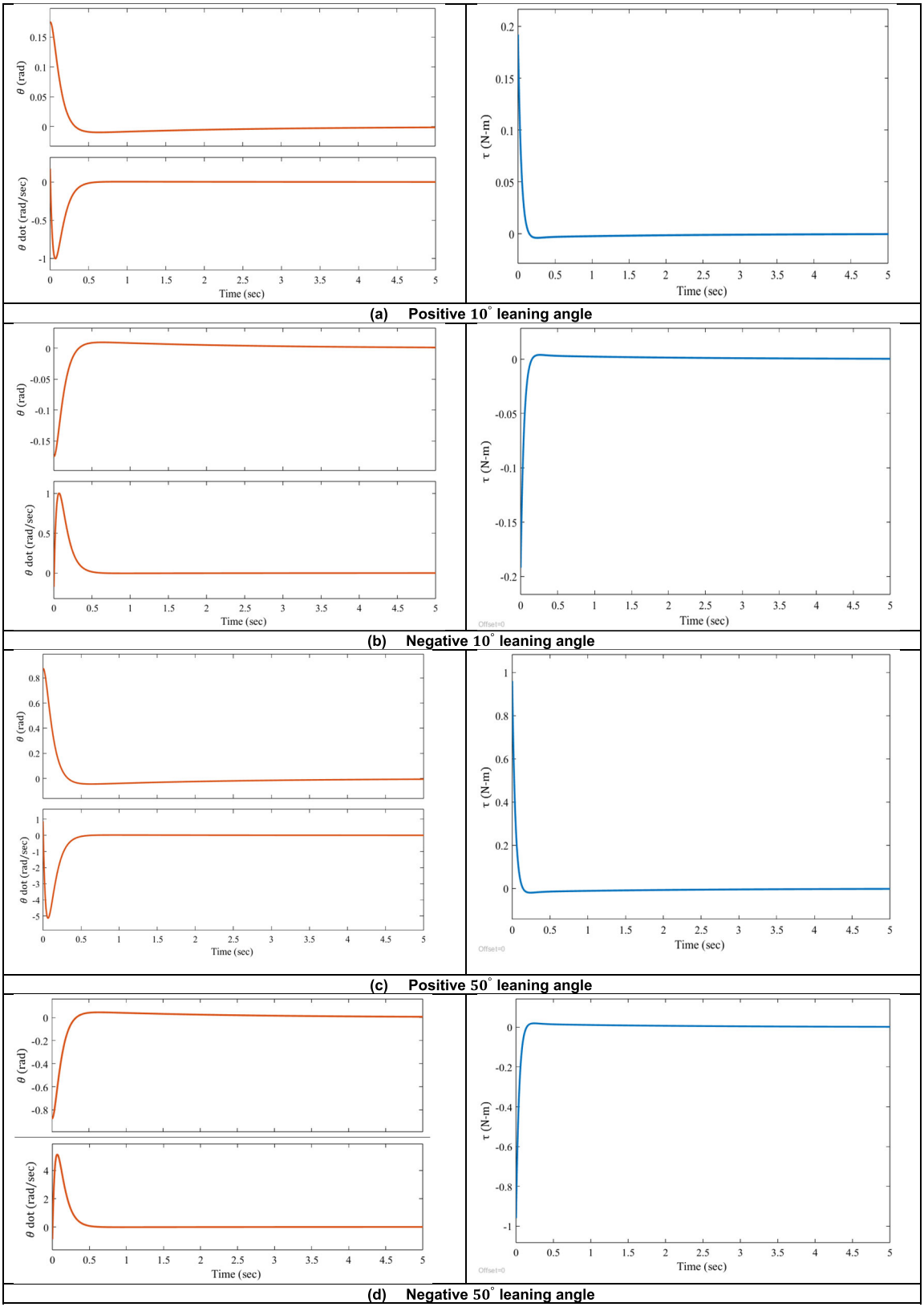


**FIGURE 7.** State and action simulation responses for various leaning angles.

maintain its performance despite uncertainties or variations in the system parameters.

It is clear from the comparison results that the proposed controller balances the motorcycle rapidly under

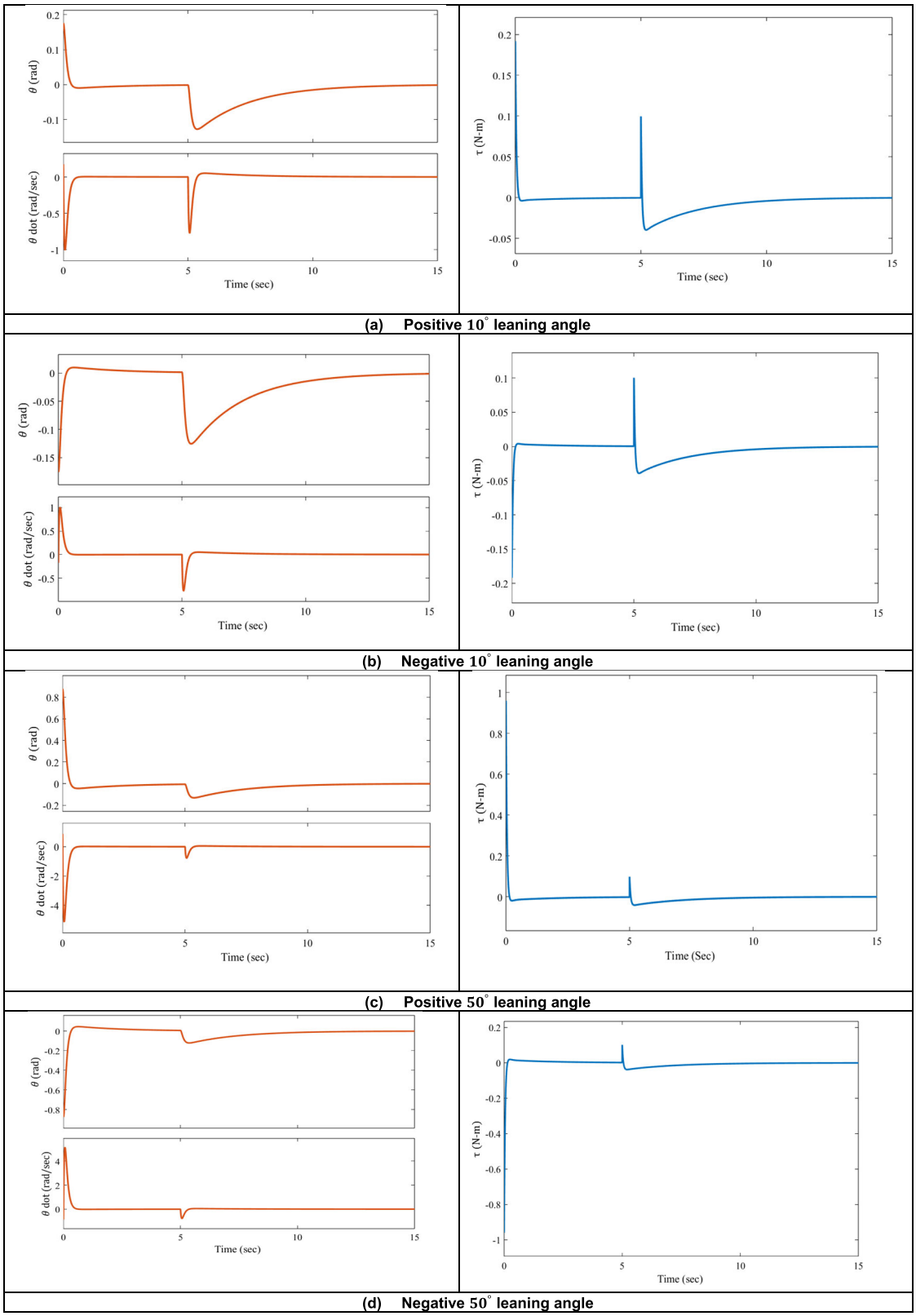


**FIGURE 8.** State and action simulation responses in case of step disturbance.

smaller and higher leaning angles with and without disturbances. To evaluate the robustness of proposed controller to model error, the responses are compared with the conventional PD controller in case of ±10% change in

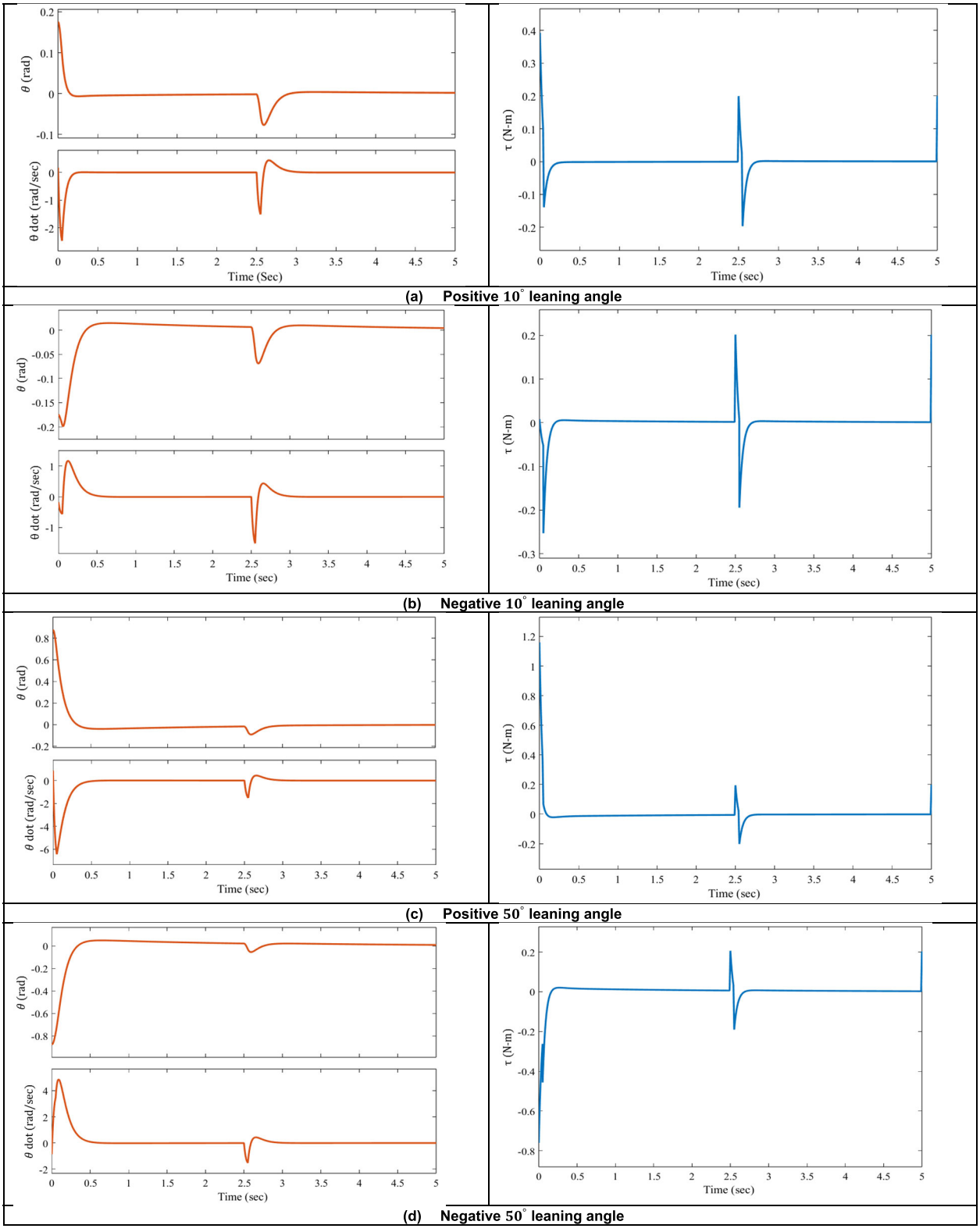


**FIGURE 9.** State and action simulation responses in case of pulse disturbance.

mass of the pendulum rod as shown in Fig. 14 and Fig.15.

Even in case of model error, the motorcycle balances very quickly and efficiently with DDPG controller. It is evident

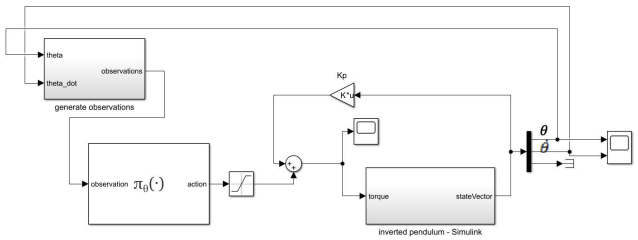


**FIGURE 10.** Simulink model of motorcycle with RL policy.

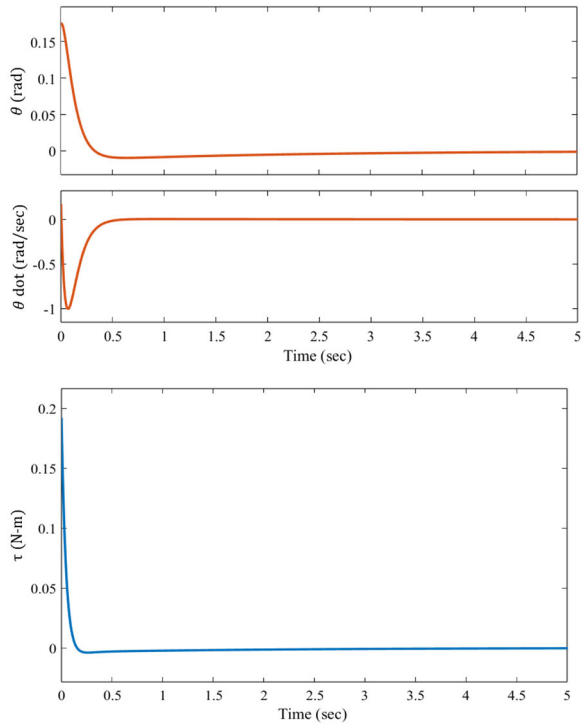


**FIGURE 11.** Policy function validation results.

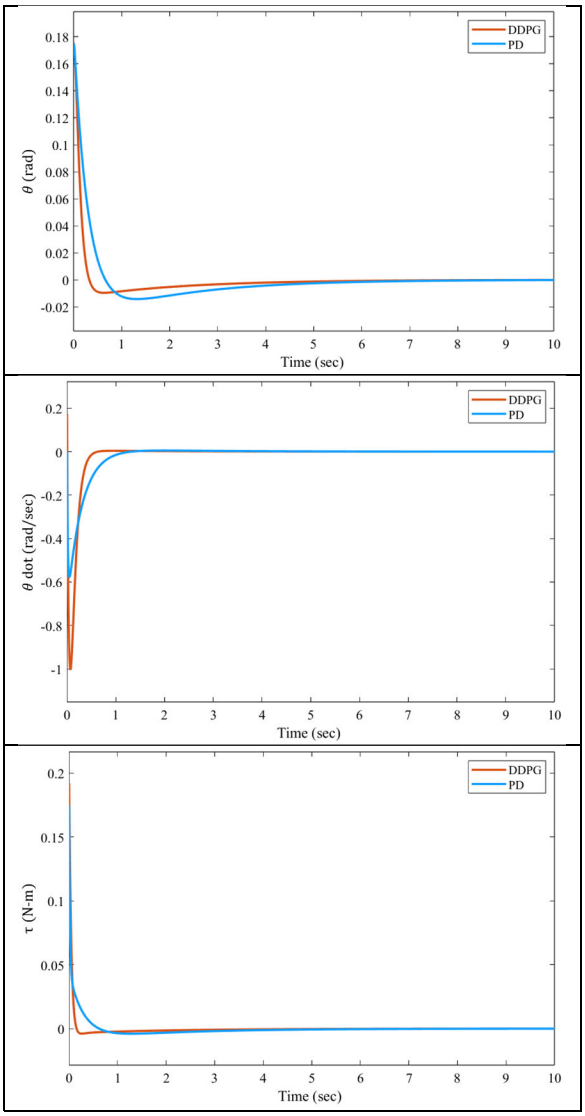


**FIGURE 12.** Comparison of responses for positive 10° leaning angle.

from all the results that the proposed controller is more efficient in balancing the motorcycle at various leaning angles and more robust against disturbances and model uncertainties. The performance of motorcycle with DDPG controller is superior compared to that of the PD controller.

The DDPG policy tested initially in simulation environments helps validate its performance before deploying it on real hardware. Now, this policy function is ready to implement for real time hardware. Once the policy function (the trained neural network) is ready for implementation on real-time hardware, it can effectively adapt and control the motorcycle based on its observations of the environment, leading to smoother, more precise, and potentially safer operation.

## IX. EXPERIMENTAL VALIDATION AND RESULTS

This policy is deployed on physical hardware. MATLAB and Simulink support packages for Arduino Hardware are required for developing hardware model. Firstly, the motorcycle subsystem is created which consist of inertia wheel, IMU sensor and battery Read as shown in Fig. 16.

The digital controller subsystem consists of a state flow chart, sensor preprocessing block, balancing switch, and the remaining components as shown in Fig. 17.

The state flow chart has four parallel states out of which check fallen, check IMU calibration and check battery are input states and calculate enable is the output state. These three input states execute concurrently and only if all are true, then control is enabled. The last step within design workflow is implementing a controller with the DDPG policy function as shown in Fig. 18. Before deploying the model, the balancing switch is moved to the off position.

Self-balancing motorcycle hardware is connected to controller with RL policy as displayed in Fig. 19. Once the model is deployed on to Arduino nano 33 IOT board, the IMU sensor is calibrated. IMU is calibrated by moving the motorcycle in 3 spatial axes. The states inside the state flow chart are observed and if they are active, the control is enabled.

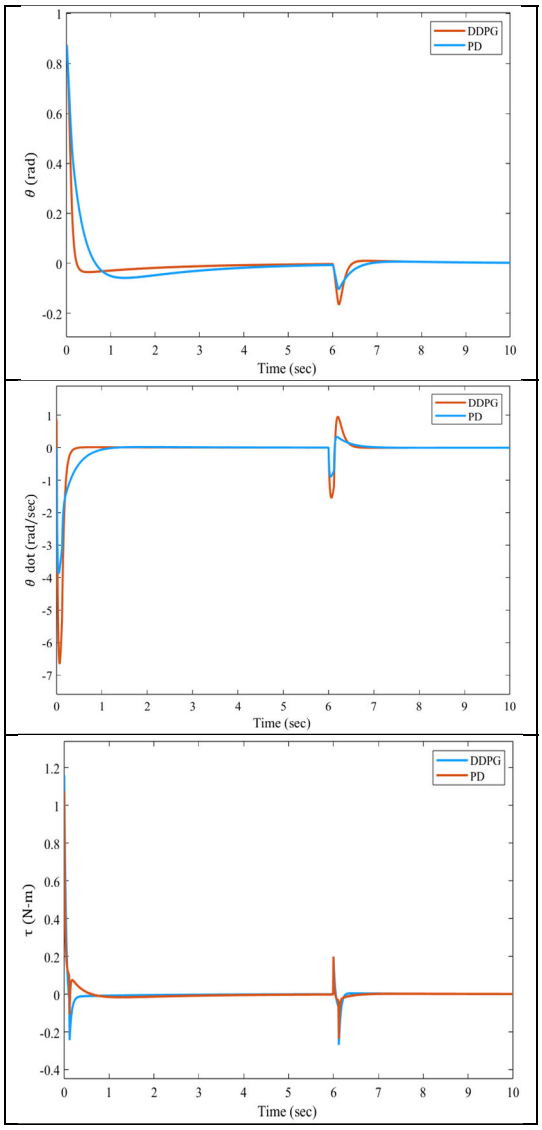


**FIGURE 13.** Comparison of responses for pulse disturbance.

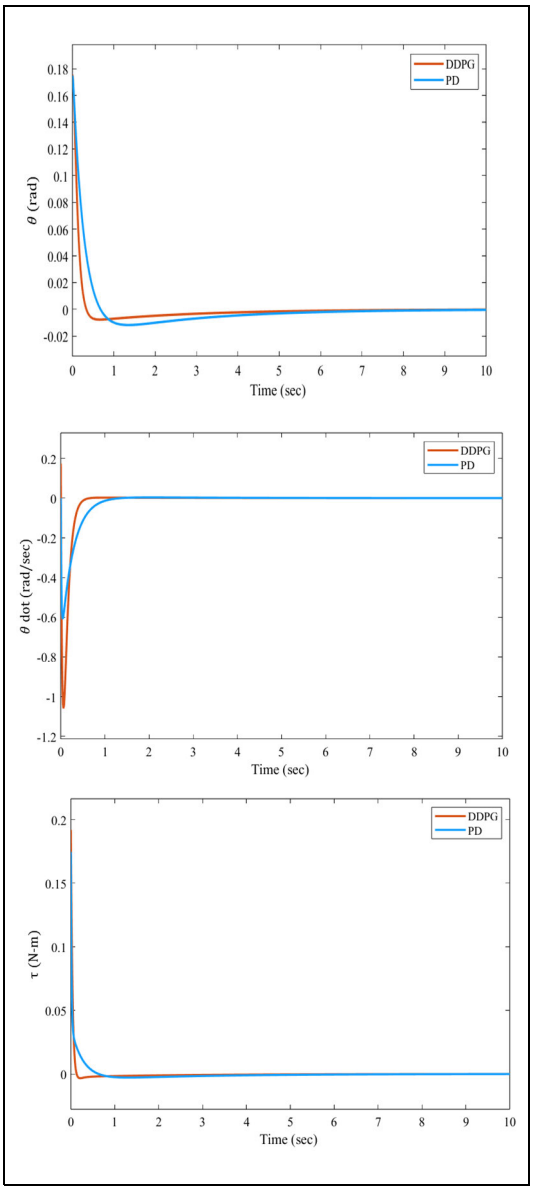


**FIGURE 14.** Comparison of responses in case of −10% model uncertainty.

The balancing switch is moved to on position to enable motorcycle balancing.

The experimentation begins by tilting the motorcycle at a positive angle of 10 degrees. Real-time state and action responses are monitored and analyzed, with the results depicted in Fig. 20. It's observed that with the proposed DDPG controller, the motorcycle rapidly returns to its upright position within a mere 1.2 seconds. This quick recovery time signifies a substantial improvement over the conventional PD controller.

Comparatively, existing literature suggests that with a PD controller, the motorcycle achieves balance only for leaning angles below 6 degrees, and even then, it takes approximately 5.6 seconds to stabilize. During this stabilization period, the controller continuously operates for about 2-4 seconds to maintain stability. However, beyond this timeframe, the controller ceases operation, and the motorcycle begins to tilt once more.

To address these limitations and further enhance motorcycle balance across a range of leaning angles, as well as to mitigate external disturbances and model uncertainties, the reinforcement learning based DDPG algorithm is introduced. This algorithm leverages the principles of deep reinforcement learning to autonomously learn optimal control policies through interaction with the environment.

The results obtained from employing the DDPG algorithm demonstrate significant improvements. The motorcycle is now able to maintain balance for an extended duration of 8-10 seconds, a notable enhancement over the existing controller's capabilities. This increased stability duration ensures better performance under varying conditions and provides greater resilience against disturbances.

Simulation results are provided for motorcycle leaning angles of $+10$, $-10$, $+50$, and $-50$ degrees. Comparison is

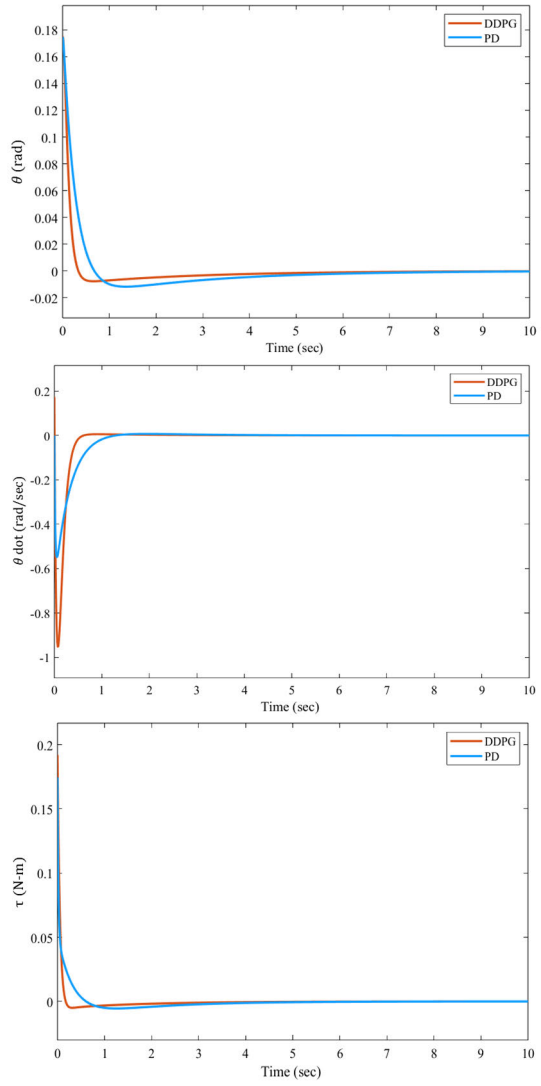


**FIGURE 15.** Comparison of responses in case of +10% model uncertainty.

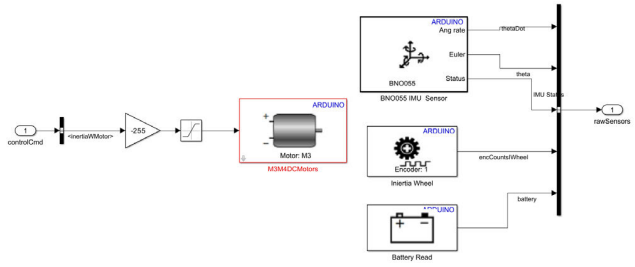


**FIGURE 16.** Hardware model of motorcycle.

made for setpoint tracking, disturbance rejection, and model uncertainty. Table 2 presents the comparative performance of DDPG-based reinforcement learning control and PD control in terms of rise time, settling time, overshoot/undershoot, steady-state error, peak velocity, and control torque for different leaning angles and control objectives. DDPG generally exhibits faster rise times and settling times compared

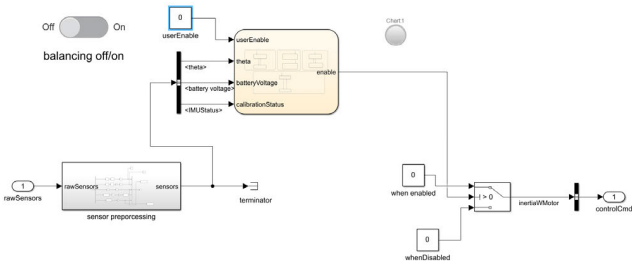


**FIGURE 17.** Hardware model of digital controller.

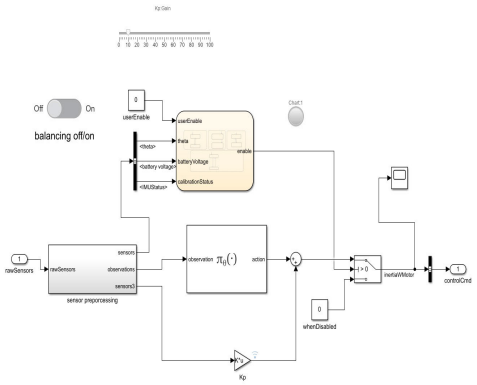


**FIGURE 18.** Hardware model of motorcycle with RL policy.

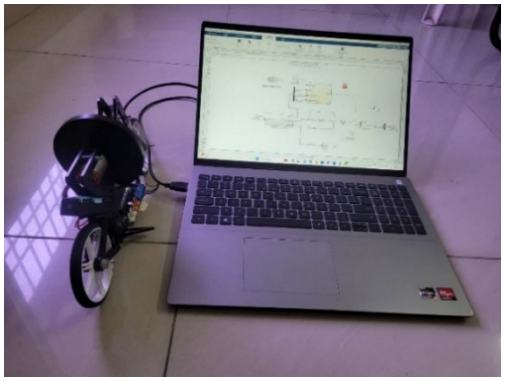

**FIGURE 19.** Experimental setup of motorcycle with RL controller.

to PD controllers across different leaning angles. DDPG shows better disturbance rejection and model uncertainty handling in terms of overshoot/undershoot. Peak velocities and control torques vary depending on the leaning angle and control method, with DDPG often providing higher velocities and torques. These results indicate the potential effectiveness of DDPG-based reinforcement learning control for self-balancing motorcycles, particularly in handling disturbances and uncertainties while achieving desired tracking performance.

Overall, the utilization of the reinforcement learning based DDPG algorithm represents a substantial advancement in motorcycle control systems. By harnessing the power of machine learning, the controller not only improves balance and stability but also enhances adaptability to diverse

**TABLE 2. Comparison of performance for various leaning angles.**

| Leaning angle | Parameters | Setpoint tracking | | Disturbance rejection | | Model uncertainty | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | +10% | | -10% | |
| | | DDPG | PD | DDPG | PD | DDPG | PD | DDPG | PD |
| Positive 10° | Rise time (sec) | 0.29 | 0.46 | 0.04 | 0.06 | 0.28 | 0.48 | 0.22 | 0.47 |
| | Settling time (sec) | 4.5 | 6 | 4.8 | 5.4 | 4.5 | 6 | 5.1 | 5.9 |
| | Undershoot (or) overshoot (rad) | -0.01 | -0.017 | -0.10 | -0.005 | -0.012 | -0.17 | -0.007 | -0.15 |
| | Steady sate error | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Peak velocity (rad/sec) | -1.01 | -0.58 | -2.55 | -1.48 | -0.95 | -0.55 | -1.06 | -0.61 |
| | Control torque (N-m) | 0.192 | 0.175 | 0.39 | 0.35 | 0.191 | 0.17 | 0.192 | 0.175 |
| Negative 10° | Rise time (sec) | 0.23 | 0.47 | 0.35 | 0.56 | 0.23 | 0.46 | 0.21 | 0.46 |
| | Settling time (sec) | 4.6 | 5.8 | 4.9 | 5.6 | 4.8 | 6.2 | 5.3 | 5.8 |
| | Undershoot (or) overshoot (rad) | 0.01 | 0.018 | 0.027 | 0.029 | 0.012 | 0.17 | 0.005 | 0.17 |
| | Steady sate error | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Peak velocity (rad/sec) | 1.01 | 0.57 | 0.55 | 0.32 | 0.95 | 0.55 | 1.06 | 0.65 |
| | Control torque (N-m) | -0.192 | -0.174 | 0.008 | 0.015 | -0.193 | -0.175 | -0.192 | -0.174 |
| Positive 50° | Rise time (sec) | 0.22 | 0.47 | 0.15 | 0.42 | 0.23 | 0.46 | 0.22 | 0.45 |
| | Settling time (sec) | 5.3 | 6.2 | 4.9 | 5.5 | 5.5 | 6.3 | 5.3 | 6.0 |
| | Undershoot (or) overshoot (rad) | -0.05 | -0.08 | -0.039 | -0.06 | -0.05 | -0.08 | -0.04 | -0.06 |
| | Steady sate error | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Peak velocity (rad/sec) | -5.13 | -2.98 | -6.64 | -3.872 | -4.9 | -2.85 | -5.40 | -3.15 |
| | Control torque (N-m) | 0.96 | 0.88 | 1.16 | 1.07 | 0.96 | 0.87 | 0.96 | 0.87 |
| Negative 50° | Rise time (sec) | 0.24 | 0.46 | 0.27 | 0.48 | 0.23 | 0.46 | 0.22 | 0.46 |
| | Settling time (sec) | 5.5 | 6.3 | 5.1 | 5.7 | 5.7 | 6.5 | 5.5 | 6.2 |
| | Undershoot (or) overshoot (rad) | 0.05 | 0.08 | 0.065 | 0.078 | 0.055 | 0.08 | 0.04 | 0.06 |
| | Steady sate error | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | Peak velocity (rad/sec) | 5.13 | 2.98 | 3.62 | 2.36 | 4.9 | 2.84 | 5.40 | 3.10 |
| | Control torque (N-m) | -0.97 | -0.88 | -0.76 | -0.67 | -0.96 | -0.87 | -0.96 | -0.87 |

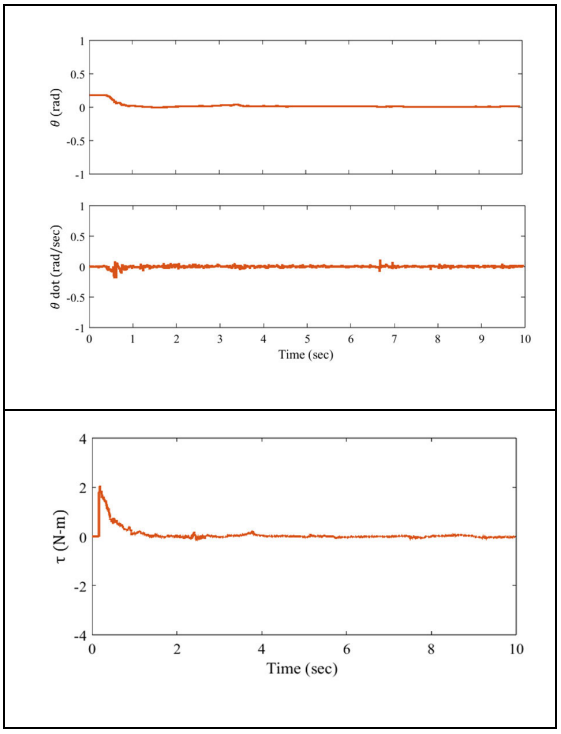


**FIGURE 20. Motorcycle balancing with DDPG controller in real-time.**

environmental conditions and uncertainties, ultimately contributing to safer and more efficient motorcycle operation.

## X. CONCLUSION AND FUTURE SCOPE

In this paper, Deep RL controller using DDPG algorithm is implemented for balancing the motorcycle on its own. Whenever the motorcycle tilts, the inertia wheel starts rotating to counter the torque. After training the DDPG agent for balancing the motorcycle, the performance is validated in both simulation and hardware. This proposed control technique ensures a more robust and stable controller for greater leaning angles with less settling time and high balancing time.

Various RL algorithms like Proximal Policy Optimization (PPO), Twin Delayed DDPG etc. can be used to optimize the motorcycle behavior for any leaning angle and to further improve the stabilizing time and robustness.

## REFERENCES

[1] S. Stasinopoulos, M. Zhao, and Y. Zhong, "Simultaneous localization and mapping for autonomous bicycles," *Int. J. Adv. Robotic Syst.*, vol. 14, no. 3, pp. 1–15, May 2017.

[2] Y. Zhang, J. Li, J. Yi, and D. Song, "Balance control and analysis of stationary riderless motorcycles," in *Proc. IEEE Int. Conf. Robot. Autom.*, Shanghai, China, May 2011, pp. 3018–3023.

[3] Y. Yu and M. Zhao, "Steering control for autonomously balancing bicycle at low speed," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Kuala Lumpur, Malaysia, Dec. 2018, pp. 33–38.

[4] Y. Sun, M. Zhao, B. Wang, X. Zheng, and B. Liang, "Polynomial controller for bicycle robot based on nonlinear descriptor system," in *Proc. 46th Annu. Conf. IEEE Ind. Electron. Soc.*, Singapore, Oct. 2020, pp. 1–6.

[5] C.-K. Chen, T.-D. Chu, and X.-D. Zhang, "Modeling and control of an active stabilizing assistant system for a bicycle," *Sensors*, vol. 19, no. 2, p. 248, Jan. 2019.

[6] X. Zheng, X. Zhu, Z. Chen, Y. Sun, B. Liang, and T. Wang, "Dynamic modeling of an unmanned motorcycle and combined balance control with both steering and double CMGs," *Mechanism Mach. Theory*, vol. 169, Mar. 2022, Art. no. 104643.
[7] K. He, Y. Deng, G. Wang, X. Sun, Y. Sun, and Z. Chen, "Learning-based trajectory tracking and balance control for bicycle robots with a pendulum: A Gaussian process approach," *IEEE/ASME Trans. Mechatronics*, vol. 27, no. 2, pp. 634–644, Apr. 2022.
[8] Y. Kim, H. Kim, and J. Lee, "Stable control of the bicycle robot on a curved path by using a reaction wheel," *J. Mech. Sci. Technol.*, vol. 29, no. 5, pp. 2219–2226, May 2015.
[9] L. Chen, J. Liu, H. Wang, Y. Hu, X. Zheng, M. Ye, and J. Zhang, "Robust control of reaction wheel bicycle robot via adaptive integral terminal sliding mode," *Nonlinear Dyn.*, vol. 104, no. 3, pp. 2291–2302, May 2021.
[10] H.-W. Kim, J.-W. An, H. D. Yoo, and J.-M. Lee, "Balancing control of bicycle robot using PID control," in *Proc. 13th Int. Conf. Control, Autom. Syst. (ICCAS)*, Las Vegas, NV, USA, Oct. 2013, pp. 145–147.
[11] K. Kanjanawanishkul, "LQR and MPC controller design and comparison for a stationary self-balancing BR with a reaction wheel," *Kybernetika*, vol. 51, no. 2, pp. 173–191, 2015.
[12] A. Owczarkowski, D. Horla, and J. Zietkiewicz, "Introduction of feedback linearization to robust LQR and LQI control—Analysis of results from an unmanned bicycle robot with reaction wheel," *Asian J. Control*, vol. 21, no. 2, pp. 1028–1040, Mar. 2019.
[13] J. Yi, D. Song, A. Levandowski, and S. Jayasuriya, "Trajectory tracking and balance stabilization control of autonomous motorcycles," in *Proc. IEEE Int. Conf. Robot. Autom.*, Orlando, FL, USA, May 2006, pp. 2583–2589.
[14] C.-L. Hwang, H.-M. Wu, and C.-L. Shih, "Fuzzy sliding-mode underactuated control for autonomous dynamic balance of an electrical bicycle," *IEEE Trans. Control Syst. Technol.*, vol. 17, no. 3, pp. 658–670, May 2009.
[15] S. Choi, T. P. Le, Q. D. Nguyen, M. A. Layek, S. Lee, and T. Chung, "Toward self-driving bicycles using state-of-the-art deep reinforcement learning algorithms," *Symmetry*, vol. 11, no. 2, p. 290, Feb. 2019.
[16] Q. Zheng, D. Wang, Z. Chen, Y. Sun, and B. Liang, "Continuous reinforcement learning based ramp jump control for single-track two-wheeled robots," *Trans. Inst. Meas. Control*, vol. 44, no. 4, pp. 892–904, Feb. 2022.
[17] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, "Residual reinforcement learning for robot control," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, Montreal, QC, Canada, May 2019, pp. 6023–6029.
[18] A. I. Glushchenko, V. A. Petrov, and K. A. Lastochkin, "On development of neural network controller with online training to control two-wheeled balancing robot," in *Proc. Int. Russian Autom. Conf. (RusAutoCon)*, Sochi, Russia, Sep. 2018, pp. 1–6.
[19] M. A. Imtiaz, M. Naveed, N. Bibi, S. Aziz, and S. Z. H. Naqvi, "Control system design, analysis & implementation of two wheeled self balancing robot (TWSBR)," in *Proc. IEEE 9th Annu. Inf. Technol., Electron. Mobile Commun. Conf. (IEMCON)*, Vancouver, BC, Canada, Nov. 2018, pp. 431–437.
[20] A. A. Bature, S. Buyamin, M. N. Ahmad, and M. Muhammad, "A comparison of controllers for balancing two-wheeled inverted pendulum robot," *Int. J. Mech. Mechatron. Eng.*, vol. 14, no. 3, pp. 62–68, 2014.
[21] A. Kharola, P. Patil, S. Raiwani, and D. Rajput, "A comparison study for control and stabilisation of inverted pendulum on inclined surface (IPIS) using PID and fuzzy controllers," *Perspect. Sci.*, vol. 8, pp. 187–190, Sep. 2016.
[22] H.-W. Kim and S. Jung, "Fuzzy logic application to a two-wheel mobile robot for balancing control performance," *Int. J. Fuzzy Log. Intell. Syst.*, vol. 12, no. 2, pp. 154–161, Jun. 2012.
[23] K.-H. Su, Y.-Y. Chen, and S.-F. Su, "Design of neural-fuzzy-based controller for two autonomously driven wheeled robot," *Neurocomputing*, vol. 73, nos. 13–15, pp. 2478–2488, Aug. 2010.
[24] C. F. Juang, Y. W. Cheng, and Y. M. Lin, "Visually interpretable fuzzy neural classification network with deep convolutional feature maps," *IEEE Trans. Fuzzy Syst.*, vol. 32, no. 3, pp. 1063–1077, Mar. 2024.
[25] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed., Cambridge, MA, USA: MIT Press, 2018.
[26] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*, 2nd ed., New York, NY, USA: Springer, 2016.
[27] *Self-Balancing Motorcycle Using Arduino Engineering Kit Rev 2—Simulink*. Accessed: Feb. 8, 2023. [Online]. Available: https://www.mathworks.com/hardware-support/arduino-engineering-kit-simulink.html
[28] *Self-balancing Motorcycle—Arduino Engineering Kit*. Accessed: Feb. 8, 2023. [Online]. Available: https://www.arduino.cc/education/engineering-kit/self-balancing-motorcycle
[29] O. Boubaker, "The inverted pendulum: History and survey of open and current problems in control theory and robotics," in *The Inverted Pendulum in Control Theory and Robotics: From Theory to New Innovations*, vol. 111, no. 1, O. Boubaker and R. Iriarte, Eds. Tunis, Tunisia: National Institute of Applied Sciences and Technology, Oct. 2017, pp. 1–22. [Online]. Available: https://digital-library.theiet.org/content/books/ce/pbce111e
[30] S. Irfan, L. Zhao, S. Ullah, A. Mehmood, and M. F. U. Butt, "Control strategies for inverted pendulum: A comparative analysis of linear, nonlinear, and artificial intelligence approaches," *PLoS ONE*, vol. 19, no. 3, Mar. 2024, Art. no. e0298093.
[31] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed., Boston, MA, USA: Pearson, 2016.
[32] B. Belousov, H. Abdulsamad, P. Klink, S. Parisi, and J. Peters, *Reinforcement Learning Algorithms: Analysis and Applications*, 1st ed., Berlin, Germany: Springer, 2019.
[33] J. Kołota and T. C. Kargin, "Comparison of various reinforcement learning environments in the context of continuum robot control," *Appl. Sci.*, vol. 13, no. 16, p. 9153, Aug. 2023.
[34] E. H. Sumiea, S. J. Abdulkadir, H. S. Alhussian, S. M. Al-Selwi, A. Alqushaibi, M. G. Ragab, and S. M. Fati, "Deep deterministic policy gradient algorithm: A systematic review," *Heliyon*, vol. 10, no. 9, May 2024, Art. no. e30697.

**K. VIJAYA LAKSHMI** was born in Visakhapatnam, Andhra Pradesh, India, in 1982. She received the B.Tech. degree in instrumentation engineering from VRSEC, Vijayawada, in 2003, and the M.Tech. degree in industrial process instrumentation from Andhra University, in 2006. She is currently pursuing the Ph.D. degree with the School of Electrical Engineering, Vellore Institute of Technology, Vellore, Tamil Nadu.

From 2006 to 2008, she was a Lecturer at VRSEC. Since 2008, she has been an Assistant Professor with the Department of Electronics and Instrumentation Engineering, Velagapudi Ramakrishna Siddhartha Engineering College, Vijayawada. Her research interests include process control, control systems, process modelling and simulation, reinforcement learning, and optimization techniques.

**M. MANIMOZHI** received the B.E. degree in electronics and instrumentation engineering and the M.E. degree in process control and instrumentation from Annamalai University, and the Ph.D. degree from VIT University, Vellore. She is currently working as a Professor with the Department of Control and Automation, VIT University. Her research interests include control and instrumentation, state estimation, sensors and signal conditioning, and reinforcement learning.