p-ISSN 2301–4156 | e-ISSN 2460–5719

Inverted Pendulum Control: A Comparative Study from Conventional Control to Reinforcement Learning

Ahmad Ataka[1], Andreas Sandiwan[2], Hilton Tnunay[3], Dzuhri Radityo Utomo[4], Adha Imam Cahyadi[5]

[1,2,4,5] Department of Electrical and Information Engineering, Faculty of Engineering Universitas Gadjah Mada, Jl. Grafika No. 2 Kampus UGM, Yogyakarta 55281 INDONESIA (tel.: 0274-552305; email: [1]ahmad.ataka.ar@ugm.ac.id, [2]andreassandiwan93@mail.ugm.ac.id, [4]dzuhri.r.u@ugm.ac.id, [5]adha.imam@ugm.ac.id)
[3] Geomatics Section, Faculty of Engineering Technology, KU Leuven, 9000 Ghent, BELGIUM (email: [3]hilton.tnunay@kuleuven.be)

[Received: 9 May 2023, Revised: 17 July 2023]
Corresponding Author: Ahmad Ataka

ABSTRACT — The rise of deep reinforcement learning in recent years has led to its usage in solving various challenging problems, such as chess and Go games. However, despite its recent success in solving highly complex problems, a question arises on whether this class of method is best employed to solve control problems in general, such as driverless cars, mobile robot control, or industrial manipulator control. This paper presents a comparative study between various classes of control algorithms and reinforcement learning in controlling an inverted pendulum system to evaluate the performance of reinforcement learning in a control problem. A test was performed to test the performance of root locus-based control, state compensator control, proportional-derivative (PD) control, and a reinforcement learning method, namely the proximal policy optimization (PPO), to control an inverted pendulum on a cart. The performances of the transient responses (such as overshoot, peak time, and settling time) and the steady-state responses (namely steady-state error and the total energy) were compared. It is found that when given a sufficient amount of training, the reinforcement learning algorithm was able to produce a comparable solution to its control algorithm counterparts despite not knowing anything about the system's properties. Therefore, it is best used to control plants with little to no information regarding the model where testing a particular policy is easy and safe. It is also recommended for a system with a clear objective function.

KEYWORDS — Reinforcement Learning, Inverted Pendulum, Root Locus, State Feedback, PD Control.

I. INTRODUCTION

The field of reinforcement learning has gained interest in recent years. Although its first development started in the 1990s [1], it has only recently come to prominence with the growth of deep learning in 2012 [2]. This merger brings forth deep reinforcement learning, combining the deep neural network capable of approximating complex functions with a reinforcement learning framework capable of solving a specific type of problem known as a control problem. This kind of problem is where a decision or action must be made for a particular system (also known as the environment). Since its huge success in solving games called Atari [3], [4], deep reinforcement learning has been applied successfully to solving various games such as chess, Go, and Dota2 [5]–[7].

Among the problems that deep reinforcement learning tried to solve are traditional control and robotics problems such as driverless car control [8], mobile robot navigation [9], legged robot control [10], and industrial manipulator control [11]. Despite its success, there are still challenges in applying deep reinforcement learning to general control problems in comparison to the game problem.

One of the main challenges is the lack of a clear reward function [12]. While a game environment normally provides a clear success criterion (such as a scoring system or a win condition), it is not generally the case in control problems. As has been extensively studied, the choice of reward is critical to the performance of deep reinforcement learning [12]. Therefore, solving a general control problem using reinforcement learning creates a new problem of designing a reward function that reflects the success criteria of the control problem. Another challenge in applying reinforcement learning to control problems is the non-episodic nature of the problem [13].

Apart from the aforementioned problems, there is also a challenge in performing training in control problems, as has been reported in [12]. While it is straightforward to perform training for the game problem, training a real control system proves to be complicated. Since reinforcement learning requires a considerable amount of training data to find the desirable policy, there is no guarantee that the policy being applied at one particular moment does not lead to catastrophic performance. Many researchers tried to overcome this problem by training the policy in the simulation prior to implementing the policy in a real system, a technique called sim-to-real [12]. It has been particularly successful in complex cases, such as training a quadruped robot to perform running tasks. However, this method depends on the availability of a physics-based simulator which needs to be sophisticated enough to mimic the performance of the real system.

Finally, some works reported the phenomena of unstable performance of deep reinforcement learning when applied to some control problems [12]. It is in contrast to conventional control methods in which stability can usually be analyzed and guaranteed if possible. Moreover, in most control problems, the goal is not only simply "solving the problem" but also producing a performance with a certain specification.

A comparative study between the reinforcement learning algorithm (namely the proximal policy optimization (PPO)) and several conventional control algorithms (namely the root locus-based compensator, the state compensator, and the proportional-derivative (PD) control for inverted pendulum control) is performed in this paper to evaluate the performance of reinforcement learning in control problems. The problem of controlling the inverted pendulum was chosen since it is one of the standard problems used to test various control algorithms [14]. It is due to the fact that an inverted pendulum is a nonlinear system with unstable equilibrium; however, at the same time, it can be easily linearized in the locality of the equilibrium. It is found that by providing sufficient training, the PPO algorithm can achieve a comparable transient and steady-

p-ISSN 2301–4156 | e-ISSN 2460–5719

state performance compared to conventional control methods despite not knowing anything about the pendulum model. The result confirms that reinforcement learning has a huge potential to produce a performance that satisfies a certain specification similar to conventional control algorithms.

This paper comprises several sections. Section II presents the mathematical model of the inverted pendulum, including the strategy to modify the control problem into the reinforcement learning formulation. Subsequently, the description of control algorithms is presented in Section III, followed by a description of the testing scenario in Section IV. Results and discussion are presented in Section V, and finally the conclusion and future works are presented in Section VI.

II. INVERTED PENDULUM MODEL

A. MODEL DERIVATION

The inverted pendulum on a cart system is shown in Figure 1. The input given to the system is the horizontal force $u = F$. The state of the system is written as following.

$$x_s = [x_c \quad \dot{x}_c \quad \theta \quad \dot{\theta}]^T. \tag{1}$$

Here, x_c refers to the cart's displacement, while θ refers to the pendulum's angle with respect to the vertical axis.

The output of the system depends on the available sensor. For some of the controllers analyzed in this paper, it is assumed that all the state information is known, in that case:

$$y = x_s. \tag{2}$$

Others only require the main parameter to be controlled, which, in this case, is the inverted pendulum angle θ.

$$y = \theta. \tag{3}$$

To model the inverted pendulum, its dynamics are analyzed using the Newton's 2nd law of motion. The system is divided into two components to simplify the analysis: the pendulum and the cart. Assuming an environment without friction, the cart's dynamics in the horizontal direction can be written as follows:

$$F - N = m_c \ddot{x}_c \tag{4}$$

where N refers to the horizontal contact force between the pendulum and the cart and m_c refers to the cart's mass.

The pendulum's dynamics in the horizontal direction can be written as follows:

$$N = m_p \ddot{x}_p. \tag{5}$$

The pendulum's acceleration, in this case, is a combination of the cart acceleration $\ddot{x}_c$ and the rotation of the pole. Suppose the pole has an angular velocity $\dot{\theta}$ and an angular acceleration $\ddot{\theta}$. The combined cart acceleration and rotating pendulum movement cause the following acceleration in the horizontal direction:

$$\ddot{x}_p = \ddot{x}_c + \ddot{\theta} l \cos\theta - \dot{\theta}^2 l \sin\theta. \tag{6}$$

Next, the following equation is obtained by analyzing the rotational dynamics of the pendulum:

$$\tau = I\ddot{\theta} + \tau_{inertial}. \tag{7}$$

Here, $\tau_{inertial}$ refers to the torque produced by the inertial force due to the cart's acceleration. Slotting in the pendulum's

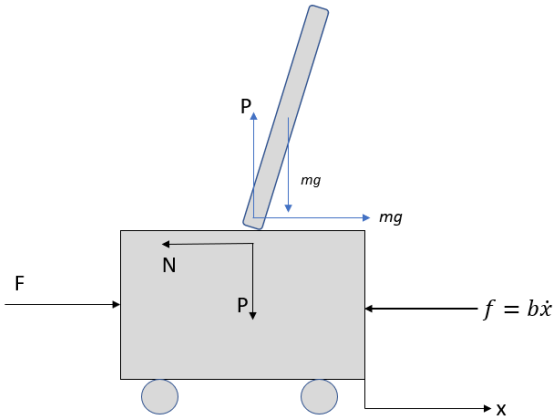

Figure 1. Free-body diagram of inverted pendulum on a cart, also known as cartpole.

moment of inertia $I = \frac{4}{3} m_p l^2$ and the torque due to the pendulum's weight $\tau = m_p g l \sin\theta$, the equation can be rewritten as

$$m_p g l \sin\theta = \frac{4}{3} m_p l^2 \ddot{\theta} + m_p \ddot{x}_c l \cos\theta. \tag{8}$$

Finally, combining three dynamics equations, the nonlinear model of the inverted pendulum is obtained as follows:

$$F = (m_c + m_p)\ddot{x}_c + m_p(\ddot{\theta} l \cos\theta - \dot{\theta}^2 l \sin\theta \tag{9}$$

$$m_p g l \sin\theta = \frac{4}{3} m_p l^2 \ddot{\theta} + m_p \ddot{x}_c l \cos\theta\,. \tag{10}$$

B. LINEARIZATION

To implement linear control, the model of the inverted pendulum needs to be linearized near its equilibrium point ($\theta = 0$). A simple linearization can be performed simply by the following modification:

$$\sin\theta \approx \theta, \tag{11}$$

$$\cos\theta \approx 1, \tag{12}$$

$$\dot{\theta}^2 \approx 0. \tag{13}$$

Therefore, the model can be written in the following form:

$$F = (m_c + m_p)\ddot{x}_c + m_p l \ddot{\theta}, \tag{14}$$

$$m_p g l \theta = \frac{4}{3} m_p l^2 \ddot{\theta} + m_p l \ddot{x}_c, \tag{15}$$

which can be written as:

$$F = (m_c + m_p)\ddot{x}_c + m_p l \frac{(m_p g l\theta - m_p l \ddot{x}_c)}{\frac{4}{3} m_p l^2} \tag{16}$$

$$F = \left(m_c + \frac{m_p}{4}\right)\ddot{x}_c + \frac{3}{4} m_p g \theta \tag{17}$$

and

$$m_p g l \theta = \frac{4}{3} m_p l^2 \ddot{\theta} + \frac{m_p l (F - m_p l \ddot{\theta})}{(m_c + m_p)} \tag{18}$$

$$g\theta = \left(\frac{4}{3} - \frac{m_p}{m_c + m_p}\right) l\ddot{\theta} + \frac{1}{(m_c + m_p)} F \tag{19}$$

$$g\theta = \left(\frac{4m_c + 3m_p}{3(m_c + m_p)}\right) l\ddot{\theta} + \frac{1}{(m_c + m_p)} F. \tag{20}$$

Then, the equation can be modified into a state equation as follows:

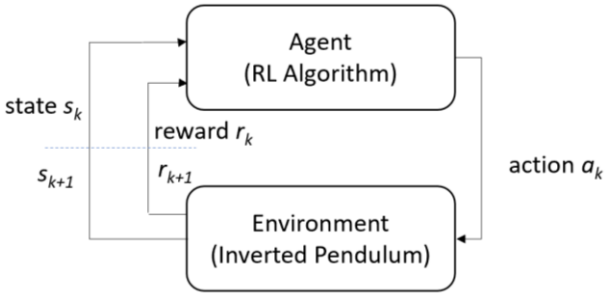

Figure 2. Inverted pendulum control as a reinforcement learning problem.

$$\dot{x}_s = Ax_s + Bu, \tag{21}$$

where,

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & a & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & b & 0 \end{bmatrix} \tag{22}$$

$$a = -\frac{3m_p g}{4m_c + m_p} \tag{23}$$

$$b = \frac{3(m_c + m_p)}{4m_c + 3m_p}\frac{g}{l} \tag{24}$$

$$B = \left[0 \quad \frac{1}{m_c + \frac{m_p}{4}} \quad 0 \quad -\frac{3}{l(4m_c + 3m_p)}\right]^T. \tag{25}$$

The linearized system can also be transformed into a complex-frequency domain:

$$F(s) = (m_c + m_p)s^2 X_c(s) + m_p l s^2 \theta(s), \tag{26}$$

$$m_p g l \theta(s) = \frac{4}{3} m_p l^2 s^2 \theta(s) + m_p l s^2 X_c(s). \tag{27}$$

So, the transfer function can be written as follows:

$$G(s) = \frac{\theta(s)}{F(s)} = \frac{c}{s^2 - d}, \tag{28}$$

where $c = \frac{d}{(m_p + m_c)g}, d = \frac{g}{l\left(\frac{4}{3} - \frac{m_p}{m_p + m_c}\right)}$.

From the transfer function, it can be seen that the pole of the linearized system is located at:

$$s_1 = \sqrt{d},\ s_2 = -\sqrt{d}. \tag{29}$$

The first pole is what makes the system unstable.

C. INVERTED PENDULUM AS A REINFORCEMENT LEARNING PROBLEM

The reinforcement learning problem deals with a system with unknown state update dynamics $s_{k+1} = F(s_k, a_k)$. Here, s_k and a_k refers to the state of the system and the action at iteration-k. Apart from that, the system also returns a reward value r_k. The reinforcement learning problem aims to find the set of actions a_k that maximize the expected return over M steps defined as:

$$R = \sum_{k=0}^{M} r_k. \tag{30}$$

The inverted pendulum control can be seen as a reinforcement learning problem as shown in Figure 2. Here, the state of the system s_k from the perspective of the reinforcement learning problem is equal to the state x_s from the perspective of modern control, albeit in the discrete form. Similarly, the action a_k is the control signal u. The only parameter left is the reward function which should be designed in such a way that it represents the performance of the inverted pendulum system in trying to stay in a vertical position. A reward function used by OpenAI in their CartPole Gym environment is employed here. The reward $r_k = 1$ if the cartpole's position satisfies $-x_{th} < x_c < x_{th}$ and the cartpole's angle satisfies $-\theta_{th} < \theta < \theta_{th}$, where x_{th} and θ_{th} represent the position and angle threshold. Then, the maximum number of steps M is chosen. In other words, a good reinforcement learning algorithm will produce action a_k over M steps which leads to a maximum return of $M\ r_k$.

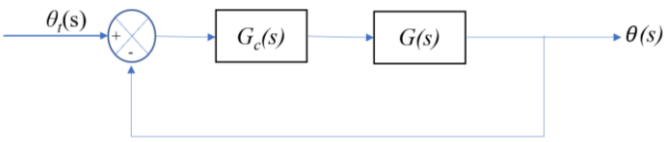

Figure 3. Block diagram of root locus-based compensator.

III. CONTROL ALGORITHMS

There are four control algorithms considered in this paper. The first three methods are chosen because they are widely used in the control literature especially for linear or linearized systems. A method based on the classical control approach (i.e., employing a transfer function) is chosen, namely the root locus method, and another method based on modern control approach (i.e., employing a state equation), namely the state compensator. The third control method is the PD control because of its popular usage in industry. Finally, the PPO algorithm is selected as one of the state-of-the-art reinforcement learning algorithms for systems with continuous action.

A. ROOT LOCUS-BASED COMPENSATOR

In this method, a compensator is added to modify the system's root locus so that the new root locus passes through the desired poles based on the requirement specifications. The compensator $G_c(s)$ is fed the error between the target output and the real output $e = \theta_t - \theta$ and produce control signal u to the system. The block diagram of the proposed controller can be seen in Figure 3.

Before designing the compensator, a root locus analysis of the pendulum's transfer function is required. The root locus of the linearized inverted pendulum system can be observed in Figure 4(a). From the root locus, it can be seen that a simple gain controller does not make the inverted pendulum stable because there is always at least one pole of the system which is located in the right half of the complex-frequency plane or exactly on the imaginary axis.

To stabilize the system, a zero of the compensators can be added into the place of pole s_2, i.e., $z_{comp} = s_2$. Then, to attract the root locus towards the left-half plane, a pole of the compensator can be added into a location $s_{comp} = 10s_2$. This method is more effective than trying to cancel the positive pole s_1 directly because in reality the positive pole is not exactly canceled, causing the system to stay unstable. The new root locus is shown in Figure 4(b).

Finally, a $K_{comp} = 300$ compensator gain is chosen to ensure that the new closed-loop poles of the system are both located in the left-half plane as shown in Figure 5. The final transfer function of the compensator is written as:

p-ISSN 2301–4156 | e-ISSN 2460–5719

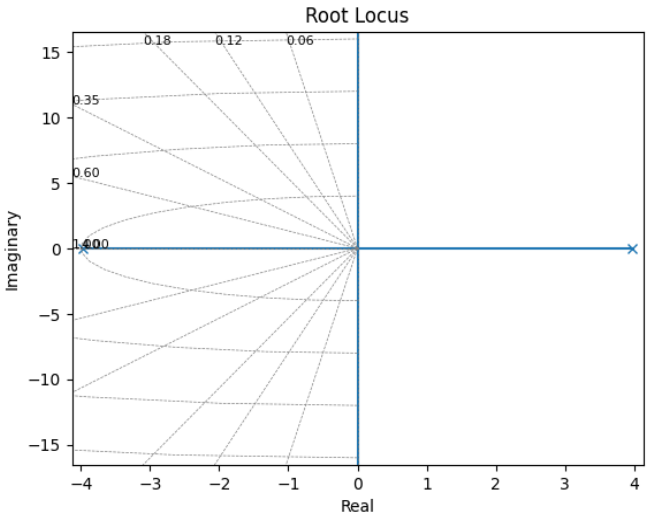

Figure 4. Root locus of (a) the inverted pendulum and (b) the modified system.

$$G_c(s) = 300\frac{s+\sqrt{d}}{s+10\sqrt{d}}. \quad (31)$$

B. STATE COMPENSATOR

In this method, a state feedback controller and state estimator based on a linearized model of an inverted pendulum is developed. The state feedback control signal is given by:

$$u = -K\,\hat{x}_s \quad (32)$$

where K stands for the state feedback constant and $\hat{x}$ stands for the estimate of state x. The feedback gain is chosen in such a way that the following equation holds:

$$\det(sI - (A - BK)) = f(s) = 0. \quad (33)$$

Here, $f(s)$ stands for the 4th order polynomial which represents the target characteristic equation of the closed-loop systems from which the poles $s_{c1}, s_{c2}, s_{c3}, s_{c4}$ can be found as follows:

$$f(s) = (s - s_{c1})(s - s_{c2})(s - s_{c3})(s - s_{c4}) = 0.$$

Here, the following values as the target poles are chosen:

$$s_{c1} = -1, s_{c2} = -2, s_{c3} = -3, s_{c4} = -4.$$

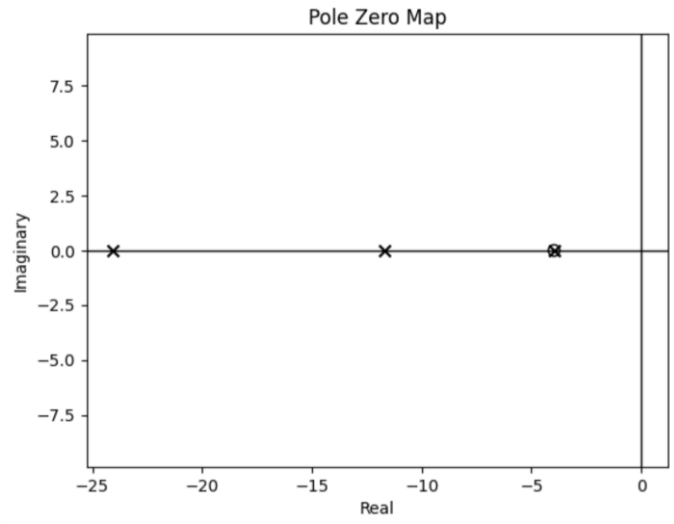

Figure 5. The pole-zero maps of the closed-loop system using root locus-based controller.

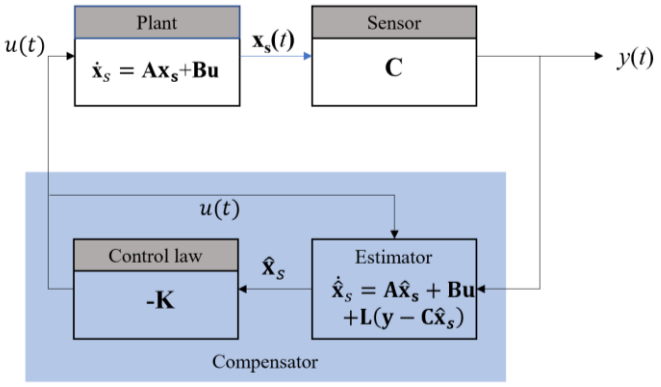

Figure 6. Block diagram of the state compensator.

The value of the state estimate can be updated using the following estimator equation:

$$\dot{\hat{x}}_s = A\hat{x}_s + Bu + L(y - C\hat{x}_s). \quad (34)$$

L stands for the estimator gain which is chosen in such a way that the following equation holds:

$$\det(sI - (A - LC)) = g(s) = 0. \quad (35)$$

Here, $g(s)$ stands for the 4th order polynomial which represents the target characteristic equation of the error estimator dynamics from which the poles $s_{e1}, s_{e2}, s_{e3}, s_{e4}$ can be found as follows:

$$g(s) = (s - s_{e1})(s - s_{e2})(s - s_{e3})(s - s_{e4}) = 0. \quad (36)$$

Here, the following values as the target poles are chosen, which are ten times greater than the poles of the controller to ensure that the state estimation error dynamics converge faster:

$$s_{e1} = -10, s_{e2} = -20, s_{e3} = -30, s_{e4} = -40.$$

The block diagram of the state compensator is provided in Figure 6.

C. PD CONTROL

In this method, the following PD controller is used to produce the control signal:

$$u = K_p\left(e + T_d\frac{de}{dt}\right). \quad (37)$$

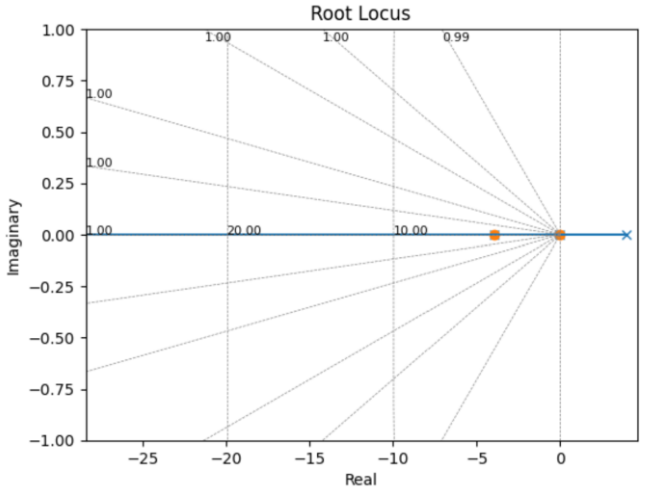

Figure 7. Root locus of the modified system after adding a PD control.

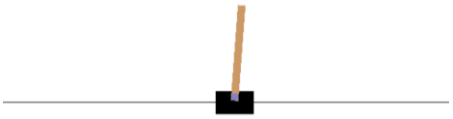

Figure 8. CartPole screenshot.

TABLE I
PARAMETERS' VALUES

Parameter	Value	Parameter	Value
m_c	1 kg	x_{th}	2.4 m
m_p	0.1 kg	θ_{th}	$3\pi/45$ rad
l	0.5 m	M	200 steps

The following equation is obtained when transforming the control signal to the complex-frequency space:

$$U(s) = K_p(1 + T_d s)E(s). \quad (38)$$

So, the zero of the PD control is located at: $s_{PD} = -\frac{1}{T_d}$.

The zero of the PD control is chosen to be located at the negative pole of the inverted pendulum: $s_{PD} = s_2 = -\sqrt{d}$. Therefore, the new root locus of the system moves toward the left-half plane of the complex-frequency space instead of moving toward the imaginary axis, as illustrated in Figure 7. Finally, the gain of the controller can be chosen to ensure that the pole of the closed-loop system is located in the left-half plane. To this end, the gain of $K_p = 30$ is employed.

D. REINFORCEMENT LEARNING

The reinforcement learning algorithm employed here is the PPO [15]. This algorithm falls under the category of policy-based reinforcement learning focusing on finding the best policy $\pi(a|s)$, which represents the likelihood of a particular action a for a particular state s. It employs an actor-critic approach, where the actor estimates the policy while the critic measures how well the action is being performed.

The algorithm starts with two networks: one is for the policy with parameter θ and the second one is for the value function V with parameter ϕ. The algorithm starts from iteration $k = 0$ by collecting a set of trajectories $D_k = \{\tau\}$ from $t = 0$ to $t = T$, where each trajectory consists of $\tau = (s_0, a_0, s_1, a_1, \ldots, s_T)$. The algorithm runs using the current policy $\pi_k = \pi(\theta_k)$. From the trajectory, the reward-to-go is calculated: $\hat{R}_t = \sum_{t'=t}^{T} r_{t'}$. The next step is to calculate the

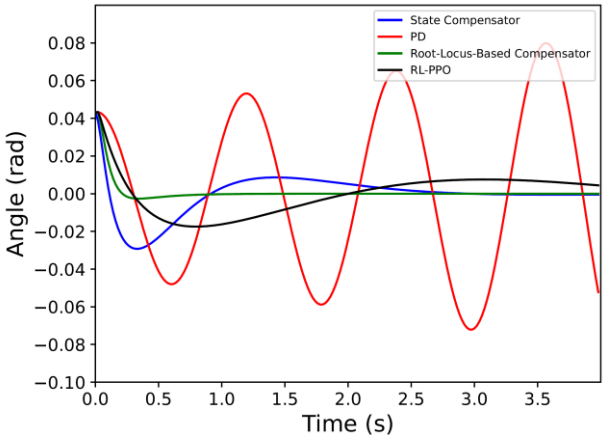

Figure 9. Plot of angle versus time produced by four algorithms.

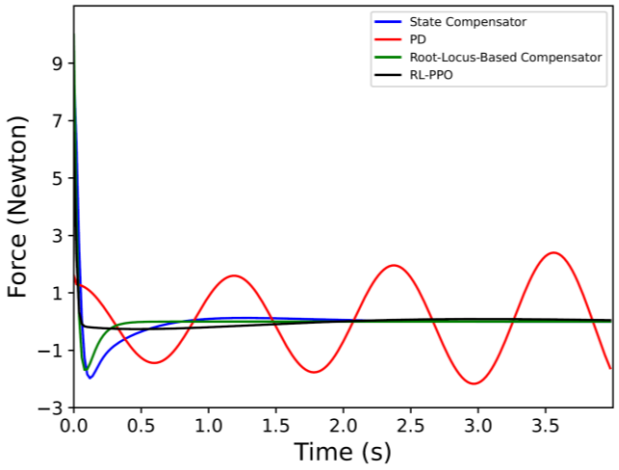

Figure 10. Plot of force versus time produced by four algorithms.

advantage estimate $\hat{A}_t$ defined as how much better it is to choose an action based on current policy π_k compared to randomly sampling an action. The advantage function is estimated based on the current value function $V_k = V(\phi_k)$.

Then, the policy network parameter is updated using stochastic gradient ascent by maximizing a PPO objective function as follows:

$$\theta_{k+1} = \arg\max_{\theta} \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^{T} \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_k}(a_t|s_t)} \hat{A}_t, g(\epsilon, \hat{A}_t) \right). \quad (39)$$

Here, the function $g(\epsilon, A)$ is defined as:

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A; & A \geq 0 \\ (1-\epsilon)A; & A < 0 \end{cases} \quad (40)$$

ϵ stands for a hyperparameter.

The next step is to update the value function network parameter by minimizing the mean squared error objective function using gradient descent. The mean squared error is given by

$$\phi_{k+1} = \arg\min_{\phi} \frac{1}{|D_k|T} \sum_{\tau \in D_k} \sum_{t=0}^{T} (V_k(s_t) - \hat{R}_t)^2. \quad (41)$$

IV. TESTING SCENARIOS

As shown in Figure 8, a modified Gym Environment simulation, called CartPole, developed by OpenAI was employed to test the performance of the control algorithms in a

p-ISSN 2301–4156 | e-ISSN 2460–5719

TABLE II
PERFORMANCE COMPARISON

Parameter	State Compensator	PD	Root Locus Compensator	PPO
Overshoot (%)	70.74	122.15	6.11	61.17
Peak time (s)	3.20×10^{-1}	2.98	3.40×10^{-1}	8.00×10^{-1}
Settling time (s)	-	-	7.00×10^{-1}	-
Steady-state error (rad)	8.21×10^{-4}	1.14×10^{-2}	-5.48×10^{-7}	5.60×10^{-3}
Total energy (N^2)	1.53×10^{2}	3.49×10^{2}	1.27×10^{2}	5.24×10^{1}

model of an inverted pendulum on a cart [16]. This Python-based environment is widely used in the reinforcement learning community because of its simplicity and the fact that it directly provides a reward function for training. The simulation used the nonlinear model of an inverted pendulum as described in Section IIA with the parameters shown in Table I. The original Python library was modified to support continuous control or action. The original state of the pendulum was randomly chosen to be $x_{c0} = 0.025$, $\dot{x}_{c0} = 0.009$, $\theta_0 = 0.043$, $\dot{\theta}_0 = 0.009$. Among the four algorithms, the PPO algorithm was the only one which relied on training. It was trained for 250,000 epochs in this experiment because at this number, the average reward seems to converge to a specific high value. Therefore, it can be assumed that the reinforcement learning algorithm has sufficient epochs to learn the most optimal policy.

The performance of the control algorithms in terms of the transient responses and the steady-state responses were compared. The transient responses under consideration include [17].

- Overshoot, which is defined as:

$$\%OV = \frac{\theta_{max} - \theta_{ss}}{\theta_{ss}} \times 100\%.$$

  Here, θ_{max} and θ_{ss} specify the peak value of the output and the steady-state value of the output.
- Peak time is defined as the amount of time required to reach the first peak of the overshoot.
- Settling time is defined as the amount of time required to reach and stay within a range of 2% about the steady-state value.

The steady-state responses, on the other hand, include the followings.

- Steady-state error, which is defined as:

$$e_{ss} = \theta_t - \theta_{ss}.$$

- Total energy over a particular time step *M*, defined as:

$$W = \sum_{k=0}^{M} u_k^2.$$

V. RESULT AND DISCUSSION

A. CONTROL PERFORMANCE

The plot of angle and force as a function of time can be observed in Figure 9 and Figure 10. The blue, red, green, and black lines represent the performance of the state compensator, PD controller, root-locus-based compensator, and reinforcement learning based on the PPO algorithm. The values of the transient response parameters (overshoot, peak time, and settling time) as well as the steady-state response parameters (steady-state error and total energy) are all summarized in Table II.

Figure 9 shows that the state compensator, root locus-based compensator, and PPO could control the inverted pendulum towards a vertical position (i.e., $\theta_t = 0$) in a stable manner. The PD control, however, seems to produce an oscillation that fails to converge to zero. It can be observed in Table II that all the algorithms except the root locus-based compensator failed to settle the output value within a range of 2% about the steady-state value before $M = 200$ steps (which is equal to $T = 4$ s). It is indicated by a dash mark on the settling time part for the three algorithms except for the root locus-based compensator.

From Figure 9 and Table II, among the three successful methods, the root locus-based compensator was able to quickly guide the pendulum's angle towards zero without considerable overshoot and it performed with quick settling time. Despite having a quick peak time, the state compensator performed worse in terms of overshoot and settling time compared to the root locus-based compensator. The PPO performed better than the state compensator in terms of overshoot, but they performed worse in terms of settling time and peak time compared to the two other methods. The reason for this performance is the fact that the reward function being used to guide the learning process focuses only on the aspect of the pendulum not falling. Recall that the reward of +1 is provided every time the pendulum's angle falls within a certain range. It means that the performance being reinforced does not cover the aspect of peak time (how quickly the angle moves towards the target setpoint), settling time (how quickly it converges), or overshoot.

From the perspective of steady-state response, especially in terms of the steady-state error, the root locus-based compensator once again performed better than two other successful methods. It is indicated by a very small value of error, as shown in Table II. PPO once again comes second followed by the state compensator. However, a different performance was observed in terms of total energy. Figure 10 demonstrates that the force produced by the PPO quickly settled down without much overshoot as compared to the force produced by the state compensator and the root locus-based compensator. This advantage of PPO is caused by the fact that it focuses only on not falling. It more relaxed target enables the algorithm to produce less energy, as shown in Table II, compared to the other methods.

B. DISCUSSION

From the previous results, it is now possible to compare how the reinforcement learning method, in this case the PPO algorithm, performs in comparison to the conventional control methods such as state compensator, PD, and root locus-based compensator in a broader sense. In terms of the background knowledge required to produce a control policy, it is clear that the reinforcement learning method does not require any knowledge regarding the pendulum's model. It is not the case for the other methods, especially the state compensator and the root locus-based compensator. Both rely heavily on the

p-ISSN 2301–4156 | e-ISSN 2460–5719

mechanical property and the dynamics model of the inverted pendulum. The PD control, in some respect, is theoretically able to perform well despite not knowing the model of the system. As has been extensively covered in literature [18], it is possible via the process of parameter tuning, such as the well-known Ziegler-Nichols method. However, there is no guarantee that optimal (or even reasonable) performance can be achieved by this kind of approach. The fact that it always relies on three parameters, for the case of proportional–integral–derivative (PID), also limits the possibility of the control policy. The reinforcement learning method, on the other hand, relies on a policy function represented by a neural network in this case. As has been covered in the literature, the neural network is a reliable function approximation that works for a wide range of functions [19].

However, despite not requiring any prior model, reinforcement learning relies on trial and error, i.e., it requires applying the current policy $\pi(s)$ to the pendulum and learn from the reward function. First of all, it can be a cumbersome process, requiring a lot of time and computation before the algorithm is able to produce the best policy $\pi^*(s)$ to produce a reasonable performance. Secondly, performing a trial and error on a plant might not be practical in some cases. Because the policy in action tends to be nonoptimal at first, the policy can produce an unpredictable or even dangerous movement.

To avoid this problem, a line of research has focused on performing the training in a simulation scenario prior to implementation [20]. It can be possible in some cases, including the inverted pendulum in this study's case. However, a good simulator in general, relies on a physics-based model which also requires a lot of computation.

Despite this challenge, one could also argue that the conventional control methods similarly have the same problem. It is due to the fact that methods such as a state compensator or a root locus-based compensator require a model of the system which needs to be retrieved from some sorts of system identification experiments. This type of experiment similarly requires some sort of "trial and error".

Another important point is the fact that the performance of the reinforcement learning algorithm depends on the reward function. The challenge in robotics, where the reward function is not readily available in general, is to define the reward function which best reflects the desired performance. In theory, there are infinite possibilities here. For our inverted pendulum case, the target of making the pendulum stays in the vertical position can be achieved by a variety of reward function. This can be a simple reward employed here, providing a flat positive reward every time the angle falls under some values. However, this reward type does not really care about transient performance or steady-state performance like other conventional control methods. To truly achieve performance based on specific considerations, it is necessary to employ a more complex reward function using a process known as reward shaping [12], [21].

Therefore, from the previous important points, several important considerations that can be drawn on when to employ reinforcement learning in general control applications. When the plant that is being controlled only provides little to no information regarding its model but testing a control policy is quite easy, reinforcement learning can be a good candidate to work on. The fact that it does not rely on any prior model and its use of a neural network as a control policy approximation is powerful compared to conventional control methods, even the PID control which has been popular in the industry, due to its capability to perform without an explicit model of the plant. If testing a control policy is easy and safe, it is also favorable to employ reinforcement learning because it requires extensive trial and error on the controlled plant. It is even more favorable to use reinforcement learning if some sorts of objective functions are available so that they can be modified into a reward function to guide the learning process.

VI. CONCLUSION

In this paper, a comparative study is performed between a reinforcement learning algorithm (namely the PPO) and several conventional control algorithms (namely the root-locus-based compensator, the state compensator, and the PD control) for inverted pendulum control. It was found that given a sufficient amount of training, the PPO algorithm was able to achieve a comparable transient and steady-state performance compared to conventional control methods despite not knowing anything about the model of the pendulum. The performance of the algorithm could be further improved by performing reward shaping techniques to include the performance specifications in the reward function. From the result analysis, it can be concluded that reinforcement learning is best suited for control problems especially if no prior model of the plant is available and performing trial and error to the model is feasible.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTION

Conceptualization, Ahmad Ataka; methodology, Ahmad Ataka, Dzuhri Radityo Utomo, and Adha Imam Cahyadi; software, Ahmad Ataka; validation, Ahmad Ataka and Andreas Sandiwan; formal analysis, Ahmad Ataka and Dzuhri Radityo Utomo; investigation, Ahmad Ataka; resources, Ahmad Ataka; data curation, Ahmad Ataka; writing—original draft preparation, Ahmad Ataka and Andreas Sandiwan; writing—review and editing, Ahmad Ataka, Andreas Sandiwan, Hilton Tnunay, Dzuhri Radityo Utomo, and Adha Imam Cahyadi; visualization, Ahmad Ataka and Andreas Sandiwan; supervision, Hilton Tnunay, Dzuhri Radityo Utomo, and Adha Imam Cahyadi; project administration, Ahmad Ataka and Andreas Sandiwan; funding acquisition, Ahmad Ataka, Dzuhri Radityo Utomo, and Adha Imam Cahyadi.

ACKNOWLEDGMENT

The authors would like to thank all undergraduate students who took the course Control Engineering, Modern Control Engineering, and Mobile Robotics during the Odd Semester 2022 for their brilliant works on controlling the inverted pendulum, which motivated the writing of this paper.

REFERENCES

[1] R.S. Sutton and A.G. Barto, *Reinforcement Learning: An Introduction*. London, England: The MIT Press, 2020.

[2] A. Krizhevsky, I. Sutskever, and G.E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Commun. ACM*, Vol. 60, No. 6, pp. 84–90, Jun. 2017, doi: 10.1145/3065386.

[3] V. Mnih *et al.*, "Playing Atari with Deep Reinforcement Learning," 2013, *arXiv:1312.5602*.

[4] V. Mnih *et al.*, "Human-Level Control Through Deep Reinforcement Learning," *Nat.*, Vol. 518, pp. 529–533, Feb. 2015, doi: 10.1038/nature14236.

p-ISSN 2301–4156 | e-ISSN 2460–5719

[5] D. Silver *et al.*, "A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go Through Self-Play," *Sci.*, Vol. 362, No. 6419, pp. 1140–1144, Dec. 2018, doi: 10.1126/science.aar6404.

[6] Ardiansyah and E. Rainarli, "Implementasi Q-Learning dan Backpropagation pada Agen yang Memainkan Permainan Flappy Bird," *J. Nas. Tek. Elekt., Teknol. Inf.*, Vol. 6, No. 1, pp. 1–7, Feb. 2017, doi: 10.22146/jnteti.v6i1.287.

[7] C. Berner *et al.*, "Dota 2 with Large Scale Deep Reinforcement Learning," 2019, *arXiv:1912.06680*.

[8] B.R. Kiran *et al.*, "Deep Reinforcement Learning for Autonomous Driving: A Survey," *IEEE Trans. Intell. Transp. Syst.*, Vol. 23, No. 6, pp. 4909–4926, Jun. 2022, doi: 10.1109/TITS.2021.3054625.

[9] X. Ruan, D. Ren, X. Zhu, and J. Huang, "Mobile Robot Navigation Based on Deep Reinforcement Learning," *2019 Chin. Control, Decis. Conf. (CCDC)*, 2019, pp. 6174–6178, doi: 10.1109/CCDC.2019.8832393.

[10] V. Tsounis *et al.*, "DeepGait: Planning and Control of Quadrupedal Gaits Using Deep Reinforcement Learning," *IEEE Robot., Automat. Lett.*, Vol. 5, No. 2, pp. 3699–3706, Apr. 2020, doi: 10.1109/LRA.2020.2979660.

[11] R. Liu *et al.*, "Deep Reinforcement Learning for the Control of Robotic Manipulation: A Focussed Mini-Review," *Robot.*, Vol. 10, No. 1, pp. 1–13, Jan. 2021, doi: doi.org/10.3390/robotics10010022.

[12] J. Ibarz *et al.*, "How to Train Your Robot with Deep Reinforcement Learning: Lessons We Have Learned," *Int. J. Robot. Res.*, Vol. 40, No. 4–5, pp. 698–721, Jan. 2021, doi:10.1177/0278364920987859.

[13] A. Sharma, R. Ahmad, and C. Finn, "A State-Distribution Matching Approach to Non-Episodic Reinforcement Learning," 2022, *arXiv:2205.05212*.

[14] A.G. Barto, R.S. Sutton, and C.W. Anderson, "Neuronlike Adaptive Elements that Can Solve Difficult Learning Control Problems," *IEEE Trans. Syst. Man, Cybern.*, Vol. SMC-13, No. 5, pp. 834–846, Sept.-Oct. 1983, doi: 10.1109/TSMC.1983.6313077.

[15] J. Schulman *et al.*, "Proximal Policy Optimization Algorithms," 2017, *arXiv:1707.06347*.

[16] G. Brockman *et al.*, "OpenAI Gym," 2016, *arXiv:1606.01540*.

[17] R.P. Borase, D.K. Maghade, S.Y. Sondkar, and S.N. Pawar, "A Review of PID Control, Tuning Methods and Applications," *Int. J. Dyn., Control*, Vol. 9, No. 2, pp. 818–827, Jun. 2021, doi: 10.1007/s40435-020-00665-4.

[18] N.S. Nise, *Control Systems Engineering*, 4th ed. Hoboken, USA: John Wiley & Sons, Inc., 2004.

[19] S. Liang and R. Srikant, "Why Deep Neural Networks for Function Approximation?" 2017, *arXiv:1610.04161*.

[20] W. Zhao, J.P. Queralta, and T. Westerlund, "Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey," *2020 IEEE Symp. Ser. Comput. Intell. (SSCI)*, 2020, pp. 737–744.

[21] J. Eschmann, "Reward Function Design in Reinforcement Learning," in *Reinforcement Learning Algorithms: Analysis and Applications*, B. Belousov *et al.*, Eds., Cham, Switzerland: Springer Cham, 2021, pp. 25–33.