**MS-THESIS-TITLE**

**A THESIS**

*submitted in partial fulfillment of the requirements*

*for the award of the dual degree of*

**Bachelor of Science - Master of Science**

*in*

**DISCIPLINE**

*by*

**STUDENT-NAME**

**(12345)**



**DEPARTMENT OF DISCIPLINE**

**INDIAN INSTITUTE OF SCIENCE EDUCATION AND**

**RESEARCH BHOPAL**

**BHOPAL – 462 066**

**April XXXX**

**CERTIFICATE**

***TO BE PRINTED ON THE OFFICIAL LETTER HEAD***

This is to certify that Prashant Shukla, BS-MS (Dual Degree) student in Department of Physics, has completed bonafide work on the thesis entitled **‘<MS-Thesis-Title>’** under my supervision and guidance.

|  |  |
| --- | --- |
| April XXXX  IISER Bhopal | <Signature>  <Prof./Dr. Advisor-Name> |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Committee Member |  | Signature |  | Date |
|  |  |  |  |  |
|  |  |  |  |  |

**ACADEMIC INTEGRITY AND**

**COPYRIGHT DISCLAIMER**

I hereby declare that this project report is my own work and due acknowledgement has been made wherever the work described is based on the findings of other investigators. This report has not been accepted for the award of any other degree or diploma at IISER Bhopal or any other educational institution. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission.

I certify that all copyrighted material incorporated into this document is in compliance with the Indian Copyright (Amendment) Act (2012) and that I have received written permission from the copyright owners for my use of their work, which is beyond the scope of the law. I agree to indemnify and safeguard IISER Bhopal from any claims that may arise from any copyright violation.

|  |  |
| --- | --- |
| April XXXX  IISER Bhopal | <Signature>  <Student Name> |

**ACKNOWLEDGEMENT**

Acknowledgement begins with an indented paragraph. Acknowledgement is here. Acknowledgement is here. Acknowledgement is here.

<Student-Name>

**ABSTRACT**

The Pursuit Evasion Differential Games poses a challenging problem defined by certain kinematic equations, with applications in many fields as economics, sports, robotics, and air combats. In such dynamic problems pursuit aims to capture the evader, whereas the evader has an adverse goal to avoid capturing. Considering the complexity of the problem we propose Reinforcement Learning as a promising approach for obtaining optimal solutions for both pursuers and evader. For single pursuer evader problems, we have used single-agent Reinforcement Learning Algorithms like DDPG and DQN. The problem becomes more challenging for multiple pursuer single evader situations where pursuer needs to cooperate with each other to find the optimal strategy to capture the evader. We propose the Multi-agent Reinforcement Learning Algorithms to be approached to tackle the multiple pursuer single evader problems. For our work, we are proposing multi-agent RL algorithms like MADDPG and neighborhood Q-learning. Results obtained for single-agent cases are very promising and show a very efficient pursuer strategy for capturing the evader.

**LIST OF SYMBOLS OR**

**ABBREVIATIONS**

|  |  |
| --- | --- |
| α | The first letter |
| ω | The last letter |
| ddcbd58d5a228364ab9da20f5cd59495.png | The Riemann Zeta function |

**LIST OF FIGURES**

|  |  |
| --- | --- |
| Not a function | 2 |

**LIST OF TABLES**

|  |  |
| --- | --- |
| Nonlinear Model Results | 5 |

**CONTENTS**

**Certificate** …………………………………………………………… **i  
Academic Integrity and Copyright Disclaimer** …………………… **ii**

**Acknowledgement** …………………………………………………... **iii**

**Abstract** ……………………………………………………………… **iv**

**List of Symbols** ………………………………………………………. **v**

**List of Figures** ……………………………………………………….. **vi**

**List of Tables** ………………………………………………………… **vii**

1. **Introduction** ……………………………………………….……… 1
   1. Basic Introduction ………………………………...………….. 1
   2. Technical Introduction ……………………………………….. 1
   3. Notation and Definitions ………………...…………………… 3
2. **Method** …………………………………………………………….5
   1. Tools used ………………………………….………………… 5
   2. Method behind the Madness …………………………………. 6
3. **Method** ……………………………………………………………5

3.1 Initial Data ………………………………………………........ 5

3.2 Processing the Data…………………………………..………. 6

1. **Conclusions** ……………………………………………………….10
   1. Observations ……………………………….………………… 10
   2. Applications ………………….………………………………. 11

**Appendices** ……………………………………………………………12

I Basic Definitions ……………………...…………………........ 13

II Additional Theorems ….……………………………..………. 13

**Bibliography** …………………………………………………………14

**1. INTRODUCTION**

**1.1 Pursuit-Evasion**

The mathematical formulation of pursuit-evasion has applications in many other fields such as surveillance, navigation, economics, analysis of biological behaviors, and conflict combat operations. Pursuit Evasion problems represent the family of problems where one group attempts to capture the other group. The aim of working on such problems is to find out the strategies for autonomous agents, in this case, pursuer and evader, so that they can work against each other in an optimal way.

Generally, it is assumed that opponents will behave in particular ways or some policy for an opponent is presumed for working on such problems. In pursuit-evasion problems pursuer aims to capture the evader whereas the evader tries to avoid getting captured. The pursuit-Evasion problem is an example of a zero-sum game where cost is the time consumed in capturing.

Certain kinematics equations were defined by the Issacs, which provides a certain surface. States of the system are constrained on these surfaces only. These equations are called **Differential Equations.** These Differential Equations are used to state problems in other research fields as well, like economics, robotics, and air combats, etc.

There are different types of engagements between pursuer and evader with the different number of pursuers and evaders.

***A. Single Pursuer and Single Evader***

Single Pursuer and single evader scenarios consist of two agents, where one is a pursuer and another one is an evader. For such scenarios, we find optimal solutions for performance functional. Pursuer and evader have opposite goals for this performance functional during their engagement. One will try to maximize the performance functional whereas the other one would try to minimize the performance functional. Because of such relation with performance functional these problems are called min-max problems. There are different possible scenarios possible for single pursuer and single evader problems.

1. Homicidal Chauffeur Game: These problems consider slow evader with high manueverability and fast pursuer with low maneuverability. In such problems, there could be 2 possible scenarios: i) Capturing of an evader is guaranteed, ii) There could be possible evader strategies using which evader can always escape from getting captured. In the first scenario, the optimal strategy for the pursuer will be to minimize the capturing time and for the evader, it will be to maximize the capturing time. Whereas for the second scenario optimal strategy for the pursuer will be to always capture the evader and for the evader, it will be to find a strategy so that it can always escape the pursuer.
2. Two Cars Differential Game: This problem was investigated by Meier, where two cars had the same minimum turn radius, while the speed of the pursuer was less than the speed of the evader, evader was said to get captured if it comes inside a certain range of pursuer[PE].
3. Pursuit Evasion in Constrained Environment: These problems were studied to understand the more real-life situations. Agents states were constrained and certain boundary conditions were introduced in this kind of problem. Agents were restricted to particular regions and extra obstacles were introduced in the environments.
4. Pursuit Evasion in Aerial Engagements: For 3-D pursuit-evasion situations like for missile-aircraft engagement, solutions were proposed by Shinar and Gutman[PE]. Results using variational methods were also obtained for more realistic pursuit-evasion scenarios were obtained by Shinar [PE].
5. Other Works: Leitmann has shown work on a pursuit-evasion differential game using Variational Technique where he used miss distance as the cost functional [PE]. Quincampoix has shown work on a differential game where a faster Lion chases a man where both man and lion can change the velocity as well as direction. He showed certain regions of escape and capture in his work.

***B. N Pursuer and Single Evader***

The problem of two pursuers and a single evader was investigated by Hagedorn and Breakwell where the evader needed to pass from the middle of two pursuers [PE]. The problem of capturing slow evader using faster pursuers has been discussed in [PE]. In [PE] a problem of N-pursuers and 1-evader was discussed where evader was guaranteed to escape. Fuzzy logic methods have also been discussed in [PE] for multi-pursuer and single evader.

**1.2 Reinforcement Learning**

Reinforcement Learning(RL) is a sub-field of machine learning in which an agent, which could be any computer algorithm, interacts with the environment/surroundings and learns how it can perform the desired task by selecting certain actions from a set of Action Space. Agents in RL are guided or supervised by Reward Function in a similar fashion, as in supervised learning outputs/labels are used to train an algorithm. The agent takes certain actions and as result receives some reward and next state as feedback from the environment. Subsequently receiving a reward and a new state in consequence of taking certain actions makes Reinforcement Learning interesting. Actions taken at a particular time step can have a very long-term effect as it affects what will be state in the next time step and what would be the reward in future time steps.

Supervised learning is also a subfield of Machine Learning. Supervised Learning algorithms require a training data set for the purpose of learning a generalized function. The training dataset must be labelled. Once the algorithms learn to make a prediction on training data it can make predictions on the input data it has never seen before. But supervised learning doesn’t suffice in the field of Artificial Intelligence. For the situations where agents need to interact with a completely new environment to make a prediction function that can predict what actions to take in particular situations supervised learning may not work. In such situations, it is not possible to get a training set, which should be a set of different possible states as an input and actions need to be executed as output, because it is impractical to get this data unless the agent interacts with the environment.

Other than Supervised Learning algorithms there are Unsupervised Learning algorithms that don’t rely on labels of the input data. Unsupervised learning algorithms tend to group the input data into certain groups by recognizing certain hidden patterns or structures inside the data. Although Reinforcement Learning doesn’t require any labels still it is not unsupervised learning. The goal of reinforcement learning is to maximize the reward for the whole episode. Unsupervised learning can find certain hidden structures from the experience of agent but that is not the final goal of reinforcement learning. Reinforcement Learning is hence considered as third paradigm of Machine Learning, whereas supervised learning and unsupervised learning are considered as other 2 paradigms.

One of the major challenge in reinforcement learning for the agent is of exploration and exploitation. Let us say agent can take 5 different possible actions, but agent keeps on taking only 3 out of those 5 actions which keeps on giving it decently good reward at each time step. This behaviour of agent is called **exploitation** as it i using its knowledge of getting good reward with those 3 actions. But it may be possible that other 2 actions may give it more cumulative reward in future rather than previous 3 outcomes. In order to explore whether agent can have more cumulative reward in future or not it need to explore those 2 actions as well. This kind of behaviour is called **exploration**. It is very important in reinforcement learning to trade-off between exploration and exploitation in order to achieve better learning. It is difficult problem as exploitation and exploration can never be achieved unless agents fails in performing the task. If the task is stochastic, then agent must try all different possible actions to get an experience so that it can make better estimate of how much reward it can expect on performing certain action.

Reinforcement Learning is highly interrelated with other fields of science. Reinforcement Learning has significantly influenced the field of study of psychology and neuroscience. In fact it is generally considered that reinforcement learning best represents the learning process of living organisms and many of the reinforcement learning algorithms are motivated from biological learning.

**1.3 Elements in Reinforcement Learning**

Reinforcement Learning has 4 major elements:

**1. Policy:** Policy can be understood as a function that maps the state of the environment into actions. Policy determines what action agent will take in particular state. Policy alone can determine the behaviour of the agent. Policies can be deterministic or stochastic. Stochastic Policies provides probabilities of taking different action in given state whereas deterministic policies provide one particular action that an agent will take.

**2. Reward:** Once the action is executed by an agent environment sends back feedback to the agent, and rewards the agent with some value. This value determines how good or how bad the new state is so that agent can learn from this feedback and improve its actions in the future. Agents aim is to maximize the reward for the complete play. Agent receiving reward is very similar to experience of pleasure and pain in biological systems. Reward received by an agent helps the agent in agent any modify its policies for future. Similar to policies reward can be deterministic or stochastic function for state and action.

**3. Value Function:** Value function is a function that maps the state or state-action pair to a particular value. This value determines how good or bad the state or state-action pair is in the long term. In short value function gives the value that determines how much cumulative reward an agent can expect in long time being in particular state or state-action pair. So a state or state-action pair may receive small reward but can have larger value associated with it. We can make analogy with here with human experience, human may experience pleasure in particular situation but in long run it might could make human displeased. There are two possible types of value functions: **s-value and q-value.** S-value tells about how much cumulative reward an agent can expect being in particular state whereas q-value tells about how much cumulative reward an agent can expect by taking certain action being in a particular state.

**4. Models:** Model is a function that predicts the possible next state and next reward after taking particular action in current state. Model helps agent in improving policy without interacting with environment directly. Two approaches used for Reinforcement Learning problems are: **model-free methods** and **mode-based methods.** Model-free methods directly interacts with environment for learning and do not use models, whereas mode-based methods uses model for learning.

Terminologies defined in Reinforcement Learning are:

(i) State[s]: It determines the current situation of environment and agent, like Pressure, Temperature and Volume determines the state of any thermodynamic system or Position and Velocity of football players in Football ground. The state of the system evolves with time. An n-dim space which contains all possible set of points representing some state s is called state space S with dimension 'n', where n represents the number of features required to represent the state of space. State-space could be discrete or continuous in nature.

(ii) Action[a]: To interact with environment algorithm of agent generates some action and when action is executed we say agent has interacted with the environment. For example, while driving a car if someone is crossing a road then the action will be to apply brakes to avoid an accident, our agent(in this case it is the driver) will generate and execute the action which is applying brakes. For executing an action agent needs to observe the current state and accordingly take the action. So we can understand it as a state(s) is mapped to some action(a). The action generated by the agent interacts with the environment and contributes to the dynamics of the system. As a result after the execution of the action, our system reaches some new state(s'). So in our car example in the new state velocity of the car would become zero. Similar to State-space we define Action Space(A) as a set of all possible actions that an agent can generate. Action Space (A) could be m-dimensional depending on the number of different actions generated by the agent, for e.g. if the driver applies a brake and simultaneously changes the direction of the car then action space will be 2-d.

(iii) Reward[r]: Once the action is executed by an agent environment sends back feedback to the agent, and rewards the agent with some value. This value determines how good or how bad the new state is so that agent can learn from this feedback and improve its actions in the future. For example, in a car driving example, if the driver would have accelerated the car which would have lead to an accident, in that case, the driver would have been rewarded negatively, on the other hand on applying brakes driver would have been rewarded positively. The reward is always a scalar quantity and it can be understood as s and a being mapped to scalar quantity r.

**Definition 1.1.** Definition is here.

Definition is here.

Definition is here.

**Remark 1.2.** Remark is here.

Remark is here.

Remark is here.

**Example 1.3.** Example is here.

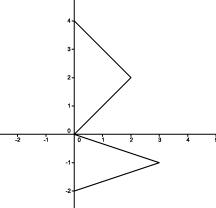
Example is here.

Example is here.

**Theorem 1.4.** *Theorem is here.*

*Theorem is here.*

*Theorem is here.*



**Fig. 1.1:** Not a function

*Proof.* Proof is here.

Proof is here.

Proof is here.

**Theorem 1.5.** *[1, Theorem 6.2] Theorem is here.*

*Theorem is here.*

*Theorem is here.*

*Proof.* Proof is here.

Proof is here.

Proof is here.

**Theorem 1.6.** *Theorem is here.*

*Theorem is here.*

*Theorem is here.*

*Proof.* Proof is here.

Proof is here.

Proof is here.

**Theorem 1.7.** *Theorem is here.*

*Theorem is here.*

*Theorem is here.*

*Proof.* Proof is here.

Proof is here.

Proof is here.

**Theorem 1.8.** *Theorem is here.*

*Theorem is here.*

*Theorem is here.*

*Proof.* Proof is here.

Proof is here.

Proof is here.

**Theorem 1.9.** *Theorem is here.*

*Theorem is here.*

*Theorem is here.*

*Proof.* Proof is here.

Proof is here.

Proof is here.

**1.3 Notation and Definitions**

**Definition 1.10.** Definition is here.

Definition is here.

Definition is here.

**Remark 1.11.** Remark is here.

Remark is here.

Remark is here.

**Example 1.12.** Example is here.

Example is here.

Example is here.

**Theorem 1.13.** *Theorem is here.*

*Theorem is here.*

*Theorem is here.*

*Proof.* Proof is here.

Proof is here.

Proof is here.

**Theorem 1.14.** *Theorem is here.*

*Theorem is here.*

*Theorem is here.*

*Proof.* Proof is here.

Proof is here.

Proof is here.

This is a reference to Example 1.3.

**Theorem 1.15.** *Theorem is here.*

*Theorem is here.*

*Theorem is here.*

*Proof.* Proof is here.

Proof is here.

Proof is here.

**2. METHOD**

**2.1 Tools Used**

**Definition 2.1.** Definition is here.

Definition is here.

Definition is here.

**Remark 2.2.** Remark is here.

Remark is here.

Remark is here.

**Example 2.3.** Example is here.

Example is here.

Example is here.

**Theorem 2.4.** *Theorem is here.*

*Theorem is here.*

*Theorem is here.*

**Tab. 2.1:** Nonlinear Models Results

|  |  |  |  |
| --- | --- | --- | --- |
| Case | Method #1 | Method #2 | Method #3 |
| 1 | 50 | 837 | 970 |
| 2 | 47 | 877 | 230 |
| 3 | 31 | 25 | 415 |
| 4 | 35 | 144 | 2356 |
| 5 | 45 | 300 | 556 |

*Proof.* Proof is here.

Proof is here.

Proof is here.

**Theorem 2.5.** *Theorem is here.*

*Theorem is here.*

*Theorem is here.*

*Proof.* Proof is here.

Proof is here.

Proof is here.

**2.2 Method behind the Madness**

**Theorem 2.6.** (Author-Name). *Theorem is here.*

*Theorem is here.*

*Theorem is here.*

*Proof.* Proof is here.

Proof is here.

Proof is here.

**Definition 2.7.** Definition is here.

Definition is here.

Definition is here.

**Remark 2.8.** Remark is here.

Remark is here.

Remark is here.

**Example 2.9.** Example is here.

Example is here.

Example is here.

**Theorem 2.10.** *Theorem is here.*

*Theorem is here.*

*Theorem is here.*

*Proof.* Proof is here.

Proof is here.

Proof is here.

**Theorem 2.11.** *Theorem is here.*

*Theorem is here.*

*Theorem is here.*

*Proof.* Proof is here.

Proof is here.

Proof is here.

**3. RESULTS**

**3.1 Initial Data**

**Definition 3.1.** Definition is here.

Definition is here.

Definition is here.

**Remark 3.2.** Remark is here.

Remark is here.

Remark is here.

**Example 3.3.** Example is here.

Example is here.

Example is here.

**Theorem 3.4.** *Theorem is here.*

*Theorem is here.*

*Theorem is here.*

*Proof.* Proof is here.

Proof is here.

Proof is here.

**Theorem 3.5.** *Theorem is here.*

*Theorem is here.*

*Theorem is here.*

*Proof.* Proof is here.

Proof is here.

Proof is here.

**3.2 Processing the Data**

**Definition 3.6.** Definition is here.

Definition is here.

Definition is here.

**Remark 3.7.** Remark is here.

Remark is here.

Remark is here.

**Example 3.8.** Example is here.

Example is here.

Example is here.

**Theorem 3.9.** *Theorem is here.*

*Theorem is here.*

*Theorem is here.*

*Proof.* Proof is here.

Proof is here.

Proof is here.

**Theorem 3.10.** *Theorem is here.*

*Theorem is here.*

*Theorem is here.*

*Proof.* Proof is here.

Proof is here.

Proof is here.

**4. CONCLUSIONS**

**4.1 Observations**

**Definition 4.1.** Definition is here.

Definition is here.

Definition is here.

**Remark 4.2.** Remark is here.

Remark is here.

Remark is here.

**Example 4.3.** Example is here.

Example is here.

Example is here.

**Theorem 4.4.** *Theorem is here.*

*Theorem is here.*

*Theorem is here.*

*Proof.* Proof is here.

Proof is here.

Proof is here.

**Theorem 4.5.** *Theorem is here.*

*Theorem is here.*

*Theorem is here.*

*Proof.* Proof is here.

Proof is here.

Proof is here.

**4.2 Applications**

**Definition 4.6.** Definition is here.

Definition is here.

Definition is here.

**Remark 4.7.** Remark is here.

Remark is here.

Remark is here.

**Example 4.8.** Example is here.

Example is here.

Example is here.

**Theorem 4.9.** *Theorem is here.*

*Theorem is here.*

*Theorem is here.*

*Proof.* Proof is here.

Proof is here.

Proof is here.

**Theorem 4.10.** *Theorem is here.*

*Theorem is here.*

*Theorem is here.*

*Proof.* Proof is here.

Proof is here.

Proof is here.

**APPENDICES**

**I Basic Definitions**

This is the first section of the appendix.

**II Additional Theorems**

This is the second section of the appendix.

**BIBLIOGRAPHY**

[1] George A. Ellott and Dorte Olesen. A simple proof of the Dauns-Hofmann theorem. *Math. Scand., 34:231-234, 1974.*

[2] Raymond M. Smullyan. *Gödel's incompleteness theorems*, volume 19 of *Oxford Logic Guides*. The Clarendon Press Oxford University Press, New York, 1992.