Design of deep Q-networks for transfer time prediction of spacecraft orbit-raising

by

Ali Hassaan Mughal

B.S., National University of Sciences and Technology, Pakistan, 2020

A THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Computer Science
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2022

Approved by:

Major Professor
Dr. Arslan Munir

Copyright

Abstract

Recently, there has been a surge in use of electric propulsion to transfer satellites to the geostationary Earth orbit (GEO). Traditionally, the transfer times to reach GEO using all electric propulsion are obtained by solving challenging trajectory optimization problems, whose solution rely on numerical schemes that are not only computationally intensive, but also lack automated implementation capabilities. This naturally creates a hindrance to their incorporation within Deep Reinforcement Learning (DRL) framework, which combines Reinforcement Learning (RL) and Deep Learning to solve trajectory planning problems in near real-time. The operation of DRL, as typically used in trajectory planning, relies on a Q-value. In the electric orbit-raising problem under consideration in this thesis, this Q-Value requires computation of transfer time in near real-time to have practical DRL training times. In our work, this Q-value is predicted by a set of deep neural networks (DNNs) instead of solving traditional optimization problems. This thesis aims at designing a set of DNNs that can serve as a Q-value(transfer time) predictor for different orbit-raising mission scenarios. To this end, we investigate different architectures for DNNs to determine the optimal DNN configuration that can predict the transfer time for each of the mission scenarios. Experimental results indicate that our designed DNNs can predict the transfer time for different scenarios with an accuracy of over 99%. We also compare the results from our designed DNNs with the contemporary Machine Learning (ML) algorithms, such as Support Vector Machine (SVM), Random Forest (RF), Decision Tree (DT) for regression. Experimental results indicate that our best-performing DNNs can provide an improvement in mean error of transfer time prediction of up to 14.05× for non-planar transfers and up to 254× for planar transfers.

Table of Contents

List of Figures

List of Tables

Acknowledgments

I would like to begin by thanking the Almighty God for giving me the knowledge, patience, perseverance, and critical thinking that helped me succeed in this path. I would then like to thank my M.Sc. advisor, Dr. Arslan Munir, for accepting me as his M.Sc. student despite my lack of research experience. One thing that I am grateful for is the creative freedom Dr. Munir gave me in choosing my own research direction and working on the problems that I was interested in. Our technical discussions always helped me in formulating some really interesting ideas that I was able to incorporate in my research. I would also like to thank all my committee members for giving me invaluable suggestions during my technical presentations. It helped me in taking a more focused approach towards the research problem. I would also like to thank them for taking the time to review my thesis and giving helpful feedback. I would like to thank Azeem Shami, Abdullah Alaklabi, Khalid Alsalmi and to the current and past members of the Intelligent Systems, Computer Architecture, Analytics, and Security (ISCAAS) lab where I worked during my M.Sc. I am also grateful to the Pakistani community here in Manhattan, Kansas who made K-State feel like a home away from home. This journey would not have been possible without the support of my family and friends in Pakistan and the USA. Last but not the least, Raja Hasnain Anwar, Kamran Janjua, Shahzaib Waseem and my elder brother Muhammad Usman for being a push and a view towards life.

This work was supported in part by the National Aeronautics and Space Administration (NASA), through the grant NASA-20-2020EPSCoR-0017. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NASA.

Dedication

To my Mom and Dad who always believed in me and provided me the comfort and the resources to pursue my dreams. To my elder siblings Umar and Usman for providing me strong support and for continuously pushing me for success even during times of low motivation. To my little brother Abubakar whose passion, makes me more passionate to follow my dreams and to my little sister Noor Fatima and my little nephew Aariz Ali, who when they smile; bring me joy and their presence; I miss every day during my studies abroad...

Chapter 1

Introduction[1]

A spacecraft pursuing a mission in the geostationary Earth orbit (GEO), a circular equatorial orbit of altitude 35,786 km, is first inserted into an intermediate transfer orbit via a launch vehicle. The next stage is to raise the orbit of the spacecraft from the transfer orbit to the GEO, which is a popular destination for telecommunication satellites. Traditional spacecraft use chemical propulsion systems for these orbit transfers. However, in recent years, the electric propulsion systems are replacing these chemical propulsion systems, to take advantage of significant propellant savings during electric orbit-raising. This advantage comes at the cost of such propulsion systems producing a low magnitude of thrust, making the transfer really long (of the order of several months); a factor that makes the underlying trajectory optimization for electric orbit-raising a challenging one.

The determination of electric orbit-raising trajectory requires the solution of a long time scale, multi-phase, nonlinear, non-convex optimal control problem that is challenging to solve. The trajectory to be computed is over a long time scale because of the low magnitude of the propulsion system thrust, resulting in a slow spiraling transfer around the Earth. The problem is multi-phase because the spacecraft passes through multiple eclipses during its transfer, and the dynamics are different in Sun-lit parts of the trajectory and in the shadow of the Earth. The absence of the Sun during eclipses prohibit the spacecraft from thrusting

[1]This work is submitted in the Elsevier Journal of Intelligent Systems with Applications with the same title as this thesis and authors as A. H. Mughal, P. Chadalavada, A. Munir, A. Dutta, and M.A. Qureshi.

because electric propulsion systems derive power from the solar arrays. The problem is non-linear owing to the dynamics being primarily governed by the inverse-square gravitational force. Finally, the problem is non-convex because of the nature of the objective functions (transfer time or fuel expenditure) that are of interest to mission designers. Nevertheless, this challenging problem has been studied in the astrodynamics literature for many decades, and a number of methods have been developed. These methods can be largely classified into direct and indirect optimization techniques, depending on whether calculus of variations is used to determine the necessary conditions of optimality (direct methods bypass this step). Numerical schemes based on these approaches need good quality initial guesses for demonstrating good convergence, and determining the good initial guesses depend on a human expert as well. Owing to a lot of computations involved in the traditional optimization problems for the electric orbit-raising problem, newer techniques that rely on deep reinforcement learning (DRL) have gained attention in recent years.

1.1 DRL in Guidance of Spacecrafts

DRL has emerged as a promising approach for optimization of guidance-based systems in recent years[1–4]. DRL has shown significant improvements in ever-challenging domains despite several uncertainties in the environment. Markov decision processes (MDPs) are one of the most common underlying mathematical frameworks leveraged by DRL. MDPs comprise an agent, an environment, and a given set of goals. Continuous-state MDPs are required for solving many problems, but present various challenges, in particular, state space explosion. A framework for faster computation of continuous-state MDPs by discretizing the space has been presented by[5]. In MDPs, the agent in a given state takes an action in the environment to reach a new state. The agent starts at an initial point and is given a set of heuristics to reach the end goal[1]. The heuristic, which is the adaptation of the policy for a particular environment or performance of an agent, improves with time. In MDP, the reward is provided to the agent based on action(s) taken either at the final state or intermediate states, depending on the environment design. A satellite takes a series of actions to reach the GEO, and

these actions need to be optimized in order to minimize a certain objective function, such as minimizing the transfer time and/or minimize the fuel consumption, en route to GEO. DRL is typically seen as an enhanced version of the more traditional reinforcement learning (RL). While the RL is more dynamic and uses trial and error to maximize the outcome, a DRL agent can use the existing knowledge and apply it to a new problem.

This thesis is motivated by the aim of reducing computation time to determine the transfer times to reach GEO. The thesis builds upon the idea of deep reinforcement learning and provides the DNNs that can serve as Q-Value predictors for the DRL to become better at trajectory guidance. In this thesis, the problem is posed as a DRL-based optimal control problem. The satellite on each step can look ahead using the optimal trajectory time and choose the best path for orbit raising. This thesis considers recent innovations in order to address the challenges of electric orbit-raising. The first innovation is the use of a regularized dynamic model that is suitable for the solution of the optimal control problem associated with electric orbit-raising. The second innovation is the modeling of the optimal control problem as a sequence of optimal control sub-problems, with each sub-problem minimizing the deviation of the spacecraft from the targeted geostationary orbit at the end of each revolution. While these innovations result in fast and robust computation of electric orbit-raising trajectory (at the cost of sub-optimality of the computed trajectories), it cannot be readily integrated within a DRL scheme. The main challenge here is that the computation of a metric of interest (such as transfer time or fuel expenditure) at any point requires the solution of all optimal control sub-problems corresponding to all future revolutions. While this process is not a challenge for the computation of a single orbit-raising trajectory, this is not an efficient mechanism to evaluate the rewards associated with all state-action pairs within a DRL scheme. The thesis addresses this challenge by training an artificial neural network that facilities the straight-forward computation of the metric without having to solve all numerous optimization problems corresponding to forthcoming revolutions. In this context, the main contribution of the thesis is a comprehensive assessment of various neural network architectures and the determination of the optimal architecture that is most suitable for the DRL scheme. While there have been some work in the literature on training

artificial neural networks for the electric orbit-raising problem[6;7], there has been a lack of a comprehensive assessment of neural network architectures, and our present thesis fills in this void.

1.2 Contributions

The main contributions of this thesis can be summarized as follows:

- Designing DNN architectures, for accurate Q-value (i.e., transfer time in our work) prediction for DRL-based electric orbit-raising for six different planar and non-planar mission scenarios for transfer to GEO.
- Finding the near-optimal setting of hyperparameters for DNNs of each raising mission scenario that minimizes the transfer time.
- Comparing the results from our designed DNNs with the contemporary Machine Learning (ML) algorithms, such as Support Vector Machine (SVM), Random Forest (RF), and Decision Tree (DT) for regression.

The orbits raising mission scenarios included in this thesis are:

- Planar and non-planar Super-GTO to GEO
- Planar and non-planar GTO to GEO
- Planar and non-planar Sub-GTO to GEO

The remainder of this thesis is organized as follows. Chapter 2 discusses prior works in literature related to spacecraft orbit-raising. Chapter 3 refers to the problem formulation while Chapter 4 details the generation of the dataset. In Chapter 5 the design of DNN's for the problem is explained following the results of these DNN's in Chapter 6.

Chapter 2

Literature Review

In recent years, researchers have been studying optimal trajectories associated with electric low-thrust orbit raising maneuvers, to reach the geostationary Earth orbit (GEO).[8–10]. The low thrust as mentioned previously is the primary concern for using a spacecraft that is all-electric, and that causes a long transfer time taken for the spacecraft to reach the Geostationary transfer orbit (GTO). In addition, the spacecraft encounters multiple eclipses in the route, this further prolongs the transfer time unless the spacecraft has the capability of storing electric energy for use during the eclipses[11]. This creates a reason to investigate closely the variety of crucial factors affecting the electric orbit-raising problem. These factors include the initial orbit of the spacecraft, dry mass i.e. the mass that is delivered, the mass of propellant, the capacity solar panel arrays have to generate thrust during transfer and the type of thrusters to be used in the spacecraft (Hall, Magneto Plasma Dynamic (MPD) or Ion Thrusters). Hence, mission designers have to investigate in detail various scenarios to obtain the least time taking trajectory.

In order to overcome the above stated issue, direct optimization methods are used. The state and control variables can be discretized with respect to time and apply quadrature rules across the discretized segments in order to set up a parameter optimization problem that will be a nonlinear programming problem[12;13], that are solved by softwares like Sparse Nonlinear OPTimizer (SNOPT), Interior Point OPTimizer (IPOPT)[14;15]. The rate of convergence for

low-thrust optimization using direct methods are better than indirect methods. However, direct methods need initial guesses which are unknown prior. This lack of knowledge about initial guesses has led to a number of studies such as shape based methods[16–21] and a number of guidance-based schemes[22–24] to approximately solve low-thrust optimization problems. The electric orbit-raising problem has been studied using a variety of dynamic models, to specifically, study the influence on the convergence of low-thrust trajectory through numerical optimization schemes[25;26]. Also, among all the regularized elements, five remain constant for time and change slowly under the perturbations in case of Keplerian motion. Hence, these are suitable for use in trajectory optimization schemes that have long-time-scale transfers.

An actor critic-based reinforcement learning model consisting of two neural networks has been explored in[2] by Kolosa. Kolosa also analyzes the effect of gravity in maneuvering near small bodies. Kolosa used a RL model to solve low-thrust trajectory optimization problems using two neural networks, an actor network and a critic network. Willis et al.[3] have studied the spacecraft near a small celestial body with unknown gravity using RL. Shaoming et al. have used a heuristic way to shape a proper reward function for the RL agent. LaFarge[27] has explored a controller, trained via RL considering multi body perturbations. The use of RL in satellite clusters is also quite appealing. Xiao et al.[28] have developed a framework to find suitable parameters for numerical optimization of such RL models.

A guidance strategy for spacecraft proximity operations using deep reinforcement learning (DRL) has been explored by Hovell and Ulrich[29]. Hovell and Ulrich have introduced a control theory alongside DRL to pose the docking scenarios. Broida and Linares[30] have presented a Rendezvous guidance technique for spacecraft in a cluttered environment using reinforcement learning (RL). Broida and Linares have implemented and evaluated a proximal policy optimization (PPO) to develop a control policy. This policy has been used to move a satellite relative to an orbit reference frame into a docking position with another. Bo et al.[31] have used the DDPG framework to target and avoid obstacles for UAVs (unmanned aerial vehicles). Charles et al.[32] have looked into the six degrees of freedom docking via reinforcement learning. The policy implemented by [32] allows the docking maneuvers as a feedback control law under uncertain environments. A safe landing site selection considering

divert maneuvers using DRL has been proposed by Keidai et al[33]. Maksim et al.[34] have presented a survey of machine learning (ML) techniques in spacecraft control detailing the work on formulation control and design of control laws in spacecraft flight and landing.

In this thesis, we present a set of (deep neural networks) DNNs that can be utilized to optimize a satellite's orbit-raising trajectory. A satellite can apply a variable force, at a required variable angle, and makes the state space for the problem continuous. The optimal DNN for each mission scenario presented in the thesis can be used to predict time from a state in the continuous space. Hyeokjoon et al.[7] present a soft actor-critic (SAC) algorithm for trajectory-raising optimization, that is a DRL based algorithm only. They have considered SAC for only LEO (low earth orbit)-GEO and GTO-GEO mission scenarios.

Chapter 3

Problem Formulation

This work aims at predicting Q-value (here, the predicted transfer time to GEO) that can be used as a directional parameter for the deep reinforcement learning (DRL) agent to reach geostationary Earth orbit (GEO) in optimal time. The proposed work breaks the problem of the minimization of time and fuel, for orbit raising, into two parts. In the first part, different architectures of RL models are developed to minimize transfer time to GEO. The second part is to predict the Q-Value, i.e., flight time, as accurately as possible, using a deep reinforcement learning model. The thesis focuses on the second part of this above stated problem.

3.1 Reinforcement Learning

Reinforcement Learning (RL) is one of the three basic machine learning paradigms, with the other two being supervised and unsupervised learning. In RL-based approaches, agents, in an environment, take actions to maximize the cumulative reward. Unlike dynamic programming, RL does not assume knowledge of the mathematical model and generally targets large Markov Decision Processes (MDPs) where the exact mathematical solution becomes infeasible. While greedy approaches follow a fixed set of paths for finding the optimal path, RL allows an agent to take actions for exploration and not follow a fixed set of paths, but a

continuous one. The agent in an RL model is typically trained using Q-Learning[1], as shown in the following equations.

$$\underbrace{Q(s,a)'}_{\text{New Q-Value}} = \overbrace{Q(s,a)}^{\text{Old Q-Value}} + \alpha \Big[\underbrace{R(s)}_{\text{Reward}} + \gamma \overbrace{\max Q'(s',a')}^{\text{Maximum predicted reward, given new state and all possible actions}} - Q(s,a) \Big] \tag{3.1}$$

Learning Rate

Discount rate

In Eq. 3.1 Q' is the updated Q value for a state s for an action a, that is, take action a in state s and update the Q value using the old Q value for the state and action. R is the reward an agent receives when it reaches a new state s' after taking an action a in state s. γ is the discount factor that determines how much future rewards are worth and α is the learning rate at which we update the Q value and get Q'.

For trajectory optimization, the trained DRL model computes the total transfer time required, for each orbit-raising scenario, to reach the GEO. Thus, there can be different transfer times corresponding to each action taken in the environment. The RL model selects the most optimal option (in terms of time) to choose the most suitable trajectory. Once the path is taken and the reward, i.e., reaching the GEO, is complete, each action taken by the satellite will increase its reward value. Figure 3.1 depicts the deep reinforcement learning model of the problem.

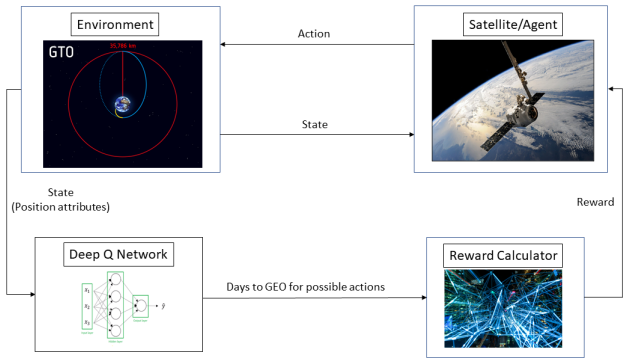

Figure 3.1: *Deep Reinforcement Learning Model for Orbit Raising*

In Figure 3.1, the Deep Q-Network (DQN) represents the optimal neural network for a particular orbit-raising transfer. The DQN predicts the number of days to reach GEO from each state for each possible action.

3.2 Predicting the Q-Value for DRL Model

The DRL model relies on the predictor function of the Q-value. The model will pass a set of actions to the predictor and get a set of (action, time-value) pairs. The pair with the optimal time-value gets selected and is passed to the model.

The predictor is a neural network trained to find the transfer time with the least mean-squared error or the error in days for a particular spacecraft's trajectory. Therefore, optimization of this neural network is crucial. The proposed work finds the optimized neural network for six different types of orbit-raising problems. These problems are broken up into three pairs of planar and non-planar orbit-raising problems.

Our goal is to train optimal (deep neural network) DNN models for each trajectory problem that can predict time to GEO from an input state. These optimal DNN models will be capable of generating an optimized solution for the DRL module of orbit-raising problem, while maintaining real-timeliness. This real-timeliness correlates to using the least number of neurons without sacrificing the prediction accuracy.

Our trained DNN model's predicted time values can be used to in DRL module to take an action, that can work as a directional input for the DRL agent and give the raising trajectory to GEO with least time. This set of actions serves as the policy that provides minimal time transfer for a specific orbit-raising. The reward has to be maximized over a set of possible actions $a \in A$ for a state s and can be formulated as:

$$R = \max_{a \in A} F(s, a), \tag{3.2}$$

where R is a reward function, F is a function that takes predicted time for each action a from a set of actions possible on a state s referred as x in the dataset chapter 4. This predicted

time to reach GEO is computed using a DNN. Each a in A will generate a different possible next state. Where a is a set of Force (F), and two control angles α (in-plane angle) and β (out-of-plane angle). To train these DNNs we need to generate data for each of the mission scenarios mentioned in Chapter 1 which is detailed in the following chapter.

Chapter 4

Generating Dataset for Orbit Raising[1]

In this chapter, we discuss the procedure that was followed to generate the data to train the deep neural networks. To this end, we compute the transfer times required from a given orbit of the spacecraft to the final orbit that is GEO by solving a sequence of optimization problems each over one revolution of the satellite. In this chapter, a brief is provided, on the states of the spacecraft using a set of dynamic coordinates introduced in work done by Sreesawet and Dutta[35].

To generate the dataset, Dr. Atri's team started with a set of given initial conditions that includes the six states of the spacecraft and the three weights for the objective function of the optimization problem. During the transfer of the spacecraft from the initial orbit to the GEO, it may encounter an eclipse when it passes through the Earth's shadow. We are propelling the spacecraft using electric propulsion and consider no on-board batteries, and we assume no thrust available during the eclipses. To this end, Dr. Atri's team determined the eclipse portion of the revolution using the cylindrical shadow model. Once the eclipse portion of the revolution is selected, we determine the optimal thrust angles in which the spacecraft have to thrust to reach GEO. Then the spacecraft's end states are calculated after each revolution using these thrust angles, as discussed earlier. The dynamic model used to

[1]This chapter is a brief version of the Dataset section in the submitted paper. This section of the actual-paper is provided by Dr. Dutta's Team at Wichita State University (WSU) as this thesis is a result of the collaborative work between WSU and KSU on the NASA project NASA-20-2020EPSCoR-0017.

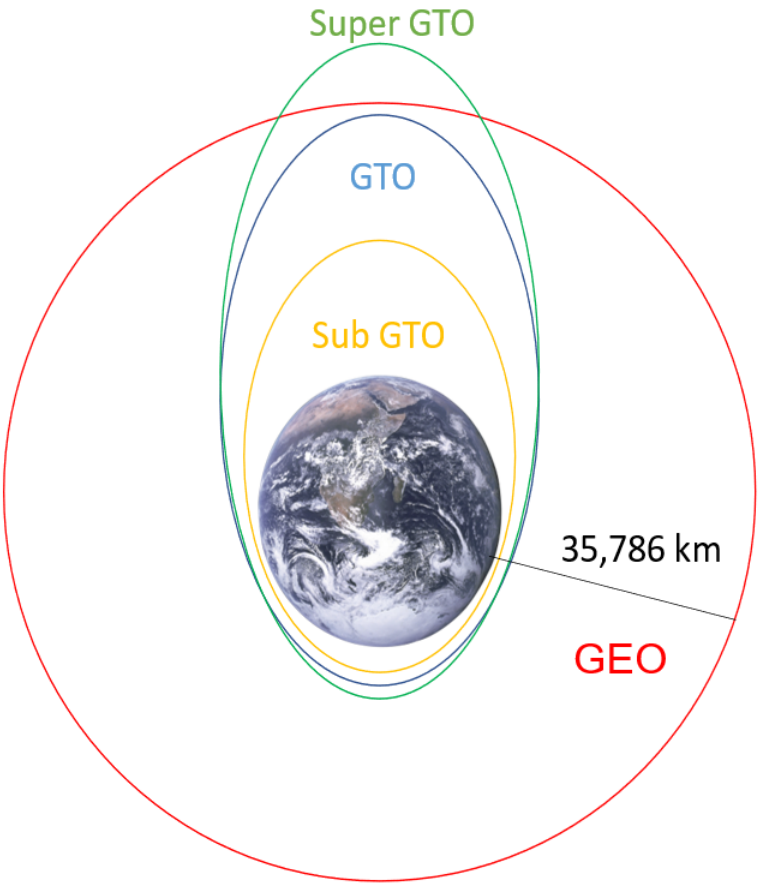

Figure 4.1: *Displayed in the figure are different orbits considered for orbit raising in this thesis.*

propagate the states of the spacecraft, the Earth's shadow model, and the optimization problem set is discussed in detail in our previous work[35;36].

After each revolution, terminal conditions were checked (whether goal orbit is reached or not). Using this final trajectory, the transfer time from each revolution to the geostationary Earth orbit (GEO) was determined. The dataset was generated from trajectories starting from six different initial orbits and all the possible weight combinations. The six initial orbits consist of three planar and three non-planar geostationary transfer orbit (GTO), Sub-GTO, and Super-GTO cases. The generated data set has **18.6k** points in planar GTO data, **483.4k** points in non-planar GTO data, **23.3k** points in planar Sub-GTO data, **567.9k** points in non-planar Sub-GTO data, **14.3k** points in planar Super-GTO data and **388.3k** points in non-planar Super-GTO, totalling to a **1495.8k** training points (1 for each revolution). Each

point has the properties listed as: **(semi-major axis (a), eccentricity (e), inclination (i),** w_h**,** w_{hxy}**,** w_e**, mass (m))**. Each training pair has seven inputs and one output. The seven inputs are the semi-major axis, eccentricity, inclination, the mass of spacecraft, and the three scalar weights of the optimization function's objective function. The output in the training pairs is the transfer time corresponding to the inputs in the respective training pairs. The mass of spacecraft is considered to account for the electric propulsion characteristics of the spacecraft as the change of mass of the spacecraft is directly proportional to the thrust of the spacecraft and inversely proportional to the specific impulse (I_{sp}). The Figure 4.1 shows the orbits(except for LEO) for which the dataset has been generated and is used to train the deep neural networks.

Chapter 5

Designing the Optimized Neural Networks

In this chapter, the proposed problem-solving methodology is discussed. It starts with an overview and continues to detail the tuning of neural networks trained for orbit-raising mission.

5.1 Overview

The dataset has trajectory data for the six mission scenarios referred in Chapter 1 and discussed in detail in Chapter 4. Our goal is to be able to predict, accurately, the time from a current state represented using the 7 variables (a, e, i, w_h, w_{hxy}, w_e, m) to a final state in geostationary Earth orbit (GEO). These variables are detailed in Chapter 4. The predicted time is a continuous value and hence a linear regression model is used[37;38]. Figure 5.1 and 5.2 depict DNN-3 with 30 neurons and DNN-5 with 40 neurons respectively.

For our experiments, we have explored a total of 40 configurations with two varying hyperparameters: number of layers and number of neurons. We have explored the total neurons in deep neural network (DNN) ranging from 10 to 110 with eight (8) values being

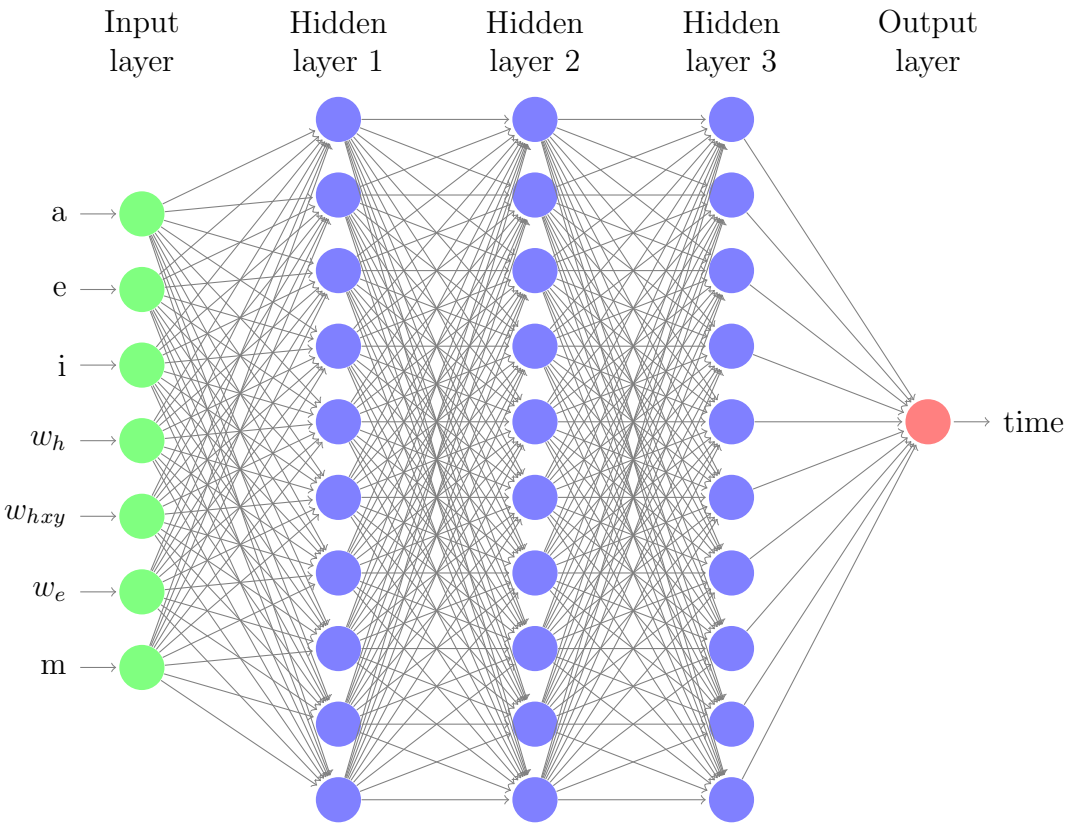

Figure 5.1: *Example of DNN-3 with 30 neurons (a 10-10-10 con guration).*

10, 20, 30, 40, 50, 70, 90, and 110. We have explored varying number of DNN layers ranging from one layer to five layers. Thus, in total, we have 8 x 5 = 40 configurations to explore for each orbit-raising scenario. A total of $5 \times 8 = 40$ combinations for each orbit raising problem were considered. The naming convention for the sake of this thesis goes as follows. DNN-X with N neurons means a DNN with X number of layers, with each layer having N/X neurons. For example, DNN-5 with N = 50 neurons implies a DNN that has a total of 5 layers, with each layer having N/X = 50/5 = 10 neurons. For N and X, where N is not divisible by X, the floor value of N/X is taken, and the excess neurons are distributed equally among the initial layers. For example, when N and X are 10 and 4, respectively would be calculated as:

$$\lfloor 10/4 \rfloor = 2$$

Each layer is assigned 2 neurons initially, then the remaining 2 neurons are distributed

to layer 1 and layer 2, thus, the layers configuration becomes 3-3-2-2, where the first two layers have three neurons each and the last two layers have two neurons each.

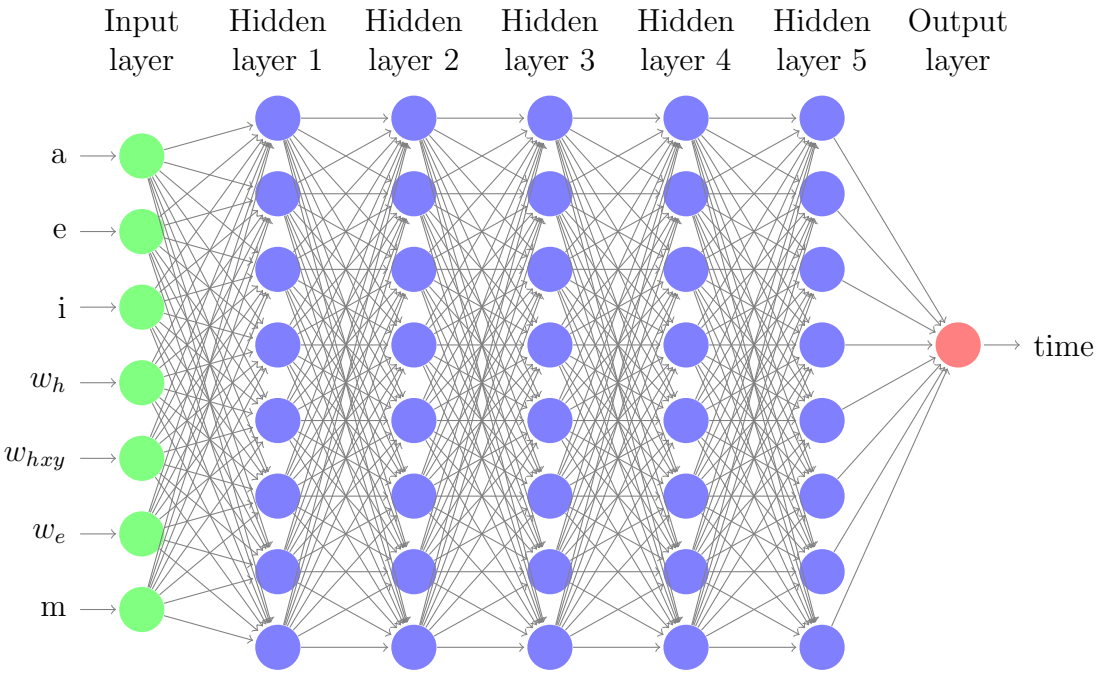

Figure 5.2: *Example of DNN-5 with 40 neurons (an 8-8-8-8-8 con guration).*

5.2 Approach

We initially used Adam optimizer[39] by Kingma and Ba. Our results for transfer time prediction and mean squared error (MSE) obtained through the Adam optimizer. The results obtained through the Adam optimizer resulted in an unacceptably high MSE. Furthermore, the time of prediction is of essence in evaluation of spacecraft orbit-raising scenarios, and hence, keeping a notch on the number of neurons in the DNN design is crucial. One such technique that is relatively faster is the Levenberg Marquardt (LM) algorithm[40] by Gavin. We, therefore, have used the LM algorithm for training the DNN to achieve a fast and near-optimal solution when compared against the Adam optimizer solution. The LM algorithm finds the optimal solution, but its effectiveness comes with a drawback of high training time/compute power and memory usage. It uses a Hessian matrix that is a matrix of sec-

ond derivative. This matrix makes it guaranteed to find the minima of the loss function. But computing a Hessian matrix becomes an in-efficient solution in terms of memory and computations for large datasets and DNNs with a large number of neurons. The training for the LM algorithm was first done using a Python implementation, but that was inefficient in terms of memory usage. The MATLAB implementation had more options to limit the amount of data it processes in each chunk of iteration of the algorithm. Therefore, MATLAB implementation was faster and had better attributes for the network to analyze and test. The results in Sec. 6. are generated using MATLAB implementation of the LM algorithm.

Initially, the Sub-GTO (geostationary transfer orbit) to geostationary Earth orbit (GEO) transfer dataset was used. Then, we tested different numbers of neurons and various configurations on this dataset to achieve a good fit for the problem. Through these experiments on Sub-GTO to GEO transfer dataset, we analyzed and found the range for maximum neurons to be 110 and a total of 5 layers to be enough to fit the dataset. We then performed a design space exploration to explore various DNN configurations with varying layer sizes and total number of neurons.

Chapter 6

Results

The Chapter details results from 40 different deep neural network (DNN) configurations for each of the 6 orbit raising mission scenarios. The results are presented as a set of both figures and tables, from figures we can analyze the pattern of DNN performance. The tables come into play for specific value comparisons. We have discussed the best neural networks overall first, then explore the best DNN configuration corresponding to the number of neurons and the number of layers the neurons are distributed into.

6.1 Non-Planar Super-GTO to GEO Transfer

In our first set of experiments, we aim to design a DNN that accurately predicts the spacecraft's transfer time for non-planar transfer from Super-GTO to GEO. Figure 6.1 depicts mean error in days of the spacecraft's transfer time predicted by different DNN configurations. Table 6.1 presents the mean error in days for 40 different DNN configurations with varying number of layers and the number of neurons for non-planar Super-GTO to GEO transfer.

The results show that the DNN-3 configuration with 110 neurons achieves the lowest mean error of 0.0256 days (37.44 minutes). DNN-2 and DNN-4 configurations with 110 neurons exhibit mean errors of 0.0294 days (41.76 minutes) and 0.0400 days (57.60 minutes),

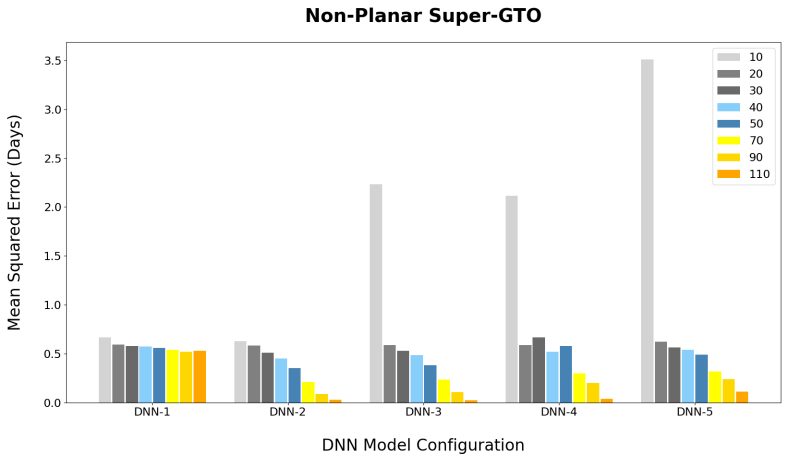

Figure 6.1: *Mean error in days (square root of MSE) for 5 di erent neural network con g-urations for non-planar Super-GTO to GEO transfer.*

respectively. This represents a 11.5% and a 53.8% increase in the mean error over the DNN-3 configuration with 110 neurons. DNN-2 configuration with 90 neurons has a mean error of 0.0891 days (128.30 minutes).

Table 6.1: *Mean error in days (square root of MSE rounded to 4 decimal points) for 5 di erent neural network con gurations for non-planar Super-GTO to GEO transfer.*

No. of Neurons	DNN-1	DNN-2	DNN-3	DNN-4	DNN-5
10	0.6660	0.6259	2.2304	2.1148	3.5103
20	0.5943	0.5849	0.5868	0.5891	0.6234
30	0.5764	0.5082	0.5308	0.6680	0.5652
40	0.5750	0.4504	0.4867	0.5196	0.5379
50	0.5578	0.3528	0.3800	0.5772	0.4905
70	0.5381	0.2106	0.2329	0.2964	0.3198
90	0.5200	0.0891	0.1091	0.1998	0.2370
110	0.5309	0.0294	0.0256	0.0400	0.1105

One key observation from Figure 6.1 is that by dividing a limited number of neurons into multiple layers, the error becomes higher. However, to achieve a good regression fit, more layers need to be added. Therefore, having more layers and a greater number of neurons results in a significant decrease in mean error. DNN-2 and DNN-3 with 30 neurons result in a mean error of 0.5082 days (731.81 minutes) and 0.5308 days (764.35 minutes), respectively.

With 40 neurons, the DNN-2 has a mean error of 0.4504 days. DNN-2 with 90 neurons and DNN-3 with 110 neurons exhibit mean errors of 0.0891 and 0.0256 days (128.30 and 36.86 minutes), respectively.

6.2 Non-Planar GTO to GEO Transfer

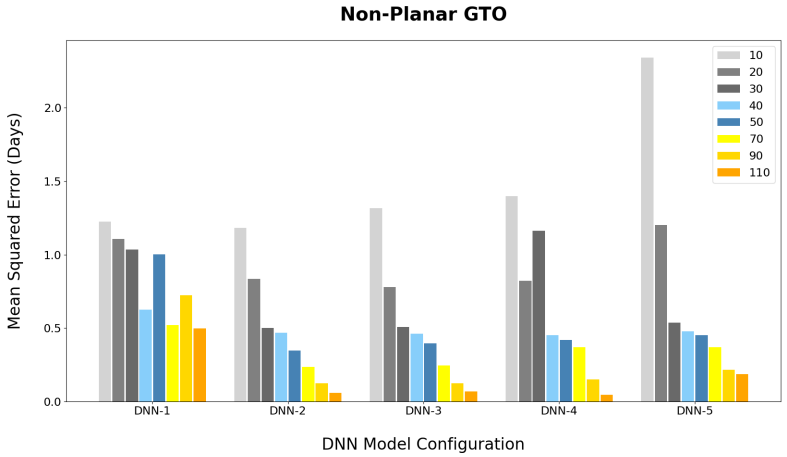

Figure 6.2: *Mean error in days (square root of MSE) for 5 di erent neural network con g-urations for non-planar GTO to GEO transfer.*

In our second set of experiments, we aim to design a DNN that accurately predicts the spacecraft's transfer time for non-planar transfer from GTO to GEO. Figure 6.2 depicts mean error in days of the spacecraft's transfer time predicted by different DNN configurations.

The results show that the DNN-4 configuration with 110 neurons achieves the lowest mean error of 0.0435 days (62.49 minutes). DNN-2 and DNN-3 configurations with 110 neurons exhibit mean error of 0.0586 days (84.24 minutes) and 0.0670 days (96.48 minutes), respectively. This represents a 34.79% and a 54.38% increase in the mean error over the DNN-4 configuration with 110 neurons. DNN-2 and DNN-3 configurations with 90 neurons have a mean error of 0.1227 days (176.69 minutes) and 0.1230 days (177.12 minutes), respectively. Table 6.2 details mean error in days of the spacecraft's transfer time predicted by different DNN configurations.

For the DNN containing 10 neurons, the most optimal performance is exhibited by DNN-

Table 6.2: *Mean error in days (square root of MSE rounded to 4 decimal points) for 5 di erent neural network con gurations for non-planar GTO to GEO transfer.*

No. Neurons	DNN-1	DNN-2	DNN-3	DNN-4	DNN-5
10	1.2230	1.1831	1.3164	1.3979	2.3424
20	1.1077	0.8336	0.7787	0.8202	1.2013
30	1.0355	0.4999	0.5080	1.1626	0.5381
40	0.6237	0.4677	0.4608	0.4499	0.4789
50	1.0026	0.3470	0.3970	0.4176	0.4509
70	0.5188	0.2349	0.2443	0.3692	0.3692
90	0.7230	0.1227	0.1230	0.1501	0.2159
110	0.4961	0.0586	0.0670	0.0435	0.1854

2 with a mean error of 1.1831 days (1703.664 minutes). For 20 and 30 neurons, DNN-2 has mean errors of 0.7780 days (1120.32 minutes) and 0.4999 days (719.86 minutes), respectively. DNN-3 with 90 neurons and DNN-4 with 110 neurons exhibit mean errors of 0.1227 days (176.69 minutes) and 0.0435 days (62.64 minutes), respectively.

6.3 Non-Planar Sub-GTO to GEO Transfer

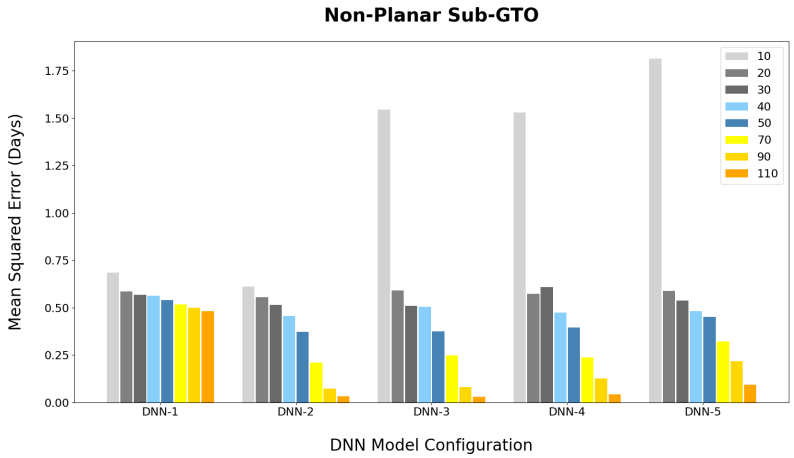

Figure 6.3: *Mean error in days (square root of MSE) for 5 di erent neural network con g-urations for non-planar Sub-GTO to GEO transfer.*

In our third set of experiments, we aim to design a DNN that accurately predicts the

spacecraft's transfer time for non-planar transfer from Sub-GTO to GEO. Figure 6.3 depicts mean error in days of the spacecraft's transfer time predicted by different DNN configurations.

The results in Figure 6.3 that the DNN-3 with 110 neurons achieves the best mean error of 0.0294 days (42.19 minutes). DNN-2 and DNN-4 configurations with 110 neurons exhibit mean errors of 0.0326 days (46.80 minutes) and 0.0423 days (60.77 minutes), respectively. This represents a 10.92% and 44.03% increase in the mean error over the best DNN-3 configuration with 110 neurons. DNN-2 with 90 neurons has a mean error of 0.0728 days (104.83 minutes), which is nearest to the mean errors of DNN-3 with 90 neurons and DNN-5 with 110 neurons 0.0802 days (115.488 minutes) and 0.0945 days (136.08 minutes), respectively. This represents an increase of 10.31% and 29.85% mean error over DNN-2 with 90 neurons. Table 6.3 details mean error in days of the spacecraft's transfer time predicted by different DNN configurations.

For a network containing 10 neurons, the most optimal performance was exhibited by DNN-2 with a mean error of 0.6099 days (878.26 minutes). For 20 and 30 neurons, DNN-2 has mean error of 0.5555 days (799.92 minutes) and 0.5083 days (731.95 minutes), respectively. DNN-2 with 90 neurons and DNN-3 with 110 neurons exhibit mean errors of 0.0728 days (104.83 minutes) and 0.0294 days (42.34 minutes), respectively.

Table 6.3: *Mean error in days (square root of MSE rounded to 4 decimal points) for 5 di erent neural network con gurations for non-planar Sub-GTO to GEO transfer.*

No. of Neurons	DNN-1	DNN-2	DNN-3	DNN-4	DNN-5
10	0.6849	0.6099	1.5446	1.5298	1.8147
20	0.5865	0.5554	0.5913	0.5724	0.5888
30	0.5685	0.5137	0.5083	0.6087	0.5366
40	0.5625	0.4568	0.5053	0.4744	0.4818
50	0.5399	0.3715	0.3746	0.3943	0.4509
70	0.5182	0.2111	0.2472	0.2379	0.3206
90	0.4999	0.0728	0.0802	0.1273	0.2166
110	0.4826	0.0326	0.0294	0.0423	0.0945

6.4 Planar Super-GTO to GEO Transfer

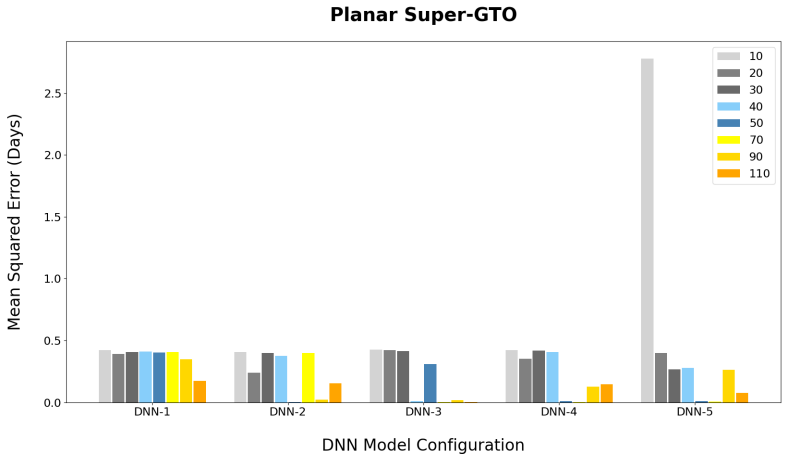

Figure 6.4: *Mean error in days (square root of MSE) for 5 di erent neural network con g-urations for planar Super-GTO to GEO transfer.*

In our fourth set of experiments, we aim to design a DNN that accurately predicts the spacecraft's transfer time for planar transfer from Super-GTO to GEO. Figure 6.4 depicts mean error in days of the spacecraft's transfer time predicted by different DNN configurations.

The results show that the DNN-3 with 70 neurons achieves the best mean error of 0.0028 days (4.03 minutes). DNN-4 with 70 neurons and DNN-3 with 110 neurons configurations exhibit mean errors of 0.0034 days (4.90 minutes) and 0.0036 days (5.18 minutes), respectively. This represents a 21.4% and 28.6% increase, respectively, in the mean errors over the best DNN-3 configuration with 70 neurons. Table 6.4 details mean error in days of the spacecraft's transfer time predicted by different DNN configurations.

DNN-2 with 50 neurons has a mean error of 0.0042 days (6.05 minutes) which is nearest to the mean error of DNN-5 with 70 neurons of 0.0053 days (7.63 minutes). Hence, DNN-2 with 50 neurons represents an increase of 23.8% mean error over DNN-2 configuration with 90 neurons.

For a network containing 10 neurons, the most optimal performance is exhibited by DNN-2 with a mean error of 0.4059 days (584.50 minutes). For 20 and 30 neurons, DNN-2 and DNN-5 have mean errors of 0.2385 days (343.44 minutes) and 0.2686 days (386.78 minutes),

Table 6.4: *Mean error in days (square root of MSE rounded to 4 decimal points) for 5 di erent neural network con gurations for planar Super-GTO to GEO transfer.*

No. of Neurons	DNN-1	DNN-2	DNN-3	DNN-4	DNN-5
10	0.4245	0.4059	0.4249	0.4213	2.7789
20	0.3914	0.2385	0.4208	0.3537	0.3985
30	0.4062	0.3984	0.4145	0.4183	0.2686
40	0.4105	0.3777	0.0097	0.4069	0.2789
50	0.4051	0.0042	0.3117	0.0096	0.0114
70	0.4084	0.4008	0.0028	0.0036	0.0053
90	0.3488	0.0212	0.0196	0.1259	0.2629
110	0.1731	0.1564	0.0036	0.1457	0.0785

respectively. DNN-3 with 90 neurons and 110 neurons exhibit mean errors of 0.0196 days (28.22 minutes) and 0.0036 days (5.18 minutes), respectively.

6.5 Planar GTO to GEO Transfer

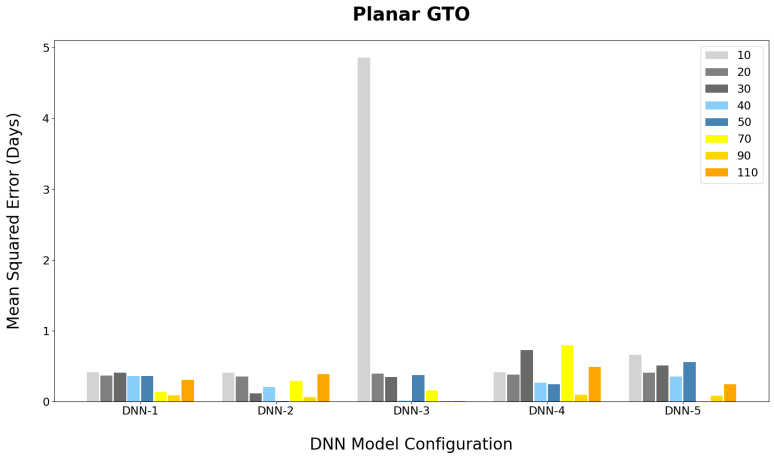

Figure 6.5: *Mean error in days (square root of MSE) for 5 di erent neural network con g-urations for planar GTO to GEO transfer.*

In our fifth set of experiments, we aim to design a DNN that accurately predicts the spacecraft's transfer time for planar transfer from GTO to GEO. Figure 6.5 depicts mean error in days of the spacecraft's transfer time predicted by different DNN configurations.

The results show that the DNN-3 with 90 neurons achieves the best mean error of 0.0030 days (4.32 minutes). DNN-3 with 110 neurons exhibits mean error of 0.0032 days (4.61 minutes). This represents a 3.33% increase in mean error over the mean error of DNN-3 with 90 neurons. DNN-5 with 70 neurons has a mean error of 0.0050 days (7.20 minutes) that is closest to the mean error of DNN-2 with 50 neurons of 0.0063 days (9.07 minutes) that is 26.0% greater than the mean error of DNN-5 with 70 neurons. Table 6.5 details mean error in days of the spacecraft's transfer time predicted by different DNN configurations.

For a network containing 10 neurons, the most optimal performance is exhibited by DNN-2 with a mean error of 0.4062 days (584.93 minutes). For 20 and 30 neurons, DNN-2 and DNN-3 have mean errors of 0.3495 days (503.28 minutes) and 0.1162 days (167.33 minutes), respectively. DNN-3 with 90 neurons and 110 neurons exhibit mean errors of 0.0030 days (4.32 minutes) and 0.0032 days (4.61 minutes), respectively.

Table 6.5: *Mean error in days (square root of MSE rounded to 4 decimal points) for 5 di erent neural network con gurations for planar GTO to GEO transfer.*

No. of Neurons	DNN-1	DNN-2	DNN-3	DNN-4	DNN-5
10	0.4098	0.4063	4.8613	0.4163	0.6590
20	0.3625	0.3495	0.3915	0.3811	0.4061
30	0.4030	0.1162	0.3441	0.7234	0.5067
40	0.3619	0.2027	0.0133	0.2655	0.3535
50	0.3607	0.0063	0.3702	0.2411	0.5537
70	0.1373	0.2901	0.1523	0.7932	0.0050
90	0.0900	0.0623	0.0030	0.0928	0.0770
110	0.3030	0.3829	0.0032	0.4858	0.2406

6.6 Planar Sub-GTO to GEO Transfer

In our sixth set of experiments, we aim to design a DNN that accurately predicts the spacecraft's transfer time for planar transfer from Sub-GTO to GEO. Figure 6.6 depicts mean error in days of the spacecraft's transfer time predicted by different DNN configurations.

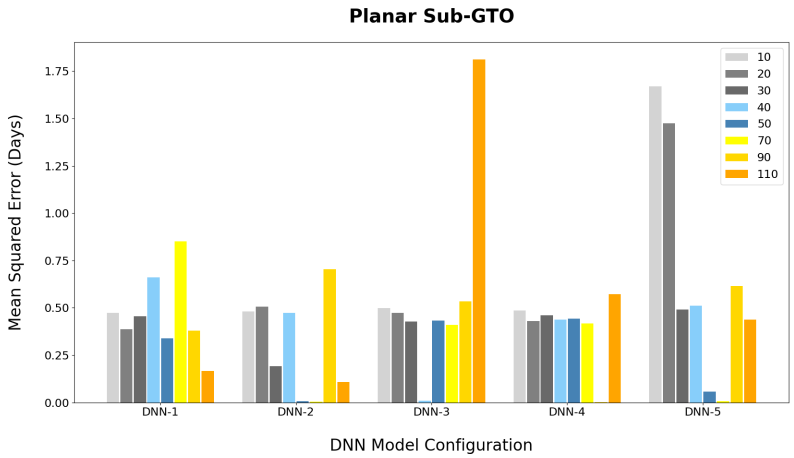

Figure 6.6: *Mean error in days (square root of MSE) for 5 di erent neural network con g-urations for planar sub-GTO to GEO transfer.*

The results show that the DNN-4 with 90 neurons achieves the best mean error of 0.0030 days (4.32 minutes). Other DNN configurations that achieve relatively low mean error are DNN-2 with 70 neurons, DNN-5 with 70 neurons, and DNN-2 with 50 neurons. DNN-2 with 70 neurons configuration exhibits mean error of 0.0039 days (5.47 minutes). This represents a 26.66% increase in mean error over DNN-3 with 90 neurons. DNN-5 with 70 neurons has a mean error of 0.0080 days (11.52 minutes) that is closest to the mean errors of DNN-2 with 50 neurons 0.0081 days (11.66 minutes) and DNN-3 with 40 neurons with a mean error of 0.0093 days (13.39 minutes); that are 1.25% and 16.25% greater than the mean error of DNN-5 with 70 neurons. Table 6.6 details mean error in days of the spacecraft's transfer time predicted by different DNN configurations.

For a network containing 10 neurons, the most optimal performance is exhibited by DNN-1 with a mean error of 0.4740 days (682.56 minutes). For 20 and 30 neurons, DNN-1 and DNN-2 have mean errors of 0.3869 days (557.14 minutes) and 0.1924 days (277.06 minutes), respectively. DNN-4 with 90 neurons and DNN-2 with 110 neurons exhibit mean errors of 0.0030 days (4.32 minutes) and 0.1088 days (156.67 minutes), respectively.

An important point to note while examining the mean error values is that the mean error values for the best-performing DNN configurations are not as huge as they seem to be. The complete trajectory is over several months, ranging to

Table 6.6: *Mean error in days (square root of MSE rounded to 4 decimal points) for 5 different neural network configurations for planar Sub-GTO to GEO transfer.*

No. of Neurons	DNN-1	DNN-2	DNN-3	DNN-4	DNN-5
10	0.4740	0.4806	0.4976	0.4866	1.6699
20	0.3869	0.5051	0.4739	0.4299	1.4739
30	0.4556	0.1924	0.4264	0.4606	0.4917
40	0.6603	0.4737	0.0093	0.4382	0.5111
50	0.3379	0.0081	0.4331	0.4427	0.0585
70	0.8507	0.0039	0.4107	0.4177	0.0080
90	0.3788	0.7039	0.5346	0.0030	0.6164
110	0.1668	0.1088	1.812	0.5709	0.4390

over 240 days. The DNN's have an accuracy of over 99% for the best networks for each of the trajectories shown above. Secondly, over-fitting these neural networks would not be a good idea. These networks are expected to give a direction to the reinforcement learning framework in exploring an unexplored part of space.

6.7 Comparison of Optimal and Predicted Trajectories

Figures 6.7, 6.8 and 6.9 provide a comparison of actual versus the predicted time for Super-GTO to GEO DNN-3 with 110 neurons, GTO to GEO DNN-4 with 110 neurons and Sub-GTO to GEO DNN-3 with 110 neurons respectively, in non-planar orbits. Results show that the predicted time follows closely to the optimal time obtained through solutions of optimization equations.

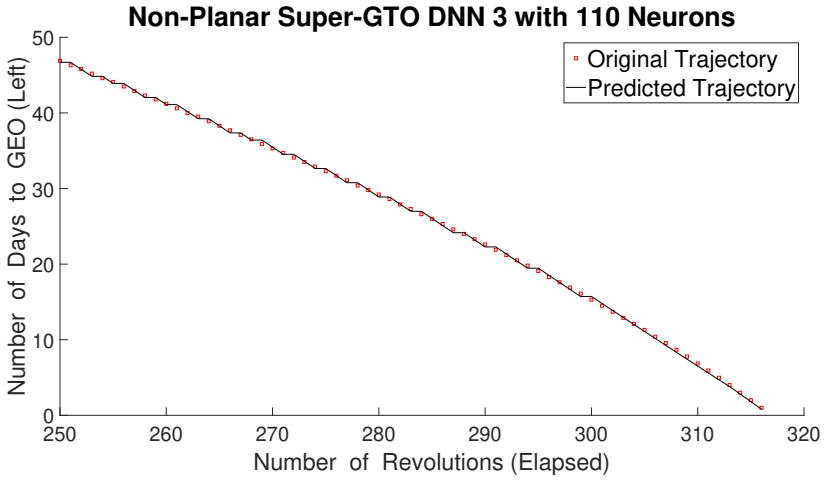

Figure 6.7: *Comparison of original and predicted transfer time for non-planer Super-GTO to GEO transfer using DNN-3 with 110 neurons.*

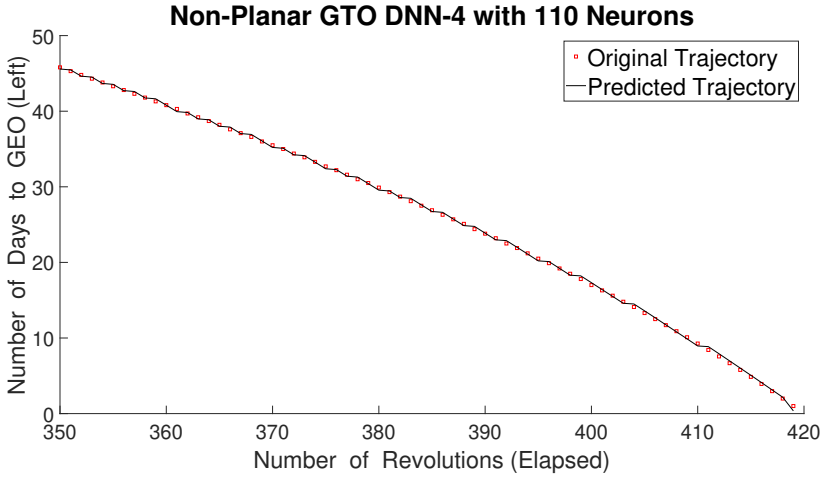

Figure 6.8: *Comparison of original and predicted transfer time for non-planer GTO to GEO transfer using DNN-4 with 110 neurons.*

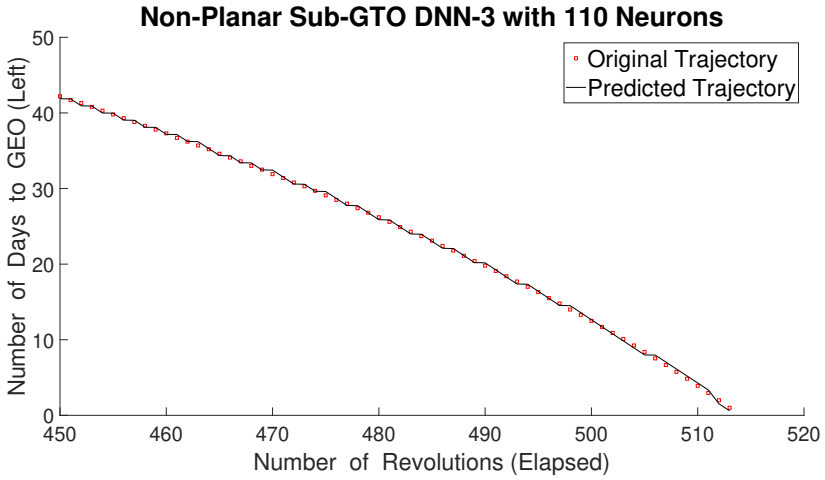

Figure 6.9: *Comparison of original and predicted transfer time for non-planer Sub-GTO to GEO transfer using DNN-3 with 110 neurons.*

Chapter 7

Comparison with Contemporary Machine Learning Algorithms

This chapter compares the results from our designed DNNs using the Levenberg Marquardt (LM) algorithm with the contemporary Machine Learning (ML) algorithms. The algorithms we provide a comparison with are: Support Vector Machine (SVM), Random Forest (RF), and Decision Tree (DT) for regression. We only provide comparison results of non-planar SuperGTO to GEO transfers to demonstrate the effectiveness of our designed DNN versus contemporary ML algorithms as other mission transfers will follow similar comparison trends.

7.1 Non-Planar SuperGTO

In this section, first we obtain the results of mean error for transfer time prediction for non-planar SuperGTO to GEO transfers using SVM, RF, and DT. After that, we provide a comparison of our designed DNNs using the LM algorithm with SVM, RF, and DT is provided. First, we and SVM algorithms is provided.

7.1.1 Support Vector Machine

Table 7.1 contains the results (mean error in days) for SVM algorithm with values of C and ϵ varying from 0.1 - 0.9 with an increment of 0.2 in each subsequent SVM trained. In total we have 25 configurations of SVMs that we have reported in this thesis.

Table 7.1: *Mean error in days (square root of MSE rounded to 4 decimal points) for each subsequent con guration for planar SuperGTO to GEO transfer.*

ϵ / C	**0.1**	**0.3**	**0.5**	**0.7**	**0.9**
0.1	0.9338	0.9284	0.9531	0.9909	1.0166
0.3	0.7065	0.7157	0.7319	0.7577	0.7937
0.5	0.6417	0.6494	0.6668	0.6905	0.7152
0.7	0.6179	0.6274	0.6410	0.6592	0.6849
0.9	**0.6032**	0.6039	0.6100	0.6345	0.6713

As it can be seen in bold text in the Table 7.1, the best performance for the mission scenario is coming from the SVM with $C = 0.9$ and $Epsilon = 0.1$ with a mean error of 0.6032 days. The second best performer is SVM with $C = 0.9$ and $Epsilon = 0.3$ with mean error of 0.6039 days that is 0.12% higher than the SVM with $C = 0.9$ and $Epsilon = 0.1$. The third best in performance is SVM with $C = 0.9$ and $Epsilon = 0.5$ with a mean error of 0.6100 days that is 1.01% higher than the second best performing SVM with $C = 0.9$ and $Epsilon = 0.3$.

7.1.2 Random Forest

Table 7.2 contains the results (mean error in days) for Random Forest algorithm with a varying number of estimators from 10 to 110. The first row in the table (header) displays the total number of estimators (#Est) we have used in Random Forest. The second row is the results presented as mean error (ME) in days.

Table 7.2: *Mean error in days (square root of MSE rounded to 4 decimal points) for 11 different con gurations of support vector machine (SVM) from 10 to 110 estimators (#Est) with an increment of 10 estimators in each subsequent con guration for non-planar SuperGTO to GEO transfer.*

#Est	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**	**110**
ME	0.4078	0.4082	**0.4041**	0.4070	0.4050	0.4072	0.4065	0.4065	0.4060	0.4066	0.4093

As it can be seen in bold text in the Table 7.2, the best performance for planar SuperGTO to GEO mission scenario is achieved with the RF algorithm with 30 estimators. The mean error of RF algorithm with 30 estimators is 0.4041 days. The second best performer, is the RF algorithm with 50 estimators with a mean error of 0.4050 days which is 0.22% higher than the mean error of RF algorithm with 30 estimators. The third best performance is achieved with 90 estimators with a mean error of 0.4060 days that is 0.25% higher than the mean error of RF algorithm with 80 estimators.

7.1.3 Decision Tree

Table 7.3 contains the results (error in days) for Decision Tree (DT) algorithm with a varying random state from 0 to 9. The first row in the table (header) displays the random state value (RSV) used to train DT algorithm. The second row is the results presented as mean error (ME) in days.

As it can be seen in bold text in the Table 7.3, the best performance for non-planar SuperGTO to GEO mission scenario is achieved with the DT algorithm with random state = 0. The mean error of DT algorithm with random state = 1 is 0.6133 days. The second

Table 7.3: *Mean error in days (square root of MSE rounded to 4 decimal points) for 10 di erent con gurations of DT algorithm from 0 to 9 RSV for non-planar SuperGTO to GEO transfer.*

RSV	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
ME	0.6139	**0.6133**	0.6182	0.6183	0.6153	0.6161	0.6177	0.6141	0.6196	0.6174

best performer, is the DT algorithm with random state = 0 with a mean error of 0.6139 days which is 0.10% higher than the mean error of DT algorithm with random state = 1. The third best performance is achieved with random state = 7 with a mean error of 0.6141 days that is 0.03% higher than the mean error of RF algorithm with random state = 0.

7.1.4 Comparison of Optimized DNN versus SVM, RF, and DT

Table 7.4 compares the mean error for our best-performing DNN (Deep Neural Network) trained using LM optimizer versus SVM, RF and DT algorithms. DNN-4 with 110 neurons is the best performing algorithm with a ME of 0.0435 . RF with 30 estimators is the second best performer with a ME of 0.4041 that is 9.29× better than the ME of DNN. SVM with C = 0.9 and Epsilon = 0.1 has a ME of 0.6032 that is 13.87× higher than the ME of the best performing DNN. DT with random state = 1 has a ME of 0.6113 that is 14.05× higher than the ME of the best performing DNN.

Table 7.4: *Comparison of mean error (actual values and times improvement) for the best DNN trained using LM optimizer, SVM, RF and DT for non-planar SuperGTO to GEO transfer. $\mathcal{I}$ denotes the improvement provided by our best-performing DNN.*

Algorithm	**DNN**	**RF**	**SVM**	**DT**
ME in Days	**0.0435**	0.4041	0.6032	0.6113
$\mathcal{I}$	**N/A**	9.29×	13.87×	14.05×

7.2 Planar SuperGTO

In this section, first we obtain the results of mean error for transfer time prediction for planar SuperGTO to GEO transfers using SVM, RF, and DT. After that, we provide a comparison of our designed DNNs using the LM algorithm with SVM, RF, and DT is provided.

7.2.1 Support Vector Machine

Table 7.5 contains the results (mean error in days) for SVM algorithm with values of C and ϵ varying from 0.1 - 0.9 with an increment of 0.2 in each subsequent SVM trained. In total we have 25 configurations of SVMs that we have reported in this thesis.

Table 7.5: *Mean error in days (square root of MSE rounded to 4 decimal points) for each subsequent con guration for planar SuperGTO to GEO transfer.*

ϵ / C	**0.1**	**0.3**	**0.5**	**0.7**	**0.9**
0.1	2.1524	2.2899	2.3458	2.2086	2.2637
0.3	1.1975	1.1996	1.2167	1.2181	1.2990
0.5	0.9009	0.8971	0.9450	0.9462	0.9854
0.7	0.7912	0.7891	0.7916	0.8567	0.8979
0.9	**0.7115**	0.7305	0.7183	0.7602	0.8097

As it can be seen in bold text in the Table 7.5, the best performance for the mission scenario is coming from the SVM with $C = 0.9$ and $Epsilon = 0.1$ with a mean error of 0.7115 days. The second best performer is SVM with $C = 0.9$ and $Epsilon = 0.5$ with mean error of 0.7183 days that is 0.96% higher than the SVM with $C = 0.9$ and $Epsilon = 0.1$. The third

best in performance is SVM with $C = 0.9$ and $Epsilon = 0.3$ with a mean error of 0.7305 days that is 1.70% higher than the second best performing SVM with $C = 0.9$ and $Epsilon = 0.5$.

7.2.2 Random Forest

Table 7.6 contains the results (error in days) for Random Forest algorithm with a varying number of estimators from 10 to 110. The first row in the table (header) displays the total number of estimators (#Est) we have used in Random Forest. The second row is the results presented as mean error (ME) in days.

Table 7.6: *Mean error in days (square root of MSE rounded to 4 decimal points) for 11 di erent con gurations of RF algorithm from 10 to 110 estimators (#Est) with an increment of 10 estimators in each subsequent con guration for planar SuperGTO to GEO transfer.*

#Est	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**	**110**
ME	0.5147	0.5086	0.5205	0.5065	0.5131	0.5141	**0.5022**	0.5035	0.5172	0.5112	0.5158

As it can be seen in bold text in the Table 7.6, the best performance for planar SuperGTO to GEO mission scenario is achieved with the RF algorithm with 70 estimators. The mean error of RF algorithm with 70 estimators is 0.5022 days. The second best performer, is the RF algorithm with 80 estimators with a mean error of 0.5035 days which is 0.25% higher than the mean error of RF algorithm with 70 estimators. The third best performance is achieved with 40 estimators with a mean error of 0.5065 days that is 0.59% higher than the mean error of RF algorithm with 80 estimators.

7.2.3 Decision Tree

Table 7.7 contains the results (error in days) for Decision Tree (DT) algorithm with a varying random state from 0 to 9. The first row in the table (header) displays the random state value

(RSV) used to train DT algorithm. The second row is the results presented as mean error (ME) in days.

Table 7.7: *Mean error in days (square root of MSE rounded to 4 decimal points) for 10 di erent con gurations of DT algorithm from 0 to 9 RSV for planar SuperGTO to GEO transfer.*

RSV	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
ME	**0.6074**	0.6168	0.6260	0.6184	0.6134	0.6157	0.6191	0.6249	0.6254	0.6131

As it can be seen in bold text in the Table 7.7, the best performance for planar SuperGTO to GEO mission scenario is achieved with the DT algorithm with random state = 0. The mean error of DT algorithm with random state = 0 is 0.6074 days. The second best performer, is the DT algorithm with random state = 9 with a mean error of 0.6131 days which is 0.94% higher than the mean error of DT algorithm with random state = 0. The third best performance is achieved with random state = 4 with a mean error of 0.6134 days that is 0.05% higher than the mean error of RF algorithm with random state = 9.

7.2.4 Comparison of Optimized DNN versus SVM, RF, and DT

Table 7.8 compares the mean error for best DNN (Deep Neural Network) trained using LM (Levenberg Marquardt) optimizer, SVM, RF and DT algorithms.

Table 7.8: *Comparison of mean error (actual values and times improvement) for the best DNN trained using LM optimizer, SVM, RF and DT for planar SuperGTO to GEO transfer. $\mathcal{I}$ denotes the improvement provided by our best-performing DNN.*

Algorithm	**DNN**	**RF**	**DT**	**SVM**
ME	**0.0028**	0.5022	0.6074	0.7115
$\mathcal{I}$	**N/A**	179.39×	216.92×	254.11×

DNN-3 with 70 neurons is the best performing algorithm with ME 0.0028. RF with 70 estimators is the second best performer with a ME of 0.5022 that is 179.39× better than

ME of DNN. DT with random state = 0 has a ME of 0.6074 that is 216.92× higher than ME of the best performing DNN. SVM with C = 0.9 and Epsilon = 0.1 has a ME of 0.7112 that is 254.11× higher than the ME of the best performing DNN.

Chapter 8

Conclusion

Spacecraft orbit-raising trajectory problem is a continuous-space problem. The work has deep neural networks (DNNs) trained on multiple trajectories for each of the six mission scenarios. These can both interpolate and extrapolate to states in the space not present in the training set. Markov Decision Processes (MDPs) can be used to solve this problem using deep reinforcement learning (DRL) agents. This thesis designs DNNs for Q-value (transfer time) prediction for six orbit-raising trajectories involving transfers from planar and non-planar Sub-GTO, geostationary transfer orbit (GTO), and Super-GTO to geostationary Earth orbit (GEO). These DNNs can guide the DRL agent in a continuous space to correctly raise its orbit. Results indicate that for each of the six trajectories, our best performing DNN designs attain an accuracy of over 99%. Generating this direction for the DRL through mathematical optimization would be computationally expensive because it has to generate the complete trajectory to reach a certain point and from that point to the GEO in order to direct the spacecraft.

We also compare the results from our designed DNNs using the Levenberg Marquardt (LM) algorithm with the contemporary Machine Learning (ML) algorithms, such as Support Vector Machine (SVM), Random Forest (RF), and Decision Tree (DT) for regression. Experimental results indicate that our best-performing DNN provides an improvement of mean error of 9.29×, 13.87×, and 14.05× over RF, DT, and SVM, respectively, for non-planar

superGTO to GEO transfers. Experimental results indicate that our best-performing DNN provides an improvement of mean error of 179.39×, 216.92×, and 254.11× over RF, DT, and SVM, respectively, for planar superGTO to GEO transfers.

The next work to be done on this problem, is to use these DNNs and use them to help in exploring the environment using deep reinforcement learning. The work is being done in ISCAAS Lab. A composite DNN reward function is also possible, where as the DNN model gets trained, it can rely less on the DNNs we trained in the thesis, and use it's own DNN to make itself better from the actual experience it has had in the environment.

Note: The generated trajectory data does not have the data for each possible state in the space. If a state that is not present in the data of trained models is reached, the model (specific to the mission scenario) might not give accurate expected time to GEO.

Bibliography

[1] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[2] Daniel S. Kolosa. A reinforcement learning approach to spacecraft trajectory optimization. page 3542, 2019.

[3] Stefan Willis, Dario Izzo, and Daniel Hennes. Reinforcement learning for spacecraft maneuvering near small bodies. 2016.

[4] Shaoming He, Hyo-Sang Shin, and Antonios Tsourdos. Computational missile guidance: A deep reinforcement learning approach. *Journal of Aerospace Information Systems*, 18 (8):571–582, 2021.

[5] Janusz Marecki, Sven Koenig, and Milind Tambe. A fast analytical algorithm for solving markov decision processes with real-valued resources. In *Proceedings of the 20th International Joint Conference on Arti cal Intelligence*, IJCAI'07, page 2536–2541, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[6] Lakshay Arora and Atri Dutta. Reinforcement learning for sequential low-thrust orbit raising problem. In *AIAA Scitech 2020 Forum*, page 2186, 2020.

[7] Hyeokjoon Kwon, Snyoll Oghim, and Hyochoong Bang. Autonomous guidance for multi-revolution low-thrust orbit transfer via reinforcement learning. 02 2021.

[8] J. T. Betts. Survey of numerical methods for trajectory optimization. 21(2):193–207, 1998.

[9] Malchow H. L. Sackett, L. L. and T. Delbaum. Solar electric geocentric transfer with attitude constraints: analysis. 1975.

[10] C. A. Kluever and S. R. Oleson. Direct approach for computing near-optimal lowthrust earth-orbit transfers. 35(4):509–515, 1998.

[11] M. W. Marasch and C. D. Hall. Application of energy storage to solar electric propulsion orbital transfer. 37(5):645–652, 2000.

[12] J. T. Betts. Very low-thrust trajectory optimization using a direct sqp method. 120 (1-2):27–40, 2000.

[13] A. L. Herman and B. A. Conway. Direct optimization using collocation based on high-order gauss-lobatto quadrature rules. 19(3):592–599, 1996.

[14] A. Dutta and L. Arora. Objective function weight selection for sequential low-thrust orbit-raising optimization problem. 168(AAS 19-567):1023–1038, 2019.

[15] K. F. Graham and A. V. Rao. Minimum-time trajectory optimization of low-thrust earth-orbit transfers with eclipsing. 53(2):289–303, 2016.

[16] A. E. Petropoulos and J. M. Longuski. Shape-based algorithm for the automated design of low-thrust, gravity assist trajectories. 41(5):787–796, 2004.

[17] B. J. Wall and B. A. Conway. Shape-based approach to low-thrust rendezvous trajectory design. 32(5):95–101, 2009.

[18] P. De Pascale and M. Vasile. Preliminary design of low-thrust multiple gravity-assist trajectories. 43(5):1065–1076, 2006.

[19] De Pascale P. Vasile, M. and S. Casotto. On the optimality of a shape-based approach based on pseudo-equinoctial elements. 61(1-6):286–297, 2007.

[20] D. M. Novak and M. Vasile. Improved shaping approach to the preliminary design of low-thrust trajectories. 34(1):128–147, 2011.

[21] E. Taheri and O. Abdelkhalik. Shape based approximation of constrained low-thrust space trajectories using fourier series. 49(3):535–546, 2012.

[22] C. A. Kluever. Simple guidance scheme for low-thrust orbit transfers. 21(6):1015–1017, 1998.

[23] A. E. Petropoulos. Simple control laws for low-thrust orbit transfers. 2003-0630.

[24] A. Petropoulos. Low-thrust orbit transfers using candidate lyapunov functions with a mechanism for coasting. 2004-5089.

[25] J. L. Junkins and E. Taheri. Exploration of alternative state vector choices for low-thrust trajectory optimization. 42(1):47–64, 2018.

[26] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. pages 161–168, 2006.

[27] Nicholas Blaine LaFarge. Autonomous guidance for multi-body orbit transfers using reinforcement learning. May 2020.

[28] Wang Xiao, Shi Peng, Wen Changxuan, and Zhao Yushan. An algorithm of reinforcement learning for maneuvering parameter self-tuning applying in satellite cluster. *Mathematical Problems in Engineering*, 2020.

[29] Kirk Hovell and Steve Ulrich. Deep reinforcement learning for spacecraft proximity operations guidance. *Journal of Spacecraft and Rockets*, 58:254–264, 2021.

[30] Jacob Broida and Richard Linares. 01 2019.

[31] Bo Li, Zhi peng Yang, Da qing Chen, Shi yang Liang, and Hao Ma. Maneuvering target tracking of uav based on mn-ddpg and transfer learning. *Defence Technology*, 17(2): 457–466, 2021. ISSN 2214-9147.

[32] Charles E. Oestreich, Richard Linares, and Ravi Gondhalekar. Autonomous six-degree-of-freedom spacecraft docking with rotating targets via reinforcement learning. *Journal of Aerospace Information Systems*, pages 1–12, 2021.

[33] Keidai Iiyama, Kento Tomita, Bhavi Jagatia, Tatsuwaki Nakagawa, and Koki Ho. Deep reinforcement learning for safe landing site selection with concurrent consideration of divert maneuvers. *ArXiv*, abs/2102.12432, 2021.

[34] Maksim Gennadievich Shirobokov, S. P. Trofimov, and Mikhail Ovchinnikov. Survey of machine learning techniques in spacecraft control design. *Acta Astronautica*, 186:87–97, 2021.

[35] Suwat Sreesawet and Atri Dutta. Fast and robust computation of low-thrust orbit-raising trajectories. *Journal of Guidance, Control, and Dynamics*, 41(9):1888–1905, 2018.

[36] Pardhasai Chadalavada, Tanzimul Farabi, and Atri Dutta. Sequential low-thrust orbit-raising of all-electric satellites. *Aerospace*, 7(6):74, 2020.

[37] J. M. Cass. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 32 (1):94–95, 1983. ISSN 00359254, 14679876.

[38] F. H. C. Marriott. *Biometrics*, 41(2):596–596, 1985. ISSN 0006341X, 15410420.

[39] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. 2015.

[40] H.P. Gavin. The levenberg-marquardt method for nonlinear least squares curve-fitting problems. September 2020.