Comparative Analysis of Reinforcement Learning and Linear Control Methods for the Inverted Pendulum Problem

Anne M. Tumlin
Department of Computer Science
Vanderbilt University
Nashville, Tennessee
anne.m.tumlin@vanderbilt.edu

***Abstract*—This project compares Reinforcement Learning (RL) and traditional linear control methods on the inverted pendulum problem. RL, a dynamic branch of machine learning, adapts and learns in dynamic and uncertain environments. However, its adaptability introduces concerns when applied to safety-critical systems, where reliability and predictability are paramount. To analyze these concerns, this project reviews various Q-learning methods within the RL framework, evaluating their effectiveness in maintaining the pendulum's balance to ensure safety and reliability.**

Conversely, traditional linear control systems are esteemed for their predictability and robustness. This project specifically focuses on the Linear Quadratic Regulator (LQR) approach. These systems are rooted in established mathematical principles and provide essential certainty and stability in critical applications. This project compares and contrasts the performance of these approaches in stabilizing the pendulum, thus shedding light on their respective strengths and weaknesses. A significant insight from this comparative analysis is the relative complexity in stabilizing RL methods as opposed to linear control systems, which tend to achieve stability more quickly. This finding illustrates the challenges and trade-offs in selecting the appropriate control method for different scenarios.

***Index Terms*—inverted pendulum, reinforcement learning, linear quadratic regulator**

I. Introduction

Recently, there has been a surge in the use of machine learning (ML) in many fields, leading to extensive research on its integration into cyber-physical systems (CPS) [18]. Reinforcement learning (RL) has emerged as a significant method for consideration due to its capacity to learn and adapt from data, thereby offering solutions that conventional algorithmic methods cannot provide. As a result, this has incentivized integrating RL methods into intricate real-world problems across various domains, including robotics, autonomous systems, and industrial automation [13, 14].

Reinforcement Learning (RL), a prominent branch of machine learning, operates on the paradigm of trial-and-error learning, characterized by its interaction with dynamic environments to achieve objectives [23]. This approach is particularly advantageous for addressing problems where the formulation of explicit rules is challenging or impractical. RL distinguishes itself by leveraging a reward-based system, where algorithms iteratively improve their decision-making strategies based on feedback from the environment, effectively 'learning' optimal behaviors through accumulated experience. Moreover, RL algorithms are adept at handling stochastic and non-linear systems, showcasing adaptability and resilience in highly uncertain and complex environments [19].

Linear control methods represent a more traditional and deterministic approach to control theory, in contrast to the adaptive and exploratory Reinforcement Learning (RL) methods. LQR (Linear-Quadratic Regulator) is a commonly utilized control paradigm [12]. This linear control methodology is highly regarded for its robustness and simplicity. LQR optimizes control actions based on a defined cost function that balances state deviations and control effort, ensuring efficiency in system response.

LQR works exceptionally well within linear system dynamics, where system behaviors can be accurately modeled and predicted. This effectiveness under linearity assumptions has made it a go-to solution in industrial applications. Its widespread adoption in various sectors can be attributed to its predictability, which ensures consistent performance and reliability in real-world applications.

While Reinforcement Learning (RL) is particularly effective in managing complex, non-linear environments, its application in safety-critical systems within cyber-physical systems (CPS) is cautiously approached. Despite the advancements and potential of RL, there remains hesitancy in its application to these systems [20]. This caution stems from challenges related to RL-based solutions' reliability, predictability, and robustness. Safety-critical systems demand a high level of assurance against failures, and the often black-box nature of RL algorithms makes it difficult to guarantee their behavior under all circumstances.

One such application of ML, particularly RL, is the inverted pendulum problem - a classic problem in control

The author would like to express gratitude to Taylor Johnson for their guidance in working with the real inverted pendulum system. Additional thanks should be given to Preston Robinette and Diego Lopez for their helpful commentary and feedback throughout the work.

theory and robotics [22]. This problem involves balancing a pendulum in the upright position by applying forces to its pivot point, exemplifying a fundamental challenge in dynamic system stabilization. The inverted pendulum is a benchmark for evaluating control strategies due to its well-understood dynamics and relevance to various engineering applications, from robotics to human motion analysis.

A. Contributions

This project aims to compare reinforcement learning and linear control methods for solving the inverted pendulum problem. Through a systematic analysis, the study aims to assess the effectiveness, reliability, and suitability of these methods in the inverted pendulum environment. The ultimate goal is to provide valuable insights that could help develop future safety-critical cyber-physical systems.

B. Organization

The paper is structured as follows: In Section II, you will find some background information about the inverted pendulum problem, including an overview of the states involved. Section III details the principles of reinforcement learning and linear control methods and the methodology used in the study. Section IV presents the results and discussion, and finally, in Section V, the paper concludes with final remarks and suggestions for future research.

II. Preliminaries

A mathematical model of the inverted pendulum system is first established. This requires identifying the system's dynamics via differential equations and constructing a state space model. To begin, the system's mechanics are analyzed by utilizing a free-body diagram.

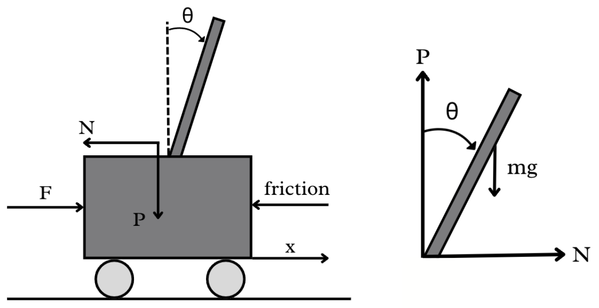

Fig. 1. The inverted pendulum system and relevant forces.

The system illustrated in Figure 1 comprises a cart with a pendulum attached. The primary control objective is stabilizing the pendulum at a vertical angle θ of 0 degrees. The nonlinear form describes this system

$$\dot{x} = f(x, u) \tag{1}$$

However, the equations of motion can be further derived for the plant. For the full tutorial on this process, see the referenced Matlab tutorial [3]. One of the governing equations of the pendulum can be derived by summing the forces in its free-body diagram in the horizontal direction.

$$(M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}cos\theta - ml\dot{\theta}^2 sin\theta = F \tag{2}$$

To obtain the second equation of motion for this system, sum the forces perpendicular to the pendulum to derive the second governing equation:

$$(I+ml^2)\ddot{\theta} + mglsin\theta = -ml\ddot{x}cos\theta \tag{3}$$

This set of equations can be simplified by linearizing them. In particular, the equations will be linearized around the point of vertical equilibrium, where θ equals π. The system is assumed to remain within a small vicinity of this equilibrium. Let ϕ denote the deviation of the pendulum's position from equilibrium, that is, $\theta = \pi + \phi$. The following approximations of the nonlinear functions in the system equations can be used, which are valid for small deviations (ϕ) from equilibrium:

$$\cos\theta = \cos(\pi + \phi) \approx -1 \tag{4}$$

$$\sin\theta = \sin(\pi + \phi) \approx -\phi \tag{5}$$

$$\dot{\theta}^2 = \dot{\phi}^2 \approx 0 \tag{6}$$

Two linearized equations of motion are derived by substituting the above approximations into the nonlinear governing equations. Note that the input, F, has been replaced by u.

$$(I+ml^2)\ddot{\phi} - mgl\phi = ml\ddot{x} \tag{7}$$

$$(M+m)\ddot{x} + b\dot{x} - ml\ddot{\phi} = u \tag{8}$$

The system has four state variables: the cart position x, cart velocity $\dot{x}$, pendulum angle θ, and pendulum velocity $\dot{\theta}$. The implemented system operates within certain physical limits. The cart position x ranges from -0.7 to 0.7 meters, with the set point x_s varying from -0.5 to 0.5 meters. The cart velocity is constrained between -1.0 and 1.0 meters per second, while the pendulum angle θ is restricted to between -30 and 30 degrees, and the pendulum velocity is not constrained. In the RL methods, further constraints are placed by the OpenAI Gym Cart Pole system, which is discussed later. The equations of motion discussed above can also be represented in state-space form if they are rearranged into a series of first-order differential equations. These equations are linear and can be put into the standard matrix form. The linearized form of the model is:

$$\dot{x} = Ax + Bu \tag{9}$$

given,

$$x = \begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} \tag{10}$$

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -a_{22} & -a_{23} & a_{24} \\ 0 & 0 & 0 & 1 \\ 0 & a_{42} & a_{43} & -a_{44} \end{bmatrix} \tag{11}$$

$$B = \begin{bmatrix} 0 \\ b_2 \\ 0 \\ -b_4 \end{bmatrix} \tag{12}$$

where $a_{22} = \frac{4\bar{B}}{D_l}$, $a_{23} = \frac{3mg}{D_l}$, $a_{24} = \frac{6B_\theta}{lD_l}$, $a_{42} = \frac{6\bar{B}}{lD_l}$, $a_{43} = \frac{6\bar{M}g}{lD_l}$, $a_{44} = \frac{12\bar{M}B_\theta}{ml^2D_l}$, $b_2 = \frac{4B_l}{D_l}$, and $b_4 = \frac{6B_1}{lD_l}$, for $D_l = 4\bar{M} - 3m$, $\bar{B} = \frac{K_gB_m}{r^2} + \frac{K_g^2K_iK_b}{r^2R_a}$, $B_l = \frac{K_gK_i}{rR_a}$, $\bar{M} = \frac{m+M+(K_gJ_m)}{r^2}$, and where g is gravity, R_a is the armature resistance, r is the driving wheel radius, J_m is the motor rotor inertia, B_m is the motor's coefficient of viscous friction, B_θ is the pendulum joint's coefficient of viscous friction, K_i is the motor torque constant, K_b is the motor back-e.m.f. constant, K_g is the gear ratio, M is the cart mass, m is the pendulum mass, and l is the pendulum length [6, 11].

After linearizing the motion equations of the inverted pendulum around its upright position, the next critical step is to ensure the system's asymptotic stability. This is achieved by determining a suitable control strategy encapsulated in the equation $u = Kx$, where u represents the control input and K denotes the feedback gain matrix.

$$\dot{x} = Ax + BKx \tag{13}$$

The selection of K is pivotal in stabilizing the linearized system, requiring careful analysis to determine the gains that ensure all eigenvalues of the system's closed-loop dynamics are in the left half of the complex plane, indicative of stability.

III. Control Methods

The previous section described the formal model of the inverted pendulum system. This section aims to introduce different types of control and reinforcement learning methods that are used to evaluate the system. Additionally, this section provides a summary of the overall framework.

A. Reinforcement Learning

Reinforcement Learning (RL) is a branch of machine learning where an agent learns to make decisions by interacting with an environment. In the context of the inverted pendulum problem, RL can be applied to balance the pendulum in its upright position dynamically. This project focuses on leveraging variations of the Q-Learning algorithm [25].

The Q-Learning algorithm iteratively updates the Q-values based on the equation:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R + \gamma \max_{a'} Q(s',a') - Q(s,a)\right] \tag{14}$$

where s is the current state, a is the chosen action, s' is the new state, R is the reward, α is the learning rate, and γ is the discount factor. The policy, ϵ-greedy, balances exploration and exploitation [8]. A more detailed overview of the Q-Learning algorithm can be seen in Algorithm 1.

Algorithm 1 Q-Learning Algorithm

```
Require: Learning rate $\alpha$, Discount factor $\gamma$
  Initialization: $Q(s,a) \leftarrow 0$
  for each episode do
    $s \leftarrow$ random state
    while state $s$ is not terminal do
      Choose action $a$ from $s$ using policy derived from Q
      Take action $a$, observe reward $r$, and next state $s'$
      $Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
      $s \leftarrow s'$
    end while
  end for
```

1) Q-Learning Implementation Strategies: This project investigated different methods for implementing Q-Learning, including using a traditional Q-Table, integrating a Deep Neural Network (DNN), and enhancing the DNN with advanced features such as experience replay and Dual DQN techniques. Python was used to develop and test these reinforcement learning methods. The following sections will further discuss the variations of the Q-learning techniques.

2) Q-Table: Initially, rewards are predefined in the Q-table, and the agent iteratively selects actions based on policies, moving to new states and updating Q-values until they converge to a stable value [10]. Q-learning, a fusion of dynamic programming and Monte Carlo methods, effectively solves the Bellman equation and forms the foundation of many reinforcement learning algorithms, particularly excelling in single-agent environments due to its simplicity and strong learning capabilities [9]. However, its application is limited in complex, large state-action spaces and multi-agent environments, primarily because it updates values only once per action and requires substantial memory for the Q-table [21].

3) Deep Q-Learning: DeepMind's deep Q-learning merges basic Q-learning with Convolutional Neural Networks (CNNs) for approximating value functions in complex states [16]. This approach can include an additional technique - experience replay. Experience replay, by storing and randomly sampling experiences, mitigates the instability often seen in neural network-based value approximations, breaking correlations among states, actions, and rewards to facilitate more stable learning [17].

4) Double Q-Learning: Hasselt developed Double Q-learning [7] to enhance Q-learning's performance in stochastic environments where conventional Q-learning often exhibits biased outcomes due to the overestimation of action values [24]. In such environments, standard Q-learning repeatedly selects the highest existing values without seeking new optimal ones. The innovative approach of Double Q-learning addresses this bias by splitting the valuation function into two distinct parts. This division aims to prevent the value deviation commonly seen in traditional Q-learning algorithms. The core algorithm remains similar to Q-learning, but it introduces a novel mechanism of using two Q-functions [10].

$$Q^A(s,a) \leftarrow Q^A(s,a) + \alpha(s,a)[R + \gamma Q^B(s',a') - Q^A(s,a)] \tag{15}$$

$$Q^B(s,a) \leftarrow Q^B(s,a) + \alpha(s,a)[R + \gamma Q^A(s',a') - Q^B(s,a)] \tag{16}$$

The two functions are updated separately and adjusted using each other's values. They learn from different experiences but are used together to make decisions. This diversifies the learning process and improves the accuracy of value estimation, resulting in more reliable outcomes in uncertain situations.

5) Training: This project uses OpenAI Gym to simulate the Cart-Pole (Inverted Pendulum) system and is coded in Python [1]. Specific standard configurations characterize this environment. The action space, represented as an array with the shape (1,), allows for binary values 0, 1, which denote the direction of force applied to the cart—0 for a push to the left and 1 for a push to the right. The observation space is a 4-dimensional ndarray, encapsulating key physical attributes: the cart's position x, cart velocity $\dot{x}$, pendulum angle θ, and pendulum velocity $\dot{\theta}$. The simulation concludes under any of the following conditions: the pendulum's angle θ deviates beyond $\pm 12°$, the cart's position x exceeds ± 2.4 units or the episode surpasses a length of 500 timesteps. The agent receives a reward of 1 for every step taken, including the termination step. However, the reward function was adjusted to include a negative reward for the failure condition. The problem is considered solved if the average reward is greater than or equal to 195 over 100 consecutive episodes.

B. Linear Control Methods

This section introduces the Linear Quadratic Regulator (LQR), the control method of choice [2]. LQR optimizes system performance by minimizing a cost function that balances state error and control effort. Take our linearized model, which is a linear time-invariant system in state-space form,

$$\dot{x} = Ax + Bu \tag{17}$$

with the infinite-horizon cost function given by,

$$J = \int_0^\infty [x^T Q x + u^T R u + 2x^T N u] dt \tag{18}$$

In the LQR framework, Q and R are weight matrices influencing the performance index J. The matrix Q should be either positive definite or positive semi-definite and symmetric, ensuring system stability and performance. Meanwhile, R is required to be a positive definite symmetric matrix. Ideally, both weighting matrices are diagonal, with their elements' magnitudes directly correlating to their impact on the performance index J, thereby balancing state accuracy and control effort [4]. The feedback control law is:

$$u = -Kx \tag{19}$$

where K is given by:

$$K = R^{-1}(B^T S + N^T) \tag{20}$$

and S is found by solving the associated algebraic Riccati equation [5]:

$$A^T S + SA - (SB + N)R^{-1}(B^T S + N^T) + Q = 0 \tag{21}$$

A significant challenge in implementing the LQR Controller lies in determining the appropriate weighting matrices, a factor that can restrict its practical application in control system design. In this project, we define the weight matrices Q and R as such:

$$Q = \begin{bmatrix} 5000 & 0 \\ 0 & 100 \end{bmatrix} R = \begin{bmatrix} 1 \end{bmatrix} \tag{22}$$

We need to define the matrices A and B to calculate the optimal state feedback matrix. We can refer to our previously defined formulations of A and B in equations 11 and 12. Additionally, we can use the calculations of [11] to define these matrices. Once we have applied real values, we can use the A and B matrices for our calculations.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -10.95 & -2.75 & 0.0043 \\ 0 & 0 & 0 & 1 \\ 0 & 24.92 & 28.58 & -0.044 \end{bmatrix} \tag{23}$$

and

$$B = \begin{bmatrix} 0 \\ 1.94 \\ 0 \\ -4.44 \end{bmatrix} \tag{24}$$

Matlab is used to implement the Linear Quadratic Regulator (LQR) optimal control system, which involves solving for the optimal state feedback matrix K. This is done using a specialized tool within Matlab's suite of control system design utilities called the LQR function [15] - which takes input matrices A, B, Q, and R. By handling the complex matrix computations and optimization routines involved in control theory, the LQR function simplifies solving for K. Once solved, the optimal state feedback matrix K is given.

$$K = \begin{bmatrix} -70.7107 & -131.5987 & -49.2563 & -27.4728 \end{bmatrix} \tag{25}$$

IV. Testing & Results

A. LQR Method Results

The outcomes of applying the Linear Quadratic Regulator (LQR) control method using Matlab are illustrated in Fig. 2. This graph clearly demonstrates the system's rapid response, highlighting that the pendulum achieves balance in approximately 1.5 seconds. This swift stabilization shows the effectiveness of the LQR method in determining the optimal state feedback matrix K. The graph illustrates the LQR controller's efficiency in achieving rapid balance. Matlab's LQR function calculates the optimal parameters with precision, resulting in a highly responsive and optimized control system.

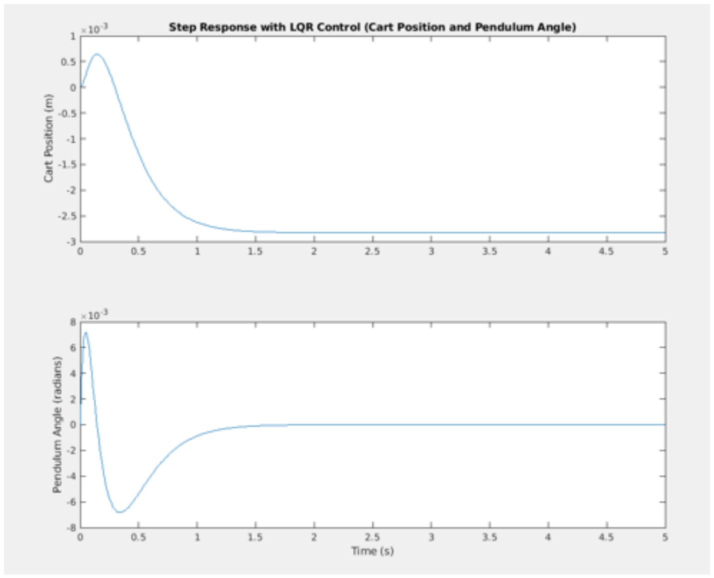

Fig. 2. Results of the LQR control method evaluating the step response of the cart position (top figure) and pendulum angle (bottom figure) over time.

B. RL Method Results

Several parameters were chosen to optimize performance and efficiency in developing the Reinforcement Learning framework utilizing Q-Learning. The implementation of Q-Table and the Deep Q-Learning methods utilized slightly different parameters.

The Q-Table method used a learning rate of 0.1 and a discount factor γ of 0.95. An epsilon-greedy approach was employed to explore the environment with an initial epsilon value of 0.2. The training process was executed for 5000 episodes.

The Deep Q-learning method used a learning rate of 0.001 and a discount factor γ of 0.9. An epsilon-greedy approach was employed to explore the environment with an initial epsilon value of 0.3 and a decay factor of 0.995. The training process was executed for 300 episodes.

The Deep Q-learning method is further implemented, containing a neural network with three fully connected layers. The first layer maps the state dimension to a 128-unit hidden layer, followed by a LeakyReLU activation. The second layer expands this to 256 units (twice the hidden dimension), again using a LeakyReLU activation—the final layer outputs Q-values matching the action dimension. The network employs Mean Squared Error Loss (MSELoss) and uses the Adam optimizer.

The link to the Github can be found at https://github.com/atumlin/CS6376_TermProject.

In Figures 3, 4, 5, and 6, the training outcomes of different Q-Learning methods are presented, showcasing the training of the models over time. These figures show the results of using Q-Table, Deep Q-Learning Network, Deep Q-Learning Network with Replay, and Dual Q-Learning Network approaches to attempt to stabilize the cart-pole system. A key observation

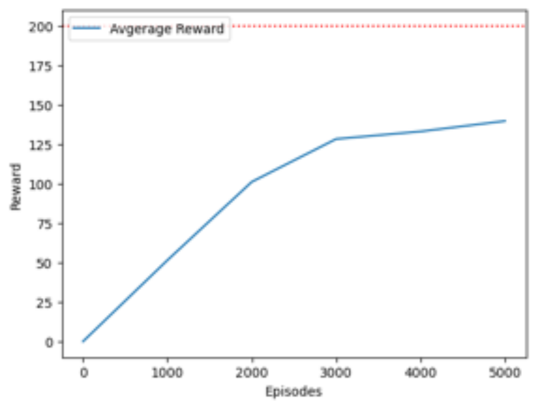

Fig. 3. Results of the training of the Q-Table Learning Method.

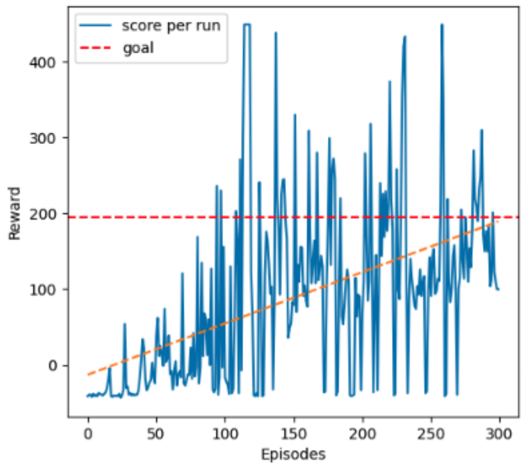

Fig. 4. Results of the training of the Deep Q-Learning Method.

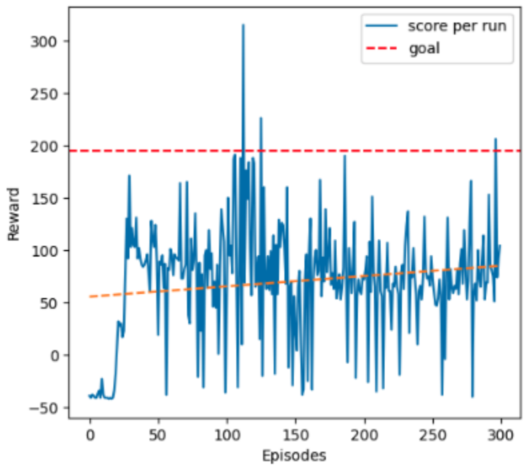

Fig. 5. Results of the training of the Deep Q-Learning Method with Replay.

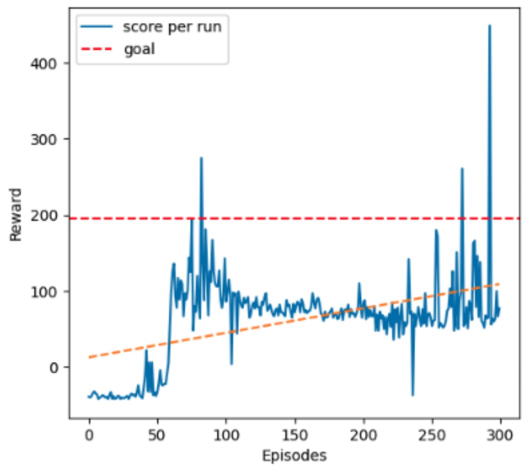

Fig. 6. Results of the training of the Dual Deep Q-Learning Method.

from these results is the apparent inability to achieve stable control of the cart-pole system. This instability could stem from factors such as potential flaws in the implementation process or suboptimal choices of hyperparameters that guide the learning process.

To understand these results in greater detail, it is recommended to view the accompanying video, which offers a more comprehensive analysis of these outcomes. The video also provides animated demonstrations of the cart-pole problem, offering a more intuitive understanding of the system's behavior under different Q-Learning strategies. https://www.loom.com/share/ce40d76ab8ec486ea6296c0f44829fed?sid=8f22cecd-5101-495a-8094-6cc2114485fe

V. Conclusion and Future Work

In this project, the effectiveness of Reinforcement Learning (RL) and Linear Control methods were compared for addressing the inverted pendulum problem. The results found that the LQR approach offered quick and efficient stabilization, whereas the RL method faced relative difficulty achieving stability. The cart-pole system, with its dynamically controllable environment, allowed for a clear understanding of each method's distinct advantages and limitations.

While RL has the potential to manage uncertain, unknown environments better than LQR in theory, this study revealed challenges in achieving balance practically with RL. These difficulties may be attributed to implementation nuances or parameter selection. The results suggest that although Q-learning can balance the pendulum, further refinement is necessary to fully realize this capability within the current research.

If continued, the project's primary focus will be to refine the RL approach to achieve balance, a task that has proven to be more straightforward with LQR. This observation highlights an essential consideration in cyber-physical systems (CPS) and robotics, where designers and engineers must carefully analyze what is more vital for the specific application - the rapid and reliable stabilization offered by LQR or the adaptable, albeit more complex, control potential of RL.

References

[1] Cart Pole - Gym Documentation.

[2] Ch. 8 - Linear Quadratic Regulators.

[3] Control Tutorials for MATLAB and Simulink - Inverted Pendulum: System Modeling.

[4] Ramashis Banerjee and Arnab Pal. Stabilization Of Inverted Pendulum On Cart Based On Pole Placement and LQR. In *2018 International Conference on Circuits and Systems in Digital Enterprise Technology (ICCSDET)*, pages 1–5, December 2018.

[5] Alberto Bemporad, Manfred Morari, Vivek Dua, and Efstratios N Pistikopoulos. The explicit linear quadratic regulator for constrained systems. 2002.

[6] Jason Brownlee. A Benchmark Control Theory Problem.

[7] Hado Hasselt. Double Q-learning. In *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.

[8] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Gabriel Dulac-Arnold, Ian Osband, John Agapiou, Joel Z. Leibo, and Audrunas Gruslys. Deep Q-learning from Demonstrations, November 2017. arXiv:1704.03732 [cs].

[9] Marina Irodova and Robert H Sloan. Reinforcement Learning and Function Approximation.

[10] Beakcheol Jang, Myeonghwi Kim, Gaspard Harerimana, and Jong Wook Kim. Q-Learning Algorithms: A Comprehensive Classification and Applications. *IEEE Access*, 7:133653–133667, 2019.

[11] Taylor Johnson. Stability analysis of simplex architecture controlled inverted pendulum. 2008.

[12] N. Lehtomaki, N. Sandell, and M. Athans. Robustness results in linear-quadratic gaussian based multivariable control designs. *IEEE Transactions on Automatic Control*, 26(1):75–93, 1981.

[13] Chong Li and Meikang Qiu. *Reinforcement Learning for Cyber-Physical Systems: with Cybersecurity Case Studies*. Chapman and Hall/CRC, 1st edition, 2019.

[14] A. Rupam Mahmood, Dmytro Korenkevych, Gautham Vasan, William Ma, and James Bergstra. Benchmarking reinforcement learning algorithms on real-world robots, 2018.

[15] MathWorks. Linear quadratic regulator (lqr) design. https://www.mathworks.com/help/control/ref/lti.lqr.html, 2023. Accessed: yyyy-mm-dd.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland,

Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, February 2015. Number: 7540 Publisher: Nature Publishing Group.

[18] Felix Olowononi, Danda B. Rawat, and Chunmei Liu. Resilient Machine Learning for Networked Cyber Physical Systems: A Survey for Machine Learning Security to Securing Machine Learning for CPS. *IEEE Communications Surveys & Tutorials*, 23(1):524–552, 2021. arXiv:2102.07244 [cs].

[19] Sindhu Padakandla. A survey of reinforcement learning algorithms for dynamically varying environments. *ACM Computing Surveys*, 54(6):1–25, July 2021.

[20] Muammer Eren Sahin, Lo'ai Tawalbeh, and Fadi Muheidat. The security concerns on cyber-physical systems and potential risks analysis using machine learning. *Procedia Computer Science*, 201:527–534, 2022. The 13th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 5th International Conference on Emerging Data and Industry 4.0 (EDI40).

[21] Lu Shoufeng, Liu Ximin, and Dai Shiqiang. Q-Learning for Adaptive Traffic Signal Control Based on Delay Minimization Strategy. In *2008 IEEE International Conference on Networking, Sensing and Control*, pages 687–691, April 2008.

[22] Hiroyshi Nishihara Shozo Mori and Katshuisa Furtua. Control of unstable mechanical system control of pendulum†. *International Journal of Control*, 23(5):673–692, 1976.

[23] Csaba Szepesvári. Algorithms for reinforcement learning. Online, 2009. Accessed: [insert date here].

[24] Hado van Hasselt, Arthur Guez, and David Silver. Deep Reinforcement Learning with Double Q-learning, December 2015. arXiv:1509.06461 [cs].

[25] Christopher J. C. H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, May 1992.