REINFORCEMENT LEARNING BASED STABILIZATION OF CART INVERTED PENDULUM SYSTEM

This thesis is submitted in the partial fulfilment of the requirements of the degree

MASTER IN CONTROL SYSTEM ENGINEERING

Submitted by

Sajay Dutta

Examination Roll Number: M4CTL23010

Registration Number: 160197 of 2021–2022

Under the Guidance of

Dr. Sayantan Chakraborty

Department of Electrical Engineering

Faculty of Engineering and Technology

JADAVPUR UNIVERSITY

KOLKATA–700032

June, 2023

Faculty of Engineering and Technology

JADAVPUR UNIVERSITY

Kolkata-700032

Certificate of Recommendation

This is to certify that **Mr. Sajay Dutta (002110804005)** has completed his dissertation entitled, "**Reinforcement Learning Based Stabilization of Cart Inverted Pendulum System**", under the direct supervision and guidance of **Dr. Sayantan Chakraborty**, Department of Electrical Engineering, Jadavpur University. We are satisfied with his work, which is being presented for the partial fulfilment of the degree of **Master in Control System Engineering** of Jadavpur University, Kolkata-700032.

....

Dr. Sayantan Chakraborty

Assistant Professor,
Electrical Engineering Department
Jadavpur University, Kolkata-700032

....

Dr. Biswanath Roy

Head of the Department,
Electrical Engineering Department,
Jadavpur University, Kolkata-700032

....

Dr. Ardhendu Ghosal

Dean,
Faculty of Engineering & Technology
Jadavpur University, Kolkata-700032

Faculty of Engineering and Technology

JADAVPUR UNIVERSITY

Kolkata-700032

Certificate of Approval

The foregoing thesis entitled "**Reinforcement Learning Based Stabilization of Cart Inverted Pendulum System**" is hereby approved as a creditable study of an Engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as a pre-requisite to the degree of Master in Control System Engineering for which it has been submitted. It is understood that, by this approval the undersigned does not necessarily endorse or approve any statement made, opinion expressed, or conclusion therein but approve this thesis only for the purpose for which it is submitted.

Final Examination for Evaluation of the Thesis

………………………………………………………

………………………………………………………

………………………………………………………

Signature of the Examiners

Declaration of Originality

I hereby declare that this thesis contains a literature survey and original research work by the undersigned candidate, as part of his Master in Control System engineering curriculum. All information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name: Sajay Dutta

Examination Roll No.: M4CTL23010

Thesis Title: Reinforcement Learning Based Stabilization of Cart Inverted Pendulum System

Signature with Date:

ACKNOWLEDGEMENT

I sincerely thank my supervisor, Dr. Sayantan Chakraborty, Department of Electrical Engineering, Jadavpur University, Kolkata, for his invaluable guidance, suggestions, encouragement, and constant support throughout my thesis work, which helped me in successfully completing it. It was a great honour for me to pursue my research under his supervision.

I would also like to thank all my classmate for continuous help and support without which this would not have been possible.

I would like to express my gratitude towards all the staffs of control systems laboratory for providing constant encouragement throughout my thesis work.

Last but not the least, I extend my words of gratitude to my parents, Mr. Manoj Kumar Dutta and Mrs. Sati Dutta, my elder brother Mr. Manab Dutta and my friend Miss Moumita Das for their endless love support to guide me through every thick and thin of life.

Sajay Dutta

Jadavpur University,

Kolkata – 700032

ABSTRACT

For the Classical Method of Control, the system must first be identified. After that, a suitable control law is developed using the system model. System identification presents many difficulties because the majority of the time the system is non-linear and the dynamics of the system are unknown.

Data Driven Control (DDC) system design technique solve this issue by directly controlling the system using just only the input output data without knowing the system model. Even when the system is non-linear, Reinforcement Learning has demonstrated encouraging outcomes in regulating the systems.

Inverted Pendulum is a well-known benchmark problem for developing new control strategies. There exist several traditional control methods for balancing an Inverted Pendulum like PID Controller, LQR Controller, State Dependent Riccati Equation (SDRE) Controller etc. All of these control techniques have major issues like either need to find out the values of different parameters in case of PID (k_P, k_I and k_D gain constant) and LQR (Q and R), or need to solve some complex algebraic equation to get values in all steps in case of SDRE. Values of these constants may affect the system significantly. It can make the system unstable or oscillatory.

The aim of this study is to implement different Reinforcement Learning algorithm to balance an inverted pendulum and compare them. Instead of using classical control algorithms that need a model of the system to be controlled, we used model-free control algorithm i.e., Q-learning, Policy Gradient, Actor-Critic. We demonstrate that reinforcement learning can be successfully used in Inverted Pendulum to balance and control it without having a detailed model. The stability control time is too long in the Q-learning Algorithm. Policy-based algorithms like Actor-Critic and Policy Gradient were also able to obtain some excellent results given the right hyperparameter set.

Keywords: PID, LQR. State Dependent Riccati Equation (SDRE), DDC, Reinforcement Learning, Q-learning, Policy Gradient, Actor-Critic.

CONTENTS

List of Abbreviations

CIPS	: Cart Inverted Pendulum System
MBC	: Model Based Control
PID	: Proportional, Integral Derivative
LQR	: Linear Quadratic Regulator
SDRE	: State Dependent Riccati Equation
MPC	: Model Predictive Control
DDC	: Data Driven Control
AI	: Artificial Intelligence
ML	: Machine Learning
RL	: Reinforcement Learning
DL	: Deep Learning
DRL	: Deep Reinforcement Learning
DNN	: Deep Neural Network
CPS	: Cyber Physical System
MDP	: Markov Decision Process
IFAC	: International Federation of Automatic Control
PPO	: Proximal Policy Optimization
TD	: Temporal Difference

List of Figures

List of Tables

Chapter 1

Introduction & Literature Survey

1.1 History and Background

In 1960's Parametric State Space Model, Optimal and Robust Control together gave rise to the Modern Control Theory [1, 2]. Model Control Theory includes both linear control system design techniques like Zero-Pole Placement, LQR Design etc. and non-linear control system design techniques like Lyapunov based design, Back Stepping, Feedback Linearization etc. The first step of all these control system design techniques is identifying the plant model. Then this plant model is used to design the controller for the system. Therefore, Modelling and Identification of plant model is very much essential and crucial for Modern Control Theory.

Though Modern Control Theory can produce controllers for nonlinear systems, but still assumes linear dynamics. But modern systems are high dimensional and wildly non-linear like self-driving car, flight path control etc. Often this kind of systems either do not have a proper model or model identification is difficult. So, researchers are trying to model controllers based on the system input output data rather than apriori system models

Every research discipline is growing more and more dependent on learning from data. It is the foundation of both statistics and machine learning (ML) or artificial intelligence (AI). It is also becoming more common in other engineering fields. One of the fields where learning from data is being developed intensively is control engineering.

In control theory, data-driven learning is nothing new. System identification [3] is one of the key advancements in the Data Driven Control (DDC) field, where data-driven learning techniques are used to model systems. In this field, the data-driven control techniques prediction error, subspace approaches, and maximum likelihood [4] are all accepted as standards. Despite several advancements in this field, the technique for designing data-driven control systems is still not fully understood.

Reinforcement learning is a machine-learning (ML) technique characterized by an agent capable of self-learning in an environment guided only by numerical rewards [5]. Although historically RL's development was primarily influenced by artificial intelligence (AI), it is indisputable that RL is a direct descendant of optimal control theory [6, 7]. Nevertheless, it was a long time before the control community realized the potential of this technique to address control, as reported in Hoskins and Himmelblau [8]. There was a reduction in research incentives and state-of-the-art development at that time, as the results showed a technique with inferior performance to the proportional–integral–derivative controller (PID) that was also algorithmically complex, data-driven, and a black-box model.

This situation began to change in 2012 due to the consolidation of deep-learning (DL) theory, in which deep neural networks (DNN) began to be used as feature extractors [9, 10]. This allowed RL in problems with high-dimensional state space (i.e., deep reinforcement learning (DRL)) such as, for example, cyber-physical systems (e.g., robot control; see Wulfmeier et al. [11], Peng et al. [12]), fixed, strict and complex environments (e.g., AlphaGO; see Silver et al. [13]; and AI in gaming; see Mnih et al. [14]), and large-scale environments (e.g., Vinyals et al. [15]).

For these reasons, as reported in [16,17], the number of publications on RL applied to control began to grow again.

Many control problems encountered in areas such as robotics and automated driving require complex, nonlinear control architectures. Techniques such as gain scheduling, robust control, and nonlinear model predictive control (MPC) can be used for these problems, but often require significant domain expertise from the control engineer. For example, gains and parameters are difficult to tune. The resulting controllers can pose implementation challenges, such as the computational intensity of nonlinear MPC.

Deep neural networks, trained using reinforcement learning, can be used to implement such complex controllers. These systems can be self-taught without intervention from an expert control engineer. Also, once the system is trained, Reinforcement Learning Policy can be deployed in a computationally efficient way.

Reinforcement learning can be used to create an end-to-end controller that generates actions directly from raw data, such as images. This approach is attractive for video-intensive applications, such as automated driving, since do not have to manually define and select image features.

An Inverted Pendulum is a pendulum whose center of mass is above its pivot point. It is inherently a highly unstable, nonlinear system which is very difficult to control. That’s why it is a benchmark and an important classical problem for control system research and use to analyze and design control laws.

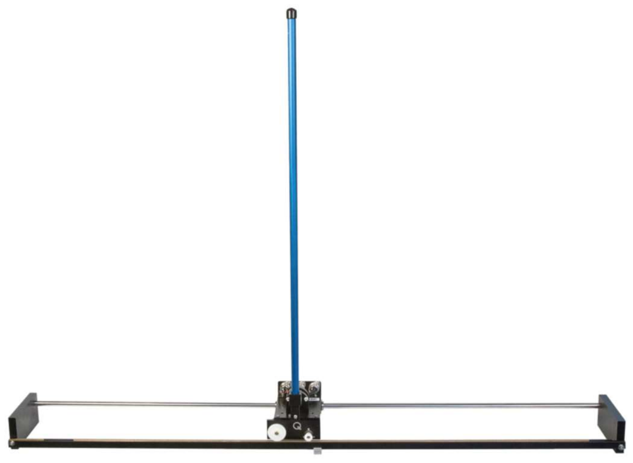


Fig. 1.1: Real inverted pendulum with an IP02 servo plant [18]

Using reinforcement learning to train a single inverted pendulum is a proof of concept that could make other reinforcement learning applications more efficient, letting computers quickly and efficiently train models before implementing them in the real world. Most of the cartpole implementations are either simulations with simplified models or model-free implementations involving virtual training with the help of artificial neural network or Reinforcement Learning.

1.2 Inverted Pendulum

The simple inverted pendulum system is constructed by mounting the pole on cart which can move horizontally using some servo mechanism. This is also called Cart and Pole apparatus. It has a stable and an unstable equilibrium point. Just like normal pendulum, the inverted pendulum oriented downwards in stable equilibrium state. It has the unstable equilibrium point in vertically upward direction. As it is highly unstable, it requires to be controlled always to keep the pendulum in upright position, otherwise it will fall. It can be kept in upright position either by applying torque at the pivot point or by moving the pivot point horizontally using a feedback loop by changing the rate of rotation of mass mounted on the pendulum parallel to the pivot axis and hence developing a net torque on the pendulum or by oscillating the pivot point vertically. An example of moving the pivot point in a feedback loop is achieved by balancing a pen on the finger.

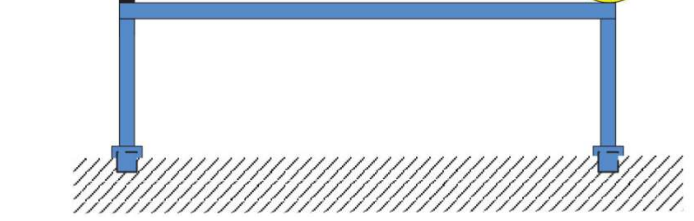


Fig. 1.2: Cart Pole System [19]

The main objective is to utilize specific reinforcement learning technique to balance an inverted pendulum, known as Cart Pole, a single inverted pendulum attached to a cart on a one-dimensional track with a four-dimensional continuous state space.

The cartpole State Space is represented by: $[x, \theta, \dot{x}, \dot{\theta}]$

Where, x = position (right positive)

θ = angle (counterclockwise positive from 0 vertical)

$\dot{x}$ = velocity

$\dot{\theta}$ = angular velocity

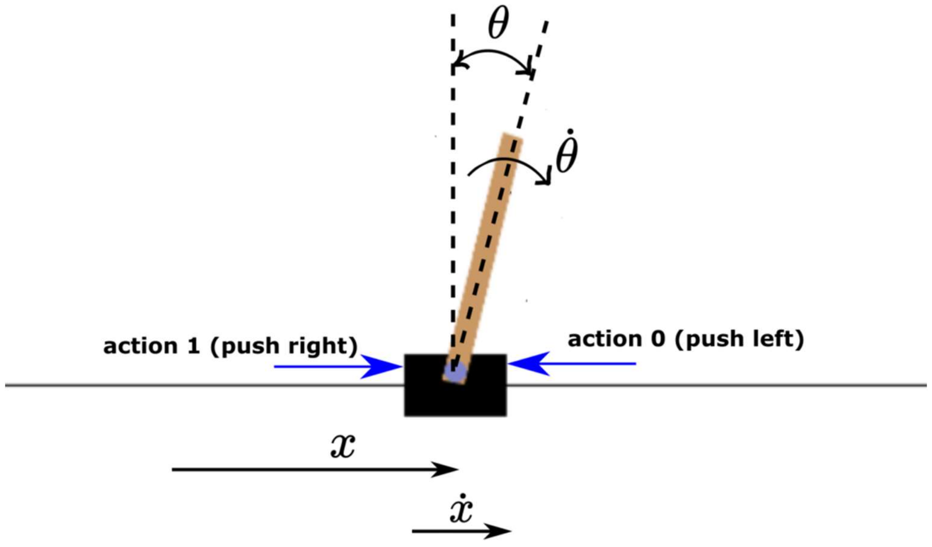


Fig. 1.3: Cart Pole State Representation

This action space is the continuous range of voltages available to the motor, moving the cart left (Action-0) or right (Action-1), respectively. The cart will also move according to the physics of the system, even if no voltage is applied. Goal is to apply the appropriate voltage to the motor in order to balance the pendulum vertically above the cart, without running the cart out of the track. The model is virtually trained, entirely through a simulation in Python or MATLAB. Then the trained model can be implemented on the real inverted pendulum.

1.3 Reinforcement Learning

Reinforcement Learning is one among the main three branch of Machine Learning paradigm alongside Supervised and Unsupervised Learning. It is a feedback-based approach that give an agent the ability to learn from experience just by interacting with the environment. In Supervised Learning, agent requires labelled data consisting of input vectors and respective desired output vectors to supervise its learning process. So, this method cannot be used where the labelled desired datasets are not available. Reinforcement Learning can solve this problem. Agents learn to behave in an unknown environment by taking some actions and seeing the result of its action. For taking a good action the agent gets positive reward and for taking a bad action the agent gets negative reward.

So, Reinforcement Learning is a decision-making technique where agent is learning to take the optimal behavior in an unknown environment to get the maximum reward. Here the data is gathered from the machine learning systems that use a trial-and-error method. Data is not part of the input output labelled data set that can be found in supervised or unsupervised machine learning.

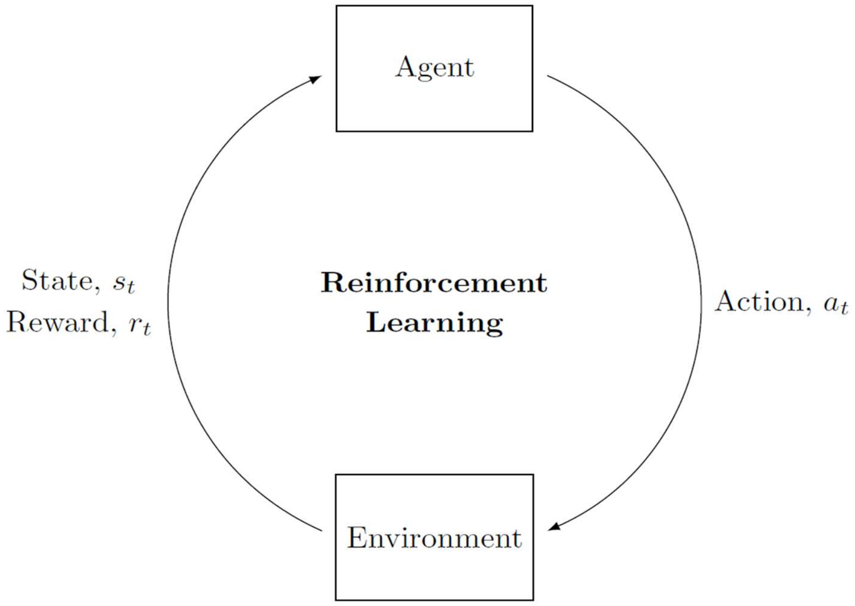


Fig. 1.4: Agent-Environment Interaction Loop

Almost every RL problem can be constructed as one kind of Markov Decision Process (MDP) made of a set of states, a set of actions and a function/policy which describes the transition behaviour. Generally, a MDP is a tuple (S, A, ρ, R, γ) where, S is a finite set of states, A is a finite set of actions, ρ is a state transition probability matrix represents the probability of moving from one state (s) to another state (s') given action a such that

$$\rho(s, a, s') = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a] \tag{1.1}$$

R is the reward function calculating the expected rewards in state s for taking action a

$$R(s, a) = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a] \tag{1.2}$$

And γ is a discount factor takes value in between 0 to 1

The main objective is to find a deterministic policy $\pi : S \to A$ such that $\pi(s) = a$ or stochastic policy $\pi : S \times A \to [0, 1]$ such that,

$$\pi(a|s) = \mathbb{P}[A_t = a \mid S_t = s] \tag{1.3}$$

An optimal policy π^* is the one that maximizes the expected rewards or minimizes the expected cost over any future state actions sequence, sometimes call the reward-to-go.

Optimal action value function maximizes the reward-to-go over all policies so that

$$Q^*(s, a) = \max_{\pi} Q_\pi(s, a) \tag{1.4}$$

And all optimal policies gain the optimal action value function

$$Q_{\pi^*}(s, a) = Q^*(s, a) \tag{1.5}$$

While this optimization MDP problem can also solved using Dynamic Programming or Linear Programming Techniques. But Reinforcement Learning poses two key advantageous features – Sampling and Function Approximation.

Large dimensional problems can be simplified by sampling actions from the action-space to optimize the performance since optimal trajectories for every possible action is not needed. For continuous action-spaces, this makes difficult problems easily solvable. Also due to the universal function approximation properties of neural networks it is possible to solve problems with large state spaces where it would not be feasible to get an analytical solution due to the curse of dimensionality.

1.4 Literature Review

A benchmark test problem in the field of control systems is the inverted pendulum system. Due to the system's inherent instability and nonlinearity, control system research must address this classical issue. The system consists of a horizontally moving cart with a pole placed on it. The pole has an unstable equilibrium point. The International Federation of Automatic Control (IFAC) Theory Committee has identified it as one of the real-world control problems for testing new and existing control techniques [20]. Computer simulations are frequently used to evaluate these controls instead of real pendulums because they can be costly to procure, time-consuming to set up, and labour-intensive to maintain. This is because the model can be simulated with a high degree of precision. Due to their ubiquity, there have been a plethora of classic control techniques used to balance them.

1.4.1 Traditional Control

It is very hard to accurately construct an exact simulation of real inverted pendulum in a computer. While things like friction, motor electrodynamics, and viscous damping coefficients can be included to make the simulation match reality better, various attempts to do this still neglect the nonlinear Coulomb friction applied to the cart, and the force on the cart due to the pendulum's action [21, 22]. Additionally, variations in manufacturing tolerances, unbalanced cart setup, motor play, and general gear wear and tear all contribute to unanticipated noise that could lead to simulated controls failing or performing worse than the simulation predicts. Finally, nonlinear controllers need to be evaluated quickly for online implementation, posing a challenge for computationally intensive controllers. There are significantly fewer inverted pendulums built and balanced in reality as a result of this major obstacle.

PID with a Kalman Filter [23, 24], LQR [25, 26], State-Dependent Riccati Equation (SDRE) [27] and power series approximation of the HJB equation [22] are a few examples of conventional controllers.

1.4.1.1 PID Control

Proportional-integral-derivative (PID) controllers are a staple of industrial process control. By first linearizing the space state equations about the unstable equilibrium, the error $e(t)$ between the current and target states can be fed into the PID controller.

$$u(t) = k_P e(t) + k_I \int_0^t e(\tau)d\tau + k_D \frac{de(t)}{dt} \tag{1.6}$$

Where, k_P, k_I and k_D are the proportional, integral, and derivative gain constants respectively.

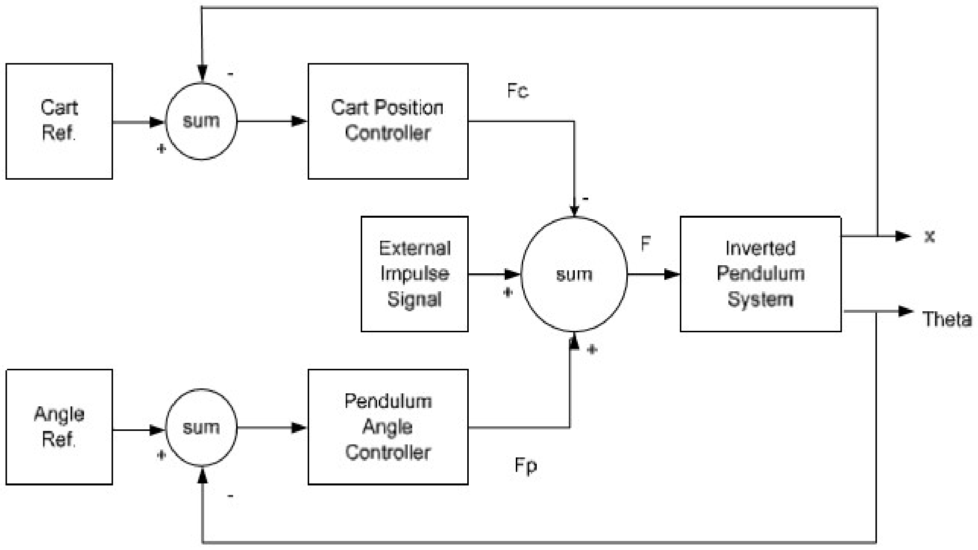


Fig. 1.5: Block Diagram of Cart Inverted Pendulum PID Controller [28]

Finding values for these constants that work for this particular system directly affects the controller's performance. The system may oscillate, become unstable, or not balance the pole at all if one or more of the gain constants is set to an excessively high or low value.

1.4.1.2 LQR Control

Linear-quadratic regulator (LQR) controllers also begin by linearizing the nonlinear state equations the same way, and using the solution P to the algebraic Riccati equation to minimize the cost functional.

$$J(x_0, u) = \int_0^\infty x^T Q x + R u^2 dt \tag{1.7}$$

Then the optimal feedback control is,

$$u = -R^{-1} B^T P x \tag{1.8}$$

Like the PID controller, LQR's performance is directly influenced by the values selected for Q and R; if these values are off, the performance may oscillate or be unstable.

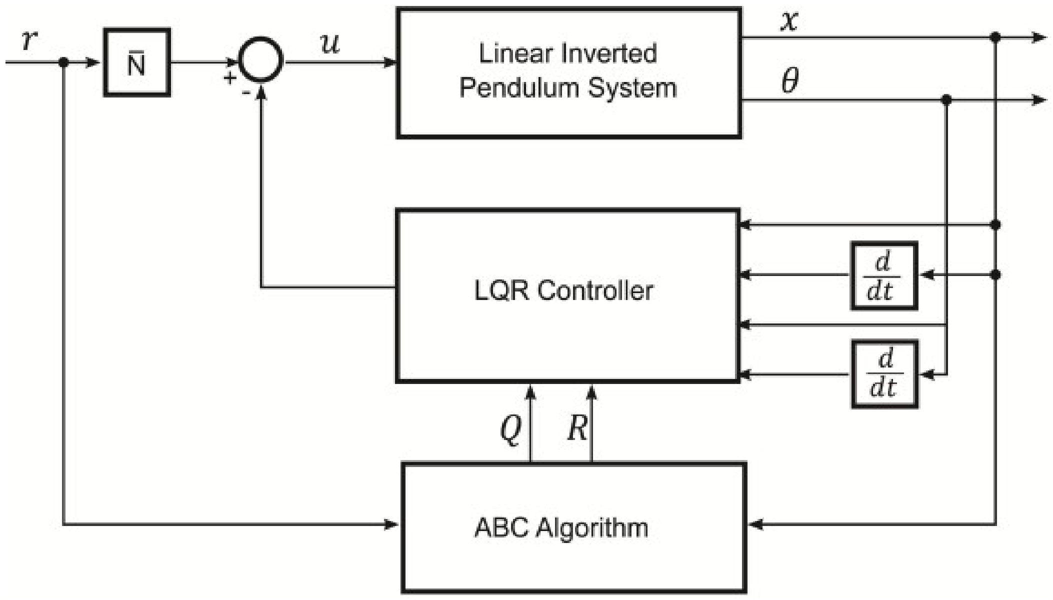


Fig. 1.6: Block Diagram of Cart Inverted Pendulum LQR Controller [29]

1.4.1.3 State Dependent Riccati Equation (SDRE) Control

A Kalman filter is used in both PID and LQR to smoothen sensor inputs and provide an approximation of $\dot{x}$ and $\dot{\theta}$ in the state, although a low-pass derivative filter may perform better if the ideal Kalman parameter values cannot be acquired [27]. The drawback of these controllers is that the linearized state equations diverge more and more from the real nonlinear equations as the state deviates from the unstable equilibrium at the origin. While PID and LQR still work well when the pole is essentially straight up, still a nonlinear controller may operate better over a larger range of values, such as during a disturbance.

Using a nonlinear formulation like the State Dependent Riccati Equation (SDRE) is one way to deal with this issue. Similar to LQR, SDRE begins by solving

$$A(x)^T P(x) + P(x)A(x) - P(x)B(x)R(x)^{-1}B(x)^T P(x) + Q(x) = 0 \tag{1.9}$$

To get $P(x) \geq 0$, which is used for the control

$$u = -R(x)^{-1}B(x)^T P(x)x \tag{1.10}$$

The main difference between LQR and SDRE is that the design matrices $Q(s)$ and $R(x)$ and the plant matrices $A(x)$ and $B(x)$ are state-dependent instead of constant. This allows the same controller to be used to swing-up and balance the pendulum, instead of switching controllers, as the linearized models would require. The major downside of this technique is that it requires solving the algebraic Riccati equation at every time step, which can be computationally expensive and limit the sampling rate. As long as the online computed resources are fast enough for real time implementation, this results in a controller that is more stable and robust.

1.4.2 Reinforcement Learning Control

A control approach needs to be resilient against model errors, sensor noise, and potential system disturbances in order to be effective. Reinforcement learning techniques can be employed as controllers because they generalize to new settings and do not require domain-specific knowledge of the system to produce satisfying outcomes. The same method can be used to balance a pole, drive a car, or play Pong. The majority of reinforcement learning algorithms are trained virtually due to the costs involved with operating in a real environment. There have been numerous attempts to use different reinforcement learning techniques to control a real inverted pendulum, but they have all been doomed by the same problem; gathering data from a real pendulum is a time and money consuming process.

It is now possible to test reinforcement learning algorithms on a variety of standardised contexts thanks to the creation of virtual environments like OpenAI Gym. The Cartpole in OpenAI Gym offers a simplified simulation of an environment with an inverted pendulum that is free of friction and has a discrete left/right action-space. The first environment was created in 1983 [30], and it significantly improved the Boxes algorithm used to teach a computer-analogue how to play tic-tac-toe by using an Adaptive Critic technique.

In the decades since, the development of reinforcement learning has given rise to a number of additional strategies that have been utilized with varied degrees of success. These include plain Policy Gradient, Actor-Critic, and Deep Q-Learning. These fundamental reinforcement learning techniques, however, are generally slow and data-intensive, necessitating thousands of trials before the policies converge. Despite this, OpenAI Gym has become the de facto method for evaluating the performance of novel algorithms like Deep Deterministic Policy Gradient [31], Trust Region Policy Optimization [32], and Proximal Policy Optimization [33]. This is due to the environment's simplicity of usage and implementation. In these algorithms are benchmarked across several OpenAI contexts [34]. However, it is challenging to replicate the findings in reality due to the discrete action space and spartan dynamics of the system. The model is often trained from scratch on a real pole in order to apply reinforcement learning to balance an actual pendulum.

1.5 Thesis Outline

This Thesis is organized as follows,

Chapter 1, includes a brief history and background of Data Driven Control. A short introduction to Inverted Pendulum and Reinforcement Learning is also given. Different Inverted Pendulum Control Techniques from Traditional Control (PID

Control, LQR Control, SDRE Control) to Reinforcement Control are discussed here.

Chapter 2, gives an introduction to Reinforcement Learning. All the terminologies related to Reinforcement Learning is defined and explained in this chapter. Different Reinforcement Learning Algorithm like Q-Learning, Policy Gradient, Actor-Critic are described in brief.

Chapter 3, deals with different modelling schemes like Mathematical Model, Transfer Function Model, State Space Model of Cart Inverted Pendulum. Further, it presents the simulation results for conventional techniques to stabilize Cart Inverted Pendulum.

Chapter 4, discusses the result of Cart Inverted Pendulum Control using different Reinforcement Learning algorithm. A comparative study has been carried out between results of different RL algorithms.

Chapter 5, concludes the contributions of the thesis and points out the scope of future work.

Chapter 2

Reinforcement Learning

2.1 Introduction

Machine Learning is one of the emerging fields in engineering. Instead of step-by-step coding, computers learn from experience using machine learning algorithm. Despite discovery of Neural Network [35], Perceptron [36], Back Propagation [37, 38] in the 1960's, it is making a rapid growth in the last 10 years only due to the increase in the processing power of computers specially graphics processing and tensors processing. This recent boom in machine learning has made the computers able to do wonderful new tasks like diagnose cancer cells more accurately [39], generate art works in style of Mozart [40], Shakespeare [41, 42], Van Gogh [43] etc. It can also play computer games like Dota [44], Go [45] etc. better than professional players. This list is expanding more and more.

Machine Learning is divided into broadly three categories –

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

In **Supervised Learning** machines are trained using labelled data for the purpose of predicting output in future.

In **Unsupervised Learning** unlabelled data is fed into machine to discover hidden structures and patterns in data.

In **Reinforcement Learning** agent takes some action in an unknown environment and get rewards from the interaction. This way agents gets experience and learn to take good action.

So, Reinforcement Learning (RL), is a Machine Learning technique, which uses trial-and-error to interact with the environment or the system by taking actions and obtaining rewards repeatedly. Rewards shows how good that action is. This inspires the agent to take the optimal action when given a state of its environment with the goal to maximize the cumulative rewards, known as the return or feedback. The main goal of RL is to optimize and automate physical tasks, but still not used in many real-world problems.

2.2 Terminologies in Reinforcement Learning

The main objective of reinforcement learning is to train an agent which can chose an optimal policy for itself to get maximum rewards. In this case, the agent is the inverted pendulum, the policy is an artificial neural network with one hidden layer that decides how much voltage to apply to the motor, and the rewards correspond to how long the pole stays balanced.

2.2.1 Policy

Policy describes how an agent would behave at a given time in unknown environment. It means policy is a function which maps a perceived state of environment to an action to be taken when in that state just like the stimulus-response rules of psychology.

Policy can be of two type – Deterministic policy and Stochastic policy.

Deterministic policy $\pi: S \rightarrow A$ describe which action to take in which state of the environment.

Stochastic policy gives a probabilistic distribution over actions for every state such that

$$\pi(a \mid s) = P[A_t = a \mid S_t = s] \tag{2.1}$$

2.2.2 Reward

Maximizing the reward is the main goal of Reinforcement Learning problems. Thus, reward shows which action is good and which action is bad for the agent. In biological system reward is similar to the experience of pleasure and pain.

The reward function $r_t = R(s_t, a_t, s_{t+1})$ depends on the current state, action and future state of the environment.

2.2.3 Value Function

The total amount of reward an agent can expect to accumulate over the future starting from the current state is described by the Value Function. The value function is function of rewards. Whereas Reward gives the immediate feedback of an action taken in a environmental state, Value indicate the long-term accumulation of rewards over the state-action trajectories considering the states that are likely to occur and rewards available in those states.

$$R_t = \sum_{t=0} r_t \tag{2.2}$$

This summation may diverge to infinity as time tends to infinity. This can be prevented by the idea of a discounted reward (to emphasize getting rewards now rather than later in the future). That is, we define $\gamma \in (0, 1]$ so that our cumulative discounted reward at time step t is

$$R_t = \sum_{k=0} \gamma^k r_{t+k} \tag{2.3}$$

It is also called the rewards-to-go as it gives the sum of discounted future rewards. Discounting not only guarantees the sum converges, but also has the additional benefit of prioritizing the current rewards over the future rewards. The value of γ can be tuned to provide the optimal combination of current and future rewards. If value of γ is small, it will prioritize the actions resulting in rewards now, while large value of γ will prioritize rewards over a longer time frame.

2.2.4 Trajectories

A Trajectory is a sequence of States and Actions of a Reinforcement Learning Experiment.

$$\tau = (s_0, a_0, s_1, a_1, \cdots) \tag{2.4}$$

From distribution ρ_0, starting state s_0 is randomly sampled. After that from next time step $t > 0$, next state is sampled out using the policy π

2.2.5 Model

A model is either the inferences to be drawn about how the environment will behave or it mimics the behaviour of the environment. Planning, or the process of choosing a course of action by considering potential future circumstances before they are actually experienced, is carried out with the help of models.

2.2.6 Agent Environment Interaction

The difficulty of learning through interaction to accomplish a goal is simply framed as the reinforcement learning problem. The agent is the learner and the decision-maker. The environment is the component of the system that it interacts with that consists of everything save the agent. The agent continues to choose actions, and the environment responds to those actions by creating new scenarios for the agent to deal with. Rewards are also created by the environment; their unique numerical values are what the agent tries to maximise over time. A task is a complete specification of an environment. It gives how rewards are determined, which is one instance of the reinforcement learning problem.

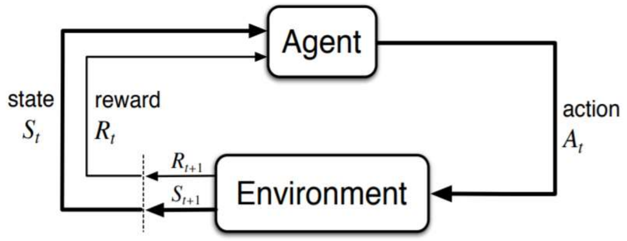


Fig. 2.1: Agent – Environment interaction in Reinforcement Learning

The agent and environment interaction happens at discrete time steps like $t = 0, 1, 2, 3, \ldots$ At every time step t, the agent gets some representation of the environments state, $S_t \in S$, where S is the set of all possible states, and on that basis selects an action, $A_t \in A(S_t)$, where $A(S_t)$ is the set of all possible actions available in that state S_t. After One time step, as a consequence of its action, the agent gets a numerical reward, $R_t \in R$ and moves itself in a new state S_{t+1} . Figure 2.1 shows the agent – environment interaction.

The agent implements a mapping from states to probabilities of choosing each potential action at each time step. This mapping is called the agents policy denoted as π_t, where $\pi_t(a|s)$ is the probability that $A_t = a$ if $S_t = s$. Reinforcement learning techniques outline how the agent modifies its policy in response to its observations. The agent's objective is to maximise its overall return over the long term.

Because it is versatile and abstract, this framework can be used to solve a wide range of issues. The optimisation and control issues have the same MDP framework in place.

2.2.7 Episodic or Continuous Task

Task is defined as an instance of a reinforcement learning problem is of two types – Episodic and Continuous.

In **Episodic Tasks**, there is a starting point (Initial State) and ending point (Terminal State) to a task. It consists of a set of States, set of Actions, Rewards.

Continuous tasks are the tasks that never ends (means no terminal state). There is no starting point and ending point. The agent keeps running until training is completed.

2.2.8 Monte Carlo and Temporal Difference Learning

Monte Carlo and Temporal Difference are the two methods by which agents can learn.

2.2.8.1 Monte Carlo Learning

Monte Carlo learning agent gets the rewards at the end of each episode and uses these cumulative rewards to look how good it performs. In another word, The Monte Carlo method uses random process sampling to tackle a variety of optimisation problems. By randomly sampling the environment for the duration of an episode, the Monte Carlo approach is applied to an MDP, and the optimum course of action is then chosen in order to maximise rewards from the environment.

$$V(S_t) \leftarrow V(S_t) + \alpha (G_t - V(S_t))$$

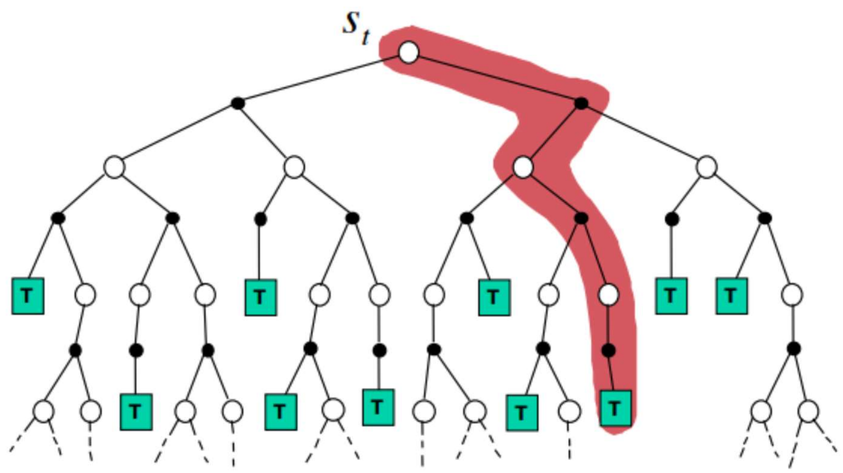


Fig. 2.2: Tree Diagram of Monte Carlo Learning

Generalised Policy Iteration represented in figure 2.2 is the algorithmic framework that Monte Carlo Learning, when applied to Control Problems, shares with Dynamic Programming.

The Monte Carlo for the Prediction problem is considered, and can be further classified into two types:

1. First Visit Monte Carlo Policy Evaluation
2. Every Visit Monte Carlo Policy Evaluation

The environment and the objective are taken into consideration while choosing the Monte Carlo Prediction type. The Monte Carlo update can be used in place of the update equation for the value function in this Generalised Policy Iteration structure:

Number of visits to state $s = N_t(s)$

Total returns from visit to state s, $R_t(s) = R_t(s) + G_t$

Average Return observed from visit to the state s, $V(s_t) = \frac{R_t(s)}{N_t(s)}$

Only episodic tasks can use Monte Carlo Methods since the method needs a Terminal State. This is excellent for the Cart Pole Balancing task since the agent receives a Failure signal at the terminal state, as it has exceeded the limitations defined by the Objective.

In The state value function V(s) in the aforementioned Monte Carlo update is determined by dividing the total return by the total number of visits, which adds extra work because it must be done for each state at each episode. By using the incremental means method, the update rule can be minimised by,

$$Mean_t(s) = Mean_{t-1}(s) + \frac{1}{N(s)}\left(R_t - Mean_{t-1}(s)\right) \tag{2.5}$$

Thus, the algorithm for Monte Carlo Method with incremental updates applied on Cart Pole Balancing problem, at the end of every episode is:

$$Q(S_t, A_t) \longleftarrow Q(S_t, A_t) + \alpha[R_t - Q(S_t, A_t)] \tag{2.6}$$

At the end of an episode in Monte Carlo Learning, the value function is changed using the episode's total return. As the agent chooses actions more effectively in the Cart Pole dilemma, the length of the episode lengthens. In such a case, a state (say, s_1) examined at the beginning of the episode will have to wait until the end of the entire episode to increase the estimate of its quality when Monte Carlo Learning is used to update the agent's value function. whenever the agent’s next state space S' contains s, the old estimate of the state s1 would be used to update the quality of the remaining states throughout the episode. Whenever the agent's next state space S' contains s, the old estimate of the state s1 would be used to update the quality of the remaining states throughout the episode. While the Monte Carlo Method is capable of determining the best course of action, the time it takes between exploring a condition and updating its estimate can cause a delayed learning process. A different class of Reinforcement Learning algorithms has been created to solve the issue of the learning process' slowing down in the past [46].

2.2.8.2 Temporal Difference Learning

Temporal Difference learning agent gets reward after it take every action. It evaluates and update value function after every action. In order to assess the quality of a state, Temporal Difference Learning (TD) is a class of Reinforcement Learning algorithms that uses bootstrapping and one-step updates to the value function.

$$V(S_t) \leftarrow V(S_t) + \alpha\left(R_{t+1} + \gamma V(S_{t+1}) - V(S_t)\right)$$

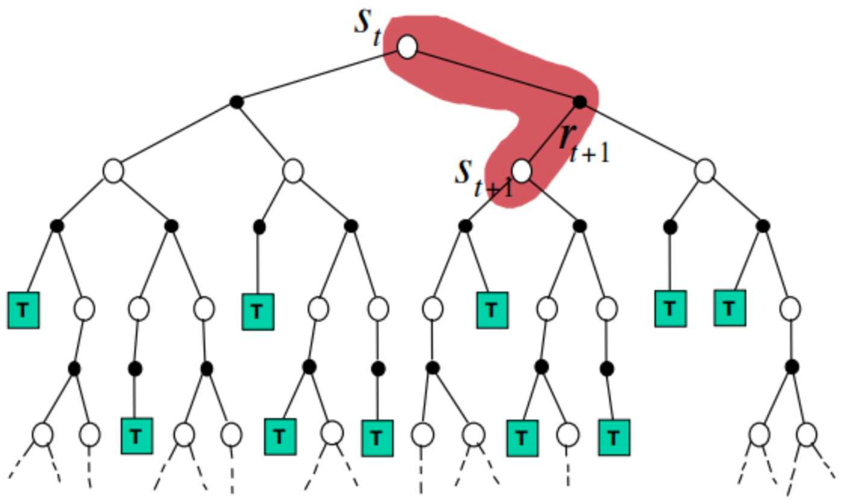


Fig. 2.3: Tree Diagram of Temporal Difference Learning

Unlike Monte Carlo Methods, the TD methods are step-by-step algorithms with online updates of value estimates. At every step of an episode, the quality of the state is updated using the reward obtained at that step and the old estimate of the quality of the next state. In other words, a guess of the state’s quality is updated towards a better guess.

The TD Learning update equation for:

1. **Prediction:**

$$V(S_t) \longleftarrow V(S_t) + \alpha[R_t + \gamma V(S_{t+1}) - V(S_t)] \tag{2.7}$$

Where, S_t = Current State

S_{t+1} = Next State

$V(S_t)$ = Quality of the agent being in State S_t

R_t = Reward obtained by the agent from state S_t

2. **Control:**

$$Q(S_t, A_t) \longleftarrow Q(S_t, A_t) + \alpha[R_t + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \tag{2.8}$$

Where, (S_t, A_t) = Current State Action set

(S_{t+1}, A_{t+1}) = Next State Action set

$Q(S_t, A_t)$ = Quality of the agent being in State S_t and taking action A_t

R_t = Reward obtained by the agent from state S_t by taking action A_t

2.2.9 Exploration and Exploitation Trade Off

Exploration is exploring the environment and getting information about the environment. On the other hand, Exploitation is exploiting the already known information about environment to get maximum reward. If an agent does not explore its environment, it won't be able to know if there are any other actions which will give it more reward. If it does not exploit the known state-actions set it won't be able to find which action will yield max reward among the known actions. So, the agent needs to balance between the exploration and exploitation and it is known as the exploration and exploitation trade off. If the environment is predictable, the agent must experiment by trying different actions in each state and gradually learn to choose the appropriate action for each state. The same action must be tested multiple times in a stochastic environment, however, in order to assess the reward that may be expected from each condition.

2.2.10 Markov Property

In a reinforcement learning situation, it is ideal to have a state signal that summarizes past sensations while still retaining all pertinent information. A signal of this type that is successful in keeping all pertinent data is said to be Markov, or to have Markov property.

Consider how the environment as a whole might react at time $t+1$ to the action taken at time t. In the most general, causal case, this response may depend on everything that has happened earlier in the past. Only by defining the entire joint probability distribution is it possible to characterize the dynamics in this situation:

$$\mathbb{P}\{S_{t+1} = s', R_{t+1} = r \mid S_0, A_0, R_1, \dots, S_{t-1}, A_{t-1}, R_t, S_t, A_t\}$$

for s', r and all possible values of past events $S_0, A_0, R_1, \dots, S_{t-1}, A_{t-1}, R_t, S_t, A_t$.

The next state and reward obtained through environment dynamics and received by the agent at time $t+1$, are said to have the Markov property if they solely depend on the system's state and the RL agent's behavior at time t. The agent keeps track of an estimate of the dynamics of the environment that are inherent to the system through State Transition Probabilities, which can be defined by:

$$\mathbb{P}\{S_{t+1} = s', R_{t+1} = r \mid S_t = s, A_t = a\}$$

for all s', r, s, a

If an environment possesses the Markov property, then we may predict the subsequent state and anticipated reward given the current state and action using the subsequent state and action's one-step dynamics [5, 30]. By repeating this equation, it is possible to demonstrate that it is possible to forecast all future states and expected rewards using only knowledge of the current state, which is equivalent to using knowledge of the entire past up to the present. Therefore, Markov states offer the greatest potential foundation for selecting actions. In other words, the greatest strategy for selecting actions based on a Markov state is just as effective as the best strategy for selecting actions based on entire histories.

2.2.11 Bellman Equation

Agent solves the Q table over time by breaking up the whole problem into multiple simple ones. Rather than solving the true value of a state-action pair in one step, the agent will update the value each time a state-action pair is visited through dynamic programming. The Bellman equation is important for Q-learning as well as other learning algorithms, such as DQN.

$$new\ Q(s,a) = Q(s,a) + \alpha[R(s,a) + \gamma \cdot \max Q'(s',a') - Q(s,a)] \tag{2.9}$$

2.2.12 Different Approaches to Reinforcement Learning

There are three different approaches to Reinforcement Learning – Value Based, Policy Based and Model Based.

2.2.12.1 Value Based

In value-based RL, the goal is to optimize the value function $V(s)$. The value function is a function that tells us the maximum expected future reward the agent will get at each state. The value of each state is the total amount of the reward an agent can expect to accumulate over the future, starting at that state.

$$V_\pi(s) = \mathbb{E}_\pi[R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \mid S_t = s] \tag{2.10}$$

2.2.12.2 Policy Based

In policy-based RL, the policy function $\pi(s)$ is directly optimized without using a value function. The policy is what defines the agent behaviour at a given time. The policy function maps each state to the best corresponding action.

$$a = \pi(s) \tag{2.11}$$

2.2.12.3 Model Based

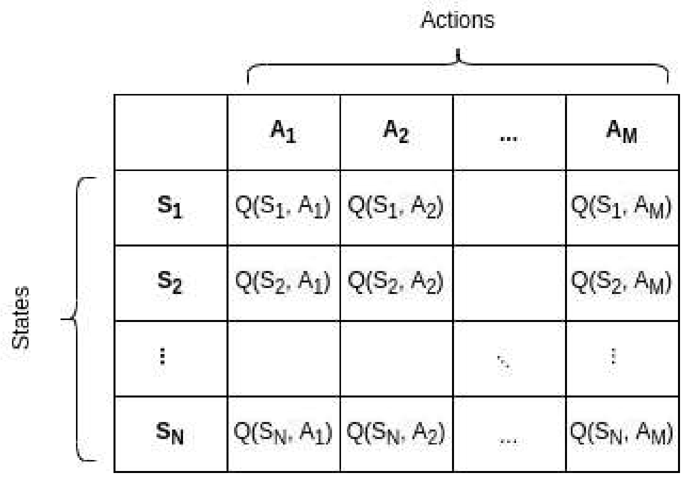
In model-based RL, the environment is modelled. This means we create a model of the behaviour of the environment. The problem is each environment will need a different model representation.

2.3 Q-Learning

Q-Learning is a value based off-policy Reinforcement Learning algorithm used to find the optimal action using the Q-table and Q-function. It uses a greedy approach to select the best action in any state using Q-table so that it gets maximum expected future reward. An exploratory approach is chosen to estimate the action value or Q-value of every possible action in every possible state or to update the Q-table.

2.3.1 Q-Table

Q-table is kind of lookup table in which the maximum expected future reward for each action at each state is stored. In Q-table rows will be the states and columns will be the actions.


Actions
States

	A_1	A_2	...	A_M
S_1	$Q(S_1, A_1)$	$Q(S_1, A_2)$		$Q(S_1, A_M)$
S_2	$Q(S_2, A_1)$	$Q(S_2, A_2)$		$Q(S_2, A_M)$
⋮			⋱	⋮
S_N	$Q(S_N, A_1)$	$Q(S_N, A_2)$	...	$Q(S_N, A_M)$

Fig. 2.4: Q-table

2.3.2 Q-Function / Action-Value Function

Q-function or Action-Value Function is used to update the Q-value (maximum expected future reward to an action in a state) in the Q-table. It takes “State” and “Action” as the two input and uses Bellman equation to give maximum expected future reward as output.

$$Q_{k+1}(s,a) = Q_k(s,a) + \eta \left[R(s,a) + \gamma \max_a Q_k(s',a) - Q_k(s,a)\right] \tag{2.12}$$

2.3.3 Algorithm

1. Create Q-table with m columns (number of actions) and n rows (number of states) and initialize all the values at 0.
2. Chose an action a in the present state s based on the current Q-value estimates.
3. Perform the action a and observe the new State s' and reward r .
4. Update the Q-value in Q-table corresponding to the state s and action a using the Q-function $Q(s, a)$.
5. Repeat the step 2, 3 and 4 until the learning is stopped.

2.4 Policy Gradient

Policy gradient is a policy-based Reinforcement Learning approach. Instead of knowing a value function that tells what is the expected sum of rewards given a state and an action, in policy-based algorithm policy function directly maps the state to an action. It maximizes the expected return for a policy. It does so by shifting the parameters θ in such a way that it increases expected return $J(\pi)$.

For a state-action trajectory $\tau = (s_0, a_0, \dots, s_{T+1})$, the gradient of $J(\pi_\theta)$ is

$$\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \nabla_\theta \underset{\tau\sim\pi_\theta}{\mathbb{E}}[R(\tau)] \\
&= \nabla_\theta \int_\tau P(\tau|\theta)R(\tau) \\
&= \int_\tau \nabla_\theta P(\tau|\theta)R(\tau) \\
&= \int_\tau \frac{P(\tau|\theta)}{P(\tau|\theta)} \nabla_\theta P(\tau|\theta)R(\tau) \\
&= \int_\tau P(\tau|\theta) \frac{\nabla_\theta P(\tau|\theta)}{P(\tau|\theta)} R(\tau) \\
&= \int_\tau P(\tau|\theta) \nabla_\theta \ln P(\tau|\theta)\ R(\tau) \\
\nabla_\theta J(\pi_\theta) &= \underset{\tau\sim\pi_\theta}{\mathbb{E}}[\nabla_\theta \ln P(\tau|\theta)\ R(\tau)]
\end{aligned} \tag{2.13}$$

Expanding $\nabla_\theta \ln P(\tau|\theta)$,

$$\begin{aligned}\nabla_\theta \ln P(\tau|\theta) &= \nabla_\theta \left[\ln \rho_0(s_0) + \sum_{t=0}^{T} (\ln P(s_{t+1}|s_t, a_t) + \ln \pi_\theta(a_t|s_t))\right] \\ &= \nabla_\theta \ln \rho_0(s_0) + \sum_{t=0}^{T} (\nabla_\theta \ln P(s_{t+1}|s_t, a_t) + \nabla_\theta \ln \pi_\theta(a_t|s_t)) \\ &= \sum_{t=0}^{T} \nabla_\theta \ln \pi_\theta(a_t|s_t)\end{aligned} \tag{2.14}$$

Gradients term that does not depend on θ are 0

$$\nabla_\theta J(\pi_\theta) = \underset{\tau \sim \pi_\theta}{\mathbb{E}} \left[\sum_{t=0}^{T} \nabla_\theta \ln \pi_\theta(a_t|s_t) R(\tau)\right] \tag{2.15}$$

It is gradient of the cost function and is used for the Policy Gradient algorithm

Now, an arbitrary advantage function can be chosen given the policy, state an action

$$\nabla_\theta J(\pi_\theta) = \underset{\tau \sim \pi_\theta}{\mathbb{E}} \left[\sum_{t=0}^{T} \nabla_\theta \ln \pi_\theta(a_t|s_t)\ A_{\pi_\theta}(s_t, a_t)\right] \tag{2.16}$$

The advantage function $A_{\pi_\theta}(s_t, a_t) > 0$ points towards direction of increasing $\pi_\theta(a_t|s_t)$

2.4.1 Algorithm

1. Set the parameterized policy $\pi_\theta(a_t|s_t)$.
2. Initialize θ and $s_0 \sim \rho_0$.
3. Sample out the trajectory using $\pi_\theta(a_t|s_t)$.
4. Calculate, $\nabla_\theta J(\pi_\theta) = \sum_{t=0}^{T} \nabla_\theta \ln \pi_\theta(a_t|s_t) R(\tau)$.
5. Update, $\theta \coloneqq \theta + \eta \nabla_\theta J(\pi_\theta)|_\theta$.
6. Repeat step 3, 4 and 5 until training is complete.

2.5 Actor-Critic

Value based methods like Q-Learning learns value function that gives a value to each state action pair. In every state the action with highest value gets selected. This method works well in situation where number of states and actions are finite.

Again, Policy based methods like Policy Gradients optimize a policy which selects the best action to a state directly without using any value function. It can be used where action set is continuous and stochastic. But the problem is that it uses the total rewards in an episode for selecting the actions, not for a single action.

Both of these problems can be solved using a new hybrid methods like Actor-Critic algorithm. Here, an Actor generates the policy that will control how the agent will behave (Policy based approach) and a Critic gives value score to the action actor takes to see the action is good or bad (Value based approach). So, to do this start with the same Policy Gradient policy, but with two neural networks. One for selecting the optimal action in any state and other for finding the value of the corresponding state using the value function $V(s_t)$ to judge it. The policy subtracts a baseline value $b(s_t)$ from the gradient of the cost function in Policy Gradient algorithm. The baseline value depends only on the current state and calculated using value function.

$$\nabla_\theta J(\pi_\theta) = \underset{\tau\sim\pi_\theta}{\mathbb{E}}\left[\sum_{t=0}^{T} \nabla_\theta \ln \pi_\theta(a_t|s_t)\left(\sum_{t'=t}^{T} R_{t'} - b(s_t)\right)\right] \tag{2.17}$$

The Expected Grad-log-Prob (EGLP) is 0 for a parameterized probability distribution P_θ over any random variable x. Means,

$$\underset{x\sim P_\theta}{\mathbb{E}}[\nabla_\theta \ \ln P_\theta(x)] = 0 \tag{2.18}$$

So, for baseline $b(s_t)$ which depends only on current state,

$$\underset{a_t\sim\pi_\theta}{\mathbb{E}}[\nabla_\theta \ \ln \pi_\theta(a_t|s_t) b(s_t)] = 0 \tag{2.19}$$

To get the best baseline value just start by defining a state-action value function which calculates the maximum expected future return if starts from state s and using the policy π_θ take an action a.

$$Q_{\pi_\theta}(s,a) = \underset{\tau\sim\pi_\theta}{\mathbb{E}}[R(\tau)|\, s_0 = s, a_0 = a] \tag{2.20}$$

The state value function which gives maximum expected future return when starts from particular state s and follow the policy π_θ

$$V_{\pi_\theta}(s) = \underset{\tau\sim\pi_\theta}{\mathbb{E}}[R(\tau)|\, s_0 = s,] \tag{2.21}$$

So, the relation between state-action value function Q_{π_θ} and state value function $V_{\pi_\theta}(s)$ is

$$V_{\pi_\theta}(s) = \underset{a\sim\tau}{\mathbb{E}}\left[Q_{\pi_\theta}(s,a)\right] \tag{2.22}$$

Now, the advantage function $A_{\pi_\theta}(s,a)$ is

$$A_{\pi_\theta}(s,a) = Q_{\pi_\theta}(s,a) - V_{\pi_\theta}(s) \tag{2.23}$$

Then, gradient of the cost function,

$$\begin{aligned}
\nabla_\theta\, J(\pi_\theta) &= \underset{\tau\sim\pi_\theta}{\mathbb{E}}\left[\sum_{t=0}^{T} \nabla_\theta \ln \pi_\theta(a_t|s_t)\;\, A_{\pi_\theta}(s_t,a_t)\right] \\
&= \underset{\tau\sim\pi_\theta}{\mathbb{E}}\left[\sum_{t=0}^{T} \nabla_\theta \ln \pi_\theta(a_t|s_t)\;\left(Q_{\pi_\theta}(s_t,a_t) - V_{\pi_\theta}(s_t)\right)\right] \\
&= \underset{\tau\sim\pi_\theta}{\mathbb{E}}\left[\sum_{t=0}^{T} \nabla_\theta \ln \pi_\theta(a_t|s_t)\;\, Q_{\pi_\theta}(s_t,a_t) - \sum_{t=0}^{T} \nabla_\theta \ln \pi_\theta(a_t|s_t)\, V_{\pi_\theta}(s_t)\right] \\
&= \underset{\tau\sim\pi_\theta}{\mathbb{E}}\left[\sum_{t=0}^{T} \nabla_\theta \ln \pi_\theta(a_t|s_t)\;\, Q_{\pi_\theta}(s_t,a_t)\right] - \underset{\tau\sim\pi_\theta}{\mathbb{E}}\left[\sum_{t=0}^{T} \nabla_\theta \ln \pi_\theta(a_t|s_t)\, V_{\pi_\theta}(s_t)\right] \\
&= \nabla_\theta\, J(\pi_\theta) - 0
\end{aligned} \tag{2.24}$$

As,

$$\begin{aligned}\underset{\tau\sim\pi_\theta}{\mathbb{E}}[\nabla_\theta \ln \pi_\theta(a_t|s_t)\, Q_\pi(s_t,a_t)] &= \underset{\tau\sim\pi_\theta}{\mathbb{E}}\left[\nabla_\theta \ln \pi_\theta(a_t|s_t) \underset{\tau\sim\pi_\theta}{\mathbb{E}} Q_{\pi_\theta}(s_t,a_t)\right] \\ &= \underset{\tau\sim\pi_\theta}{\mathbb{E}}\left[\nabla_\theta \ln \pi_\theta(a_t|s_t)\, A_{\pi_\theta}(s_t,a_t)\right] \\ &= \nabla_\theta\, J(\pi_\theta)\end{aligned} \tag{2.25}$$

And,

$$\begin{aligned}\underset{\tau\sim\pi_\theta}{\mathbb{E}}[\nabla_\theta \ln \pi_\theta(a_t|s_t)\, V_\pi(s_t)] &= \underset{\tau\sim\pi_\theta}{\mathbb{E}}\left[\underset{a_t\sim\pi_\theta}{\mathbb{E}}\left[\nabla_\theta \ln \pi_\theta(a_t|s_t)\, V_{\pi_\theta}(s_t)\right]\right] \\ &= \underset{\tau\sim\pi_\theta}{\mathbb{E}}\left[\underset{a_t\sim\pi_\theta}{\mathbb{E}}[\nabla_\theta \ln \pi_\theta(a_t|s_t)] V_{\pi_\theta}(s_t)\right] \\ &= \underset{\tau\sim\pi_\theta}{\mathbb{E}}\left[0 \cdot V_{\pi_\theta}(s_t)\right] \\ &= 0\end{aligned} \tag{2.26}$$

2.5.1 Algorithm

1. Set the parameterized policy $\pi_\theta(a_t|s_t)$.
2. Initialize θ and $s_0 \sim \rho_0$.
3. Sample out the trajectory using $\pi_\theta(a_t|s_t)$.
4. Find V_{π_θ} to rewards.
5. Calculate, $Q_{\pi_\theta}(s_t,a_t) = \sum_k \gamma^k r_{t+k}$.
6. Calculate, $A_{\pi_\theta}(s_t,a_t) = Q_{\pi_\theta}(s_t,a_t) - V_{\pi_\theta}(s_t)$.
7. Calculate, $\nabla_\theta\, J(\pi_\theta) = -\sum_t \nabla_\theta \ln \pi_\theta(a_t|s_t)\, A_{\pi_\theta}(s_t,a_t)$.
8. Update, $\theta \coloneqq \theta + \eta \nabla_\theta\, J(\pi_\theta)|_\theta$.
9. Repeat step 3 to 9 until training is completed.

Chapter 3

Cart-Inverted Pendulum

3.1 Introduction

An Inverted Pendulum is a pendulum whose centre of mass is above its pivot point. It is inherently a highly unstable, nonlinear system which is very difficult to control. That's why it is a benchmark and an important classical problem for control system research and use to analyse and design control laws.

The system is constructed by mounting the pole on cart which can move horizontally using some servo mechanism. This is also called Cart and Pole apparatus. It has a stable and an unstable equilibrium point. Just like normal pendulum, the inverted pendulum oriented downwards in stable equilibrium state. It has the unstable equilibrium point in vertically upward direction. As it is an inherently unstable system, it requires a control action throughout its operation to keep the pendulum in upright position. It can be kept in upright equilibrium position either (a) by applying a torque at the pivot point or (b) by moving the pivot point horizontally using a feedback loop by changing the rate of rotation of mass mounted on the pendulum parallel to the pivot axis and hence developing a net torque on the pendulum or (c) by oscillating the pivot point vertically. An example of moving the pivot point in a feedback loop is achieved by balancing a pen on the finger.

3.2 Free Body Diagram

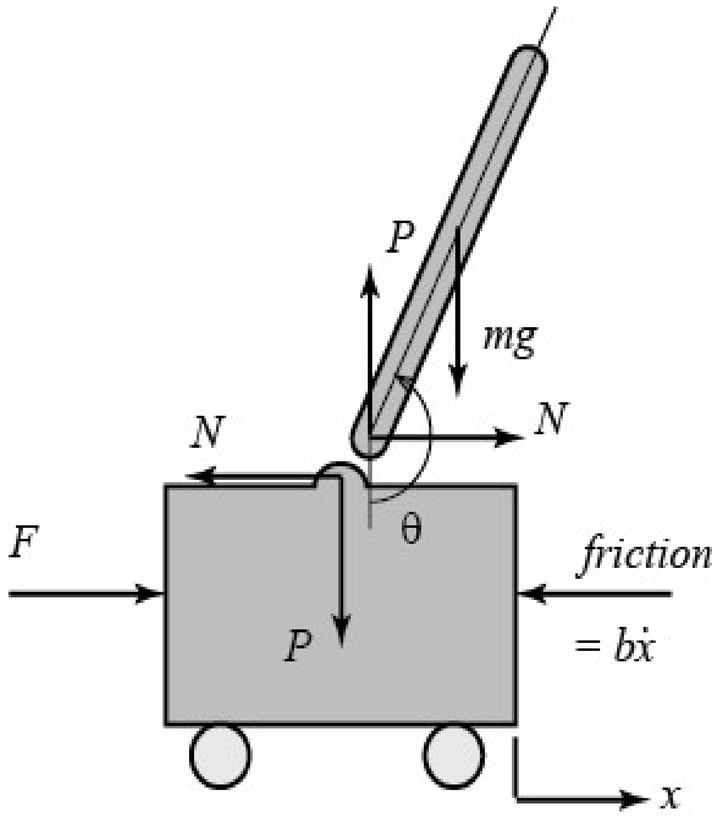


Fig. 3.1: Free Body Diagram of Cart Inverted Pendulum [47]

Here, M = Mass of the Cart

m = Mass of the Pendulum

b = Friction Co-efficient of Cart

l = Length to Pendulum Centre of Mass

I = Moment of Inertia of the Pendulum

F = Force applied to the Cart

x = Cart Position Co-ordinate

θ = Pendulum Angle

g = Gravitational Acceleration

3.3 Mathematical Model

Adding the forces in the horizontal direction in the free body diagram of the cart

$$M\ddot{x} + b\dot{x} + N = F \tag{3.1}$$

Adding the forces in the vertical direction in the free body diagram of the cart does not yield any information.

Form the forces in the horizontal direction in the free body diagram of the pendulum, the expression of N is evaluated as

$$N = m\ddot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta \tag{3.2}$$

By substituting equation (3.2) in equation (3.1), one of the two governing equation is derived as,

$$(M+m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F \tag{3.3}$$

To get the second governing equation of the system, adding and solving the forces in the vertical direction

$$P\sin\theta + N\cos\theta - mg\sin\theta = ml\ddot{\theta} + m\ddot{x}\cos\theta \tag{3.4}$$

Using the sum of the moments about the centroid of the pendulum to get rid of the P and N,

$$-Pl\sin\theta - Nl\cos\theta = I\ddot{\theta} \tag{3.5}$$

Combining the last two equation (3.4) and (3.5),

$$(I + ml^2)\ddot{\theta} + mgl\sin\theta = -ml\ddot{x}\cos\theta \tag{3.6}$$

So, the two mathematical governing equation of the Cart Inverted Pendulum system is,

$$(I + ml^2)\ddot{\theta} + mgl\sin\theta = -ml\ddot{x}\cos\theta \tag{3.7a}$$

$$(M + m)\ddot{x} + b\dot{x} + ml\ddot{\theta}\cos\theta - ml\dot{\theta}^2\sin\theta = F \tag{3.7b}$$

3.4 Linearized Model

Classical control techniques for analysis and design can apply only in linear system. So, to linearize the Cart Inverted Pendulum system equations along the vertical upright position ($\theta = \pi$), assume that the pendulum does not deviate more than $20°$ from the vertically upright position. Let, ϕ is the small deviation from the equilibrium, so that, $\theta = 180° + \phi$. So, approximately,

$$\cos\theta = \cos(180° + \phi) \approx -1$$

$$\sin\theta = \sin(180° + \phi) \approx -\phi$$

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

Substituting this to equation (3.7a), (3.7b) and assuming u is the input force. Then the linearized version of the governing equations of the Cart Inverted Pendulum system is,

$$(I + ml^2)\ddot{\phi} - mgl\phi = ml\ddot{x} \tag{3.8a}$$

$$(M + m)\ddot{x} + b\dot{x} - ml\ddot{\phi} = u \tag{3.8b}$$

3.5 Transfer Function Model

To obtain the transfer function from the linearized system equation, take the Laplace Transform of system equations assuming zero initial conditions,

$$(I + ml^2)\Phi(s)s^2 - mgl\Phi(s) = mlX(s)s^2 \tag{3.9a}$$

$$(M + m)X(s)s^2 + bX(s)s - ml\Phi(s)s^2 = U(s) \tag{3.9b}$$

Transfer function representation is a single input single output representation. To get the first transfer function where output is $\Phi(s)$ and input is U(s), need to eliminate X(s) from above equations and solve it.

From equation (3.9a),

$$X(s) = \left[\frac{I + ml^2}{ml} \quad - \frac{g}{s^2}\right] \Phi(s)$$

(3.10)

Substitute this to equation (3.9b),

$$(M + m)\left[\frac{I + ml^2}{ml} \quad - \frac{g}{s^2}\right] \Phi(s)s^2 + b\left[\frac{I + ml^2}{ml} \quad - \frac{g}{s^2}\right] \Phi(s)s - ml\Phi(s)s^2 = U(s)$$

(3.11)

Rearranging this,

$$\frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s^2}{s^4 + \frac{b(I + ml^2)}{q}s^3 - \frac{(M + m)mgl}{q}s^2 - \frac{bmgl}{q}s}$$

(3.12)

Where, $q = [(M + m)(I + ml^2) - (ml)^2]$

By cancelling the both a pole and a zero in origin transfer function becomes,

$$P_{pend}(s) = \frac{\Phi(s)}{U(s)} = \frac{\frac{ml}{q}s}{s^3 + \frac{b(I + ml^2)}{q}s^2 - \frac{(M + m)mgl}{q}s - \frac{bmgl}{q}} \qquad \left[\frac{rad}{N}\right]$$

(3.13)

Similarly, the transfer function with the cart position X(s) as input and U(s) as output is,

$$P_{cart}(s) = \frac{\mathrm{X}(s)}{U(s)} = \frac{\frac{(I + ml^2)s^2 - gml}{q}}{s^4 + \frac{b(I + ml^2)}{q}s^3 - \frac{(M + m)mgl}{q}s^2 - \frac{bmgl}{q}s} \qquad \left[\frac{m}{N}\right]$$

(3.14)

3.6 State Space Model

Linearized equations of motion can rearrange to get the state space form as follow,

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & \frac{-(I+ml^2)b}{(M+m)I+Mml^2} & \frac{m^2gl^2}{(M+m)I+Mml^2} & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \frac{-mlb}{(M+m)I+Mml^2} & \frac{mgl(M+m)}{(M+m)I+Mml^2} & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{I+ml^2}{(M+m)I+Mml^2} \\ 0 \\ \frac{ml}{(M+m)I+Mml^2} \end{bmatrix} u$$

(3.15a)

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u$$

(3.15b)

The C matrix has 2 rows as both the cart's position and the pendulum's position are part of the output.

3.7 Problem Description

Table 3.1: Parameters of Cart Inverted Pendulum

Mass of the Cart (M)	1 kg
Mass of the Pendulum (m)	0.1 kg
Friction of the Cart (b)	0.1 N/m/sec
Length to the Pendulum Centre of Mass (l)	0.5 m
Inertia of Pendulum (I)	0.006 kg*m^2
Gravitational Acceleration (g)	9.81 m/sec^2

So, for above specification the transfer function become

$$P_{cart}(s) = \frac{X(s)}{U(s)} = \frac{3.09 \times 10^{-5}\, s^2 - 0.0004893}{3.155 \times 10^{-5} s^4 + 3.09 \times 10^{-6}\, s^3 - 0.0005382\, s^2 - 4.839 \times 10^{-5}\, s} \tag{3.16a}$$

$$P_{pend}(s) = \frac{\Phi(s)}{U(s)} = \frac{4.99 \times 10^{-5}\, s}{3.155 \times 10^{-5} s^3 + 3.09 \times 10^{-6}\, s^2 - 0.0005382\, s - 4.839 \times 10^{-5}} \tag{3.16b}$$

And the state space model will be

$$\begin{bmatrix} \dot{x} \\ \ddot{x} \\ \dot{\phi} \\ \ddot{\phi} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -0.0981 & 0.7753 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -0.1582 & 17.06 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0.981 \\ 0 \\ 1.582 \end{bmatrix} u \tag{3.17a}$$

$$y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \\ \phi \\ \dot{\phi} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix} u \tag{3.17b}$$

3.8 Stabilization of CIPS using Conventional Techniques

3.8.1 PID Controller

Two PID controller is used – one is for position control and another one is angle control of the Cart Inverted Pendulum System. General equation for both the PID controller is,

$$u(t) = k_P e(t) + k_I \int_0^t e(\tau) d\tau + k_D \frac{de(t)}{dt} \tag{3.18}$$

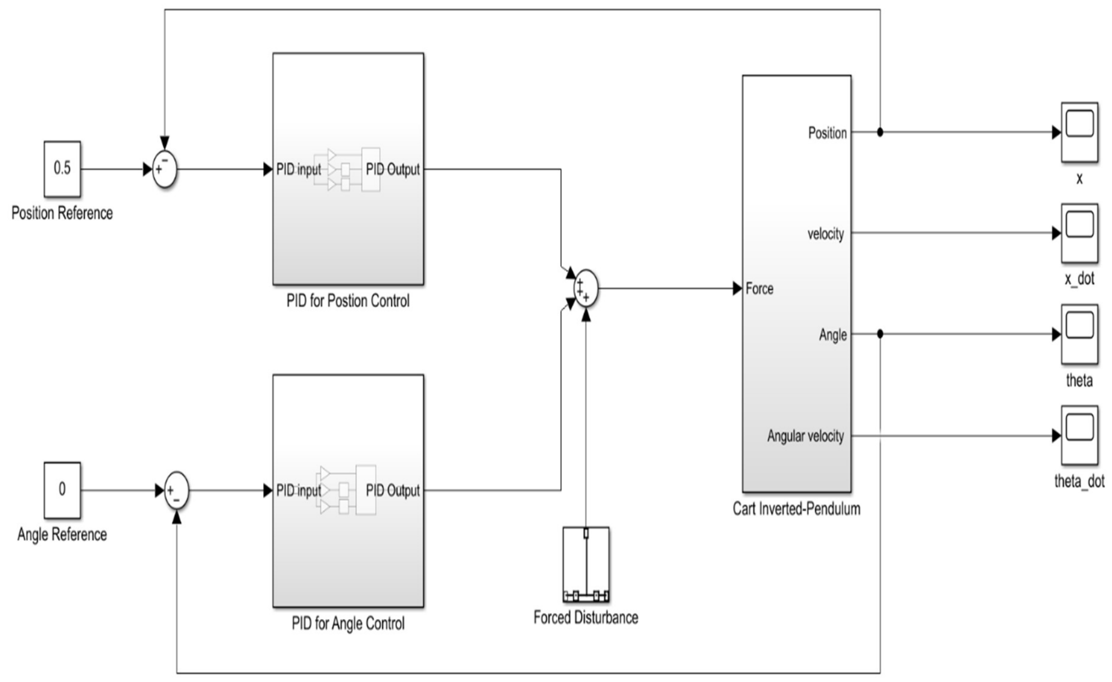


Fig. 3.2: Simulink Model of Cart Inverted Pendulum with PID Control

Using GA based tuning method, the value of gain constants for PID Controller has been calculated.

Table 3.2: Value of Gain Constants for PID Controller

Gain Constants	**PID for Position Control**	**PID for Angle Control**
Proportional Gain (k_P)	396	36
Integral Gain (k_I)	296	206
Derivative Gain (k_D)	106	37

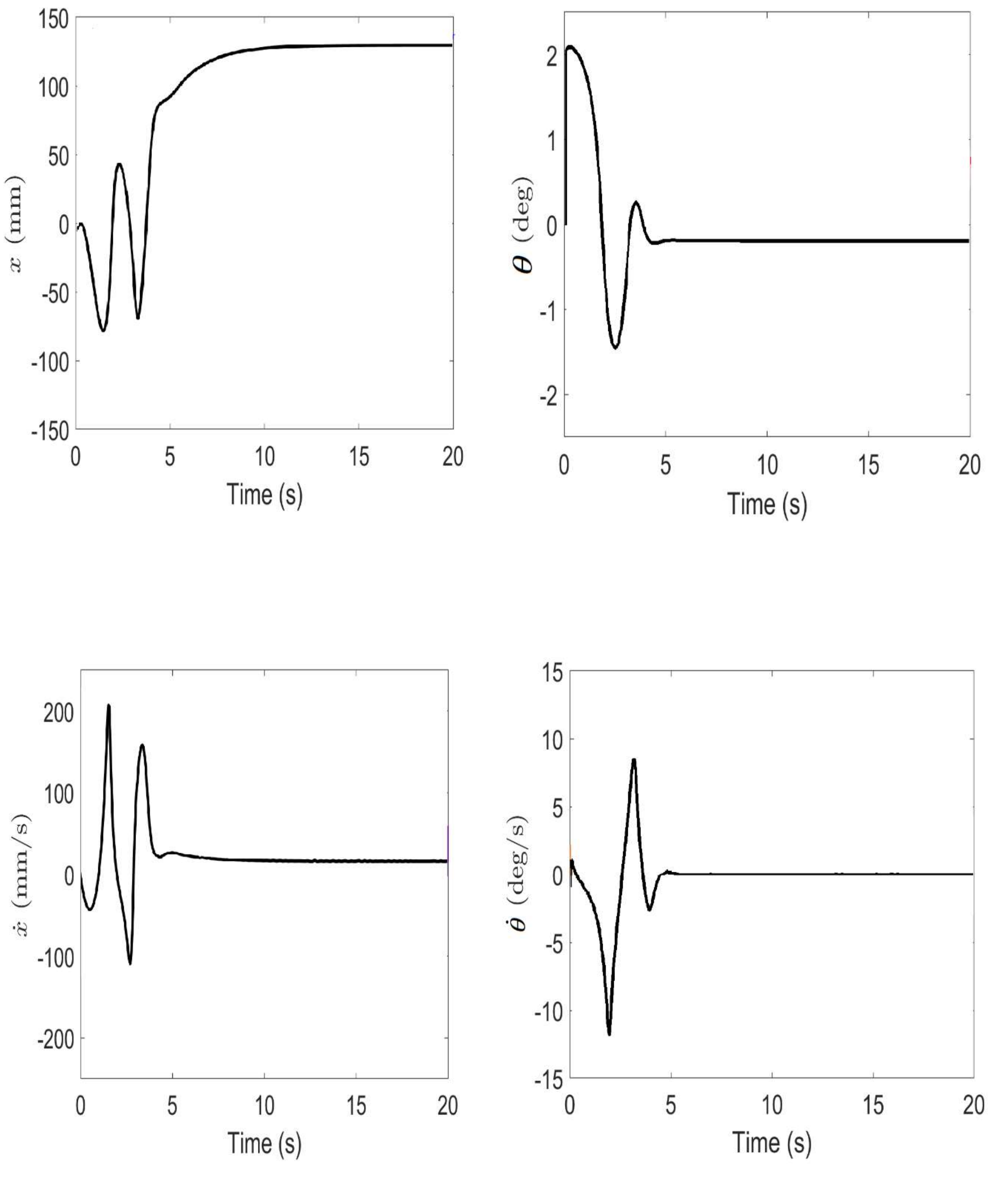


Fig. 3.3: State Response of Cart Inverted Pendulum System using PID Control

3.8.2 Model Predictive Controller (MPC)

Model predictive control (MPC) is an optimal control technique in which the calculated control actions minimize a cost function for a constrained dynamical system over a finite, receding, horizon.

At each time step, an MPC controller receives or estimates the current state of the plant. It then calculates the sequence of control actions that minimizes the cost over the horizon by solving a constrained optimization problem that relies on an internal plant model and depends on the current system state. The controller then applies to the plant only the first computed control action, disregarding the following ones. In the following time step the process repeats.

In practice, despite the finite horizon, MPC often inherits many useful characteristics of traditional optimal control, such as the ability to naturally handle multi-input multi-output (MIMO) plants, the capability of dealing with time delays (possibly of different durations in different channels), and built-in robustness properties against modeling errors. Nominal stability can also be guaranteed by using specific terminal constraints. Other additional important MPC features are its ability to explicitly handle constraints and the possibility of making use of information on future reference and disturbance signals, when available.

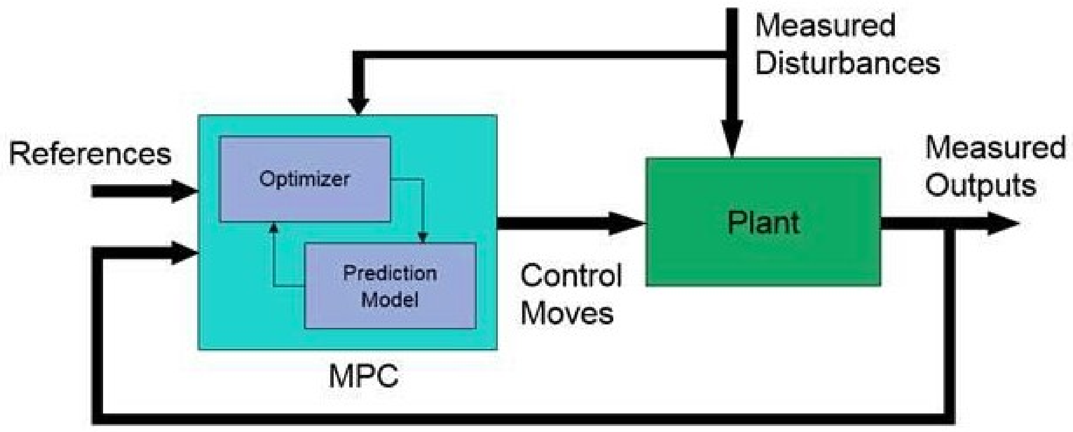


Fig. 3.4: MPC Basic Control Loop

In this scheme a single MPC controller is used with:

- One manipulated variable: variable force F.
- Two measured outputs: Cart position x and pendulum angle θ.
- One unmeasured disturbance: Impulse disturbance dF.

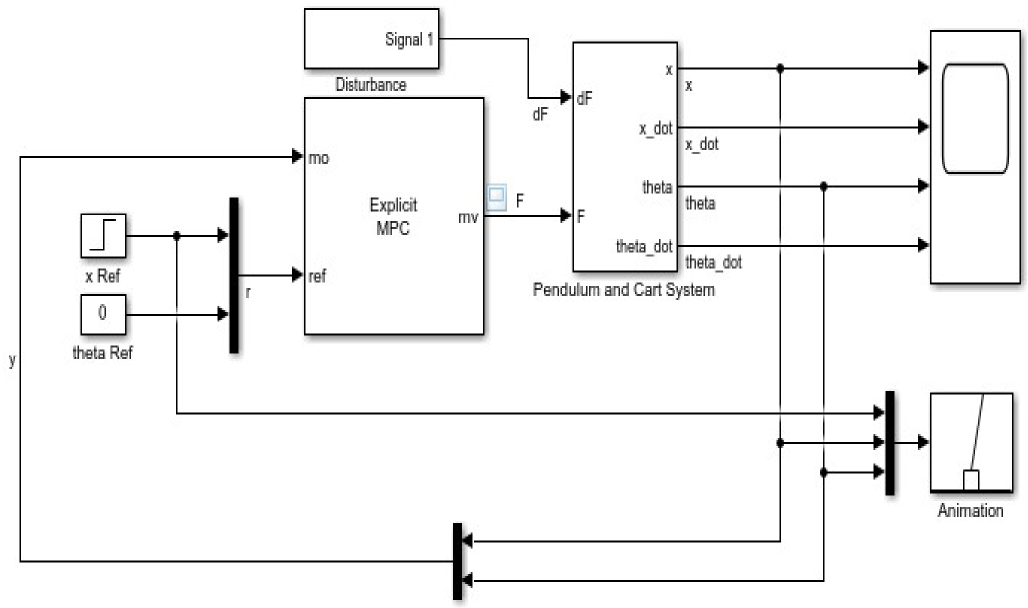


Fig. 3.5: Simulink Model of Cart Inverted Pendulum with MPC Control

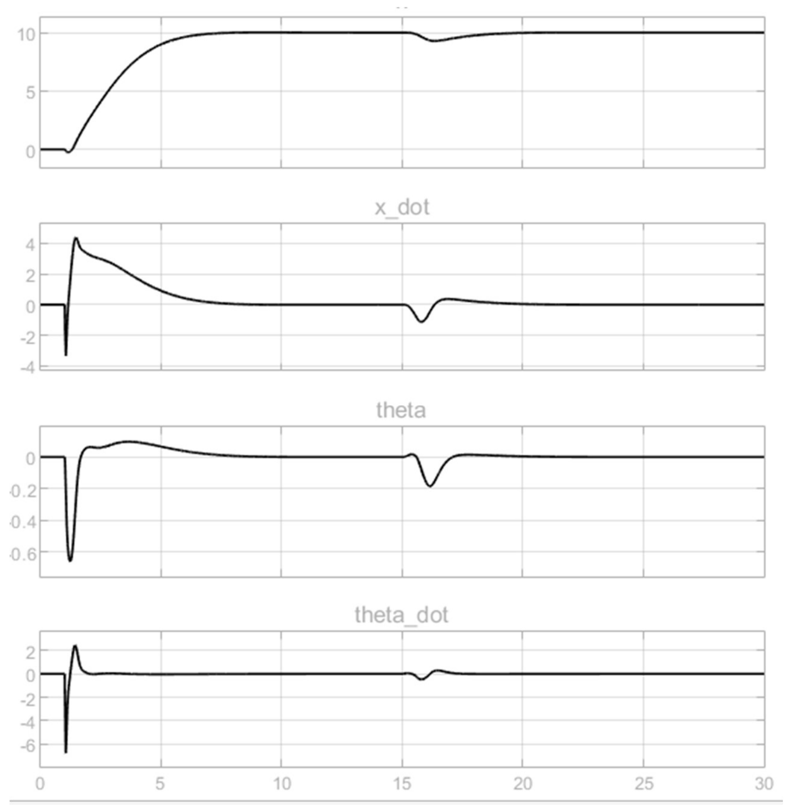


Fig. 3.6: State Response of Cart Inverted Pendulum System using MPC Control

3.8.3 LQR Controller

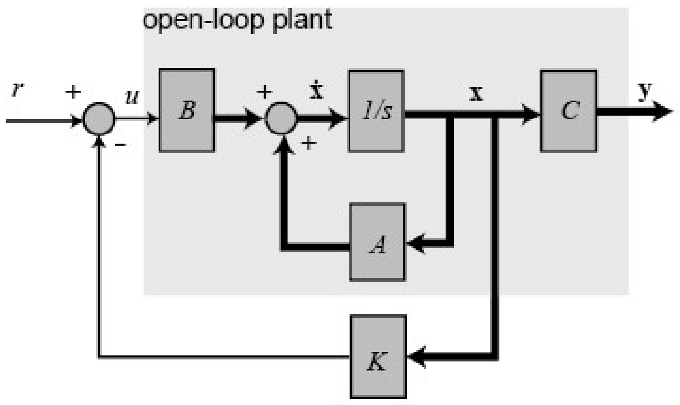


Fig. 3.7: Block Diagram of Cart Inverted Pendulum System using LQR Control

$$\text{Here, } Q = \begin{bmatrix} 5000 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \text{ and } R = 1$$

$$K = [-70.7107 \quad -47.8411 \quad 168.7206 \quad 42.1634]$$

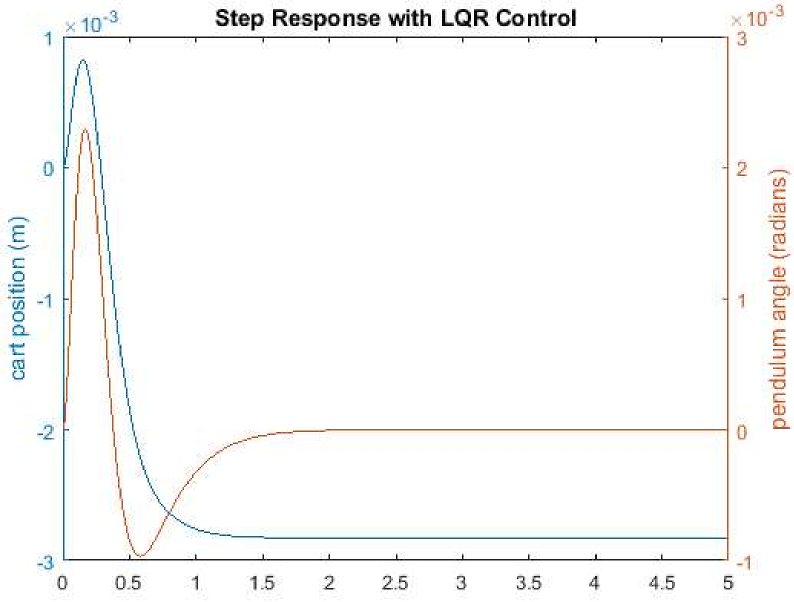


Fig. 3.8: State Response of Cart Inverted Pendulum System using LQR Control

Chapter 4

Results

4.1 Training

We trained all of the above-described algorithm taking different hyper-parameters (learning rate η and the discount factor γ is varied). Training was said to be successful if the agent can keep the pendulum upright ($|\theta| < 12°$) without going out of the track ($|x| < 2.4$m) for all the 500 steps.

Training was done virtually and all the above-mentioned algorithms were able to balance the virtual inverted pendulum. But different algorithms took different amounts of time to learn how to balance the inverted pendulum. Actor Critic performed better among all the algorithms. For different discount factor γ, the learning time was also different. The average learning time was minimum if discount factor γ is taken as 0.99. More than half of the simulation was successfully able to balance the cart pole. Q-learning was able to balance the pole, but failed in most of the cases. Again, Policy Gradient most of the time got stuck in local optimum and did not improve the policy performance fully. Actor-Critic resolves all this issues by dynamically changing hyper-parameters. It was also seen that with a very high learning rate $\eta = 1000$, the agent would never learn the optimal policy. The result shows that that the pendulum angle range is quite wide and covers the whole permitted state space from -12° to +12° during the episode corresponding to the optimal policy.

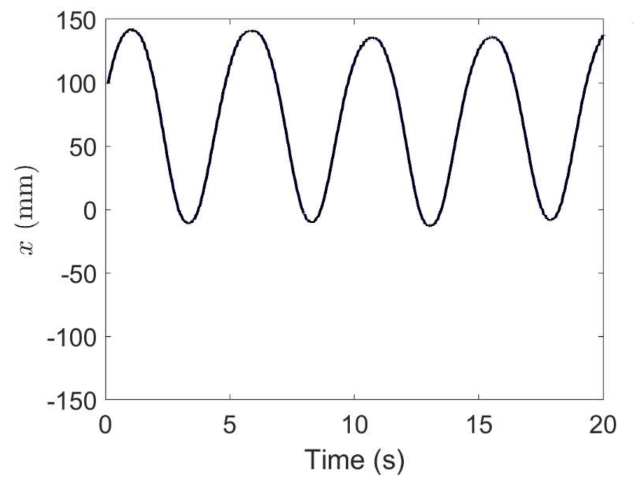
Table 4.1: Cart Inverted Pendulum Environment Properties

Property	**Description**	**Value**
Gravity	Acceleration due to gravity in meters per second squared	9.8
MassCart	Mass of the cart in kilograms	1
MassPole	Mass of the pole in kilograms	0.1
Length	Half the length of the pole in meters	0.5
MaxForce	Maximum horizontal force magnitude in newtons	10
Ts	Sample time in seconds	0.02
ThetaThresholdRadians	Pole angle threshold in radians	0.2094
XThreshold	Cart position threshold in meters	2.4
RewardForNotFalling	Reward for each time step the pole is balanced	1
PenaltyForFalling	Reward penalty for failing to balance the pole	Discrete: -5 Continuous: -50
State	Environment state, specified as a column vector with the following state variables: • Cart position • Derivative of cart position • Pole angle • Derivative of pole angle	$\begin{bmatrix}0\\0\\0\\0\end{bmatrix}$

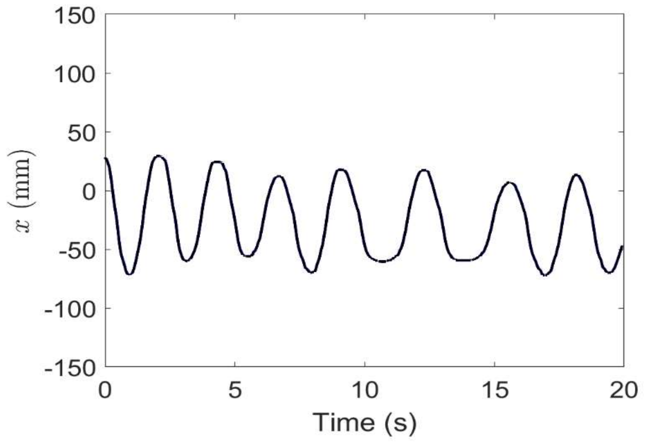
Table 4.2: Training results for all three policy-based algorithms from Chapter 2 using different discount factors γ

Algorithm	**Discount Factor (γ)**	**Percentage of Success**	**Average no of Trials**
Q-Learning	0.9	50	420
	0.99	75	330
	0.999	60	360
Policy Gradient	0.9	55	400
	0.99	80	310
	0.999	70	345
Actor-Critic	0.9	65	315
	0.99	90	275
	0.999	75	290

Q-Learning



Policy Gradient



Actor- Critic

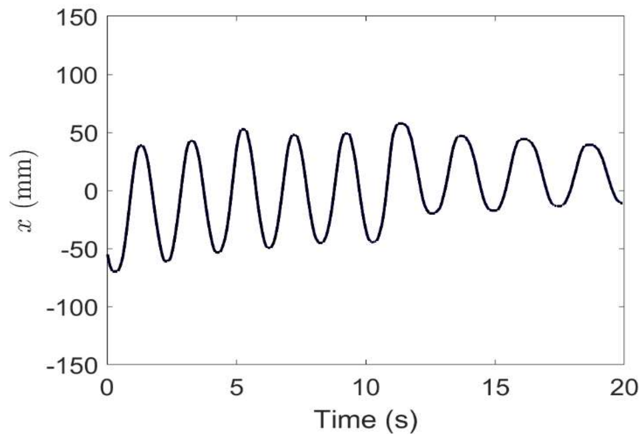
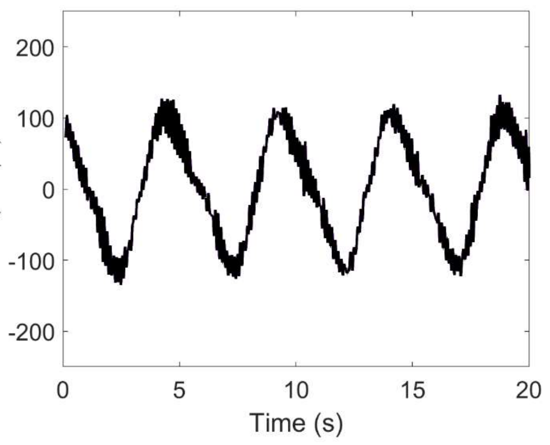
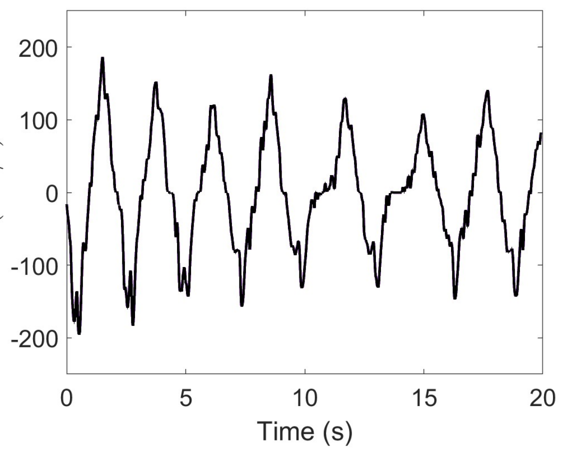


Fig. 4.1: State response of position x throughout a continuous 20-second period for each algorithm

Q-Learning



Policy Gradient



Actor-Critic

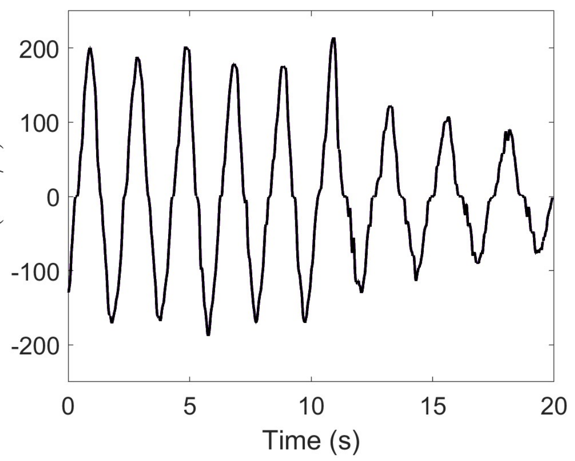


Fig. 4.2: State response of velocity $\dot{x}$ throughout a continuous 20-second period for each algorithm

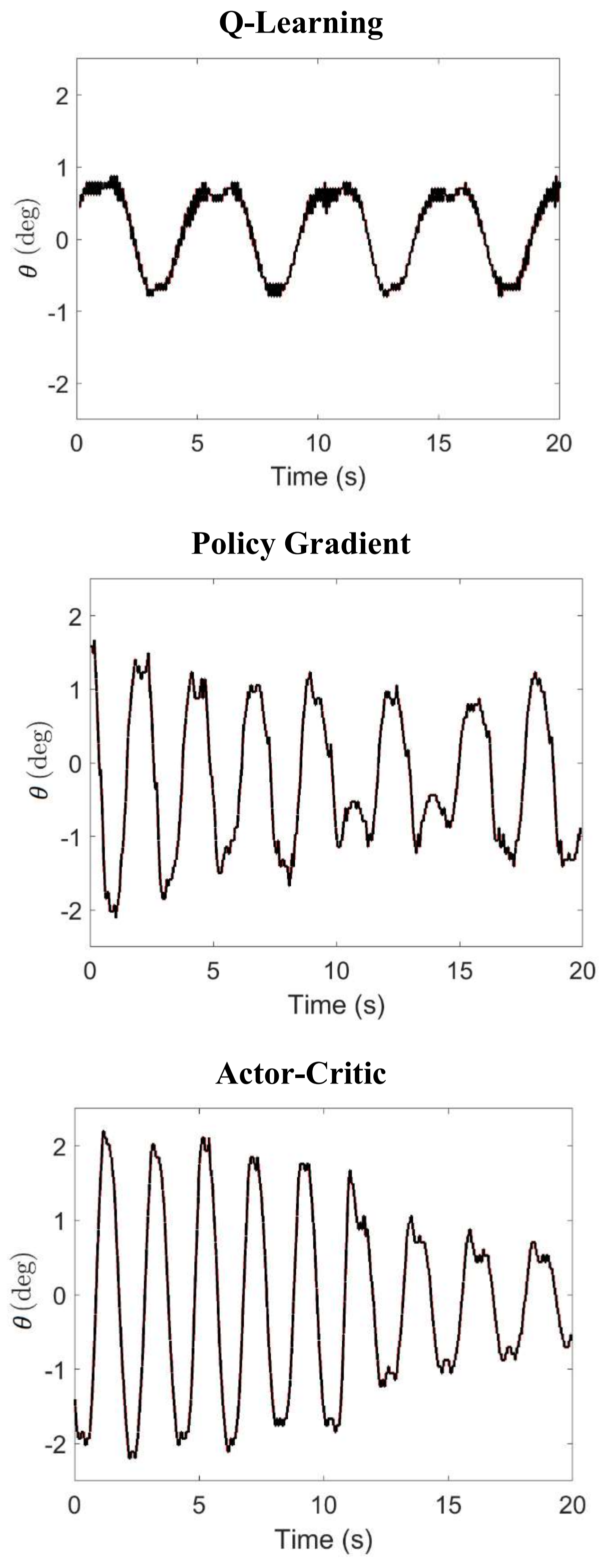


Fig. 4.3: State response of angle θ throughout a continuous 20-second period for each algorithm

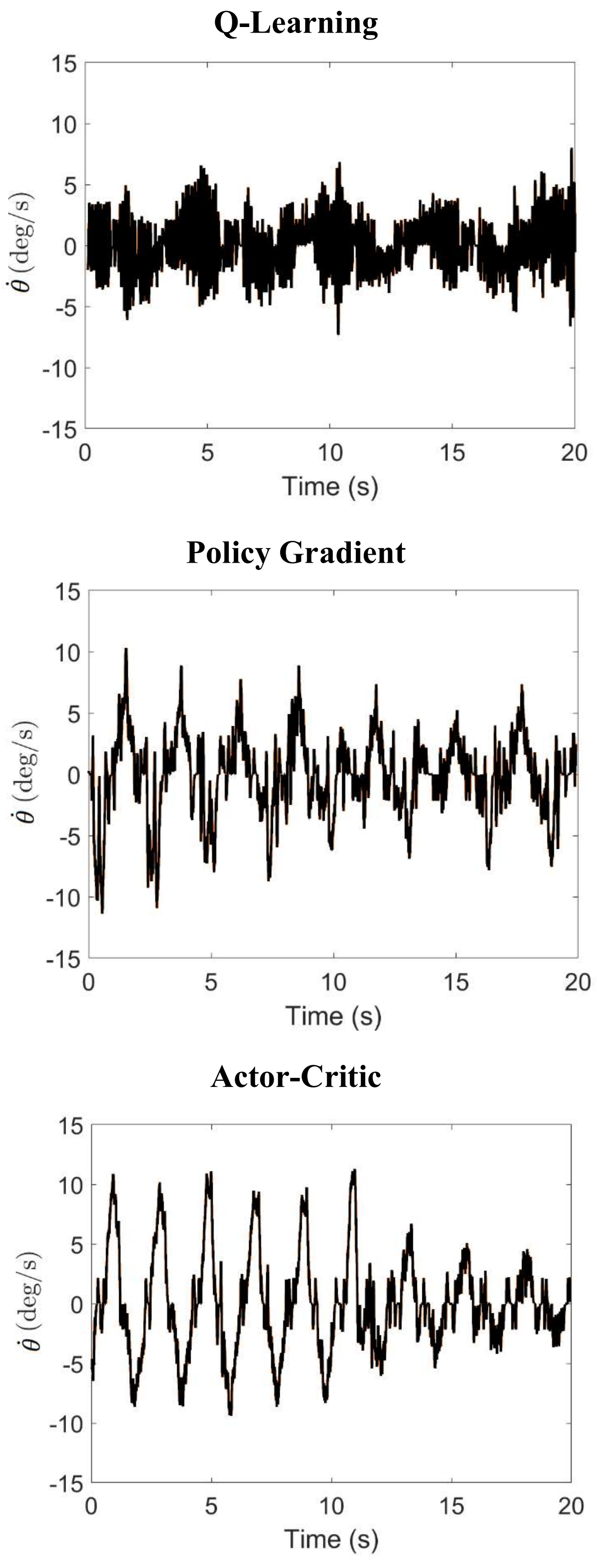


Fig. 4.4: State response of angular velocity $\dot{\theta}$ throughout a continuous 20-second period for each algorithm

4.2 Normalized Rewards

In each of the algorithm we used normalized discounted rewards to go (described in equation). The discount factor γ ensured that the rewards were finite. Depending on the value chosen it emphasis on current rewards over future rewards. But still using discount factor, the cumulative rewards might go to infinity if number of trials were large or magnitude of reward was large. In that case the training might crush. So, rewards were normalized to deal with these issues. Normalizing the rewards also beneficial for selecting actions. As it would encourage and discourage the actions equally. So, instead of encouraging all actions, encourage the good action and discourage the bad actions only.

In Actor-Critic algorithm, the critic is trying to estimate the normalized discounted rewards (an estimate of the value function $V_{\pi_\theta}(s)$) given only the state at each time step. Though it has many common parameters with the actor network which defines the distribution of each action (action is sampled randomly), finding a good estimation is pretty much impossible. Given two identical states in which the pole falls and the trial ends sometimes ago, the normalized discounted rewards-to-go associated will differ in each time step depending on the length of the trial. The critic has no knowledge of how long the trial is, so has no way to accurately estimate the value function. As a result, the critic loss will increase, meaning less weight will be given on minimizing the actor loss.

By only discounting but not normalizing the rewards, solves the issue completely. After discounting only identical states have identical rewards when they occur the same amount of time before a trial ends. Since the critic only has access to the state of the environment, it will predict the same value function given the same state, enabling it to learn the function much better. The critic network is not trying to estimate the state-action value function $Q(s, a)$. Rather, it is trying to estimate the expected return for a given state, which is the normalized discounted reward for the action. Value estimates for the same states should be the same, regardless of when the trial occurs. Instead of encouraging actions that lead to longer lifetimes like Policy Gradient, Actor-Critic encourages actions that perform better than expected, regardless of whether the action ends up being particularly good. Similarly, this discourages actions that are worse than expected, even if they do not cause the pole to fall. If the current action is worse than the average action for that state, take another action.

Despite not being able to learn value function, performance of the normalized discounted reward was far better than unnormalized discounted reward for larger values of discount factor γ. We saw that the unnormalized rewards only learned to balance less than half times and took more time to get there. Large values of γ might result in considerably larger discounted rewards towards the beginning of a long trials where even a small percentage error will result in a massive critic loss, which will give it far too much weight during optimization. Normalizing the discounted rewards keeps the values in a much smaller range, so the optimizer can focus primarily on updating the policy parameters instead.

Table 4.3: Training results for normalized and unnormalized Actor-Critic with different discount factor γ

Algorithm	**Discount Factor (γ)**	**Percentage of Success**	**Average no of Trials**
Actor-Critic Normalized	0.9	65	315
	0.99	90	275
	0.999	75	290
Actor-Critic Unnormalized	0.9	35	700
	0.99	55	565
	0.999	40	615

4.3 Discussions

As we've shown, every algorithm that has been put to the test can successfully train a virtual model to balance an inverted pendulum. But not every set of hyperparameters produced a model that consistently learned to balance. It took some trial and error for this optimization method to identify values that will train the most reliably.

Neural Network calculates the value function as part of the policy gradient or actor-critic algorithm, and a probabilistic approach is used to choose the optimum action for the current state using the state value function. The Policy Gradient or Actor-Critic approach performs poorly with some set of parameters even though it manages to stay inside the Cart-Pole state restrictions after the parameters are tweaked to obtain optimal policy. On the other hand, it can be shown from the data that the optimal policy is only reached through significant steady-state oscillations, which result in significant power losses for the entire system. The findings demonstrate that the pendulum angle range is quite wide and covers the whole permitted state space from -12° to +12° during the episode corresponding to the optimal policy. During the optimal episode, the linear position of the cart with respect to the pendulum's centre also varies greatly. This significant variance has an impact on the pendulum's swing and the cart's velocity, and it wastes a significant amount of energy when the pole is being swung up. Although there are significant errors in the pendulum's stabilisation, it is observed that the trajectories of the cart's position and the pendulum's angle are nearly periodic. This suggests that these algorithms can be optimised by choosing better hyper parameters which increases the likelihood of improved performance.

It's not always the ideal method to tackle a task just using reinforcement learning. Reinforcement Learning algorithms are excellent at learning new policies to control an unknown system, but there are no assurances that they will eventually converge to the best ones. Finding a more effective conventional solution is frequently possible using prior knowledge of the environment. Building a model of the system using conventional approach, then reinforcement learning, imitation learning, or transfer learning can be employed to achieve the best results. RL algorithms perform best in environments when it is difficult to deploy conventional techniques or when the environment is simply too vast.

It may now be concluded that although the conventional control techniques seem to yield better result as compared to RL based ones, the applicability and efficacy of the former are largely dependent on the available model of the plant. It is obvious that the better the model of the plant the better will be the result for such schemes. However, in most of the practical situations it often involves a great extent of mathematical complexity to develop an accurate model and further it is not always a straight-forward task to obtain a suitable controller for such complex plant models. RL based design, on the other hand, relies on a well-established algorithm to identify the suitable control action without necessitating the requirements of a plant model.

Chapter 5

Conclusion

5.1 Contribution of the Thesis

1. Data Driven Control system design has been studied. It’s capabilities and limitations and scope for application in non-linear control systems are discussed in brief.

2. Different Reinforcement Learning Algorithms are described in details and used to design controller for balancing Cart Inverted Pendulum system. The results by these RL based controllers are compared with each other to comment on their relative pros and cons.

3. Various conventional control schemes for balancing Cart Inverted Pendulum system is considered and the results are compared with the RL control schemes to justify the viability of the latter.

4. Limitations of balancing a Cart Inverted Pendulum using Reinforcement Learning are also highlighted.

5.2 Scope of Future Work

1. Further studies may be carried out to find new ways to overcome the limitations of balancing Cart Inverted Pendulum system using Reinforcement Learning and may even gain better stability and robustness.

2. New control scheme can be developed for balancing Inverted Pendulum by combing the Traditional Model Based Control with Reinforcement Learning Control. Therefore, the limitations of both the Traditional Control and the Reinforcement Learning Control can be removed.

3. The findings may lead to successful implementation to a real time CIPS.

4. It might be reasonable to extend the uses of Reinforcement Learning Control to other control system application as well where reinforcement learning is not typically applied, even if the focus of this dissertation has been on the problem of balancing an Cart Inverted Pendulum System.

REFERENCES

[1] R. E. Kalman, “A new approach to linear filtering and prediction problems”, transactions ASME, series D, Journal of Basic Engineering, Vol. 82, pp. 34–45, 1960.

[2] R. E. Kalman, “Contributions to the theory of optimal control”, Boletin de la Sociedad Matematica Mexicana, Vol. 5, pp. 102–119, 2009.

[3] L. Ljung, “System Identification: Theory for the User”, Englewood Cliffs, NJ, USA: Prentice-Hall, 1987.

[4] M. Verhaegen and V. Verdult, “Filtering and System Identification: A Least Squares Approach”, Cambridge, U.K.: Cambridge Univ. Press, 2007.

[5] R. S. Sutton and A. G. Barto, “Reinforcement Learning: An Introduction”, MIT Press: Cambridge, MA, USA, Vol. 1, 1998.

[6] R. Bellman, “Dynamic Programming”, Princeton University Press: Princeton, NJ, USA, Vol. 95, 1957.

[7] R. Bellman, “A Markovian decision process”, J. Math. Mech., Vol. 6, pp. 679–684, 1957.

[8] J. Hoskins and D. Himmelblau, “Process control via artificial neural networks and reinforcement learning”, Comput. Chem. Eng., Vol.16, pp. 241–251, 1992.

[9] G. Hinton, N. Srivastava and K. Swersky, “Neural networks for machine learning lecture 6a overview of mini-batch gradient descent”, 14, 2, 2012.

[10] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors”, arXiv:1207.0580, 2012.

[11] M. Wulfmeier, I. Posner, and P. Abbeel, “Mutual alignment transfer learning”, Proceedings of the Conference on Robot Learning (PMLR), Mountain View, CA, USA, pp. 281–290, 2017.

[12] X. B. Peng, M. Andrychowicz, W. Zaremba and P. Abbeel, “Sim-to-real transfer of robotic control with dynamics randomization”, Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA), Brisbane, Australia, pp. 3803–3810, 2018

[13] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton et al., “Mastering the game of go without human knowledge”, Nature, Vol. 550, pp. 354–359, 2017.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski et al, “Human-level control through deep reinforcement learning” Nature, Vol. 518, pp. 529–533, 2015.

[15] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev et al. “Grandmaster level in StarCraft II using multi-agent reinforcement learning”, Nature, Vol. 575, pp. 350–354, 2019.

[16] R. Nian, J. Liu and B. Huang, “A review on reinforcement learning: Introduction and applications in industrial process control” Comput. Chem. Eng., Vol. 139, pp. 106886, 2020.

[17] L. Buşoniu, T. D. Bruin, D. Tolić, J. Kober and I. Palunko, “Reinforcement learning for control: Performance, stability, and deep approximators”, Annu. Rev. Control, Vol. 46, pp. 8–28, 2018.

[18] D. Bates, “Virtual Reinforcement Learning for Balancing an Inverted Pendulum in Real Time”, PhD Thesis submitted to the Graduate Faculty of North Carolina State University Raleigh, North Carolina, 2021.

[19] T. Abut and S. Soygader, “Two-loop controller design and implementations for an inverted pendulum system with optimal self-adaptive fuzzy-proportional–integral–derivative control”, Transactions of the Institute of Measurement and Control, 2021.

[20] E. Davison and I. F. Theory of Automatic Control Technical Committee, “Bench-mark Problems for Control System Design: Report of the IFAC Theory Committee”, International Federation of Automatic Control, 1990.

[21] R. Florian, “Correct equations for the dynamics of the cart-pole system", 2005.

[22] E. Kennedy, “Swing-up and Stabilization of a Single Inverted Pendulum: Real-Time Implementation". PhD thesis, North Carolina State University, 2015.

[23] B. Abeysekera and W. K. Wanniarachchi, “Modelling and Implementation of PID Control for Balancing of an Inverted Pendulum", pp. 43-53, 2018.

[24] J. L. C. Miranda, “Application of Kalman Filtering and PID Control for Direct Inverted Pendulum Control", MA thesis, California State University, Chico, 2009.

[25] A. Krizhevsky et al., “ImageNet Classification with Deep Convolutional Neural Networks", Advances in Neural Information Processing Systems Ed. by F. Pereira et al., Vol. 25. Curran Associates, Inc., 2012.

[26] S. Ozana et al. “Design and implementation of LQR controller for inverted pendulum by use of REX control system", 12th International Conference on Control, Automation and Systems, pp. 343-347, 2012.

[27] P. Dang and L. Frank,“Controller for swing-up and balance of single inverted pendulum using SDRE-based solution", 31st Annual Conference of IEEE Industrial Electronics Society, IECON, 2005.

[28] V. A. Arya, R. B. Aswin and A. E. George, “Modified Model Reference Adaptive Control for the Stabilization of Cart Inverted Pendulum System”, International Research Journal of Engineering and Technology (IRJET), Vol. 5, 2018.

[29] A. Baris and R. Coban, “Linear Quadratic Optimal Control of an Inverted Pendulum on a Cart using Artificial Bee Colony Algorithm: An Experimental Study”, Çukurova University Journal of the Faculty of Engineering and Architecture, Vol. 32(2), pp. 109-123, 2017.

[30] A. G. Barto et al. “Neuronlike adaptive elements that can solve difficult learning control problems". IEEE Transactions on Systems, Man, and Cybernetics SMC Vol. 13.5, pp. 834-846, 1983.

[31] T. P. Lillicrap et al., “Continuous control with deep reinforcement learning”, Cs. Lg., arXiv: 1509.02971, 2015.

[32] J. Schulman et al., “Trust Region Policy Optimization", CoRR, arXiv: 1502.05477, 2015.

[33] J. Schulman et al., “Proximal policy optimization algorithms", CoRR, arXiv: 1707.06347, 2017.

[34] Y. Duan et al., ‘Benchmarking Deep Reinforcement Learning for Continuous Control", ICML, arXiv: 1604.06778, 2016

[35] A. G. Ivakhnenko and V. G. Lapa, “Cybernetic Predicting Devices”, JPRS 37, 803. CCM Information Corporation, 1965.

[36] F. Rosenblatt, “The Perceptron, a Perceiving and Recognizing Automaton Project” Para. Report: Cornell Aeronautical Laboratory. Cornell Aeronautical Laboratory, 1957.

[37] A. E. Bryson, “A gradient method for optimizing multi-stage allocation processes", Proceedings of the Harvard University Symposium on digital computers and their applications, 1961.

[38] H. J. Kelley, “Gradient Theory of Optimal Flight Paths". ARS Journal, pp. 947-954, 1960.

[39] S. M. McKinney et al, “International evaluation of an AI system for breast cancer screening", Nature 577.7788, pp. 89-94, 2020.

[40] C. Payne, MuseNet, url: https://openai.com/blog/musenet/, 2019.

[41] G. Branwen, GPT-3 Creative Fiction, https://www.gwern.net/GPT-3, 2020.

[42] T. B. Brown et al. “Language Models are Few-Shot Learners”, arXiv: 2005.14165 [cs.CL], 2020.

[43] L. A. Gatys et al., “A Neural Algorithm of Artistic Style”, arXiv: 1508.06576, [cs.CV], 2015.

[44] OpenAI,S OpenAI Five, https://blog.openai.com/openai-five/, 2018.

[45] D. Silver et al., “Mastering the game of Go with deep neural networks and tree search", Nature, Vol. 529, pp. 484-503, 2016.

[46] R.S. Sutton, "Learning to predict by the methods of temporal differences.", Machine learning, Vol. 3.1, pp. 9-44, 1988.

[47] URL: “https://ctms.engin.umich.edu/CTMS/index.php?example=InvertedPendulum&section=SystemModeling”

[48] R. K. Thomas, "Getting in tune with Ziegler-Nichols," PhD, in the Academic Viewpoint column, Control Engineering magazine, pp. 28, 2007.

[49] C. De Persis and P. Tesi, “Formulas for Data-Driven Control: Stabilization, Optimality, and Robustness”, IEEE Transactions on Automatic Control, Vol. 65, 2020.

[50] Z. S. Hou, and Z. Wang, “From model-based control to data-driven control: Survey, classification and perspective”, Inf. Sci., Vol. 235, pp. 3–35, 2013.

[51] A. Bazanella, L. Campestrini, and D. Eckhard, “Data-Driven Controller Design: The H2 Approach”, Springer, 2011.