POLITECNICO DI TORINO

Master's Degree in Aerospace Engineering

Politecnico di Torino

TEXAS

The University of Texas at Austin

Master's Degree Thesis

Low-Thrust Cislunar Trajectory Optimization leveraging Neural Networks for autonomous and real time applications

Supervisors

Prof. Maruthi Akella

Prof. Manuela Battipede

Candidate

Simone Amorosino

A.Y. 2021/2022

Summary

In recent years the need of an augmented spacecraft autonomy has considerably increased along with the renewed interest in Lunar missions.
Low-Thrust Cislunar trajectory optimization is typically achieved leveraging classical optimal control methods. These methods can be time expensive, challenging for mission analysts and they often require nontrivial computational effort, due to the non-linear nature of the three body dynamics problem.
The aim of the thesis is to develop a neural network (NN) to re-compute the optimal control history of a transfer maneuver autonomously on board, in shorter time and with low computational effort.
The main core of the work logic is the collection of different datasets of optimal transfer maneuvers between typical lunar orbits. A classical optimal control problem (OCP) with a minimum control energy cost function is built to provide optimal solutions. Using an indirect method the OCP is reduced to a Two Point Boundary Value Problem, with initial and final states fixed on the orbits, that is solved by the Matlab bvp4c solver. A state's discretization on both orbits is taken into account to consider different transfers. Moreover in order to cover different kind of mission plans, both time fixed and free transfer time formulations are studied.
As is common knowledge one of the main drawbacks of an indirect approach is the sensitivity on the initial guesses. Therefore the implementation of an indirect method on board could be unfeasible if a real time trajectory planning is needed, because the solver may could not converge rapidly to an optimal solution without reasonable initial guesses.
As a typical approach in the literature, during the data collection, random initial guesses are assigned several times and the same OCP is solved iteratively, saving the minimum energy solution. The initial random guesses that lead the method to converge are then slightly perturbated in order to check if the solution corresponds with a suitable local minimum, considering the well-known inability to find a global minimum in this chaotic dynamics. This latter solution is then used as initial guess for a similar OCP, as in a continuation method, to expedite the data collection.
Finally, the neural network can be trained on this data in order to map a relationship between an input vector, made of initial states and final states, and an

output vector characterized by initial costates and the transfer time in the free time problem. The NN can work as a function call on board, it can be used as a fast tool to provide adequate initial guesses to bvp4c, reducing drastically the computational time of the method and making possible to plan an orbital transfer without ground station involvement.

Acknowledgements

It is hard to express in few words my gratitude for all the people that stayed with me along the way, but I will give it a try. I would like to thanks the professor Maruthi Akella, without him this unforgettable experience abroad would not be possible. A special mention goes to Manuela Battipede, that made me develop a huge passion for orbital mechanics and pushed me to pursue this thesis experience. A special thanks goes to Stefano, a friend that I found in university's projects calls during Covid and with whom later I shared an indelible experience across the ocean. Thanks to Salvatore and his precious advice.
I bumped into lots of special people in these years, a sincere thanks goes to them and also to lifetime friends.
Finally, an heartfelt thanks goes to my family that made all of this possible. Despite the distance, they always supported me in my choices, feeding my passions and encouraging me to follow my own road.

"Life is our resume. It is our story to tell, and the choices we make write the chapters. Can we live in a way where we look forward to looking back?"

Matthew McConaughey

Ringraziamenti

È difficile esprimere in poche righe la mia gratitudine verso chi mi ha accompagnato in questo percorso, ma ci proverò lo stesso. Vorrei ringraziare il professor Maruthi Akella senza il quale quest'indimenticabile esperienza all'estero non sarebbe stata possibile. Una menzione speciale va alla professoressa Manuela Battipede che mi ha fatto appassionare alla meccanica orbitale e mi ha spronato fin da subito ad intraprendere questo percorso di tesi.
Un grazie va a Stefano, un amico che ho trovato dietro uno schermo nei primi mesi di Covid e con il quale poi ho condiviso un'esperienza indelebile oltreoceano. Grazie a Salvatore e ai suoi preziosi consigli. Un pensiero va anche a tutti gli amici incontrati lungo il cammino e a quelli di una vita.
Infine un grazie di cuore va alla mia famiglia, che ha reso possibile tutto questo e, nonostante la lontananza, ha sempre supportato le mie scelte, alimentando le mie passioni e incoraggiandomi a seguire la mia strada.

Table of Contents

List of Figures

Acronyms

NASA
National Aeronautics and Space Administration

ESA
European Space Agency

JAXA
Japan Aerospace eXploration Agency

CSA
Canadian Space Agency

LEO
Low Earth Orbit

MEO
Medium Earth Orbit

GEO
Geostationary Orbit

SOI
Sphere Of Influence

CSSA
Cislunar Space Situational Awareness

CDA
Cislunar Domain Awareness

CSTM

Cislunar Space Traffic Management

2BP

Two Body Problem

3BP

Three Body Problem

CR3BP

Circular Restricted Three Body Problem

JWST

James Webb Space Telescope

SOHO

Solar and Heliospheric Observatory Satellite

CoM

Center of Mass

DROs

Distant Retrograde Orbits

NRHO

Near Rectilinear Halo Orbit

OCP

Optimal Control Problem

NLP

Nonlinear Programming Problem

DoF

Degrees of Freedom

BVP

Boundary Value Problem

DAE

Differential Algebraic Equations

TPBVP

Two Point Boundary Value Problem

w.r.t.

With respect to

ODE

Ordinary Differential Equation

ANN

Artificial Neural Networks

NN

Neural Networks

Introduction

The idea of the thesis comes from the recent need for augmenting spacecraft onboard autonomy capabilities. Planning a maneuver between periodic orbits in a Three Body Dynamics environment is a demanding activity even on ground.
Therefore, trying to compute an optimal maneuver live on board is surely challenging. A lot of ongoing research is going on in these last years in order to provide feasible solutions. In [1] and [2] for example, a Reinforcement learning (RL) approach is investigated. Since the scientific community is still divided upon the reliability of RL in such applications, in this thesis we are going to explore the possibility to exploit feedforward neural networks as a possible on board solution to re-compute the optimal control. The idea is to map a relationship to find initial costates, or adjoints, in order to leverage the Pontryagin's minimum principle and dynamics equations to rebuild the optimal trajectory and its controls. This neural networks prediction is then propagated and given to the Two Point Boundary Value Problem (TPBVP) solver as initial solution, in order to improve the convergence in brief time and making possible to compute a transfer in real time applications.
The structure of the thesis is the following:

- **Chapter 1**: is an overview of the **Cislunar environment** from an astrodynamical point of view. Thus, a particular focus is given to the Circular Restricted Three Body Problem (CR3BP). A second order controlled differential system is obtained in order to have a physical model to describe the motion of a small mass (e.g. a spacecraft) subject to the gravitational pools of larger masses. In the second part of the chapter, typical periodic orbits (i.e. Lyapunovs, Distant Retrograde Orbits and Halos) are studied along with their peculiar characteristics.

- **Chapter 2**: is a survey of **Trajectory optimization** methods and techniques. Firstly, important concepts of the classical optimal control theory are recalled, with the definition of the Optimal Control Problem (OCP). This problem is reduced to a TPBVP thanks to an indirect method formulation that leverages Pontryagin's minimum principle to find the optimal control vector. Since the problem does not have closed analytical solution, the most common numerical

techniques are presented in the second part of the chapter (i.e. single shooting, multiple shooting, collocation).

- **Chapter 3**: in the first part the foundation of **Neural Networks** is pointed out. Then, some important advice on data collection and data split, that must be followed in order to avoid typical problems (e.g. over-fitting) and enhance the learning process, are given to the reader. Finally, at the end of the chapter is explained how neural networks can be implemented in Matlab's environment leveraging the Deep Learning Toolbox ®.

- **Chapter 4**: the first trajectory optimization problem considered in the thesis is a **Lyapunov L_1 - Lyapunov L_2** transfer. As first step, a time-fixed formulation is studied in order to familiarize with the optimal transfer maneuvers and the initial and final orbits. Then, a free time formulation is implemented. The data collection algorithms, the neural networks performance and the final results are described for both the formulations. As final section of the chapter, a promising forward-backward technique to solve the error propagation is discussed.

- **Chapter 5**: the second trajectory optimization problem considered in this work is a **DRO-Halo** transfer, thus a three-dimensional maneuver. In this case we take into account just the free time formulation which is the more plausible in a typical mission. As in the previous chapter, data collection, net performance and final results are presented. At the end of the chapter the forward-backward method is discussed.

Chapter 1

Cislunar environment

For millennia humanity has gazed in awe at the Moon. Prior to the space race in the 1960s, however, we could only look up to our neighbour satellite with telescopes. Aiming at the Moon was the most natural step in the early age of space era.
The NASA's Apollo program broke new ground in space exploration, enabling missions beyond our planet boundaries for the first time. Nowadays the Moon is the only celestial body in which we left our footprint.
Although plenty of missions have been conducted in order to study our satellite, we have never walked again the Lunar surface in 50 years. After the space race the main space agencies all over the globe shifted their focus on Low Earth Orbits (LEO) missions, due to the grown of both political and economical interests.

Figure 1.1: View of the Moon - NASA

Despite this, the scientific community's enthusiasm in studying the Moon has not waned. A total of 382 kg of lunar soil, rock, and core samples were returned to Earth by astronauts between 1969 and 1972. Recently [3], using samples from Apollo 11, 12 and 17 researchers showed the growth of terrestrial plant in diverse Lunar regoliths. The study of this challenging growth also helps biologists to better understand the behaviour of terrestrial plants in hostile and extreme environmental conditions on Earth. On the other hand in order to establish a permanent presence on the Moon, is crucial to understand how in-situ resources could be exploited to support life in an extraterrestrial habitat. As a matter of fact is well known that, in recent years, NASA is making a comeback, once again aiming at the Moon, leading the Artemis program in conjunction with ESA, JAXA and CSA.
Accordingly with the increasing economical and scientific interest on the Lunar realm [4] a deeper knowledge of Cislunar Space Situational Awareness (CSSA), Cislunar Domain Awareness (CDA), and Cislunar Space Traffic Management (CSTM) is highly needed.

In order to achieve CSSA, CDA, CSTM and for the purpose of this thesis, an excursus on Cislunar[1] dynamical environment is essential. Typically when dealing with common missions near the Earth (e.g. LEO, MEO, GEO), the gravitational effect of the Moon is neglected. That is because the Lunar sphere of influence[2] is small compared to the Earth one. For this reason a satellite orbiting the vicinity of the Earth just feels the gravitational effect of our planet (i.e. the Two Body Problem (2BP), based on Newton's Gravitational Theory can be used to model the physics of the problem with high accuracy).
Aside from the near-Lunar region, the dynamical environment in the Cislunar realm is different, with small accelerations that are not always directed towards a single central gravitational body. The force environment in the Cislunar space is illustrated in Fig.(1.2). in order to provide a comparison to the near-Earth region. [7] Besides the regions in the vicinity of the Earth and the Moon, with a cutoff of 0.009 m/s^2 in the middle region one can see the variety of accelerations that is far from being completely flat.

[1]Cislunar space from the Latin preposition Cis ("on this side of") and Lunar, indicates the space contained within the Moon's orbit. Practically it can be seen as the sphere obtained rotating this orbit around the Earth. Despite this, typically when discussing Cislunar topics, we need to consider a larger volume of space. In multi-year transfer trajectories some transfers between Earth and Cislunar space extend well beyond the Moon, to distances in excess of 1.5 million kilometers (35 GEO; GRAIL, THEMIS/ARTEMIS). [5]

[2]In celestial mechanics is useful to determine the region where the gravitational influence of a particular body prevails over the influence of other bodies. From this need comes the concept of the sphere of influence (SOI). [6]

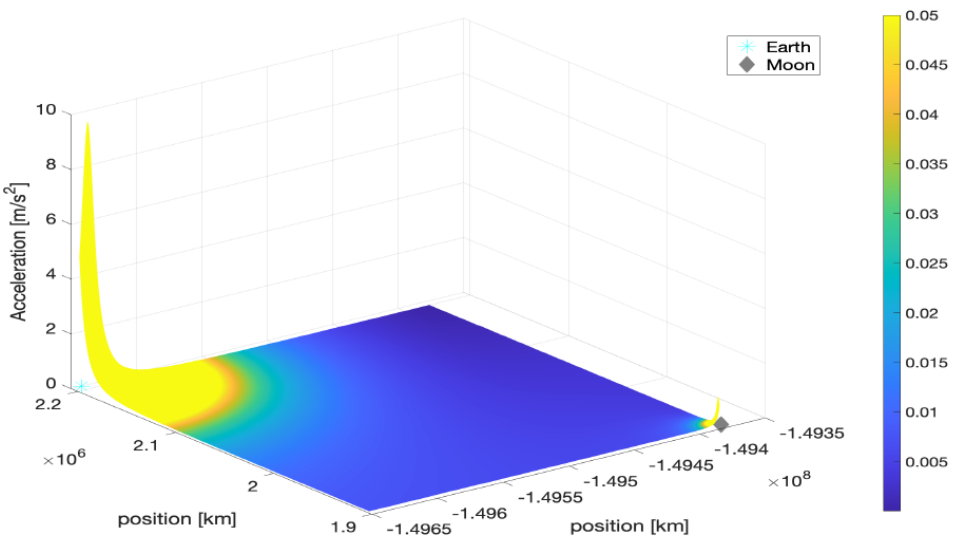


Figure 1.2: Acceleration due to combined effect of Earth, Solar, Lunar and Jupiter gravity and Solar radiation pressure for a specific spacecraft, expressed in $\frac{m}{s^2}$. [7]

From this brief introduction is clear that Cislunar dynamics differs from the classical Keplerian Theory (i.e. 2BP). Adding the gravitational effect of a third object (i.e. the Moon) to a classical system, leads to significant and challenging variation. The main differences between the 2BP and the Three Body Problem (3BP) can be summarized as follows:

- Trajectories are no longer circular, elliptical, etc;
- Trajectories are no longer planar;
- Trajectories are no longer easy to geometrically describe.

In addition trajectories in a 3BP evolve over time, and this leads to difficulties regarding the choice of the reference frame. There are two main options: [5]

- 3BP trajectories are visualized in the inertial frame;
- 3BP trajectories are visualized in a rotating non-inertial frame that moves with both Earth and Moon as they rotate about their center of mass.

The latter one is the most widely used and it lays the foundation for the Circular Restricted Three Body Problem (CR3BP).

1.1 Circular Restricted Three Body Problem

For centuries mathematicians and physicists have been fascinated by the 3BP.
The problem can be stated as follows: "Three particles move in space under their mutual gravitational attraction; given their initial conditions, determine their subsequent motion". [8]
Newton was the first to try to determine the differential equations but the problem remained relatively unsolved until 1890, when Poincare made innovative studies. Although the 18 first order differential equation can be reduced to 6 through use of conservation equations and calculus [9], the problem has still not been solved because of the lack of conservation quantities for further simplification.

From that came the need for a more restricted model that would eventually predict the physics of the problem, at least for limited cases.
In the CR3BP two bodies revolve around their common center of mass in circular orbits under the influence of their mutual gravitational attraction. [10] The purpose of the problem is to figure out how a third body (i.e. a spacecraft attracted by the previous two without influencing their motion) moves as it is affected by the gravitational pulls of both larger masses, that are typically defined as *primaries*. Therefore the main difference between 3BP and CR3BP is that in the latter the third body has a mass much smaller in comparison to other two bodies and also the restricted problem requires circular orbits for the primaries.
Despite the assumptions of the CR3BP seem unfeasible, there are different examples in our solar system that can fit the model (i.e. Sun-Earth-Moon, Sun-Earth-spacecraft, Jupiter-Jupiter's moon-spacecraft, Earth-Moon-spacecraft,...).

1.1.1 Equations of motion

Now we are going to consider a spacecraft in the Cislunar environment. The vector differential equation for the spacecraft's motion is written in an inertial frame I according to Newton's Second Law: [11]

$$m_{sc}\frac{d^2\vec{R}}{d\tau^2} = -Gm_{\oplus}m_{sc}\frac{\vec{R}_1}{R_1^3} - Gm_{\leftmoon}\, m_{sc}\frac{\vec{R}_2}{R_2^3} + \vec{T} \tag{1.1}$$

where: m_{sc} is the mass of the spacecraft, $m_{\oplus}$ is the Earth's mass and $m_{\leftmoon}$ the mass of the Moon. In addition $\vec{R}$ is the spacecraft vector position, $\vec{R}_1$ is the vector Earth-spacecraft and $\vec{R}_2$ the Moon-spacecraft vector, $\vec{T}$ is the thrust vector, G is the Gravitational costant and $d\tau$ is the time unit.
At this point usually the first step to obtain the CR3BP formulation is a standard

nondimensionalization. As already stated the mass of the spacecraft is negligible. Different characteristic can be written as in Tab.(1.1):

Characteristics	Formula
Mass	$m^* = m_\oplus + m_\leftmoon$
Length	$l^* = R_{\oplus\rightarrow\leftmoon}$
Time	$\tau^* = \sqrt{\frac{l^{*3}}{Gm^*}}$

Table 1.1: The three main characteristics of the CR3BP

where $R_{\oplus\rightarrow\leftmoon}$ (sometimes defined as R_{12}) is the Earth-Moon distance and the characteristics time is computed from Kepler's third law in order to make the non-dimensional gravitational constant G^* equal to 1.
New non-dimensional quantities can be defined as:

$$\vec{r} = \frac{\vec{R}}{l^*}, \quad \vec{r}_i = \frac{\vec{R}_i}{l^*} \quad \{i\} \in \{1{,}2\} \quad \mu = \frac{m_\leftmoon}{m^*} \quad \text{and} \quad t = \frac{\tau}{\tau^*} \tag{1.2}$$

In addition $\vec{T}$ is normalized as:

$$\vec{T} = T_{max}\vec{u} \quad |\vec{u}| \leq 1 \tag{1.3}$$

where T_{max} is the maximum thrust and $\vec{u}$ is the control vector.

Dividing now Eq.(1.1) by the mass m_{sc} and using Eq.(1.2) and Eq.(1.3) to simplify we get:

$$\frac{d^2\vec{r}}{dt^2} = -\frac{(1-\mu)}{r_1^3}\vec{r}_1 - \frac{\mu}{r_2^3}\vec{r}_2 + \left(\frac{l^{*2}}{m^*G}\right)\frac{T_{\max}}{m_{sc}}\vec{u} \tag{1.4}$$

In the last term we can regroup:

$$\varepsilon = \left(\frac{l^{*2}}{m^*G}\right)\frac{T_{\max}}{m_{sc}} \tag{1.5}$$

where $\left(\frac{l^{*2}}{m^*G}\right)$ is simply a normalization constant and so ε is the $\frac{T_{\max}}{m_{sc}}$ ratio in dimensionless units.

At this point the goal, as already mentioned, is to rewrite Eq.(1.4) in a rotating frame R. The angular velocity of this latter frame is the velocity of rotation of the two primaries (that are fixed in this frame) around their center of mass, which is also the origin of the frame as in Fig.(1.3).
Both the Earth and the Moon lie in the x axis with coordinates respectively of $(-\mu,0,0)$ and $(1-\mu,0,0)$. The Eq.(1.4) can be rewritten as:

$$\left(\frac{d^2\vec{r}}{dt^2}\right)_I = \left(\frac{d^2\vec{r}}{dt^2}\right)_R + 2\vec{\omega} \times \left(\frac{d\vec{r}}{dt}\right)_R + \vec{\omega} \times \vec{\omega} \times \vec{r} \tag{1.6}$$

where $\vec{\omega}$ (normalized equal to 1) is the angular velocity of R w.r.t. the inertial frame I. Now the first term of the right-hand side of Eq.(1.6) can be expanded

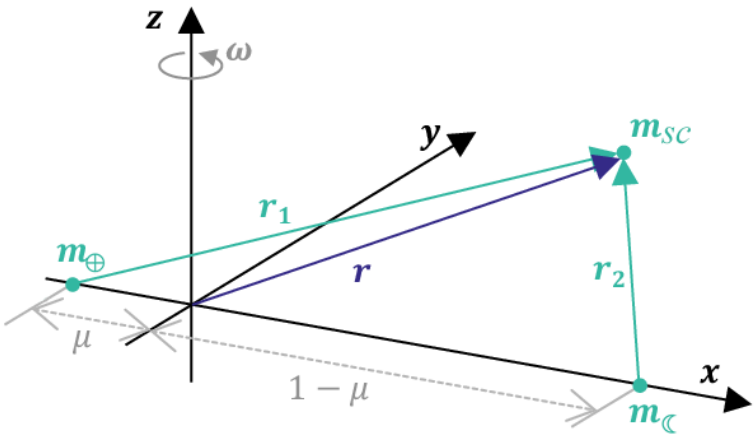


Figure 1.3: CR3BP typical reference frame [12]

recalling:

$$\begin{aligned} \vec{r} &= x\vec{x} + y\vec{y} + z\vec{z} \\ \left(\frac{d\vec{r}}{dt}\right)_R &= \dot{x}\vec{x} + \dot{y}\vec{y} + \dot{z}\vec{z} \\ \left(\frac{d^2\vec{r}}{dt^2}\right)_R &= \ddot{x}\vec{x} + \ddot{y}\vec{y} + \ddot{z}\vec{z} \end{aligned} \tag{1.7}$$

From Eq.(1.6) with Eq.(1.7) we obtain:

$$\left(\frac{d^2\vec{r}}{dt^2}\right)_I = (\ddot{x} - 2\dot{y} - x)\vec{x} + (\ddot{y} + 2\dot{x} - y)\vec{y} + \ddot{z}\vec{z} \tag{1.8}$$

Finally manipulating Eq.(1.4) and Eq.(1.8) we get the second order controlled differential system:

$$\begin{cases} \ddot{x} = 2\dot{y} + x - \frac{1-\mu}{r_1^3}(x+\mu) - \frac{\mu}{r_2^3}(x-1+\mu) + \varepsilon u_1 \\ \ddot{y} = -2\dot{x} + y - \frac{1-\mu}{r_1^3}y - \frac{\mu}{r_2^3}y + \varepsilon u_2 \\ \ddot{z} = -\frac{1-\mu}{r_1^3}z - \frac{\mu}{r_2^3}z + \varepsilon u_3 \end{cases} \tag{1.9}$$

where

$$\begin{aligned} \vec{r_1} &= (x+\mu)\vec{x} + y\vec{y} + z\vec{z} \\ \vec{r_2} &= (x-1+\mu)\vec{x} + y\vec{y} + z\vec{z} \end{aligned} \tag{1.10}$$

1.1.2 Lagrange Points

In the system (1.9) $\vec{u}$ is the control vector (u_1, u_2, u_3) in the rotating frame R and so when $\varepsilon = 0$ we have an uncontrolled free CR3BP.
Although this latter system has no closed analytical form, we can use it to find the location of special equilibrium points. The Lagrange points, also called Libration points, named from Joseph-Louis Lagrange's studies, are locations in space where the spacecraft (m_{sc}) would have zero velocity and zero acceleration. [13]
The spacecraft in these points would appear permanently at rest relative to both the primaries in the reference frame Fig.(1.3) and would appear in circular motion around primaries in the inertial frame I.
To determine the Lagrange points locations we must force the following conditions in the system (1.9):

$$\dot{x} = \dot{y} = \dot{z} = 0 \quad \text{and} \quad \ddot{x} = \ddot{y} = \ddot{z} = 0 \tag{1.11}$$

This yields to:

$$\begin{cases} 0 = x - \frac{1-\mu}{r_1^3}(x+\mu) - \frac{\mu}{r_2^3}(x-1+\mu) \\ 0 = y - \frac{1-\mu}{r_1^3}y - \frac{\mu}{r_2^3}y \\ 0 = -\frac{1-\mu}{r_1^3}z - \frac{\mu}{r_2^3}z \end{cases} \tag{1.12}$$

From the third equation:

$$\left(\frac{1-\mu}{r_1^3} + \frac{\mu}{r_2^3}\right)z = 0 \tag{1.13}$$

we can state that $z = 0$, since the term between round brackets can not be equal to zero for μ definition. Therefore the first conclusion is that these equilibrium points lie in the orbital plane. In addition from the second equation of (1.12):

$$0 = \left(1 - \frac{1-\mu}{r_1^3} - \frac{\mu}{r_2^3}\right)y \tag{1.14}$$

assuming $y \neq 0$, we obtain:

$$\frac{1}{r_2^3} = \frac{1}{\mu}\left(1 - \frac{1-\mu}{r_1^3}\right) \tag{1.15}$$

We can also manipulate the first equation of (1.12) rewriting that as:

$$0 = x\left(1 - \frac{1-\mu}{r_1^3} - \frac{\mu}{r_2^3}\right) + \mu\left(\frac{1}{r_2^3} - \frac{\mu}{r_2^3} - \frac{1-\mu}{r_1^3}\right) \tag{1.16}$$

Substituting Eq.(1.15) in Eq.(1.16) the dependency on x vanishes and we get:

$$\frac{1}{r_1^3} = 1 \tag{1.17}$$

From Eq.(1.17) into Eq.(1.15) we obtain:

$$\frac{1}{r_1^3} = \frac{1}{r_2^3} = 1 \Rightarrow r_1 = r_2 = 1 \tag{1.18}$$

Recalling Eq.(1.10) with $z = 0$ and Eq.(1.18) and expliciting the norm of both vectors we have:

$$\begin{aligned} 1 &= (x + \mu)^2 + y^2 \\ 1 &= (x - 1 + \mu)^2 + y^2 \end{aligned} \tag{1.19}$$

Equating the right-hand sides of these latter two equations leads to:

$$x = \frac{1}{2} - \mu \Rightarrow y = \pm\frac{\sqrt{3}}{2} \tag{1.20}$$

These are the coordinates of the first two Lagrangian points, L_4 and L_5. Recalling that the distance between the Earth and the Moon, $l^* = R_{\oplus\to\leftmoon}$, is the characteristic length of the system (i.e. the Earth-Moon distance is equal to 1 in the non-dimensional normalized frame Fig.(1.3)), and Eq.(1.18), the two primaries and these two Lagrange points lie at the vertices of equilateral triangles, as illustrated in Fig.(1.4).

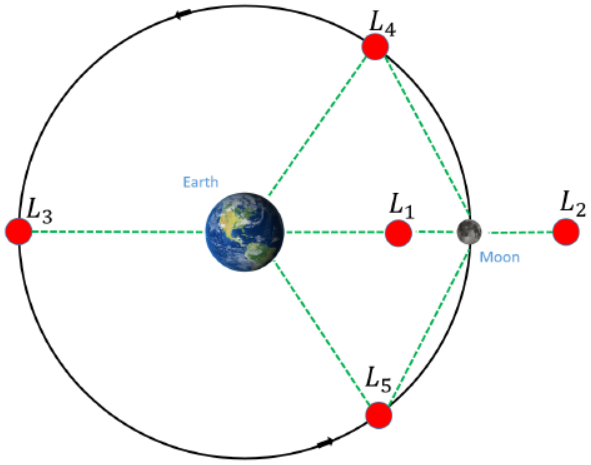


Figure 1.4: Lagrange points in the Earth-Moon system [14]

In order to find the others points (L_1, L_2, L_3) we have to consider the other case of study in Eq.(1.14), i.e. $y = 0$, and also $z = 0$ that still remains valid. Computing the magnitude of Eq.(1.10) we obtain:

$$\begin{aligned} |r_1| &= |x + \mu| \\ |r_2| &= |x + \mu - 1| \end{aligned} \tag{1.21}$$

and substituting these into the first equation of (1.12) leads to:

$$0 = x - \frac{1-\mu}{(x+\mu)^2} - \frac{\mu}{(x+\mu-1)^2} \tag{1.22}$$

The roots for this latter equation can be computed numerically, after the selection of the parameter μ. In the Earth-Moon system this leads to the numerical results reported in Fig.(1.5).

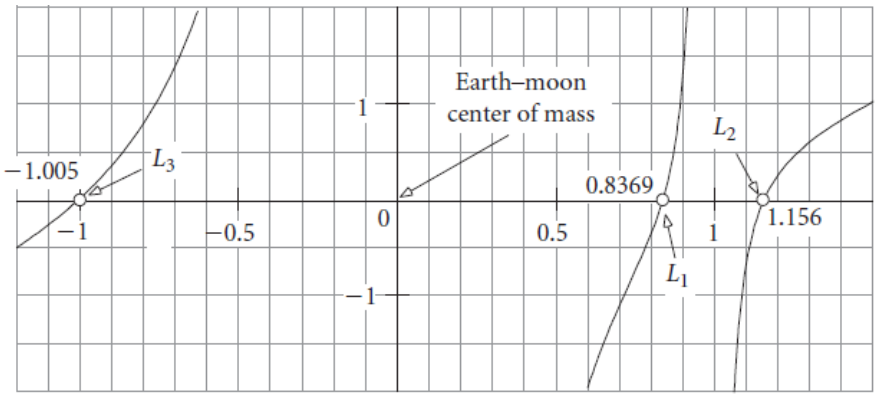


Figure 1.5: Graph for equation (1.22) for Earth-Moon data [13]

Studying the stability of Lagrangian points [15] it turns out that L_1, L_2 and L_3 are *unstable*. If a spacecraft is placed in these regions, an hypothetical perturbation could make it drifts away from the starting point. Despite these points are unstable and require station-keeping, different interesting orbits exist in their neighbourhood as explained in Sec.(1.2). Plenty of missions leverage these points [16], e.g. SOHO is now based at the L_1 point of the Earth-Sun system, which provides an uninterrupted view of the sun. L_2 has been leveraged by JWST for its peculiarities. This is indeed a great location for astronomy since a spacecraft can connect with Earth easily, maintaining the Sun, Earth, and Moon behind itself for solar power (with sufficient shielding), and providing a clear view of deep space.
On the other hand L_4 and L_5 are *stable*, a spacecraft nudged out of position near these points would return to its starting point. In the Earth-Moon system, indeed, these latter points are destabilized by the influence of the sun's gravity and

station-keeping maneuvers are required.
Usually objects orbiting at the L_4 and L_5 points are known as Trojans, after the three massive asteroids Agamemnon, Achilles, and Hector, which can be found in the Jupiter-Sun system's L_4 and L_5 points.

1.1.3 Jacobi Constant

Introducing the potential function:

$$V_\mu(x,y,z) = -\frac{1-\mu}{r_1} - \frac{\mu}{r_2} - \frac{1}{2}\left(x^2+y^2\right) \tag{1.23}$$

the system (1.9) can be rewritten as:

$$\begin{cases} \ddot{x} - 2\dot{y} + V_{\mu_x} = \varepsilon u_1 \\ \ddot{y} + 2\dot{x} + V_{\mu_y} = \varepsilon u_2 \\ \ddot{z} + V_{\mu_z} = \varepsilon u_3 \end{cases} \tag{1.24}$$

where $V_{\mu_x} = \frac{\partial V_\mu}{\partial x}, V_{\mu_y} = \frac{\partial V_\mu}{\partial y}$ and $V_{\mu_z} = \frac{\partial V_\mu}{\partial z}$.
Now multiplying the first equation by $\dot{x}$, the second by $\dot{y}$, the third by $\dot{z}$ and forcing $\varepsilon = 0$, to deal with an uncontrolled system, we get:

$$\begin{cases} \ddot{x}\dot{x} - 2\dot{y}\dot{x} + V_{\mu_x}\dot{x} = 0 \\ \ddot{y}\dot{y} + 2\dot{x}\dot{y} + V_{\mu_y}\dot{y} = 0 \\ \ddot{z}\dot{z} + V_{\mu_z}\dot{z} = 0 \end{cases} \tag{1.25}$$

Summing the left and right sides of these equations and rearranging terms we obtain:

$$\frac{1}{2}\frac{d}{dt}(\dot{x}^2+\dot{y}^2+\dot{z}^2) + \frac{d}{dt}(V_\mu) = 0 \tag{1.26}$$

or:

$$\frac{d}{dt}\left(\frac{V^2}{2} + V_\mu\right) = 0 \tag{1.27}$$

that leads to the Jacobi constant:

$$2C = V^2 + 2V_\mu \tag{1.28}$$

where V^2 is the kinetic energy per unit mass and in V_μ, Eq.(1.23), the first two terms are the gravitational potential energies of the primaries, while the last term[3] $-\frac{1}{2}(x^2+y^2)$ may be interpreted as the potential energy of the centrifugal force per

[3]Note that this term in a dimensional non-normalized formulation would be preceded by ω^2, that here it is equal to 1

unit mass induced by the rotation of the reference frame.
The constant C is well-known in astrodynamics and it is called Jacobi constant. This term can be interpreted as the total energy of the secondary particle relative to the rotating frame. C is a constant of motion of the secondary mass just as angular momentum and energy in a classical 2BP.
Since V^2 is always greater than or equal to zero, recalling Eq.(1.28) we can state:

$$2C - 2V_\mu \geq 0 \tag{1.29}$$

with Eq.(1.23):

$$2C + \frac{2(1-\mu)}{r_1} + \frac{2\mu}{r_2} + \left(x^2 + y^2\right) \geq 0 \tag{1.30}$$

A secondary body, i.e. a spacecraft in our case, is not allowed to run a trajectory that does not respect this disequality constraint.
The boundaries between allowed and forbidden regions, that a spacecraft is not allowed to cross, can be found setting Eq.(1.30) equal to zero:

$$2C + \frac{2(1-\mu)}{r_1} + \frac{2\mu}{r_2} + \left(x^2 + y^2\right) = 0 \tag{1.31}$$

For a given value of the Jacobi constant the zero velocity curves can be found. Note that C can assume only negative values at the zero velocity curves, since the other three terms in Eq.(1.31) are always positive. Large negative values of C suggest that the spacecraft is far away from the CoM of the system (i.e. large $(x^2 + y^2)$ value) or is close to the Earth or the Moon (i.e. small values of r_1 or r_2).
In Fig.(1.6) forbidden regions are illustrated for increasing values of Jacobi's constant. For a low value of C_0 (a) a spacecraft cannot reach the Moon or escape the Earth-Moon system. Using the L_1, L_2 and L_3 coordinates in Eq.(1.31) leads to greater value of C_1, C_2 and C_3. The spacecraft is able to reach respectively L_1 (b) ,L_2 (c) and L_3 (d) arriving at these points with zero velocity. C_2 depicts the minimal energy required for a spacecraft to leave the Earth–Moon system via a short corridor around the moon, as shown in (c). As C increases, the tunnel becomes wider, and at C_3, escape in the opposite direction from the moon becomes conceivable.
L_4 and L_5 are surrounded by the last remnants of the restricted zones (e). With a higher Jacobi constant, an Earth-launched spacecraft can reach the entire Earth–Moon system and beyond (f).

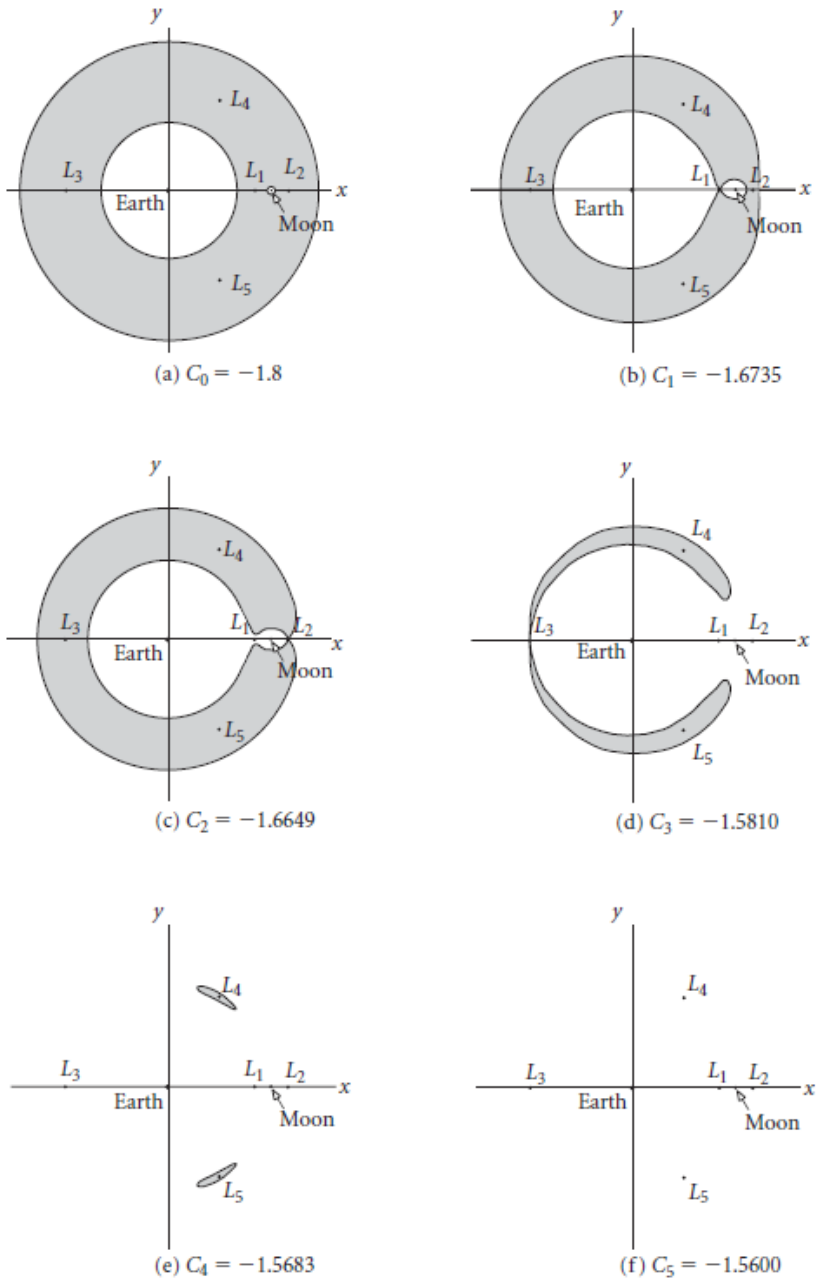


Figure 1.6: Forbidden regions (shaded) within the Earth–Moon system for increasing values of Jacobi's constant (km^2/s^2). [13]

1.2 Periodic orbits

The CR3BP is characterized by an infinite number of periodic solutions. Generally orbits and transfer maneuvers with similar features are collected into families. In this thesis we are going to consider the most useful orbits that are typically used in a three body mission planning. Periodic orbits in the CR3BP have been widely investigated and some examples of common families include Lyapunov and Halo orbits around the collinear libration points, and families of orbits around the Moon, such as Distant Retrograde Orbits (DROs). The L_1 and L_2 libration points lie approximately 58000 km and 64000 km from the center of the Moon and consequently orbits around these points are often leveraged for trajectory design involving lunar proximity operations. [17]

1.2.1 Lyapunov Orbits

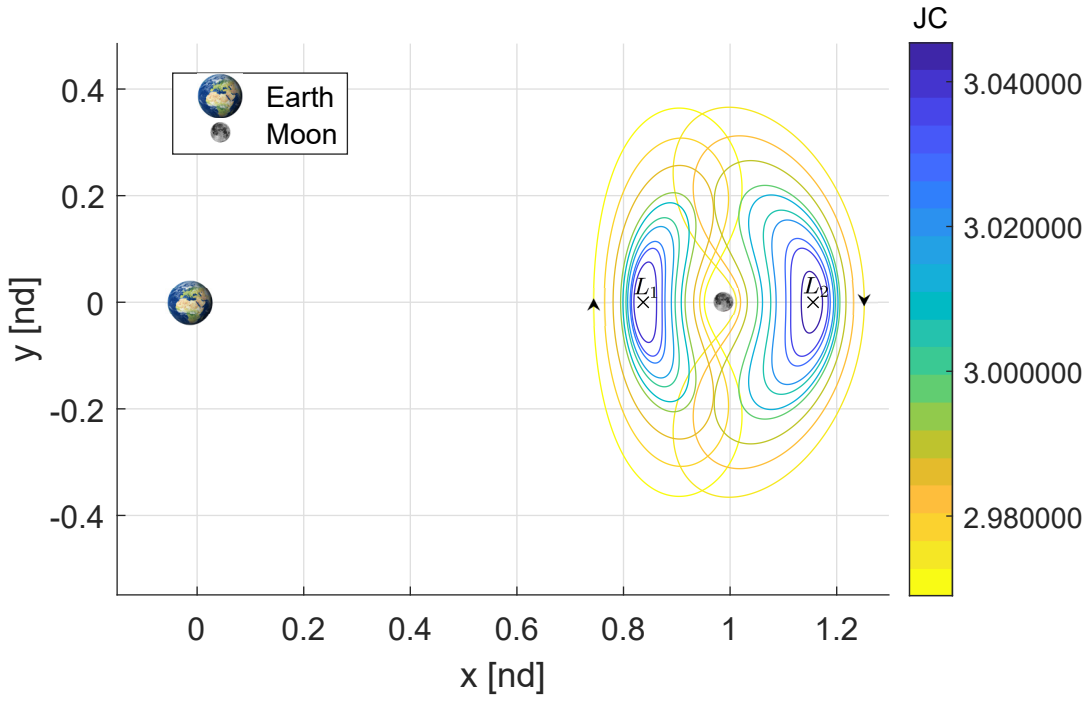
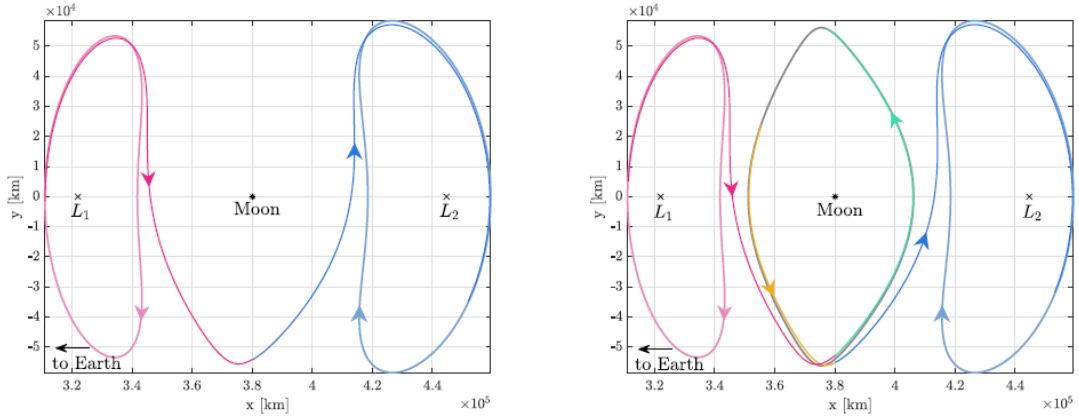


Figure 1.7: L_1 and L_2 Lyapunov orbits for different C levels.

Lyapunov orbits are typically planar orbits around all the five lagrangian points.

In this thesis we are going to consider just the Lyapunovs around L_1 and L_2. Transfers between these two kind of orbits will be investigated due to the growing interest that Artemis mission has brought in the last years. Connecting these two Lyapunovs orbits, indeed, one could assure continuous observation of both lunar sides. In Fig.(1.7) orbits with different C levels are illustrated (Earth and Moon size not in scale).
Despite just few of the infinite possible solutions are reported, it can easily seen that in some region L_1 and L_2 Lyapunov orbits merge into each other.
The CR3BP dynamics is highly chaotic and both these Lagrangian points are unstable as explained in the section before. Nevertheless this instability makes transfers between different natural orbits easier. With a small maneuver, in fact, a spacecraft can exit its unstable orbit and join an unstable or stable natural path, usually referred as manifold, [18] in order to make a transfer to a different periodic natural orbit minimizing the fuel cost as depicted in Fig.(1.8).
A lot of ongoing research is trying to find transfer trajectories that trade fuel cost against transfer time leveraging natural manifold. [19]
During mission design we usually look for the lower possible mission's cost, i.e. fuel minimization, but sometimes space missions require to follow a tight timeline and from this comes the need to find transfer maneuvers that do not attempt to heavily leverage dynamics at the expense of a greater fuel consumption but a remarkable time advantage.
Although in this thesis we are going to follow this last path, trying to not leverage the dynamics directly with the invariant manifolds, we will discuss considerable results obtained in this kind of transfer that will lead inevitably and indirectly to follow natural dynamics when trying to minimize the control energy cost function.



(b) L_1-L_2 Heteroclinic Connection

(c) L_1-DPO-L_2 Heteroclinic Chain

Figure 1.8: Heteroclinic connection and heteroclinic chain between L_1 and L_2 Lyapunov orbits, C = 3.11295. [17]

1.2.2 Distant Retrograde Orbits

DROs are highly stable Moon-centered planar orbits characterized by retrograde motion, a spacecraft would travel this orbit in the opposite direction to the one in which the moon orbits the planet. The term "distant" refers to the fact that the majority of these orbits pass far away from the Moon and Lagrangian points, covering a large region of the Cislunar realm.
In Fig.(1.9) few DROs with different C levels are reported.

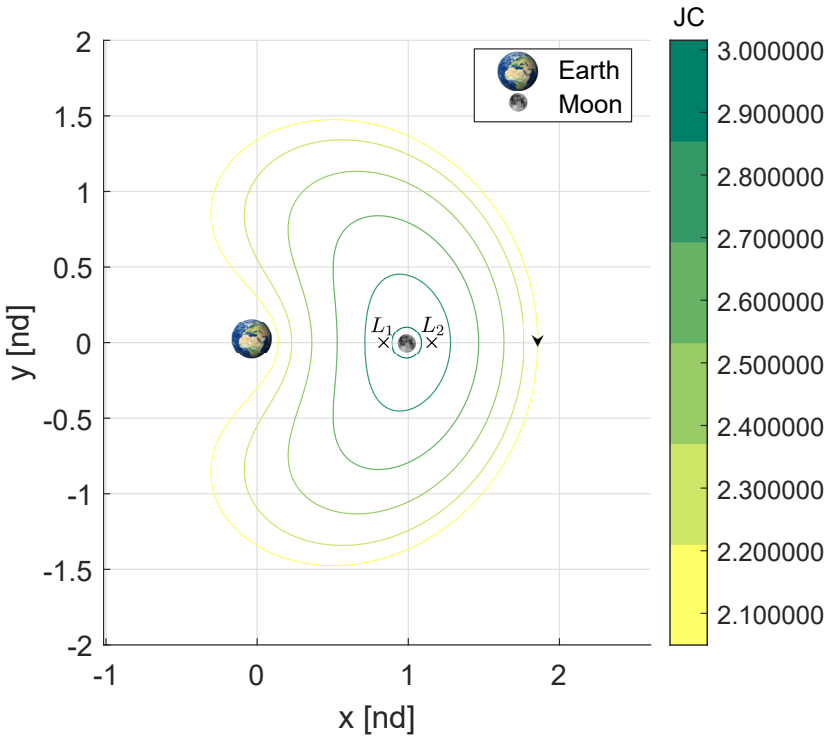


Figure 1.9: DROs for different C levels.

One feature that makes these orbits desirable in a Cislunar mission design is the already mentioned high Lyapunov stability. A spacecraft that would eventually encounter any perturbation along the orbit would return to its initial position after a certain time, and it would not drift away like in previous Lyapunovs.
Although DROs have been proposed in different conceptual mission design, like in a first concept for Jupiter Icy Moons Orbiter, the only orbiters that used them, to the author knowledge in 2022, are the chinese Chang'e 5 [20] and the Orion spacecraft of Artemis 1 mission. In addition in the NASA Lunar Gateway technical report [21] is highlighted that: "the Gateway shall be capable of performing a single round trip transfer to Distant Retrograde Orbit (DRO) and back within 11 months".

1.2.3 Halo Orbits

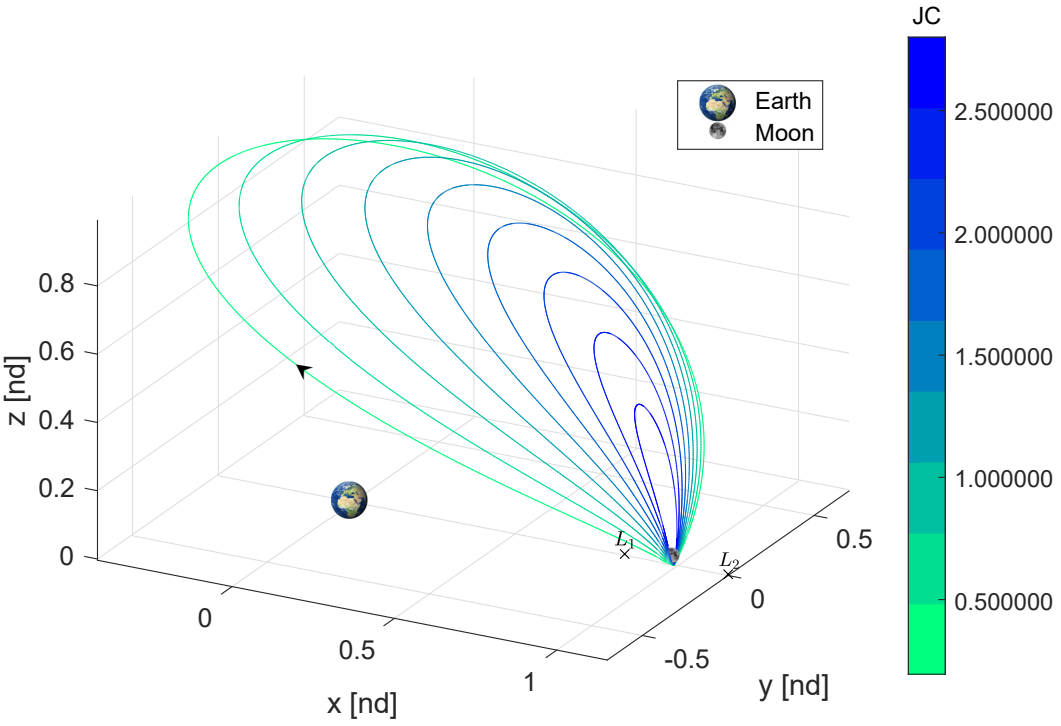


Figure 1.10: Halo orbits around L_1 for different C levels.

Halo orbits are periodic three dimensional orbits around one of the three collineare Lagrange points (L_1, L_2, L_3). In Fig.(1.10) a family of L_1 Halo orbits is illustrated.
In 1966 Robert W. Farquhar was the first to use the name Halo [22] and the first to propose the use of this kind of orbit to solve one of the major problem in Apollo missions design, the lack of direct communication from the Earth to the back side of the Moon. This problem still be considerable nowadays. A spacecraft in an Halo orbit, indeed, would be in continuous view of both the far side of the Moon and the Earth. While Farquhar used analytical expression to represent Halos, in 1984 Kathleen Howell was able to compute a larger precise set of Halos numerically. [23]
Despite the Halo orbit solves the communication problem of lunar missions they still have the same drawback of all the libration points orbits, the instability.
This leads to the need to perform station keeping maneuvers in order to maintain the initial nominal orbit. In Fig.(1.11) different Lunar Gateway's orbits are depicted.

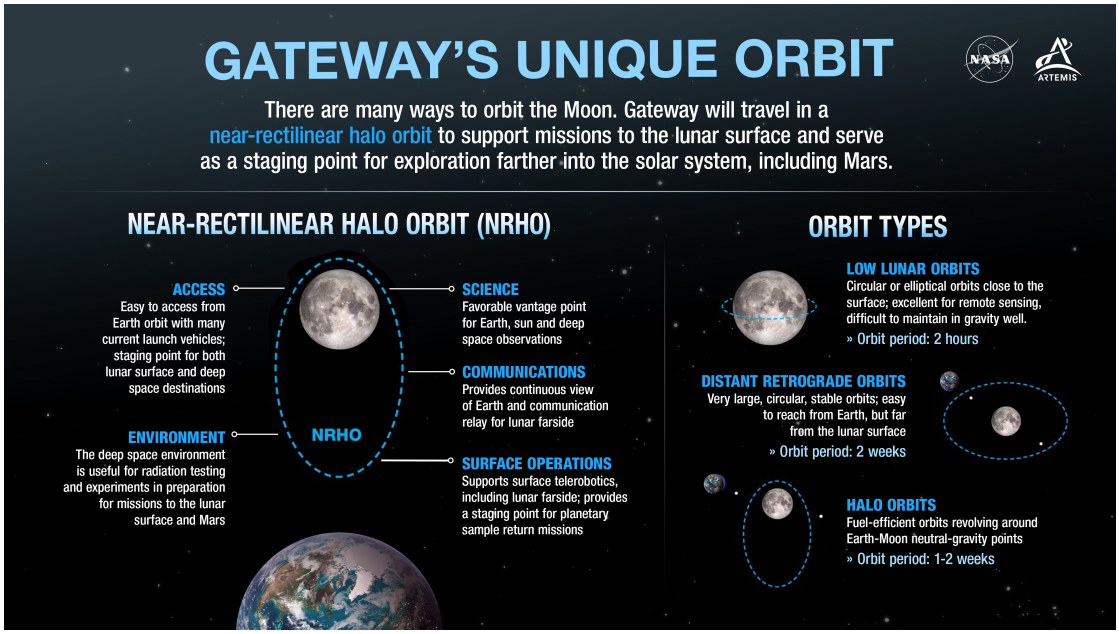


Figure 1.11: Infographic depicting NRHO, Gateway's unique near-rectilinear halo orbit.[24]

From the infographic it can be seen as both Halos and DROs have been taken into account in the Gateway's mission design. Precisely an Halo Orbit is usually referred as Near-Rectilinear Halo Orbit (NRHO) when its inclination is high and approximately polar. The mission leverages the best from both the last two kind of orbits we have discussed. Science and communications tasks are achieved thanks to the (NRHO) that could also orbit the vicinity of the Moon with its low perilune altitude. On the other hand a Distant Retrograde Orbit, that is stable, is the perfect choice to accomplish all engineering duties without the need for fuel expensive station keeping maneuvers.
Other than in Cislunar missions, an Halo orbit has been also leveraged in recent times from JWST that in January 2022 entered its Halo design orbit around the Sun-Earth L_2 point. [25]

Chapter 2

Trajectory optimization

Optimization problems have always been a major challenge in all engineering fields. In particular space engineering has been a fertile land, along with the development of numerical methods, to enhance the invention of innovative optimization algorithms, methods and techniques. The spacecraft trajectory optimization problem can be described as the process of designing a trajectory that satisfies some criteria, such as the initial and terminal conditions, while minimizing (or maximizing) a particular cost function. In 1998 Betts made the first sincere effort to categorize techniques for spacecraft trajectory optimization [26], dividing the methods in two main families: the direct and the indirect methods, that will be discussed in this chapter.
As reported in [27] four steps can be used to break down the process of optimizing the trajectory of a spacecraft. As usual a *model* is required to describe the physics

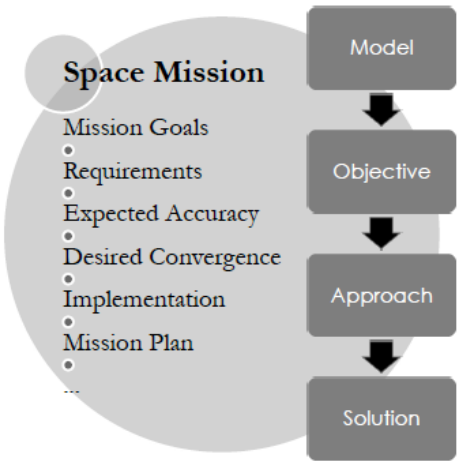


Figure 2.1: General scheme of spacecraft trajectory optimization process [27]

of the process and in our case the CR3BP is the better choice as discussed in the previous chapter. The *objective* is highly affected by mission goals and requirements. Typically in orbital mechanics optimization problems we must have to minimize the transfer time or the fuel cost (i.e. minimize the propellant). Depending on the mission constraints, different cost function can be taken into account as objective of the optimization process. The *approach* step concerns the type of methods and techniques which are dedicated to solving the trajectory design problem. Here we can found both analytical approaches, based on the classical optimal control theory or numerical approaches related to space transfers. Finally the *solution* can be a closed form analytical solution, if the analytical approach has been used, otherwise it can be found leveraging numerical techniques and algorithms.
As already mentioned, dealing with Cislunar trajectory optimization we must face the nonlinearity of the dynamics and so typically trajectory optimization problems don't have closed analytical solutions.
In addition one of the main challenges in our case is the use of a continuous model. In comparison with impulsive models, mathematical models based on this idea are more accurate but also more complex because the trajectory is handled while taking into account non-zero inputs (i.e. $\vec{u}(t) \neq 0$ in the dynamics). The comparison is the same as high-thrust and low-thrust trajectory design. As illustrated in the

Propulsion system	Thrust (N)	I_{sp} (s)
Chemical engine	$0.1 - 10^6$	140-460
Cold gas thruster	$0.05 - 200$	50-250
Resisto-jet	$0.002 - 0.1$	150-8000
Arcjet	$0.002 - 0.7$	400-1500
Ion thruster	$1 \times 10^{-5} - 0.2$	1500-5000
Hall thruster	$1 \times 10^{-5} - 1$	1500-6000
Pulsed plasma,thruster	$5 \times 10^{-5} - 0.01$	500-2000
Solar sail	$0.001 - 0.1$	∞

Figure 2.2: Characteristics of typical propulsion systems [27]

Fig.(2.2) high thrust chemical engines have low I_{sp} and this leads to a greater fuel consumption. On the other hand low thrust providers, such as Resistojet, Arcjet, Ion thrusters and Hall thrusters have a remarkable I_{sp} that promises a decrease of the fuel consumption at the expense of a greater transfer time. The majority of the Cislunar missions leverage low thrust engines due to the important mission time. A notable drawback of the low thrust approach is the need to model the actuators dynamics. Nowadays thruster are indeed built as ON/OFF devices, leading to several discontinuities on the control that can make numerical techniques challenging. Usually smoothing functions are used to solve this numerical problem, [28] but we are not going to take into account this issue in the thesis.

2.1 Optimal Control Problem

Optimal control theory is an outcome of the calculus of variations. [29]
The OCP can be defined in different ways. We can state the problem as: [30]
Minimize the Lagrange performance index:

$$J = \phi\left[x_0, x_f, t_0, t_f\right] + \int_{t_0}^{t_f} L[x(t), u(t), t]dt \tag{2.1}$$

subject to the system dynamics constraints:

$$\dot{x}(t) = f[x(t), u(t), t] \tag{2.2}$$

where $x = \{x_1, \ldots, x_n\}^T$ is the **state vector** and $u = \{u_1, \ldots, u_m\}^T$ is the **control vector**. In addition the boundary conditions are written as:

$$\psi(x_0, x_f, t_0, t_f) = 0 \tag{2.3}$$

Inequality path constraints can also be added to the formulation if needed, in this work we are going to consider only equality constraints. In Eq.(2.1) is reported the Bolza formulation of the OCP. Here we are going to adopt the Lagrange formulation ($\phi = 0$), this leads to a new Lagrange performance index:

$$J = \int_{t_0}^{t_f} u^2 dt \tag{2.4}$$

that usually is referred as minimum control energy cost function.
Please note that the specific impulse I_{sp} and the spacecraft mass are assumed constant. We choose to start from a minimum energy solution because the fuel mass objective cost function leads to instantaneous ON/OFF thrust switching which is challenging for numerical solution methods.

2.1.1 Direct Vs Indirect optimization

Generally speaking, the optimal control problem can be expressed as either an *indirect* or *direct* optimization problem. In direct methods the control is an optimization variable and the optimality is achieved by satisfying the Karush Kuhn Tucker (KKT) conditions. [31] On the other hand in indirect methods Pontryagin's minimum principle is used to derive first-order optimality conditions, which are then used to find the control vector. Unless the problem is trivial we always need to transcript the OCP into a nonlinear programming problem (NLP). The NLP is solved using classical Newton's based methods that try to iteratively solve a series of approximate linear problems.

In direct formulation there are fewer constraints than optimization variables and the sub problems are always undetermined.
Let consider a trajectory discretized into N nodes. The state vector is composed by 3 positions and 3 velocities while the control vector is a three-dimensional vector. Thus, in every node we have 9 parameters for a total of $9N$ variables to optimize in the problem. Since we have to force state continuity between each pair of nodes, we have $6(N-1)$ dynamics constraints and 12 constraints considering the initial and final states. At this point we can evaluate the DoF of the problem as:

$$DoF_{direct} = 9N - 6(N-1) - 12 = 3N - 6 \tag{2.5}$$

When $N = 2$ we have Lambert's problem, otherwise when $N > 2$ we have infinite possible solutions. On the other hand in the indirect formulation each node has 12 parameters (6 for the state and 6 for the *adjoints* or *costates* that are state dual variables defined within the formulation). Thus there are $12N$ variables to optimize. As constraint we have $6(N-1)$ state dynamics constraints and $6(N-1)$ additional constraints on costates dynamics from the Pontryagin's minimum principle. Adding also the 12 constraints on initial and final states we have removed all of the DoF and we have a single solution that obeys the constraints. However, in the CR3BP, the problem is highly nonlinear and it is common to find multiple locally optimal solutions. Finally, every time we free up a constraint (such as the endpoints or time of flight) the optimizer will use that DoF to minimize the objective function.

2.2 Indirect methods

Recalling the OCP defined in Sec.(2.1) the augmented performance index with the constraints, Eq.(2.2) and Eq.(2.3), can be written as:

$$J^* = \phi + \mu^T \psi + \int_{t_0}^{t_f} [L + \lambda^T (f - \dot{x})] dt \tag{2.6}$$

where μ and λ are two kinds of Lagrange multipliers added to enforce boundary constraints and system dynamics respectively.
Usually λ are also known as costates variables. The problem is reduced to identify a stationary point of J^*, imposing $\nabla J^* = 0$. The set of optimization variables in

our case is $[x, u, \lambda, \mu]$ and this leads to:

$$\begin{aligned}
\frac{\partial J^*}{\partial \boldsymbol{\lambda}} = 0 \quad &\Rightarrow \quad \dot{\mathbf{x}} = \mathrm{f}(\mathbf{x}(t), \mathbf{u}(t), t) \\
\frac{\partial J^*}{\partial \mathbf{x}} = 0 \quad &\Rightarrow \quad \dot{\boldsymbol{\lambda}} = -\left(\frac{\partial \mathbf{f}}{\partial \mathbf{x}}\right)^T \boldsymbol{\lambda} - \left(\frac{\partial L}{\partial \mathbf{x}}\right)^T \\
\frac{\partial J^*}{\partial \mathbf{u}} = 0 \quad &\Rightarrow \quad \left(\frac{\partial L}{\partial \mathbf{u}}\right)^T + \left(\frac{\partial \mathbf{f}}{\partial \mathbf{u}}\right)^T \boldsymbol{\lambda} = 0 \\
\frac{\partial J^*}{\partial \mu} = 0 \quad &\Rightarrow \quad \boldsymbol{\psi}(x_0, x_f, t_0, t_f) = 0
\end{aligned} \tag{2.7}$$

The first two equations represent the dynamics of states and costates respectively. The boundary conditions are also obtained in the last equation.
Usually to obtain a more compact notation the Hamiltonian is defined as:

$$H(\mathbf{x}, \mathbf{u}, \boldsymbol{\lambda}, t) = L(\mathbf{x}(t), \mathbf{u}(t), t) + \boldsymbol{\lambda}(t)^T \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t) \tag{2.8}$$

and so the dynamics can be rewritten as:

$$\begin{cases} \dot{\boldsymbol{x}} = H_\lambda \\ \dot{\boldsymbol{\lambda}} = -H_x \\ H_u = 0 \end{cases} \tag{2.9}$$

also known as Eulero-Lagrange equations or first order optimality conditions.
The OCP is reduced to a Boundary Value Problem (BVP) on a differential-algebraic system of equations (DAE). In addition, in our case, DAE can be reduced to an ODE system processing the algebraic control equation deriving from Pontryagin's minimum principle ($H_u = 0$), from whom we can obtain an explicit expression for u as function of the costates. In our case of fixed x_0 and x_f constraints can be written as:

$$\begin{cases} x(t_0) - x_0 = 0 \\ x(t_f) - x_f = 0 \end{cases} \tag{2.10}$$

reducing the original problem to a Two Point Boundary Value Problem (TPBVP). In this work the bvp4c Matlab solver, a finite difference code that implements the three-stage collocation Lobatto IIIa formula, is used in order to solve it and it will be discussed in next sections. As first iteration fixed final time transfers are studied in this thesis. Then, as second step, the final time t_f is let free.
If we consider a free final time t_f problem, a new free parameter has to be defined as:

$$\tau = \frac{t}{t_f} \tag{2.11}$$

Substituting into the state equations, they become:

$$\frac{d\vec{x}}{d\tau} = t_f \cdot \vec{f}[\vec{x}(\tau), \vec{u}(\tau), \tau] \tag{2.12}$$

and the cost function becomes:

$$J = t_f \int_0^1 u^2 d\tau \tag{2.13}$$

In this case an additional boundary condition is required:

$$H\left(t_f\right) = -\frac{\partial \Phi}{\partial t_f} = 0 \tag{2.14}$$

where $\Phi = \phi + \mu^T \psi$.
The Hamiltonian at the final time t_f must be equal to zero because the terminal cost and the terminal boundary conditions are independent of time.

2.3 Direct methods

Direct methods are based on reducing the OCP into a NLP. The core of the method is the *parameterization* of all continuous variables and the *transcription* of the differential equations describing the dynamics, into a finite set of equality constraints. Classical transcription methods are collocation, single and multiple shooting that will be discussed later. The art behind the direct methods is the way we set up the optimization problem, because later on the hard work is simply done behind the scene by the solver.
First of all the parameterization is based on the discretization of the continuous variables on a mesh settled up on the time domain.
The time is indeed discretized as:

$$t_0 = t_1 < t_2 < \ldots < t_N = t_f \tag{2.15}$$

Thus, the states and the controls are discretized over the previous mesh by defining

$$\mathrm{x}_k = \mathbf{x}\left(t_k\right) \tag{2.16}$$

and

$$\mathrm{u}_k = \mathbf{u}\left(t_k\right) \tag{2.17}$$

We obtain the vectors:

$$\begin{aligned} \mathbf{x}(t) &= \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \\ \mathbf{u}(t) &= \{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N\} \end{aligned} \tag{2.18}$$

Thus, a new vector of variables can be defined:

$$\mathbf{X} = \{t_f, \mathbf{x}_1, \mathbf{u}_1, \ldots, \mathbf{x}_N, \mathbf{u}_N\} \tag{2.19}$$

All the optimization variables are concatenated into one vector, and the problem is reduced to:

$$\begin{aligned} &\text{minimize} : J(\mathbf{X}) \\ &\text{subject to} : h(\mathbf{X}) = 0 \end{aligned} \tag{2.20}$$

2.4 Numerical techniques

As already stated when we deal with an OCP we have an infinite dimension optimization problem since the variables are functions. This is the reason why the OCP is reduced into a TPBVP, where we optimize over real numbers rather than functions. Usually this process is called transcription and the final problem, that we must solve with numerical techniques, is referred as non-linear programming problem NLP. The most used NLP solvers in trajectory optimization are directly based upon the methods presented below.

2.4.1 Single shooting

The single shooting method is the simplest way to solve a boundary value problem (BVP) by reducing it to an initial value problem. The name of the method comes from the representative case in Fig.(2.3).

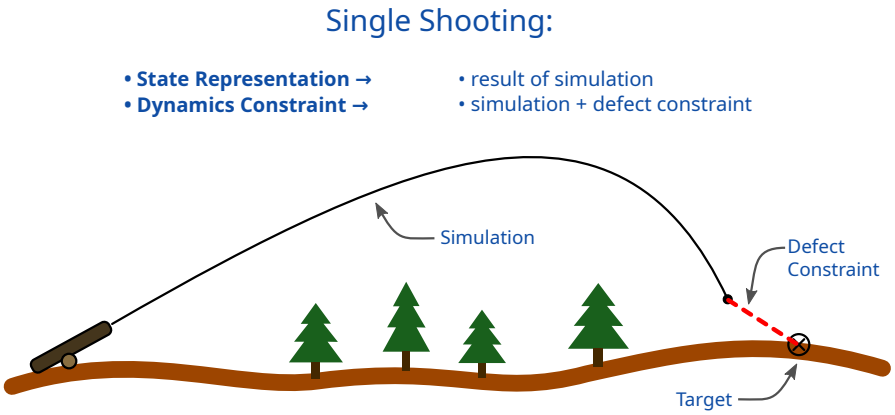


Figure 2.3: Single shooting scheme [32]

Lets imagine to aim a cannon at a target. Once we shot the cannon ball is ruled by non linear dynamics (gravity,drag, etc.) in the vertical direction. Thus, we can simulate the trajectory, which is represented as a single entire segment, and see the defect constraint, that is the difference between the desired position of the target and the point reached by the cannon ball. In mathematical terms the idea behind single shooting method is to write iteratively the final state with a Taylor's expansion around the initial conditions:

$$x_f \approx x_f^* + \frac{\partial x_f}{\partial x_0} \cdot \Delta x_0 \tag{2.21}$$

then we can solve for Δx_0 and update the initial conditions:

$$\Delta x_0 = \left[\frac{\partial x_f}{\partial x_0}\right]^{-1} \cdot \left(x_f - x_f^*\right) \tag{2.22}$$

With all the aforementioned concepts one can deduce that the single shooting method suits perfectly only simple problems, that can be approximated by linear functions. Otherwise the method struggles and other techniques are required.

2.4.2 Multiple shooting

The multiple shooting technique is the natural evolution of the previous method. The idea behind this approach is reported in Fig.(2.4). The trajectory is divided

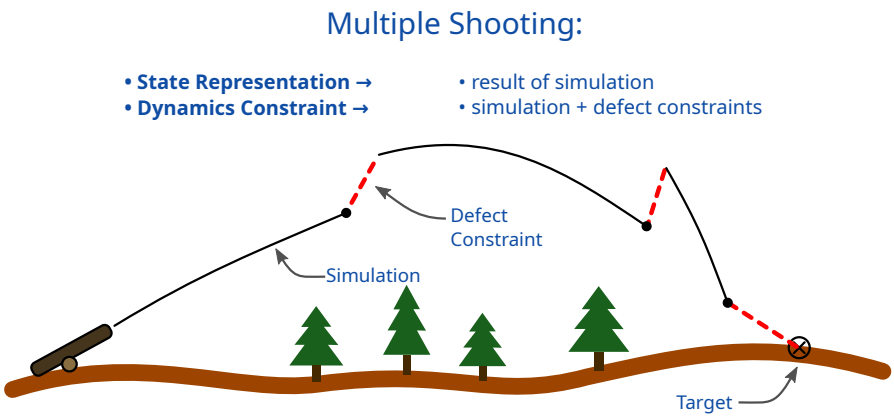


Figure 2.4: Multiple shooting scheme [32]

in N segments in order to obtain portions where the linear assumptions are more likely to be valid. Of course the solution depends on the number of nodes. A defect constraint is added between nodes and matching conditions are imposed in order to obtain a whole solution.

2.4.3 Collocation

In collocation methods the interval of integration is divided in subintervals by N nodes, or collocation points, as in Fig.(2.5). The idea is to represent the ordinary differential equation (i.e. the state and control's path) using approximate continuous functions as polinomial splines that obey dynamics at collocation points. These methods are often leveraged to solve ODE. There are numerous collocation-based techniques, which can be identified by the node spacing and basis function selection. For example, a small number of high-order polynomials are used in orthogonal collocation techniques, which were created for calculating satellite trajectories.
On the other hand in direct collocation, a large number of low-order polynomials are used to approximate the solution.

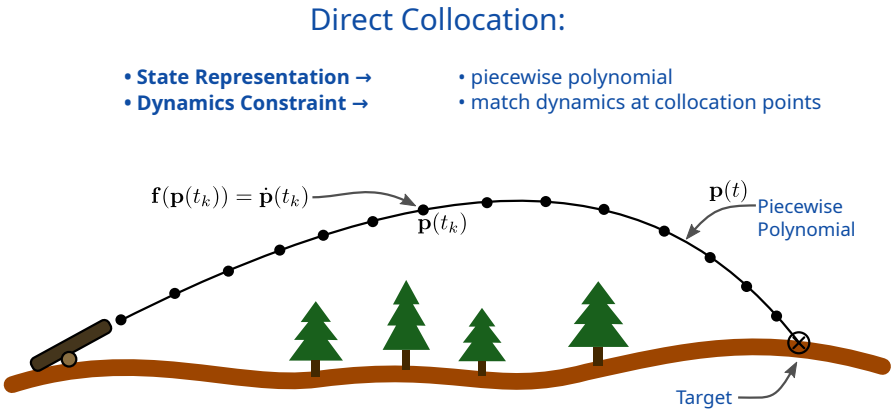


Figure 2.5: Collocation scheme [32]

Lets consider for example the following ODE:

$$y'(t) = f(t, y(t)), \quad y(t_0) = y_0 \tag{2.23}$$

that must be solved in the interval $[t_0, t_0 + c_k h]$ with c_k that represents the collocation point, $0 \leq c_1 < c_2 < \ldots < c_n \leq 1$. [33]
Using a collocation method a polynomial p of order n will approximate the solution y satisfying:

$$\begin{aligned} p(t_0) &= y_0 \\ p'(t_k) &= f(t_k, p(t_k)) \end{aligned} \tag{2.24}$$

over the collocation points $t_k = t_0 + c_k h$. Thus we have n+1 conditions, that are indeed the same number of coefficients of the polynomial p of order n.
The collocation methods can be classified as implicit Runge-Kutta methods, of course the opposite does not always hold.

2.5 bvp4c

bvp4c is the finite difference Matlab's code, used in this thesis, that implements the three-stage Lobatto IIIa formula, a collocation formula. Recalling that an implicit Runge-Kutta method can be written as:

$$y_{n+1} = y_n + h\sum_{i=1}^{s} b_i k_i \tag{2.25}$$

with:

$$k_i = f\left(t_n + c_i h, y_n + h\sum_{j=1}^{s} a_{ij}k_j\right), \quad i = 1, \ldots, s \tag{2.26}$$

and recalling also the Butcher tableau which is a useful tool to visualize the coefficients:

$$\begin{array}{c|cccc} c_1 & a_{11} & a_{12} & \ldots & a_{1s} \\ c_2 & a_{21} & a_{22} & \ldots & a_{2s} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_s & a_{s1} & a_{s2} & \ldots & a_{ss} \\ \hline & b_1 & b_2 & \ldots & b_s \\ & b_1^* & b_2^* & \ldots & b_s^* \end{array} \tag{2.27}$$

The Lobatto IIIA methods can be divided into a second-order method, represented by the following tableau:

$$\begin{array}{c|cc} 0 & 0 & 0 \\ 1 & 1/2 & 1/2 \\ \hline & 1/2 & 1/2 \\ & 1 & 0 \end{array} \tag{2.28}$$

that is usually known as trapezoidal rule:

$$y_{n+1} = y_n + \frac{1}{2}h\left(f\left(t_n, y_n\right) + f\left(t_{n+1}, y_{n+1}\right)\right) \tag{2.29}$$

or the fourth-order method:

$$\begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ 1/2 & 5/24 & 1/3 & -1/24 \\ 1 & 1/6 & 2/3 & 1/6 \\ \hline & 1/6 & 2/3 & 1/6 \\ & -\frac{1}{2} & 2 & -\frac{1}{2} \end{array} \tag{2.30}$$

This latter one is the collocation formula implemented in **bvp4c**. The solver calculates a numerical solution taking into account both the algebraic equations of

the collocation points in every subintervals and the boundary conditions. Once is run, the solver must reduce the error on each subintervals, thus the mesh is adapted iteratively in order to respect the tolerance criteria. The input of the solver are indeed the initial mesh and the value of the solution at those mesh points. In Fig.(2.6) the work logic of bvp4c is illustrated.

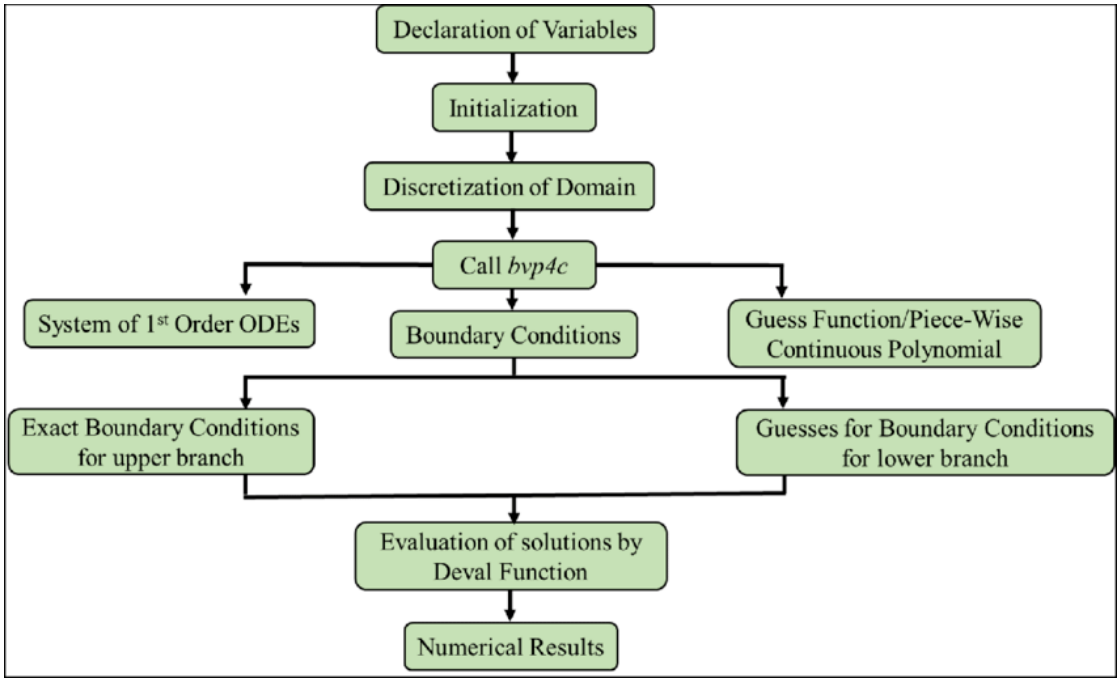


Figure 2.6: Procedure of three-stage Lobatto IIIa formula [34]

Chapter 3

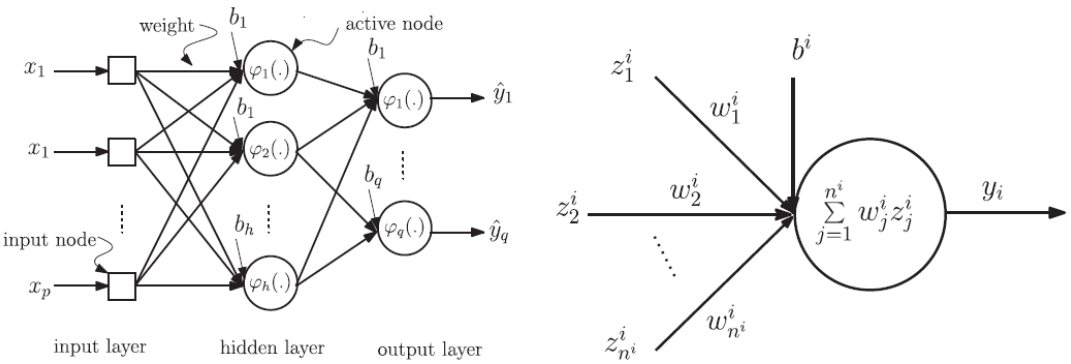
Neural Networks

Neural networks (NN) have broken new ground in scientific community in the last decades, paving the way for the future of automation. Artificial Neural Networks (ANN) are inspired by biological NN that constitute human brains. [35]
The main elements in ANN are either called nodes or neurons, and they are typically grouped in layers. Like the synapses in a human brain, each link has the ability to send a signal to neighboring neurons. Each neuron's output is calculated using a non-linear function of the total of its inputs, often called activation function. The connections between neurons are called edges. Nodes and edges have a weight that adjusts as the training of the network proceeds. This weight can either increase or decrease the signal at a connection. Moreover sometimes we need the neuron to activate just when a threshold value is crossed, and from that comes the need for biases.
The output y_i of a node can be written [36] as:

$$y_i = \varphi_i \left(\sum_{j=1}^{n^i} w_j^i z_j^i + b^i \right) \tag{3.1}$$

where n^i is the total incoming connections, z^i is the input, w^i is the weight, b^i is the bias, and φ_i is the activation function at the i-th node that typically bounds the output in the range [0,1] or [-1,1]. A representative scheme of the hypothetical structure of a neural network is illustrated in Fig.(3.1).

Besides the use of a neural network can have lots of objectives, it has been demonstrated that NN are an incredible tool for functions approximations [38] and this is the application that we are going to use in this work. Neural networks learn thanks to the training phase. When processing samples from a dataset that each contains a known "input" and "output", neural networks learn (or are trained) by creating probability-weighted associations between the two that are then stored inside the data structure of the network itself. The network then



(a) Three-layer feedforward neural network (b) Node of the network

Figure 3.1: Three-layer feedforward neural network (a), where input layer has p input nodes, hidden layer has h activation functions, and output layer has q nodes. [36]

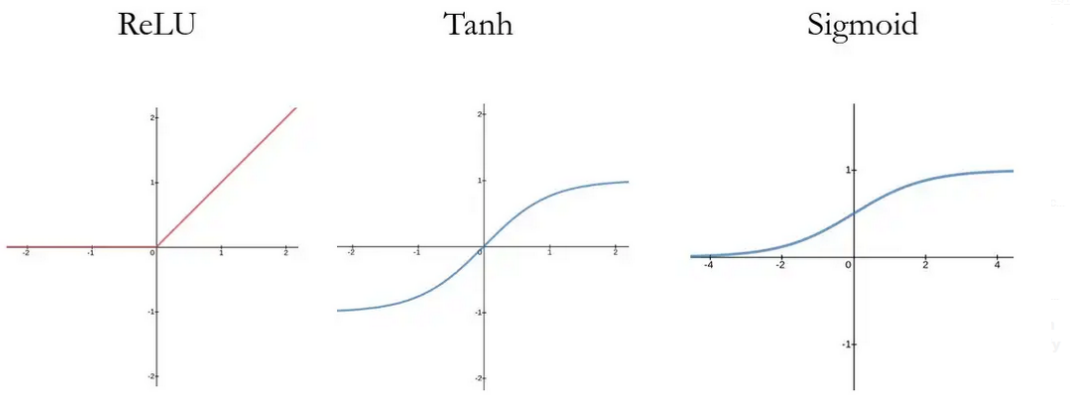


Figure 3.2: Examples of activation functions [37]

adjusts its weighted associations according to a learning rule. The power of neural networks relies on the non linear nature of activation functions. Some common functions are depicted in Fig.(3.2).
This latter method of proceeding is usually known as supervised learning and it is essentially the minimization between the desired output y_i and the model's output $\hat{y}_i$. For this purpose, several cost function can be designed. For instance, in regression problems, mean squared error is one of the commonly used cost function and it is written as:

$$c_f\left(\mathbf{y}_i, \hat{\mathbf{y}}_i\right) = \frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{q} \left(y_{ij} - \widehat{y}_{ij}\right)^2 \tag{3.2}$$

where N are the data pairs.

3.1 Dataset split

One of the key features when we try to build a neural network is the division of the dataset in training, validation and test sets. If we do not split our data in the aforementioned groups, and we evaluate the performance of the net on the single group, thus the training one, we can easily fall back into a biased score. That is the reason why an held out dataset, the test one, is usually used to give an estimate of model skill. [39]
Our data should be divided into three separate dataset splits for the training and testing phases of our model.

3.1.1 Training set

From this data the model learns in order to uncover any hidden characteristics or patterns. As the neural network design is continually given the same training data throughout each epoch, the model keeps learning the characteristics of the input. The training set should comprise a diverse collection of inputs so that the model may be trained in all circumstances and anticipates any unobserved data sample that may arise in the future.

3.1.2 Validation set

The validation set is a collection of data, separated from the previous training one, that is used to validate the net's performance throughout the learning process. Since the design of a neural network implies a trial and error method to tune all the different hyperparameters, it is useful to have a dataset that informs us whether or not our training is on the correct path. So far we have understood that as the training of the net proceeds learning from the training set, a different set is used in order to evaluate the performance of the net at the same time. The major goal of dividing the dataset into a validation set is to avoid the *overfitting* of our model that will be discussed in the following sections.

3.1.3 Test set

After the training is complete, the model is tested using a different set of data called the test set. In terms of accuracy, precision, and final model performance, this set offers an objective measure.

In Fig.(3.3) the work logic behind the learning process of a neural network is reported. At this point, after we pointed out the importance of splitting the dataset,

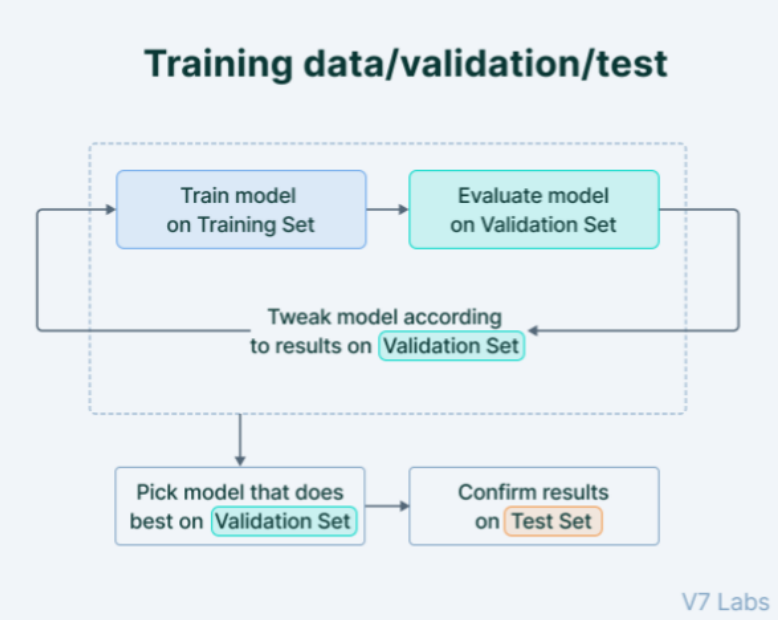


Figure 3.3: Work logic behind the dataset split [40]

we must focus on how this division should be performed. For example a machine learning model needs a bigger validation set if there are more hyperparameters to modify in order to maximize model performance. The number of hyperparameters of the neural network functions grows together with the dimension of the data, making the model increasingly complex. In these cases, a sizable portion of the data should be maintained as a training set and the remainder as a validation set. At the end there is not an optimal rule to choose the different percentages for the three sets because this depends highly from our requirements and needs. However machine learning model will exhibit large variations in training if a small percentage is given to training data. On the other hand if test and validation dataset have a small percentage we clearly lose information on the net's performance evaluation.

In Fig.(3.4) examples of data split are illustrated. As we can see the training data represent always the major percentage.
Another important feature to choose is the technique for the data splitting. The most common methodology is currently the *random picking*. Essentially the dataset is firstly shuffled in order to avoid any correlation between the order of collection of the solutions, then, depending on the percentages of the three sets, data are randomly picked and divided in training, validation and test data. This latter approach is the one followed in this thesis.

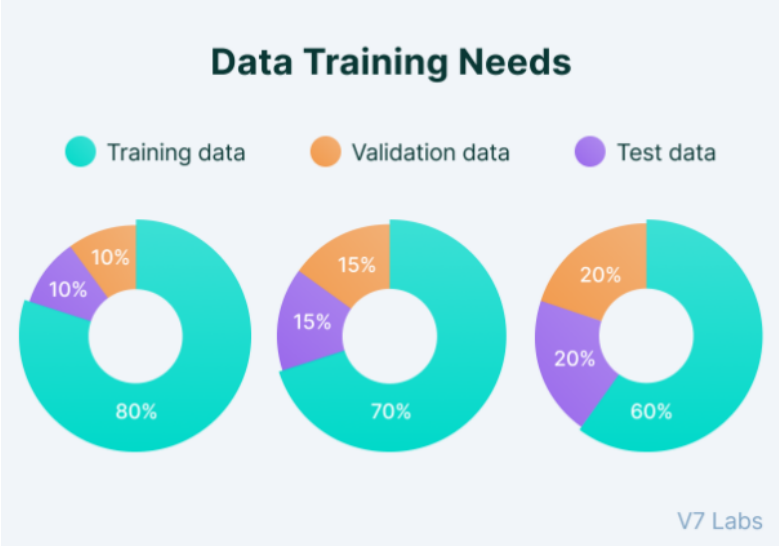


Figure 3.4: Some examples of data split [40]

3.2 Over-fitting

The ideal goal for a neural network is to perform well on unseen inputs and not only on the data that we used to train the net. For this reason we usually say that the net should be able to generalize new data.
However reaching a good generalization performance by the net is not a trivial acitivity. This depends, for example, on the data split percentages, the hyperparameters tuning and also on the dataset nature itself.
The Fig.(3.5) is really interesting and can help us to clarify the aforementioned concepts.

Three different typical machine learning applications are presented, the regression problem, the classification problem and the deep learning which is the most interesting in our case. In each of these problem, after the neural network has been implemented, we can have:

- **Under-fitting**: when the model has bad performance on both the training set and the validation one. This can due, for example, to a poor training dataset. In the regression problem above the quadratic trend of the data is approximated by a linear one, thus the model is too simple and poor.

- **Optimal-fitting**: when the model performs well on the training set and can also generalize with good performance new data. In the regression problem above, a quadratic function suits better the data in comparison with the linear one of the previous case. In the deep learning problem the trends of training

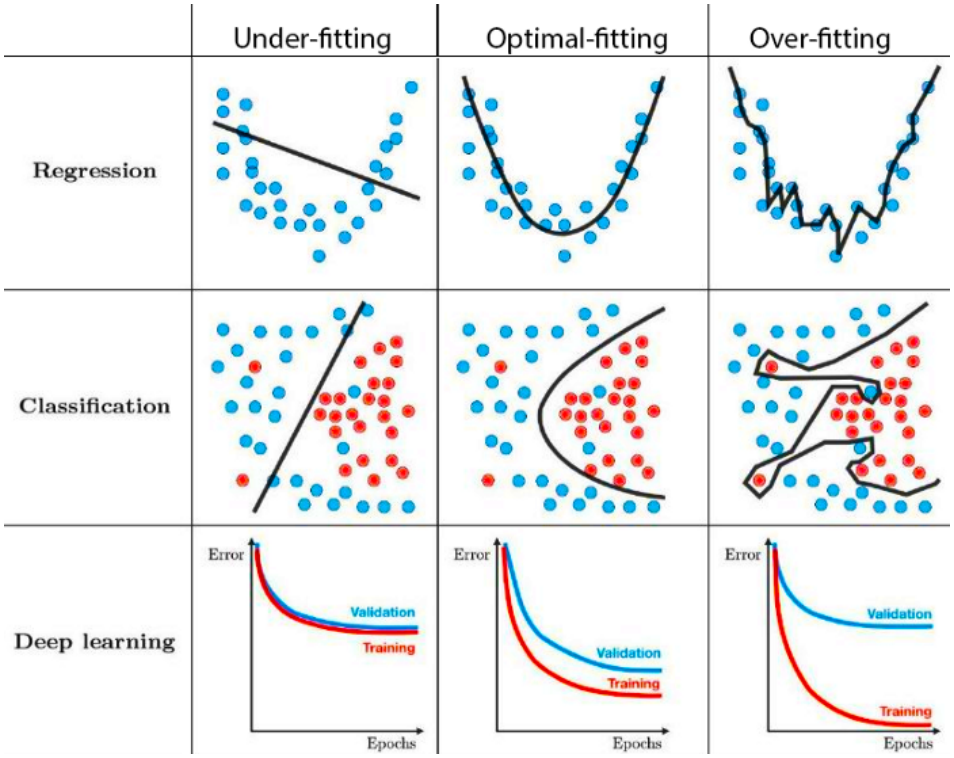


Figure 3.5: Overfitting visualization [41]

and validation sets reach a smaller error even if the training one is slightly better;

- **Over-fitting**: when the model learns every detail of the training dataset and it is excessively complex. This leads to capture also the noise of the training data. Thus, the training error is very small while the validation one is larger because the model cannot generalize well.

There are different ways to handle both under and over-fitting as reported in [41]. A good path to avoid the **under-fitting** is to increase the model complexity. An underfitted model is excessively simple, it is not able to capture the pattern of the data and this leads to poor performance.
We have different options to increase the model complexity in order to encourage the net to learn:

- increase the number of neurons and/or hidden layers of the net;
- increase the number of hyperparameters;
- change the optimizer.

If significant improvements can not be observed at the end of this process, one should consider to increase the training epochs.
On the other hand the **over-fitting** can be avoided in several ways that are reported in the above quoted literature reference. In this thesis every time we dealt with over-fitting problems we solved them getting more representative data. Increasing the dataset dimension adding diverse data can help the net to generalize.
In contrast with the under-fitting case we can also decrease the complexity of the net in the following ways:

- decrease the number of neurons and/or hidden layers of the net;
- decrease the number of hyperparameters;
- change the optimizer.

3.3 Neural networks in Matlab

In this thesis the Matlab's Deep Learning Toolbox ® is leveraged to build different neural networks as we are going to discuss in the following chapters.
The number of neurons, hidden layers, dataset split percentages and way of division, activation functions, optimizer and hyperparameters can be change from both the main or from the graphic interface that the toolbox supplies.
As the net's structure is defined along with all the aforementioned parameters, the training phase can start. At this point a graphic interface similar to the one reported in Fig.(3.6) is displayed. On top of the figure the structure that we built is displayed in a scheme. From this graphic we can also supervise the training of the net, the epochs, the time and performance. In this thesis, as already stated the performance are evaluated through the mean squared error.
The train of the net stops as the maximum epochs is reached or when some stopping criteria on the hyperparameters are encountered, thus for example when the net begins to overfit. In addition the user can also stop the training if needed.
At the end or during the training phase, we can click on *performance* to open a graphic plot similar to the one presented in Fig.(3.7). From this latter we can notice the trends of training, validation and test data throughout the training phase.
All the consideration about under and over-fitting can be done looking at this graph. As already explained in the previous section, a good neural network should minimize the gap between the trends of training, validation and test data in order to generalize unseen inputs, reaching at the same time a low mean squared error value.

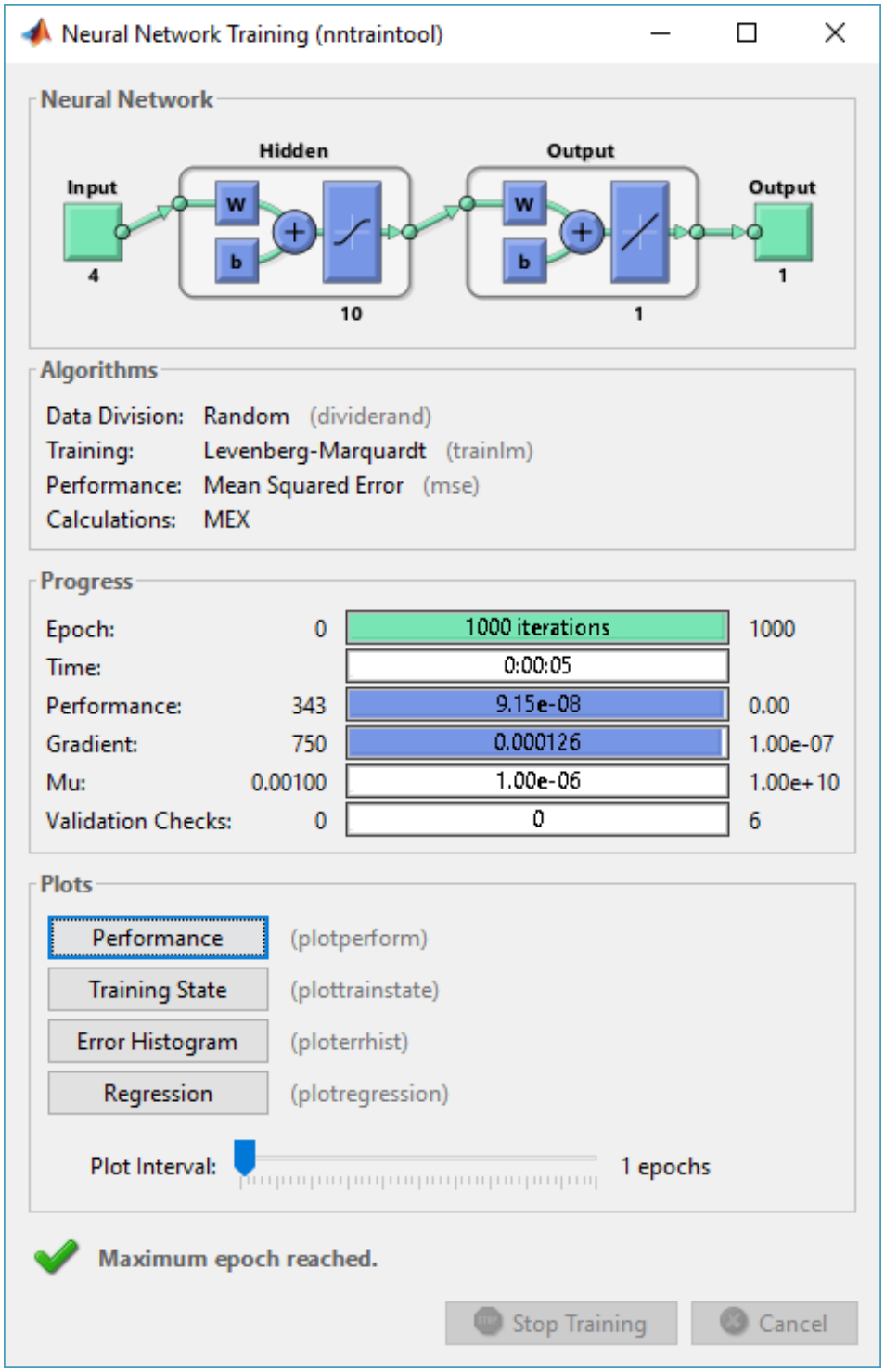


Figure 3.6: Visualization of Matlab's NN interface [42]

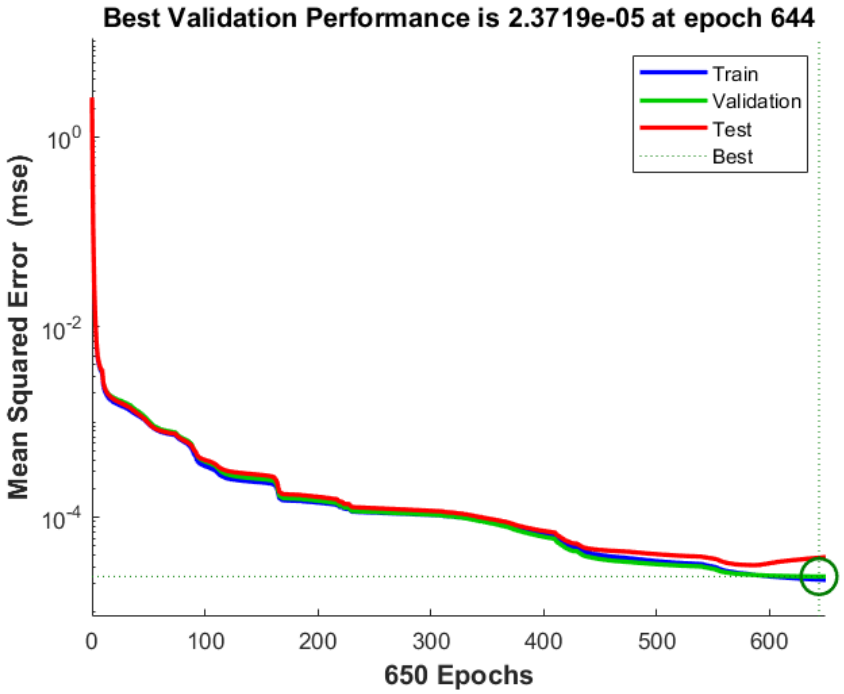


Figure 3.7: Example of a performance plot

3.3.1 Levenberg-Marquardt algorithm

The Levenberg-Marquardt (LM) is an iterative algorithm used to solve non linear least squares problems. The method is a mixture between a classical Gauss-Newton algorithm and the method of gradient descent. The main difference in comparison with these methods is that the LM is more robust.
In our application the performance function Eq.(3.2) is a sum of squares so the Hessian matrix can be written approximately as:

$$H = J^T J \tag{3.3}$$

and the gradient as:

$$g = J^T e \tag{3.4}$$

where e is the vector errors and J is the Jacobian matrix that includes the derivatives of e w.r.t. weight and biases. At this point J can be found using a backpropagation technique. This methodology to approximate the Hessian matrix is way more faster and less complex.
This approximation is used in the LM algorithm as in a Newton like update:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \left[\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}\right]^{-1} \mathbf{J}^T\mathbf{e} \tag{3.5}$$

Depending on the value of μ we can have:

- $\mu = 0 \rightarrow$ Newton's method with an approximate value of H;
- μ is large $\rightarrow$ Gradient's descent with a small step size;

Since Newton's approach is quicker and precise when used in close proximity to an error minimum, the goal is to pass as soon as possible to Newton's method.
The optimal choice for the damping parameter μ has been supported by a number of more or less heuristic considerations. When, from one step to another, the performance function decrease, the value of μ is decreased. On the other hand when a step fails, leading to an increase of the performance function, then the value of μ is increased.
Finally, we can state that the power of the Levenberg-Marquardt algorithm lies in the possibility to update the damping parameter μ between the value of Gauss Newton's method and gradient descent depending on the situation. [43]
For example, if the initial guess is similar to the optimal solution then the Newton's method would lead to faster convergence w.r.t the Levenberg-Marquardt. On the other hand, when initial guess is far away to the optimal solution, the LM behaves firstly like a steepest descent approach moving toward the optimal solution area, then the minimum is found with small value of the damping parameter. [44]
This method is helpful since it suits perfectly our need to solve a system of nonlinear

equations.
However, we must remind that, as every optimization algorithm, the LM is designed to find a local minimum rather than a global one.

Chapter 4

Lyapunov L_1 - Lyapunov L_2

In this chapter several transfer maneuvers between two Lyapunovs are investigated. Plenty of CR3BP orbits are gathered for different C-levels and can be found at [45]. From this dataset a Lyapunov around L_1 is selected as initial orbit while a Lyapunov around L_2 with similar C-level is chosen as final orbit. These kidneys-shaped orbits are propagated for a period and illustrated in Fig.(4.1).

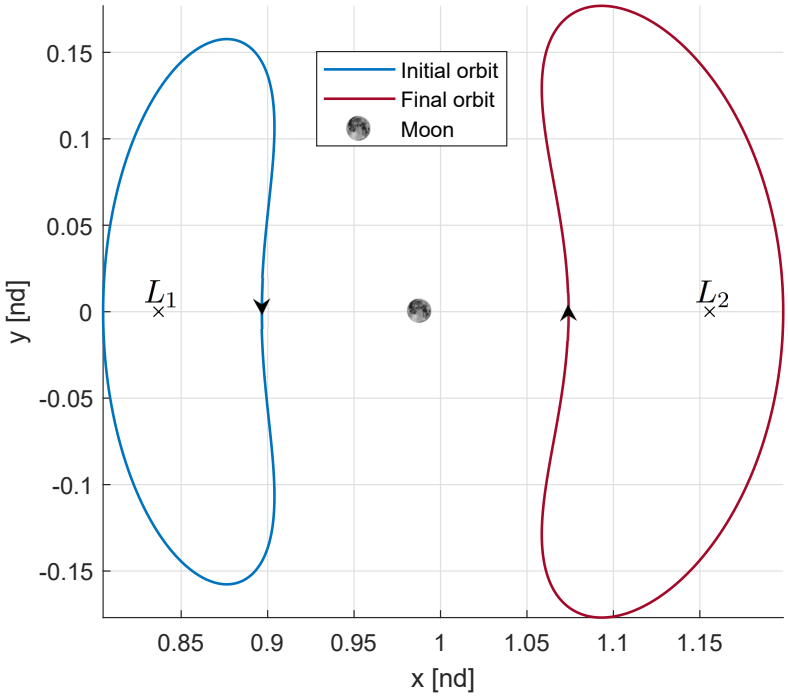


Figure 4.1: Lyapunovs considered

The characteristics of both these orbits are reported in Tab.(4.1).

Orbit	Jacobi Constant (LU^2/TU^2)	Period (days)
Lyapunov L_1	3.096	13.97
Lyapunov L_2	3.029	18.20

Table 4.1: Characteristics of the Lyapunovs considered

In order to familiarize with the problem as first step a time fixed formulation is considered. A spacecraft, indeed, may need to undertake an unforeseen transfer maneuver between two orbits at some point during its journey, for both engineering and scientific purposes, with strict timeline requirements.
On the other hand, another usual mission requirement is the need to perform a transfer with the minimum control energy cost, regardless the transfer time. From this latter consideration comes the need to study also a free time formulation.
In both the formulations the states boundary conditions (i.e. initial points and final points) are fixed. Some discretizations on both the orbits are taken into account to collect different transfer maneuvers and are further explained in the following sections.
As expected one of the major challenge dealing with CR3BP non-linearity is the wide variety of possible transfers between the same two points. In Sec.(1.2) we already mentioned the existence of families of orbits. At this point the engineer's know-how is an essential feature to collect data with reasonable criteria. As a matter of fact, the neural network may have trouble in the learning process if transfers between similar points are gathered as completely unrelated trajectories. One example that can clarify this latter consideration is the distinction between Inner and Outer orbits[1]. In the Fig.(4.2) two couple of points are considered and both kind of transfer trajectories are obtained through the time fixed formulation. It can be easily seen the significant difference in terms of geometry and transfer time, thus if we decide to collect for example an inner orbit between two points, we are encouraged to do the same for a similar transfer, preventing an important noise in the data. In this thesis we decided to focus on Inner orbits that have a significant application in plenty of Lunar missions. Nevertheless one should consider to collect Outer orbits depending on the mission plan.

[1]Usually Inner orbits are classified as orbits in the vicinity of the Moon while Outer orbits are the ones that extend well beyond the Near lunar region.

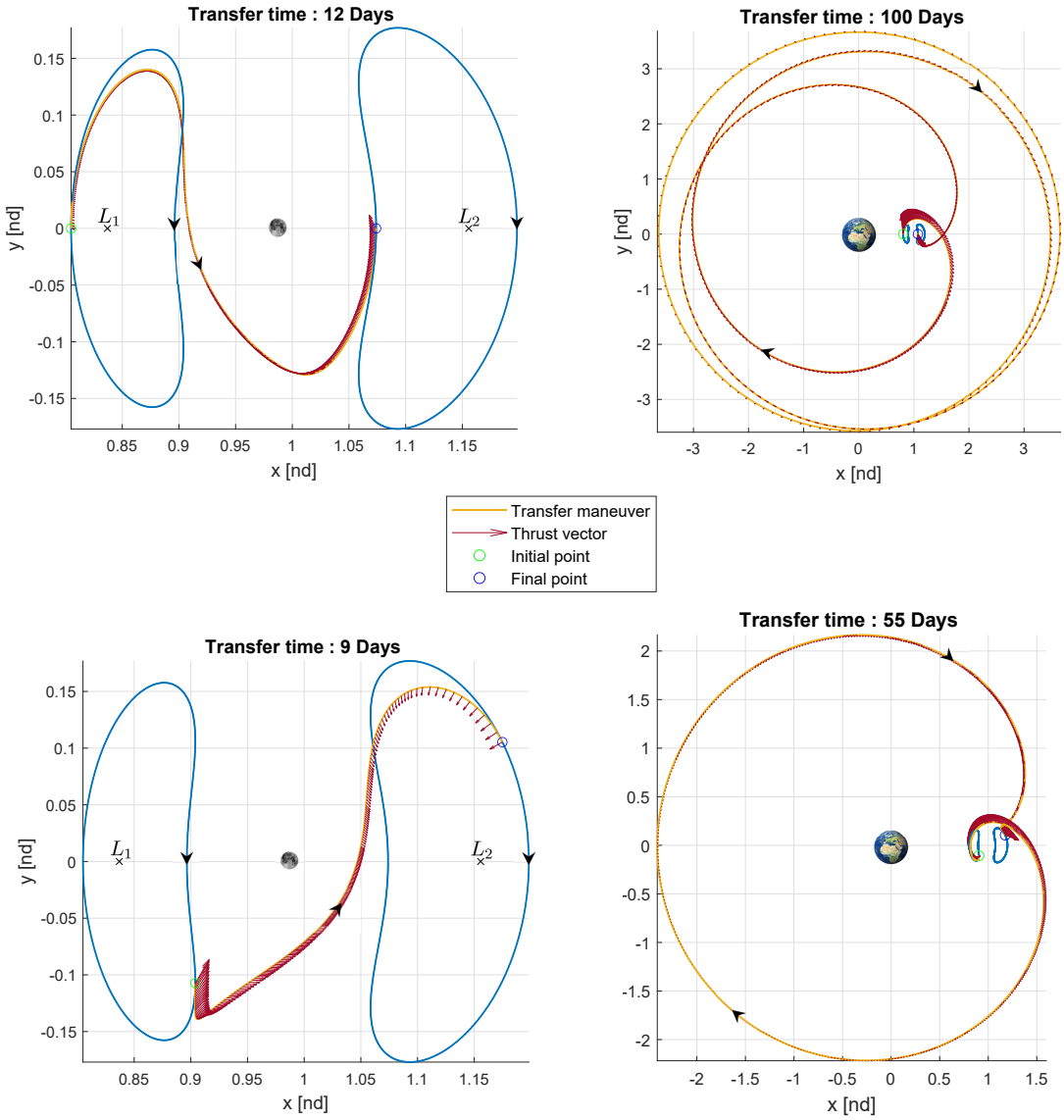


Figure 4.2: Examples of Inner orbits (left) and Outer orbits (right) obtained from the time-fixed formulation

4.1 Time fixed formulation

In this formulation multiple time fixed minimum energy problems are solved using the indirect method. Since initial states, final states and transfer time are defined, the goal of the net in this case is to map the following relationship:

$$[x_0, x_f, t_f]^T \rightarrow [\lambda_0]^T \tag{4.1}$$

where:

$$\begin{aligned} \boldsymbol{x_0} &= [x_0, y_0, z_0, \dot{x_0}, \dot{y_0}, \dot{z_0}] \\ \boldsymbol{x_f} &= [x_f, y_f, z_f, \dot{x_f}, \dot{y_f}, \dot{z_f}] \\ \boldsymbol{\lambda_0} &= [\lambda_{1_0}, \lambda_{2_0}, \lambda_{3_0}, \lambda_{4_0}, \lambda_{5_0}, \lambda_{6_0}] \end{aligned} \tag{4.2}$$

Thus the input of the net is a I_{13x1} while the output is a O_{6x1}. Once the data are gathered and the neural network trained, the methodology followed in this work to simulate a real-time on-board application is the propagation of both the states and costates dynamics from (2.9). From the costates one can also rebuild the control history leveraging the Pontryagin's minimum principle, trying to reproduce the optimal trajectory calculated offline.

4.1.1 Data collection

First of all we define an appropriate time span. From a literature review on typical lunar mission we found reasonable to set an interval from 3 to 14 days for inner transfers. The time span is then divided in 300 nodes. In order to consider a wide variety of transfers, the initial orbit is also discretized in 100 nodes (i.e. initial states) while the final orbit is discretized in 3 nodes (i.e. final states). Thus, at the end of the data collection process we have 90000 optimal trajectories to train the net. Whenever we attempt to solve a TPBVP we have to provide good initial guesses (i.e. x_0 and λ_0) to the solver. Since initial point is fixed in the space, this problem reduces to find suitable initial costates. Due to the lack of physical meaning of these dual-variables this is a non-trivial activity. A lot of on going research on this field is going on in these last decades. Indirect methods have a narrow convergence compared to direct methods. One approach to find good initial guesses for the indirect method could be to provide the solution of a direct method, which is less sensitive to the initial guesses. However direct methods do not have costates (or adjoints) in their formulations. Thus the main problem in this latter methodology is the compatibility between the two methods. In addition commercial solvers for direct methods, such as Matlab's *fmincon* are computationally and time expensive just for a single transfer maneuver. Thus, the most prevalent approach in CR3BP literature dealing with indirect methods, is to provide random generated

initial costates to the solver. The work logic behind the algorithm for the data collection in the fixed time formulation is reported in Alg.(1).

Algorithm 1 Data collection for time fixed formulation

1: **for** i=linspace(1,length($LyapL_1$),100) **do**
2: **for** j=linspace(1,length($LyapL_2$),3)) **do**
3: ▷ Pick the first transfer time
4: ▷ **Solve TPBVP** 500 times with random initial guesses
5: ▷ **Collect the solution** with the minimum energy value
6: **for** k=linspace(3,14,300) **do**
7: ▷ **Solve TPBVP** using the previous solution as initial guess
8: **end for**
9: **end for**
10: ▷ Clean data removing outliers
11: **end for**

The most useful expedient in the algorithm is the one at the point 7. In order to expedite the data collection, within the inner loop on the time span, we found effective to provide the previous solution as initial guess. This methodology operates as a numerical continuation method. Essentially the TPBVP that we solve at the $(k+1)_{th}$ step is a slight perturbation of the one that we solved at the k_{th} step.

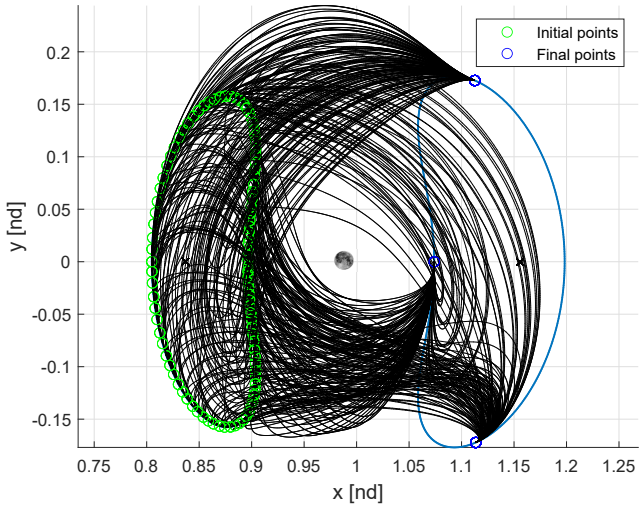


Figure 4.3: 360 of the 90000 orbits collected in the fixed time formulation

It goes without saying that this method gets more accurate as the mesh is more dense. In this case it is useful to implement this approach in the inner loop, on the 300 nodes of the time span. Following this procedures leads to remarkable advantages. First of all we save computational time and effort, but more importantly we gather the 90000 transfer trajectories with criteria, preventing the aforementioned problems of different families. At the end of the algorithm the respective initial states, costates, transfer time and final states (4.2) are collected for the neural network.

4.1.2 Neural Network performance

Using the Matlab's Deep Learning Toolbox ® a feedforward Neural Network is implemented. Prior to the learning phase, all the inputs are normalized in the range [-1,1] because of their high variance. This step is done for every NN of the thesis. The performance of the net are evaluated through the mean squared error (mse) and also with the relative error on the initial costates. As already discussed in Chapter 3 we can operate on different parameters to obtain high performance by the net. After several trial and error, adjusting the number of hidden layers and neurons, based on the net's response, we get the final structure of the net that is illustrated in Fig.(4.4).

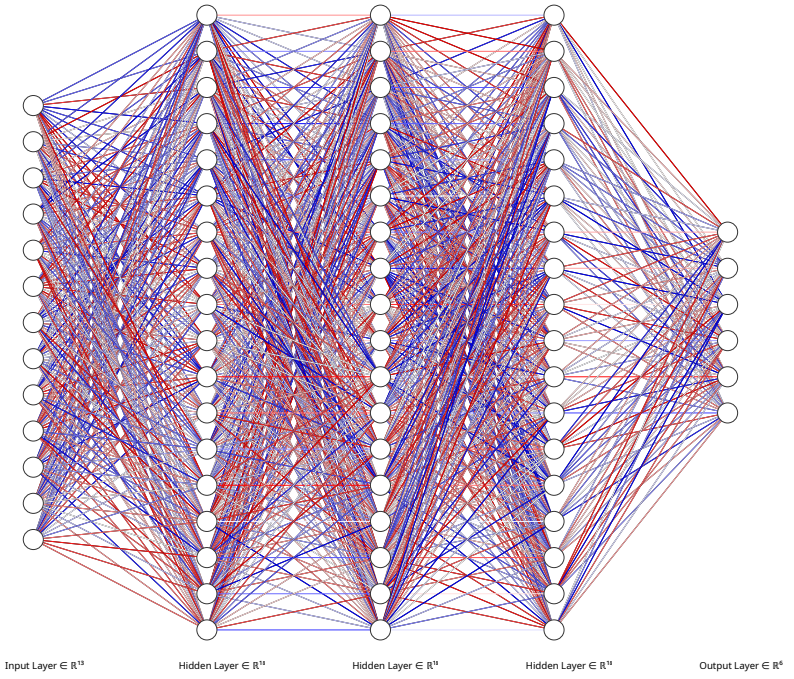


Figure 4.4: Scheme of the neural network for the fixed-time formulation

The network's structure can be summarized as follows:

- **Architecture**: 3 hidden layers with 18 neurons each;
- **Activation functions**: tanh for the hidden layers, linear for the output layer;
- **Optimizer**: Levenberg-Marquardt;
- **Batch division**: 70% training, 15% validation, 15% test;
- **Results**: MSE $\sim$ 2.37e-05; 94% of the test data has a relative error $\leq 5\%$
- **Training time**: 30 min.

In Fig.(4.5) the performance of the net are reported as a mse graph in function of time epochs. Train, validation and test data all follow the same path and at the end of the training their trajectories overlay each other. This trend highlights the quality of the net that can also generalize new data with small error since the test data goes along with the train one except in the last epochs, when indeed the network is forced to stop.

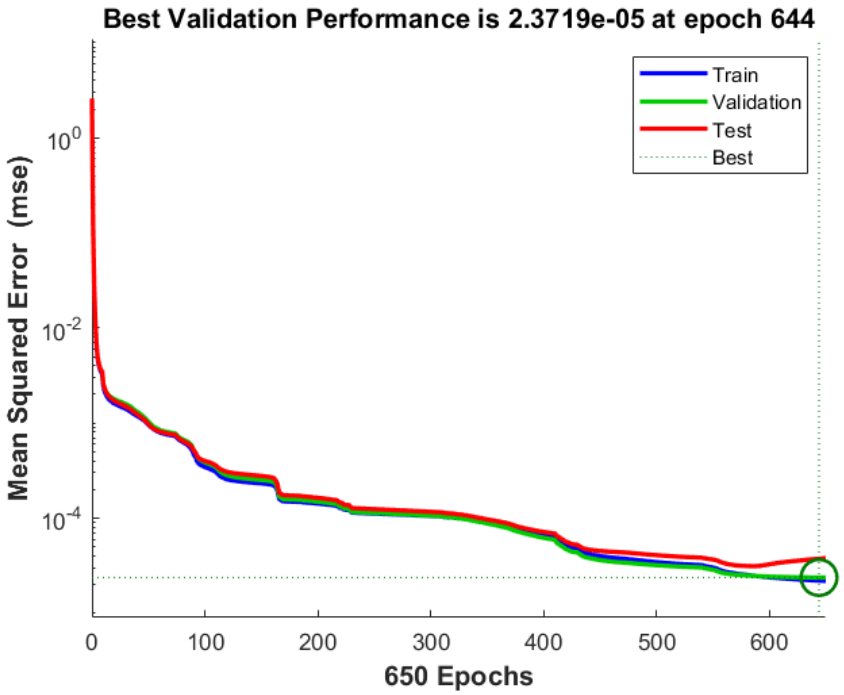


Figure 4.5: NN trained on 90000 transfers beetween 2 Lyapunovs

4.1.3 Results

Once the neural network is trained we can measure its performance.
Let's consider for example a real time application on board, summarized by the following work logic:

1. **Requirement**: The spacecraft must perform a transfer between an initial point on the Lyapunov around L_1 and a final point on the Lyapunov around L_2 with a precise transfer time and minimizing the control energy, thus we know $[x_0, x_f, t_f]^T$;

2. **NN recall**: the net will predict the initial costates vector $[\lambda_0]^T$ with an error depending on the NN performance obtained on ground;

3. **NN prediction propagation**: in order to re-build the optimal control history and the transfer trajectory we can leverage the dynamics equations (2.9) and propagate them forward.

This latter procedure works fine whenever the error on the NN prediction is small. In Fig.(4.6) both transfer trajectory and control history of a 7 days maneuver are reported. In this case the error on the initial costates is 0.1361% and this leads to strong perfomance. From the plots we can see that the NN prediction goes along with the optimal trajectory path except in the last section.
The controls'plot is also indicative of the remarkable performance, since both the path of optimal and NN prediction overlay each other.
However the goal of the thesis is to reproduce the optimal control history calculated offline on ground, trying to follow the optimal path all along the transfer rather than to minimize just the error on the final state. This is the first reason that suggest the need for a 4_{th} step on the work logic presented above. In addition to this consideration, we must evaluate also the performance of the net when the error on the initial costates is greater.
In Fig.(4.7) the error on the prediction of the net is 2.9072%. In this case the net's output gathers error in the propagation and this leads to a considerable mismatch through the second half of the transfer orbit that can be noticed also in the control's plot.

4. **Recall the TPBVP solver**: using the propagation of the neural network's prediction as initial guess for the on board solver considerably decreases the computational time and effort to solve the TPBVP. We can see the results of this latter approach in Fig.(4.8) for the case of an error of 2.9072% on the λ_0 prediction. The transfer maneuver obtained from both NN and bvp4c recalls, that is plotted with a red solid line, overlays with the optimal one (i.e. the red-dashed line) and this highlights that the results are promising.

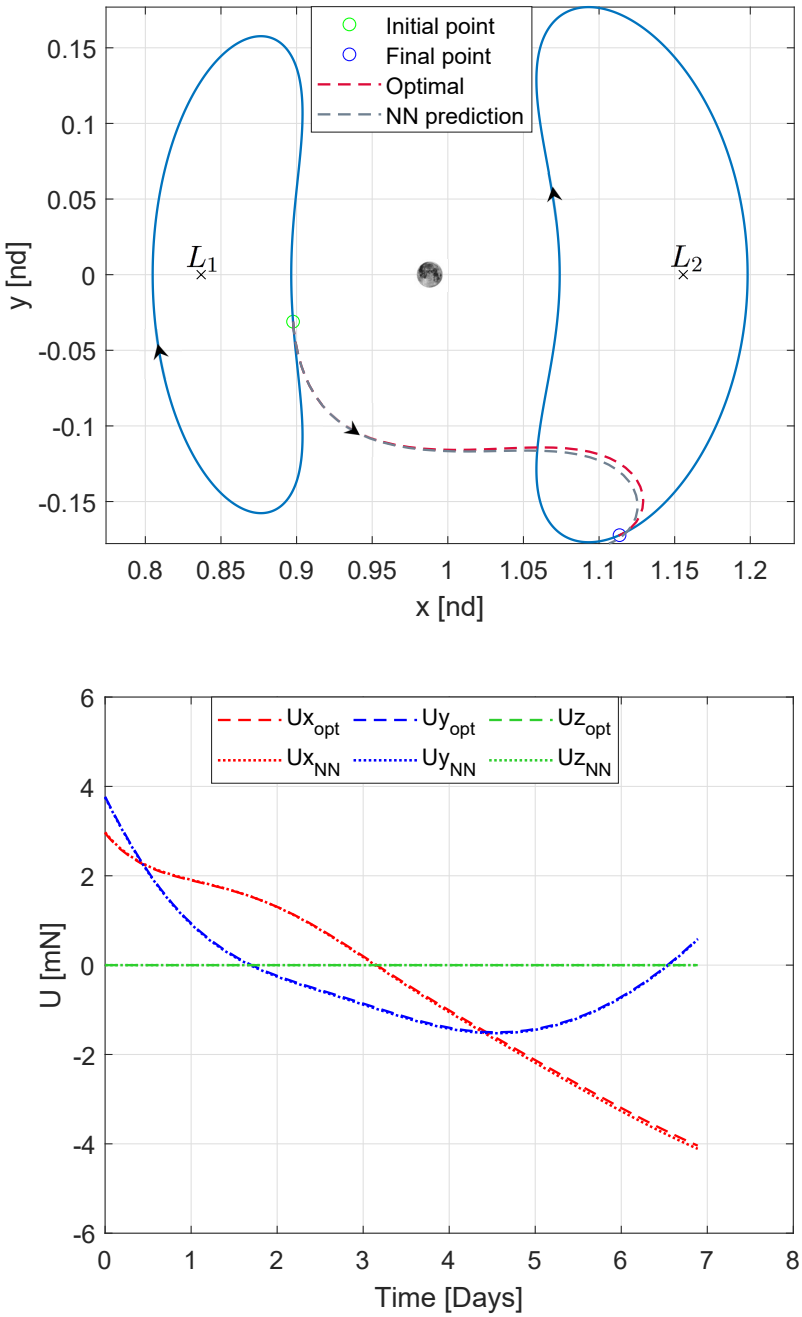


Figure 4.6: Transfer maneuver and control, propagation performed with an error of 0.1361% on the initial costates

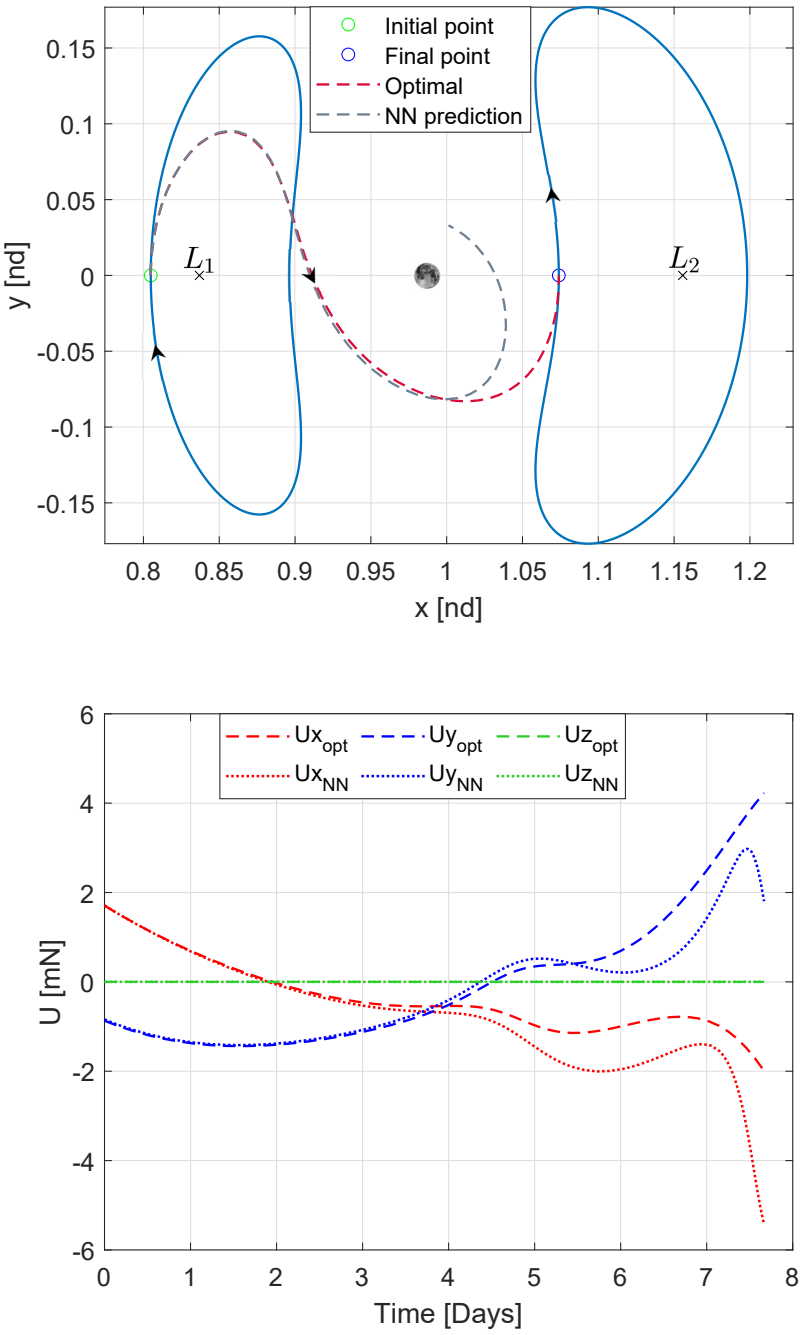


Figure 4.7: Transfer maneuver and control, propagation performed with an error of 2.9072% on the initial costates

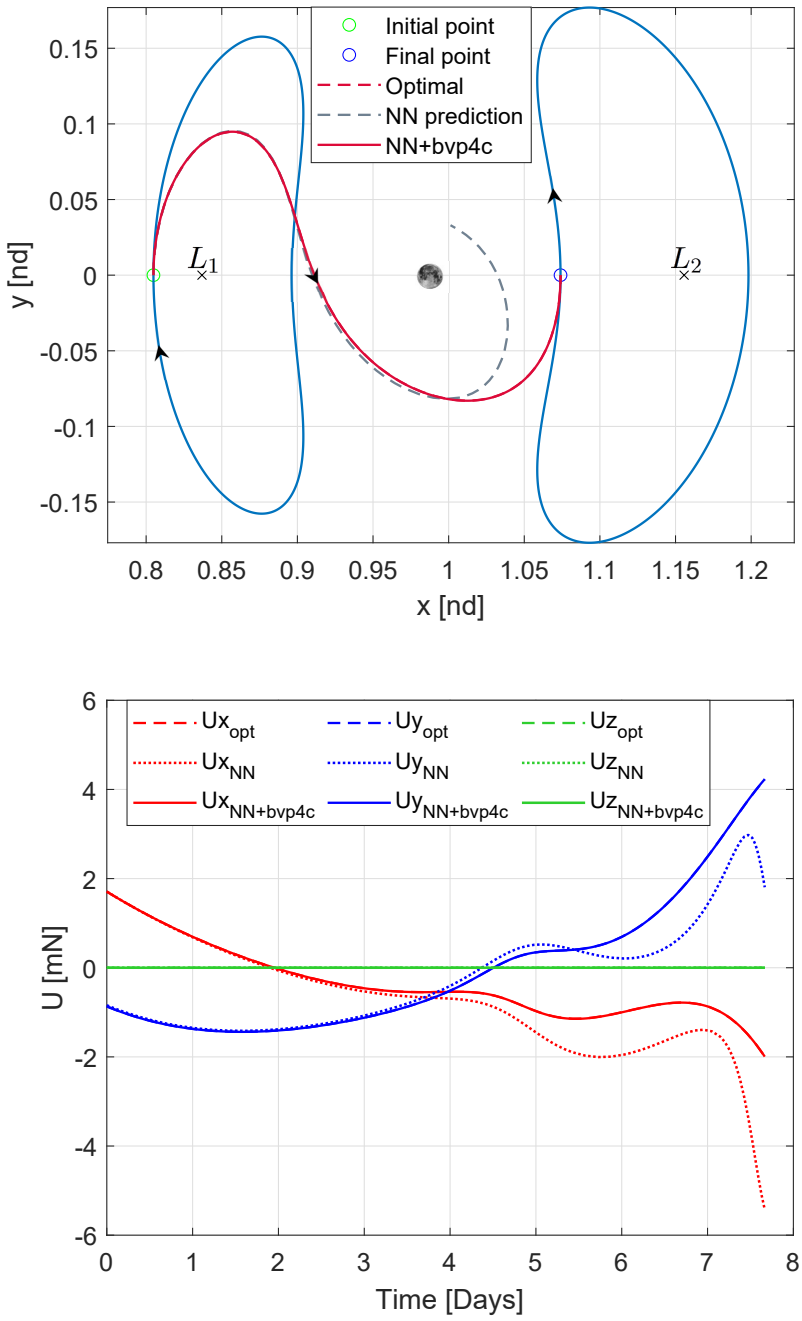


Figure 4.8: Transfer maneuver and control, propagation performed with an error of 2.9072% on the initial costates

The time required to rearrange the guess and to run bvp4c in this latter case is approximately of 0.3 seconds on an Intel ® Core™ i7-7700HQ CPU @ 2.80GHz. Of course the on-board solver will be quite different from bvp4c but the idea behind the procedure is the same. In Fig.(4.9) transfers with different errors on λ_0 prediction are illustrated. The two on the top converge to the optimal trajectory while the two on the bottom fail. Sometimes indeed when the transfer time is large and the error on the initial costates is important, the NN's propagation highly diverges from the optimal solution and consequently bvp4c struggles to follow the optimal trajectory. A different approach with both forward and backward propagation to mitigate this error is discussed in the time free formulation.

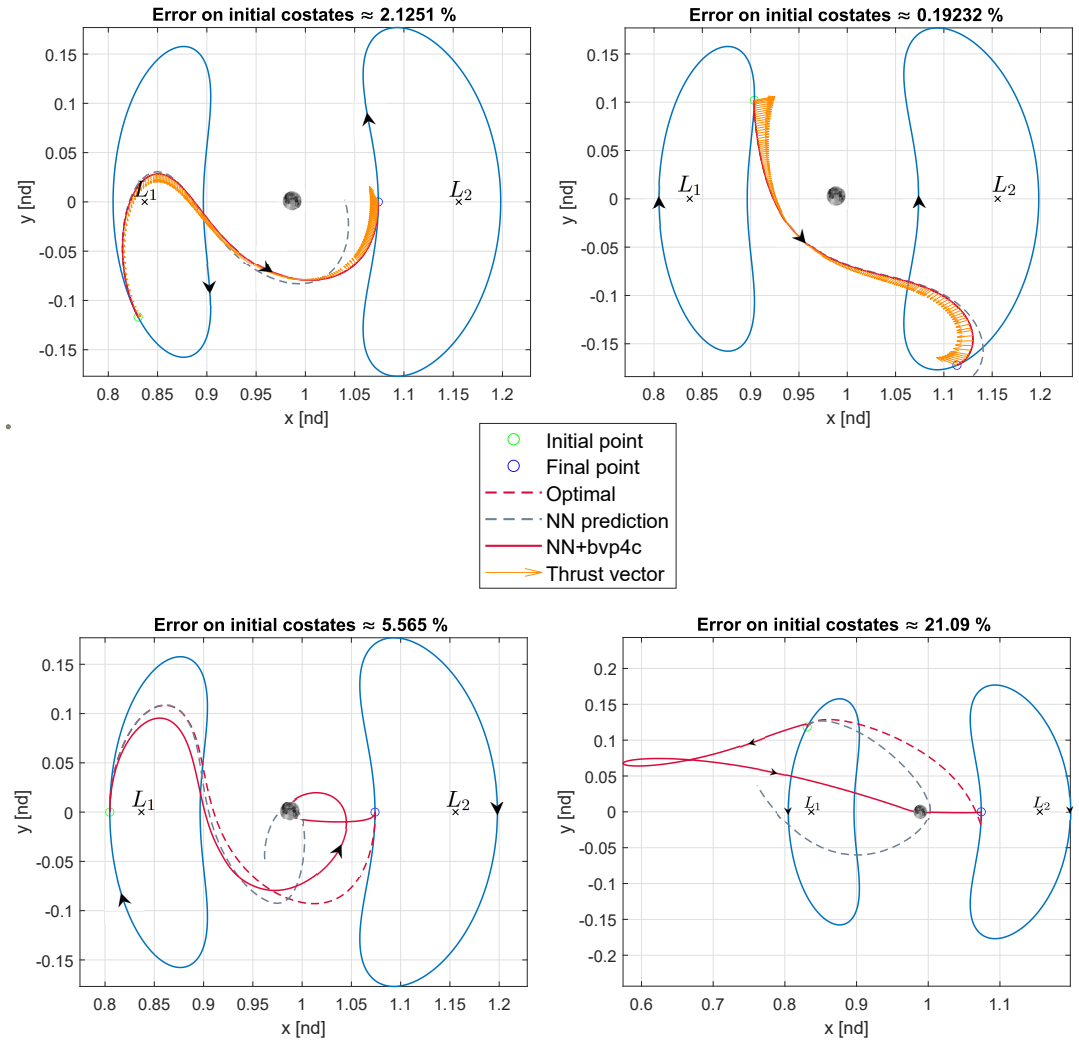


Figure 4.9: Transfer maneuvers for different errors on initial costates

4.2 Free time formulation

In this section multiple time free minimum energy problems are solved using always an indirect method. Since the transfer time is no longer a fixed parameter, but we need it to propagate the costates as seen in the previous section, the net has to map a different relationship:

$$[x_0, x_f]^T \rightarrow [\lambda_0, t_f]^T \tag{4.3}$$

This time the input of the net is a I_{12x1} while the output is a O_{7x1}.
A new free parameter is defined as:

$$\tau = \frac{t}{t_f} \Rightarrow \tau \in [0,1] \tag{4.4}$$

Substituting into the state equations and the cost function we get:

$$\frac{d\vec{x}}{d\tau} = t_f \cdot \vec{f}[\vec{x}(\tau), \vec{u}(\tau), \tau] \qquad J = t_f \int_0^1 u^2 d\tau \tag{4.5}$$

In this case an additional boundary condition is required:

$$H\left(t_f\right) = -\frac{\partial \Phi}{\partial t_f} = 0 \tag{4.6}$$

usually referred as transversality condition, where $\Phi = \phi + \mu^T \psi$.
The Hamiltonian at the final time t_f must be equal to zero because the terminal cost and the terminal boundary conditions are independent of time. [46]

4.2.1 Data collection

In order to train a NN a dataset is needed. Initial states and final states are fixed as in the previous formulation but the main difference is that this time we have a single optimal maneuver between two points rather than a series of transfers with different transfer times. As a matter of fact in this formulation the solver will try to find the optimal trajectory that minimizes the energy regardless the transfer time. Of course the distinction between inner and outer orbits is still an important issue that will be considered in the algorithm.
Since we have an univocal orbit between two points we decided to select a more dense mesh in terms of initial and final states to collect several trajectories. Thus, 50 points are considered in the initial Lyapunov orbit while all the available points in the final orbit are selected (i.e. states available from the propagation of the data at [45]). At the end of the data collection process we are going to have 32000 transfer trajectories to train the net. In Alg.(2) the steps followed in order to build the dataset are presented.

Algorithm 2 Data collection for free time formulation

1: ▷ **Solve TPBVP** between the first pair of states 500 times with random initial guesses
2: ▷ **Save solution** with minimum energy and its random initial guesses
3: ▷ **Perturb** slightly these initial guesses
4: ▷ **Check** if the solution still be a local minimum
5: **if** solution is not a local minimum **then**
6: ▷ **Increase** the number of random iterations and return to 1
7: **end if**
8: **for** i=linspace(1,length($LyapL_1$),50) **do**
9: ▷ Using the solution of point 2 as initial guess
10: ▷ **Solve TPBVP** between i-th state and $LyapL_2$ first point
11: **for** j=2:length($LyapL_2$) **do**
12: ▷ **Solve TPBVP** using the previous solution as initial guess
13: ▷ **Collect the solution**
14: **end for**
15: **end for**

The procedure is quite different compared to Alg.(1). This time as first step, prior to enter the loops for the data collection, we search for an optimal solution between the first pair of states, using always randomized initial guesses and iterating the process 500 times, saving the minimum energy solution. Then, the random initial guesses that led to this latter optimal solution are slightly perturbated, and several TPBVP are solved in order to check if the solution represents effectively a local minimum. From Fig.(4.10) we can see how with 200 perturbations in the $\pm 1\%$ range, the guess used remains a good local minima in comparison with the chaos of other solutions.
If this check fails the algorithm returns to point 1, increasing the number of random iterations. On the other hand if the solution represents a local minimum we can consider it as optimal and we are encouraged to use it as initial guess for a similar TPBVP. As a matter of fact a continuation method is leveraged also in this formulation. We already stated that the mesh of the second orbit is composed by all the possible final states from the Nasa's dataset [45]. Thus, the best choice is to use the continuation method on the inner loop, where initial state is the same, the mesh is more dense and we solve a TPBVP for every final point of the Lyapunov L_2 orbit, using the previous solution as initial guess. The continuation method presented has to be reset at every iteration i of the outer loop (i.e. when the point of initial orbit's mesh changes) otherwise the algorithm would eventually lead to unlikely large transfer times.
In Fig.(4.11) the family of the orbits gathered is illustrated. A clear difference in comparison with the data collection of the time fixed formulation, Fig.(4.3), is the

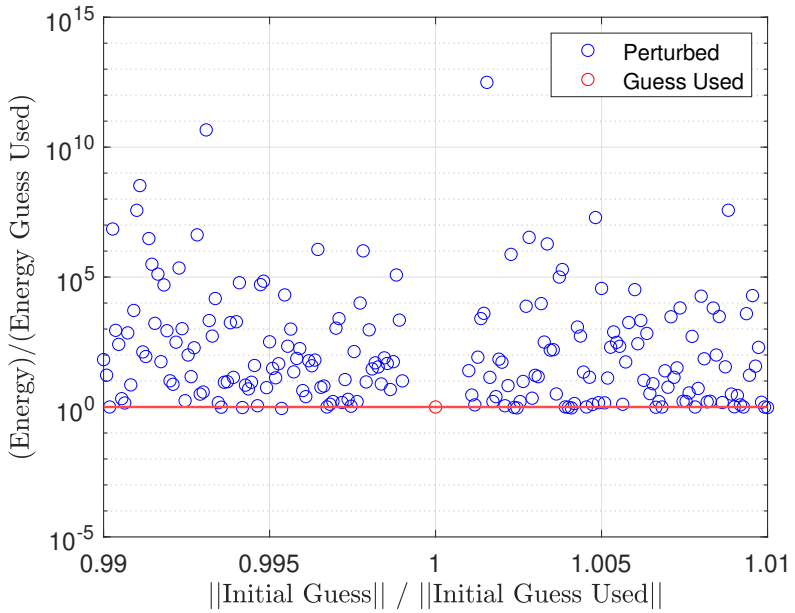


Figure 4.10: Energy distribution for a slight perturbation of optimal initial guess

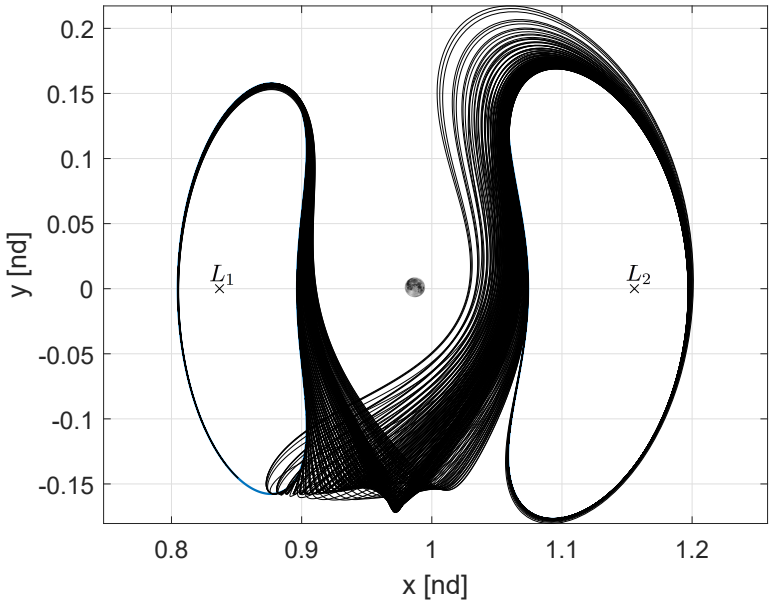


Figure 4.11: 360 of the 32000 orbits collected in the free time formulation

well-ordered pattern of the time free formulation.
When the time is a free variable, the optimizer can look for a trajectory with a shorter or longer total flight time that minimizes the cost function. Since our cost function is the control energy, which is related to the minimum fuel function, it is obvious that with this formulation the solver is put on the path to converge to solutions that highly leverage natural dynamics.
Although manifolds and heteroclinic connections have not been investigated directly in the formulation, the family obtained in the figure, and the results presented in the following sections, suggest that natural path are implicit solutions when we do not constrain the transfer time.

4.2.2 Neural networks performance

After the data collection another neural network is built with the Matlab's Deep Learning Toolbox ®. As already stated the input of the net is a I_{12x1} (i.e. initial and final states) while the output is a O_{7x1} (i.e. initial costates and transfer time). In Fig.(4.12) the network's architecture is illustrated. Note that the color of the edges which is correlated to their value (negative edges in blue, positive edges in red, default edges in black) is given just as an example of what we should expect within a NN.

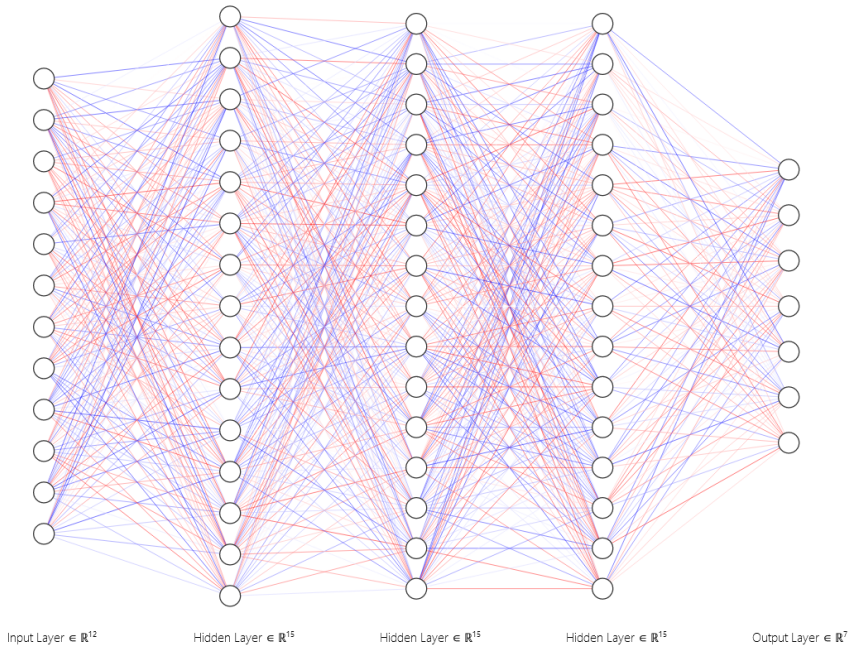


Figure 4.12: Illustrative scheme of the net.

The network's structure can be summarized as follows:

- **Architecture**: 3 hidden layers with 15 neurons each;
- **Activation functions**: tanh for the hidden layers, linear for the output layer;
- **Optimizer**: Levenberg-Marquardt;
- **Batch division**: 70% training, 15% validation, 15% test;
- **Results**: MSE $\sim$ 9.53e-08; 100% of the test data has a relative error $\leq 1\%$
- **Training time**: 50 min.

Once again the performance of the net are evaluated with the mse and the relative error on the output $[\lambda_0, t_f]$. In this case the net reached slightly better accuracy, with the 100% of the test data that has a relative error $\leq 1\%$.

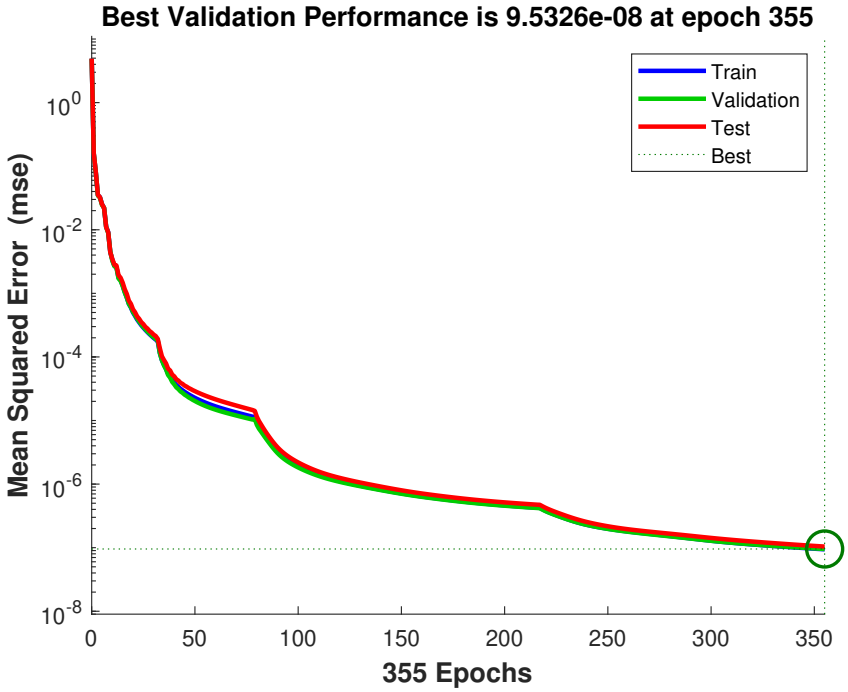


Figure 4.13: NN trained on 32000 transfers beetween 2 Lyapunovs

Certainly this behaviour is due to the considerations on the pattern of the data collected, explained in the previous section. A more homogeneity of the solutions gathered, indeed, bring the net to overcome the performance of the network of the time fixed formulation, which was trained on a more chaotic dataset.

4.2.3 Results

The nature of trajectories obtained through the indirect formulation is highly dependent on the mission requirements and the cost function. Let's take for example a transfer maneuver between two points of the Lyapunovs, calculated with the time free formulation Fig.(4.14). It is obvious that in a fixed time formulation a transfer trajectory between the same two points would lead to a solution with different shape and geometry, similar to the ones presented in Fig.(4.9).
Furthermore in the time fixed formulation, when we need to perform the transfer in a short time, (i.e. Inner orbits) the thrusters are usually switched on all along the maneuver. On the other hand in the free time formulation usually we can have sections where thrusters are switched off and the spacecraft leverages natural path. These latter sections are usually referred as *coasting phases* [2] and can be noticed in Fig.(4.14). The level of thrust decreases as the spacecraft inserts in the natural path and drifts to the final point driven by the CR3BP dynamics.
In this first example the transfer time is 21 days and the error on the NN prediction is 0.010%. The on-board work logic used is the same of the previous formulation:

1. **Requirement**: The spacecraft must perform a transfer between an initial point on the Lyapunov around L_1 and a final point on the Lyapunov around L_2 without any particular constraint on the transfer time, but minimizing the control energy, thus we know $[x_0, x_f]^T$;

2. **NN recall**: the net will predict the vector $[\lambda_0, t_f]^T$ with an error depending on the NN performance obtained on ground;

3. **NN prediction propagation**: in order to re-build the optimal control history and the transfer trajectory we can leverage the dynamics equations (2.9) and propagate them forward;

4. **Recall the TPBVP solver**: the on board solver can be run leveraging as initial guesses the propagation of net's prediction.

In Fig.(4.15) different transfers are illustrated. All the maneuvers converge to the optimal ones and the final trajectory, once again represented by a red solid line, completely overlays the optimal one. From this figure we can also see that this

[2]Note that in this work the thrusters dynamics is not taken into account.
Usually thrusters are ON/OFF devices, they can be either switched on at maximum thrust or be completely switched off. This implies discontinuity in the controls and the use of switching functions that are challenging for numerical methods.
In this thesis the controls are continuous functions, thus we can just refer to *coasting phases* when the level of thrust is approximately zero.

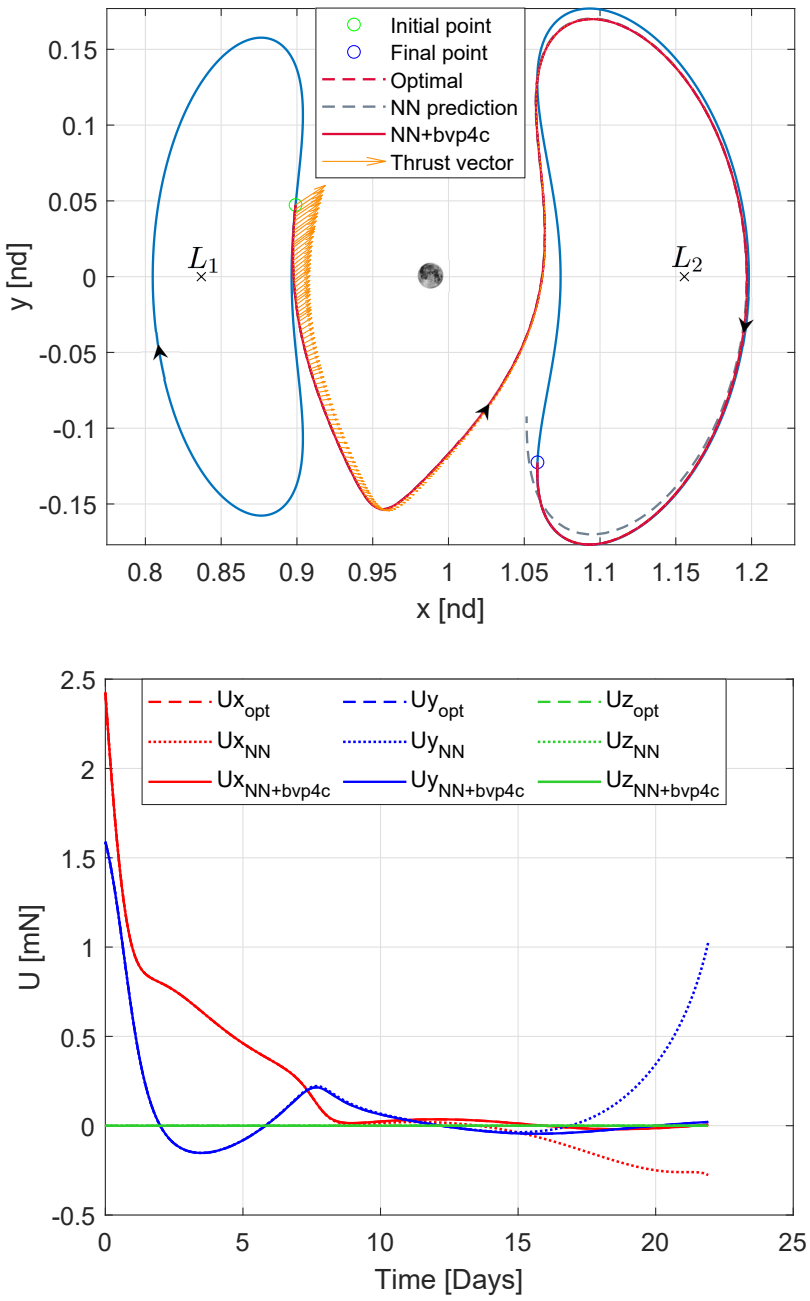


Figure 4.14: 21 days transfer maneuver and control, propagation performed with an error of 0.010% on the NN prediction

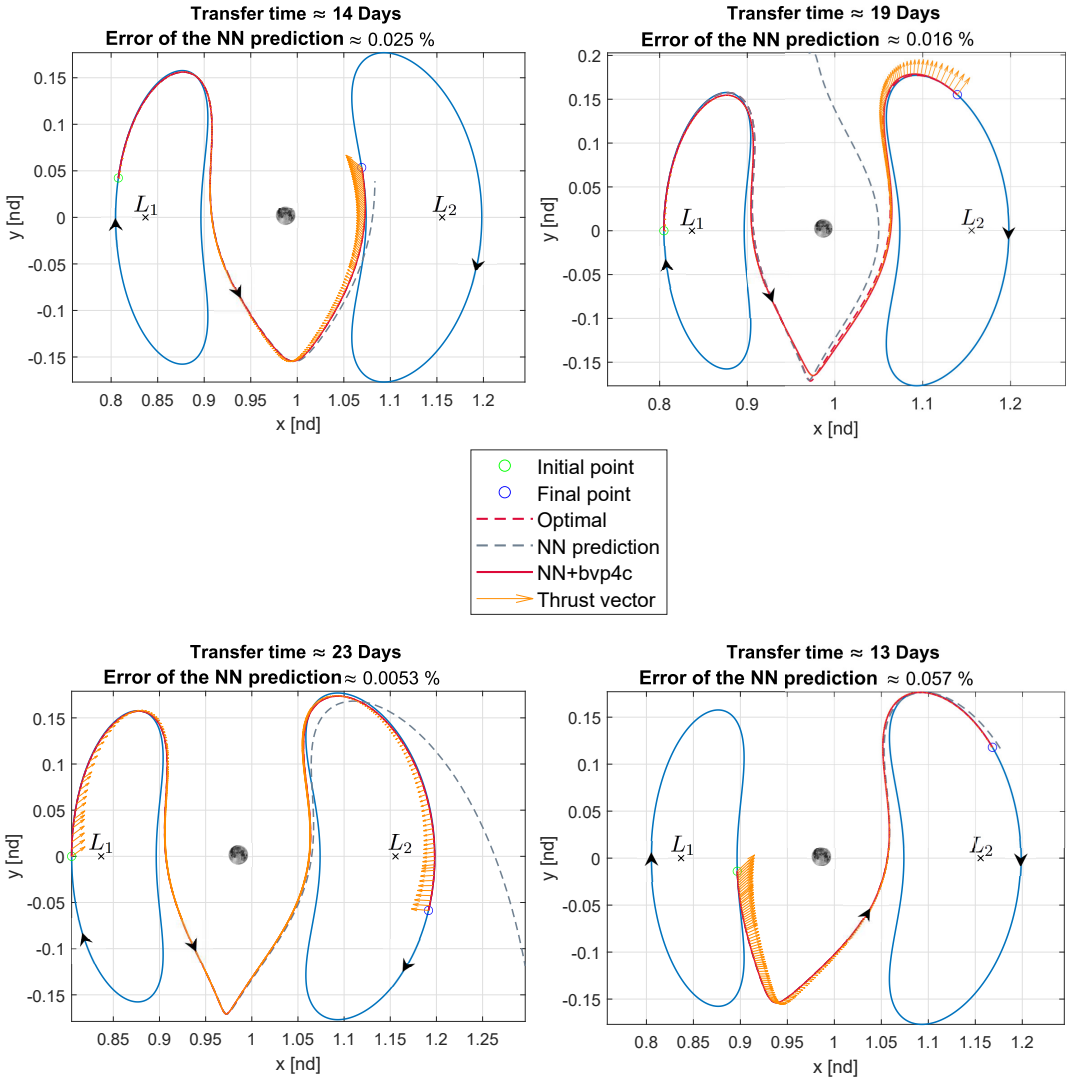


Figure 4.15: Transfer maneuvers for different errors of the NN prediction

approach leads to convergence a wide variety of maneuver even though the propagation highly diverges from the optimal one and the error on the final state is remarkable. Nevertheless in some cases the TPBVP solver struggles to follow the optimal trajectory even with this last methodology, as in Fig.(4.16).
This latter behaviour may could due to the high sensitivity of that particular solution to initial guesses and to the numerical error propagation of the NN's prediction. For this reason a different approach must be investigated in order to improve convergence also for these particular cases.

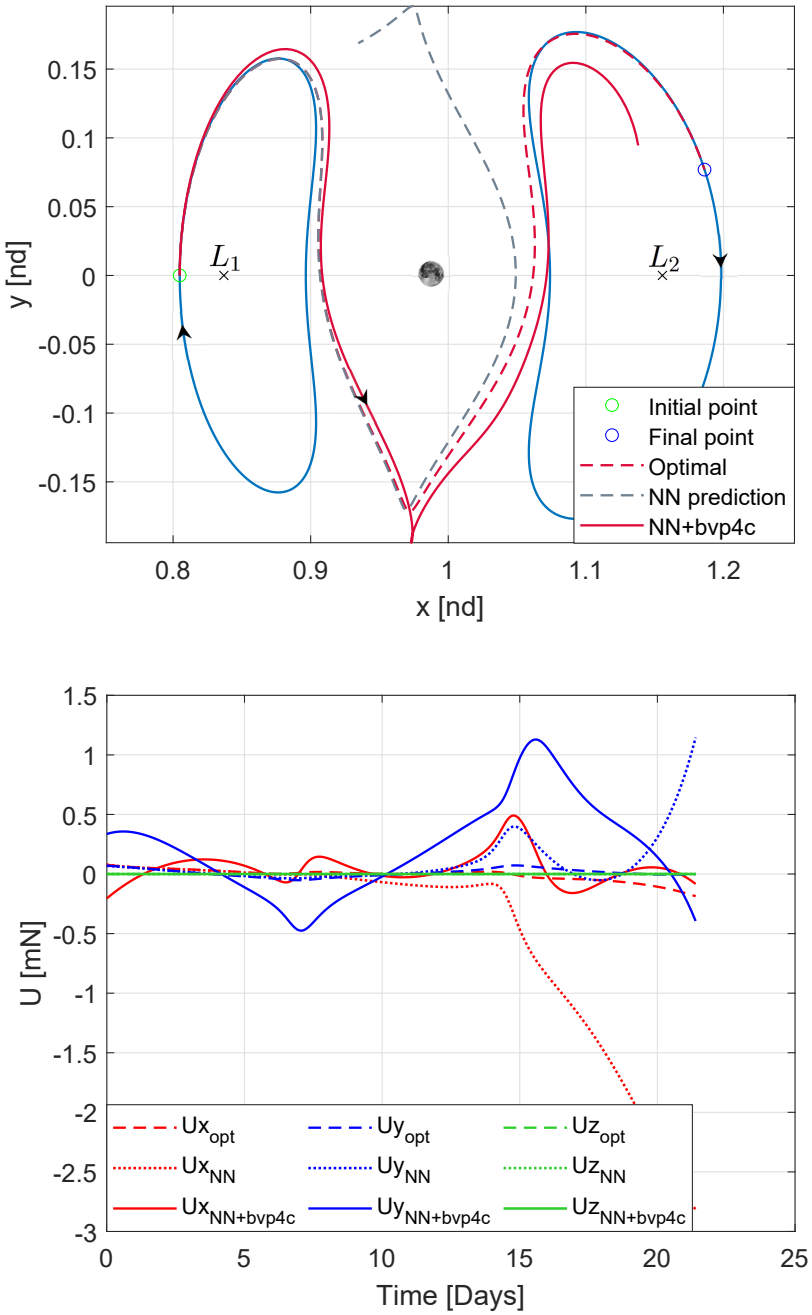


Figure 4.16: 21 days transfer maneuver and control, propagation performed with an error of 0.011% on the NN prediction

4.3 Forward-Backward propagation

So far both the neural networks built had to map a relationship to obtain initial costates (and transfer time in the free time formulation). At this point it is clear that one of the major drawbacks of this approach is the growth of the error of the net's prediction once it is propagated forward. A possible solution to improve convergence is to include also final costates in the output of the net.
The net should map the following relationship:

$$[x_0, x_f]^T \rightarrow [\lambda_0, \lambda_f, t_f]^T \tag{4.7}$$

The idea behind this approach is schematized in Fig.(4.17).

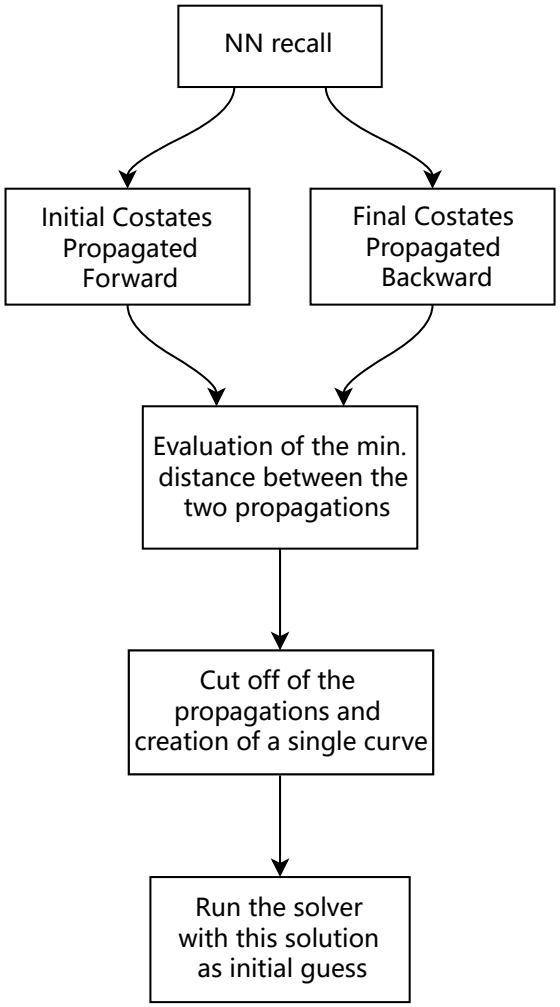


Figure 4.17: On board work logic for a NN that maps both initial and final costates.

The main goal of this approach is to mitigate the error propagation that occurs once the initial costates are propagated forward. If the net predicts with adequate accuracy both initial and final costates, we can propagate them forward and backward respectively, decreasing the error propagation. From the author experience most of the time the two propagations intersect each other, thus we can split them in the intersection point and create a single curve. In general we cut off the propagations where their distance is minimum. Although with this approach we can face some discontinuities, the solver runs smoothly and converges to optimal solutions even if we do not conjoin forward and backward propagation.
Neverthless, future works could investigate this latter aspect, with, for example, the application of spline functions.

4.3.1 Neural network performance

A neural network is built in the same Matlab's environment. This time the input of the net is a I_{12x1} (i.e. initial and final states) while the output is a O_{13x1} (i.e. initial costates, final costates, transfer time).
In the Fig.(4.18) the net's architecture is illustrated.

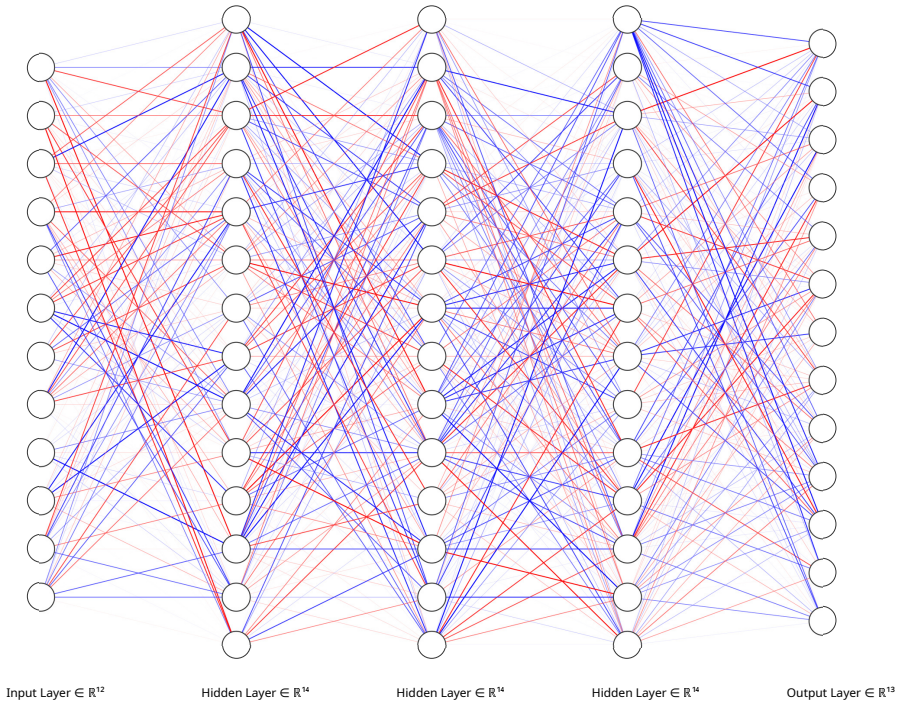


Figure 4.18: Illustrative scheme of the net.

The network's structure can be summarized as follows:

- **Architecture**: 3 hidden layers with 14 neurons each;
- **Activation functions**: tanh for the hidden layers, linear for the output layer;
- **Optimizer**: Levenberg-Marquardt;
- **Batch division**: 70% training, 15% validation, 15% test;
- **Results**: MSE $\sim$ 3.89e-06; 99% of the test data has a relative error $\leq 1\%$
- **Training time**: 1h 10 min.

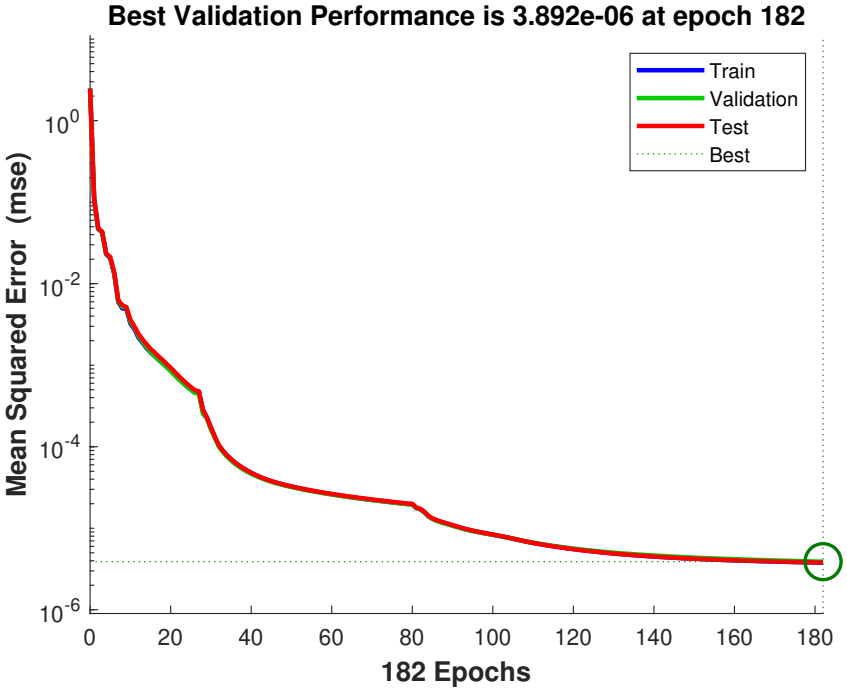


Figure 4.19: NN trained on 32000 transfers beetween 2 Lyapunovs

4.3.2 Results

We can consider the transfer failed in Fig.(4.16) as a representative case to test this method. A transfer maneuver between the same two points is considered. The results are illustrated in Fig.(4.22). On top of the figure forward and backward propagations are depicted and as we can see they follow the optimal trajectory.

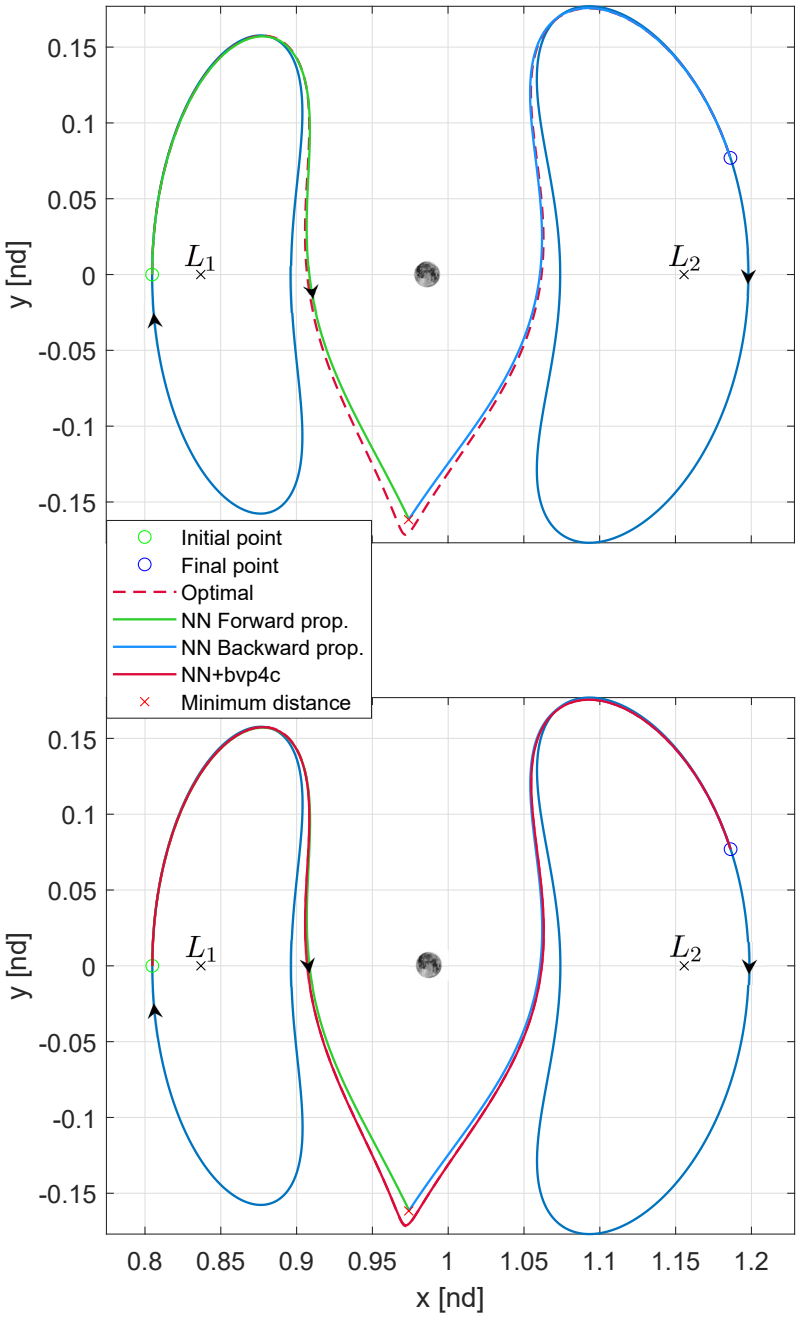


Figure 4.20: 21 days transfer maneuver with Forward propagation on top, Forward+Backward propagation on bottom.

The two propagations are split approximately in the middle, where indeed their distance is minimum. The resulting curve is given to the TPBVP solver as initial guess, leading to convergence as it can be seen on the bottom of the figure. The elapsed time to operate forward and backward propagations, to rearrange the initial guess and to run the solver, thus the time needed to execute the work logic of Fig.(4.17), is approximately of 1.12 seconds on an Intel ® Core™ i7-7700HQ CPU @ 2.80GHz.
In Fig.(4.21) the controls are reported.

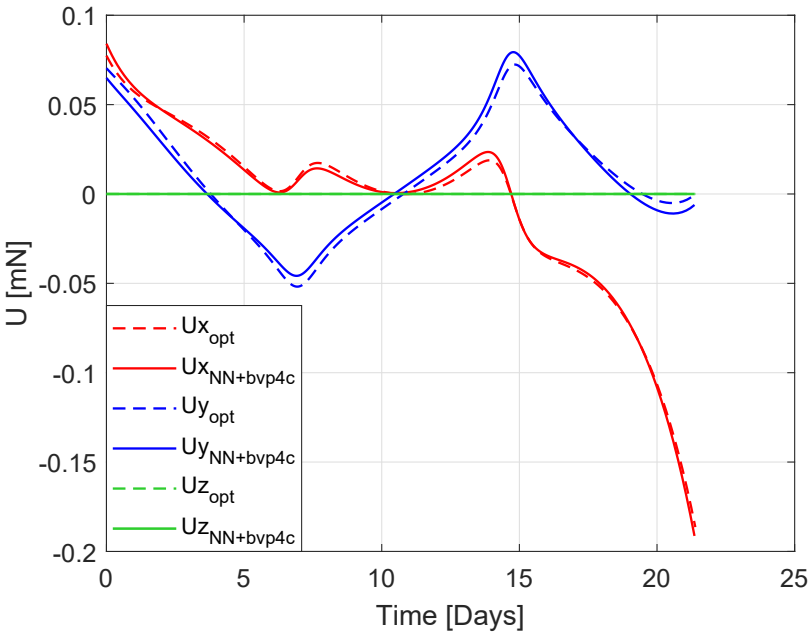


Figure 4.21: 21 days transfer maneuver controls with Forward+Backward propagation.

As we can see the controls path follows the optimal one with a minimal error. Sometimes following the "classical" methodology of forward propagation it can occur that the trajectory computed by the solver matches the final state but its path does not follow the optimal one along the maneuver, such as on the top of Fig.(4.22). As already stated the goal of this work is to reproduce entirely the optimal control path computed on ground when a transfer maneuver is required live on board. Applying the forward-backward methodology, illustrated on the bottom of the same figure, we can clearly see as all the trajectories (i.e. both propagations, optimal, and the one computed by the solver) overlay each other, matching also the final state at the end. The thrust vector is important at the beginning of the maneuver and it decreases its module as the spacecraft enters a natural path.

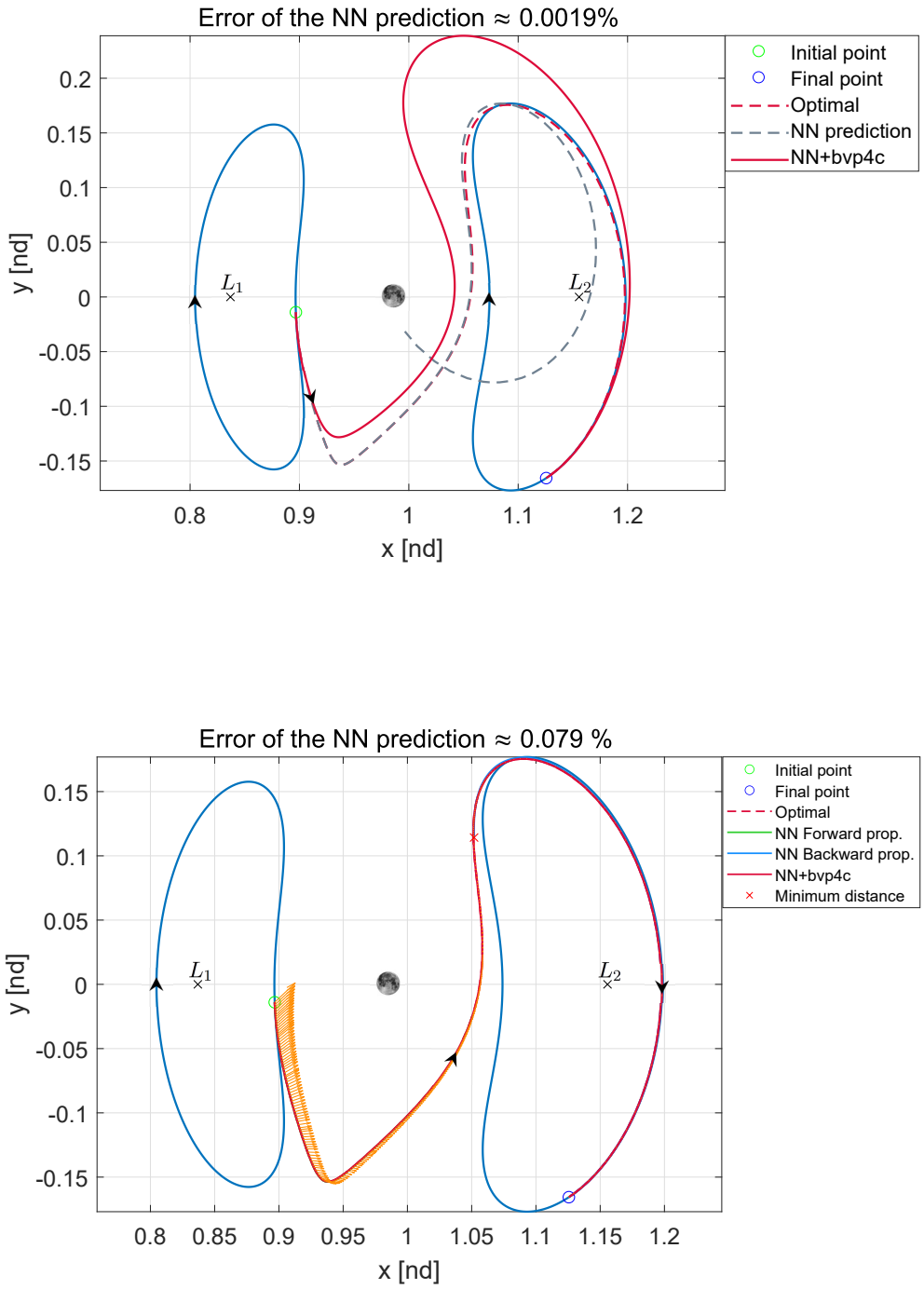


Figure 4.22: 18 days transfer maneuver Forward propagation on top, Forward+Backward propagation on bottom.

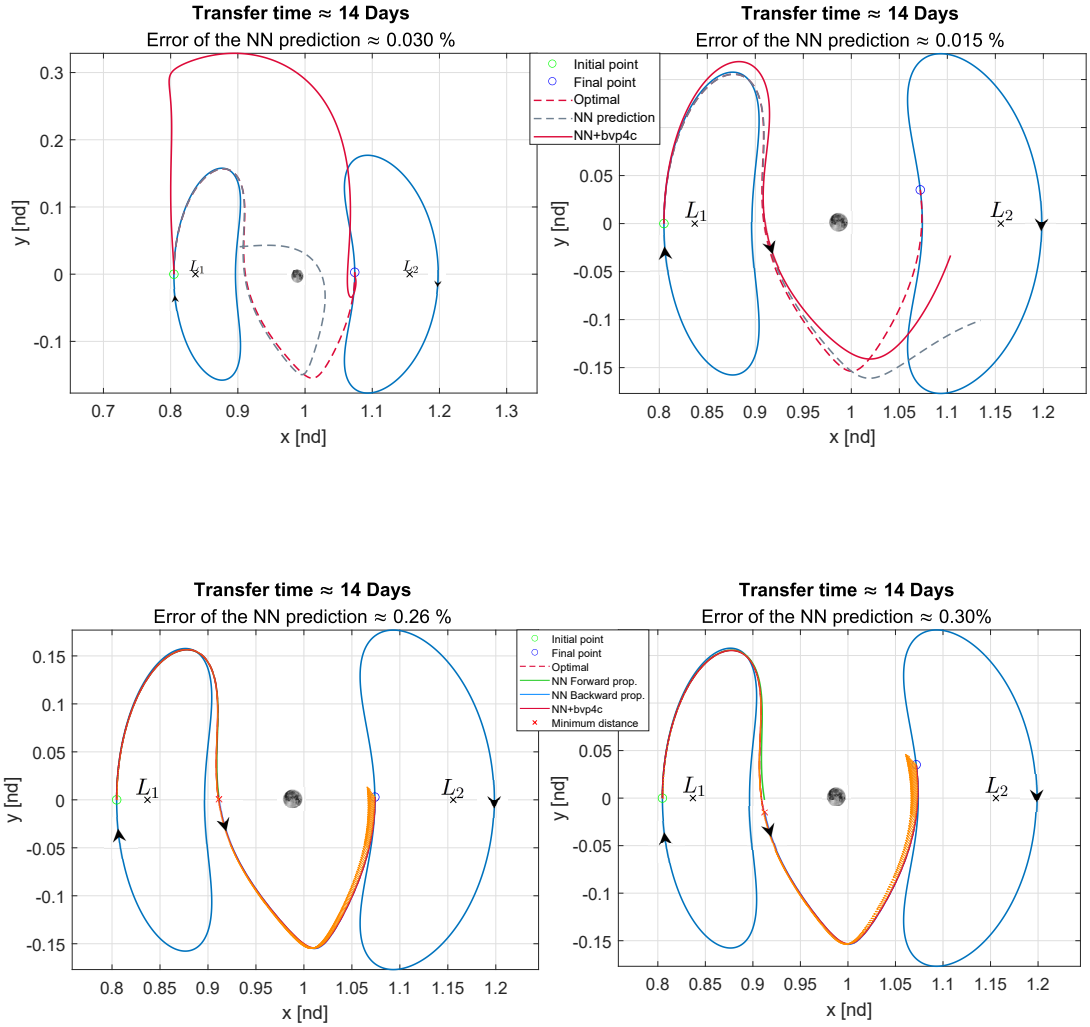


Figure 4.23: Examples of transfer maneuvers with Forward propagation on top, Forward+Backward propagation on bottom.

Some other examples of this methodology are illustrated in Fig.(4.23).
On top of the figure two different failed trajectories are reported. On the bottom both the corrected maneuver are illustrated.
Clearly the the error of the NN prediction is computed on $[\lambda_0, t_f]^T$ for the top cases, while for the bottom ones on $[\lambda_0, \lambda_f, t_f]^T$.
We can conclude that the result of this forward-backward approach are promising even though a better investigation is required. The main problem is the discontinuity that we have to face when we split the two propagations. If these latters gather a significant error the discontinuity could be too large and the solver may could fail to find an optimal solution. With this method our goal is to provide an estimated

solution of the trajectory to the solver, in order to lead it to convergence in shorter time.

Chapter 5

DRO - Halo

In this chapter several transfer maneuvers between a Distant Retrograde Orbit (DRO) and a Halo Orbit are investigated. The two orbits considered are taken from [45] and propagated for a period. In Fig.(5.1) both the orbits are illustrated. The main characteristics of the orbits considered are reported in Tab.(5.1).

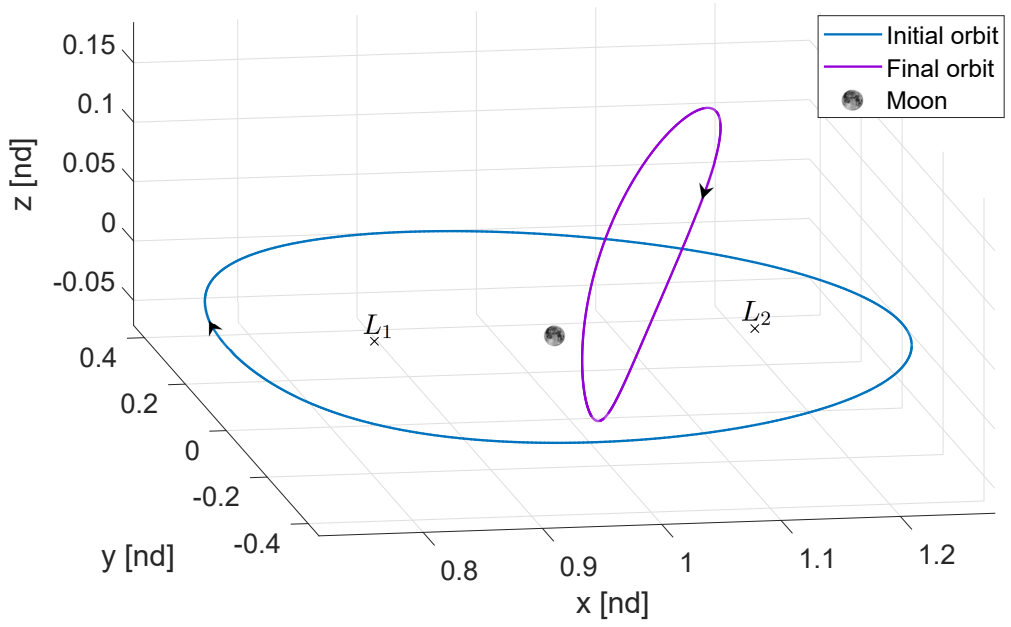


Figure 5.1: DRO and Halo considered

Orbit	Jacobi Constant (LU^2/TU^2)	Period (days)
DRO	2.875	21
HALO	3.035	13

Table 5.1: Characteristics of the DRO and Halo considered

The DRO lies in the plane that contains both the Lagrangian points and the Moon. On the other hand the Halo Orbit has an important inclination.
Recalling Fig.(1.10), we can also state that this particular Halo is almost polar, thus we can also refer to it as a Near Rectilinear Halo Orbit (NRHO). The main reason to study a maneuver between this orbit is to test the feasibility of our methodology in a more complex transfer geometry, which is not more planar but tridimensional. In Fig.(5.5) an optimal transfer maneuver between the orbits is illustrated. Once again we consider only Inner orbits, that have a lower transfer time. For this reason and for the remarkable plane changed required, the thrusters are ON for almost the entire maneuver, there is a lack of coasting phases.

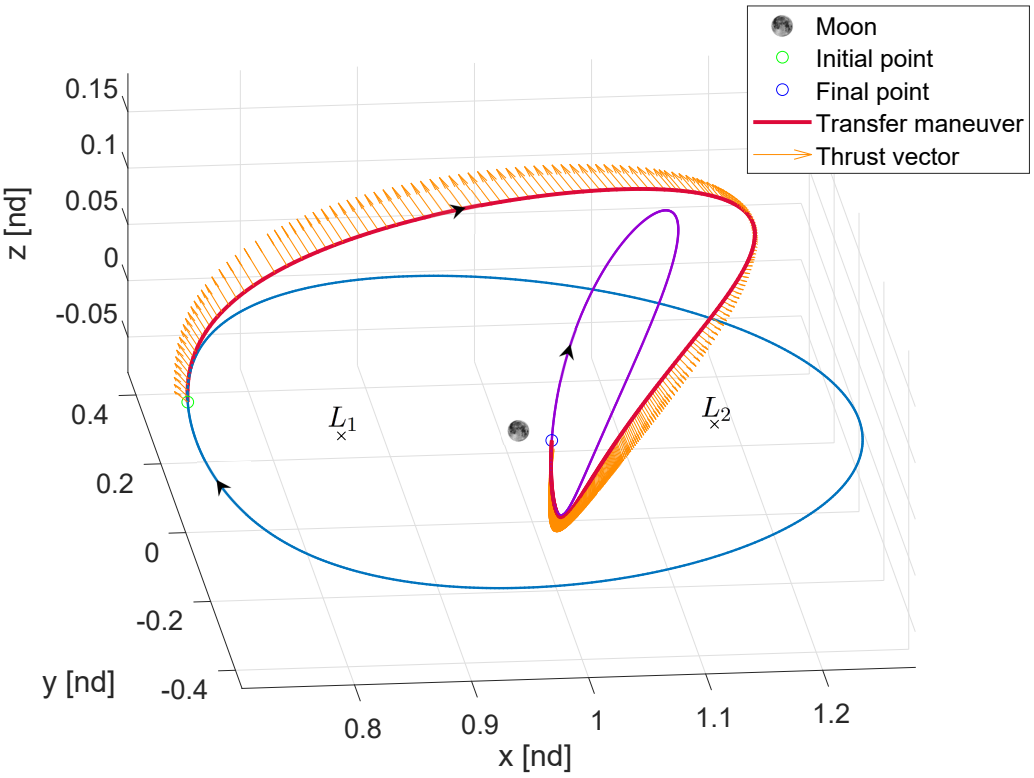


Figure 5.2: Example of a 19 day transfer maneuver between the orbits

5.1 Free time formulation

In this chapter we are going to consider just the time free formulation that is more plausible during a mission plan. Multiple time free minimum energy problems are solved using the same formulation of Sec.(4.2).

5.1.1 Data collection

For the data collection, initial and final states are once again fixed. The initial DRO orbit is discretized in 50 points while the final Halo in every single point obtained from the propagation at [45]. Thus, we have 30000 transfer trajectories to train the net. In Fig.(5.3) 200 of the 30000 transfer maneuvers are depicted.

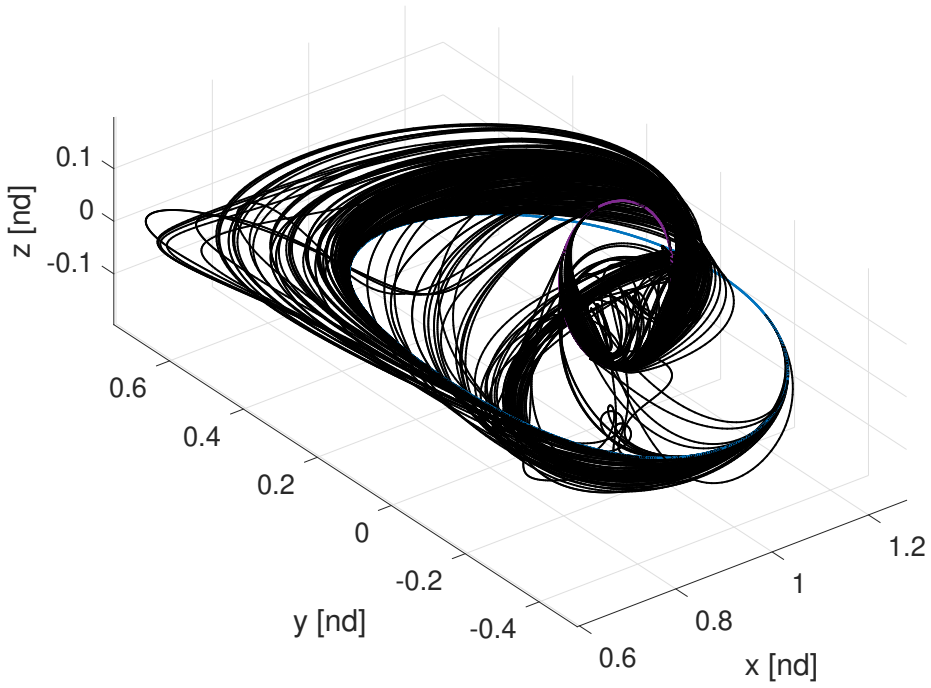


Figure 5.3: 200 of the 30000 orbits collected in the free time formulation for DRO-Halo

The pattern is clearly disorganized in comparison with the one followed for the Lyapunovs. This is because the continuation method to expedite data collection, that is the same of Alg.(2), sometimes struggles to find optimal solutions.
In these latter cases the algorithm is restarted and several iterations with randomized initial guesses are performed. Then, when an optimal solution that obeys constraints

is found, the continuation method is executed starting from this latter solution.

5.1.2 Neural network performance

A neural network is built in Matlab, the input of the net is a I_{12x1} (i.e. initial and final states) while the output is a O_{7x1} (i.e. initial costates and transfer time). Thus, we do not take into account the forward backward propagation at the moment. The network's structure can be summarized as follows:

- **Architecture**: 3 hidden layers with 17 neurons each;
- **Activation functions**: tanh for the hidden layers, linear for the output layer;
- **Optimizer**: Levenberg-Marquardt;
- **Batch division**: 70% training, 15% validation, 15% test;
- **Results**: MSE $\sim$ 9.34e-04; 97% of the test data has a relative error $\leq 5\%$
- **Training time**: 45 min.

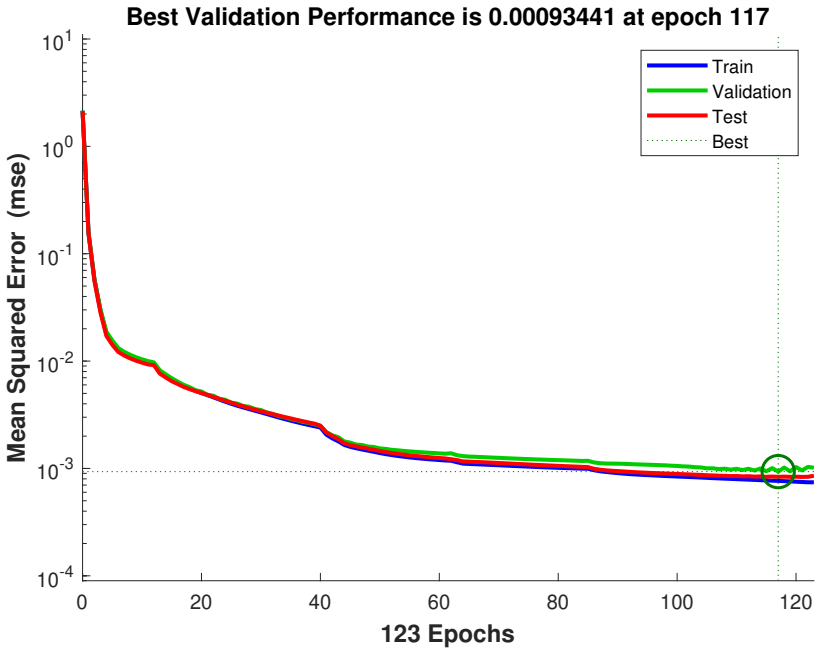


Figure 5.4: NN trained on 30000 transfers beetween a DRO and a Halo

As we can see from the Fig.(5.4) the learning process is stopped as the validation data starts to overfit. This latter behaviour can be recognized from the wavy trend in the final epochs.

5.1.3 Results

Once the neural network is trained we can test its performance simulating the possible on board application as:

1. **Requirement**: The spacecraft must perform a transfer between an initial point on the DRO to a final point on the Halo without any particular constraint on the transfer time, but minimizing the control energy, thus we know $[x_0, x_f]^T$;
2. **NN recall**: the net will predict the vector $[\lambda_0, t_f]^T$ with an error depending on the NN performance obtained on ground;
3. **NN prediction propagation**: in order to re-build the optimal control history and the transfer trajectory we can leverage the dynamics equations (2.9) and propagate them forward;
4. **Recall the TPBVP solver**: the on board solver can be run leveraging as initial guesses the propagation of net's prediction.

In Fig.(5.5) a 13 days transfer between the orbits is illustrated.
Propagating the NN prediction that has an error of 0.5%, the optimal trajectory is retraced for almost the entire transfer, even though the final point is not matched. By the way this prediction is a perfect initial guess for the TPBVP solver, in fact recalling bvp4c leads to convergence to the optimal solution and to the final state matching. In Fig.(5.6) a longer transfer maneuver of 18 days is reported. The results are once again promising. Although the propagation of the prediction diverges from the optimal path and, in addition, the maneuver passes in the vicinity of the Moon that is a critical zone with high non linearity of the dynamics, the solver converges to the optimal solution.
In Fig.(5.7) and Fig.(5.8) two other maneuvers are presented. Please note that in all these plots the thrust vector is not illustrated just for a better clarity of the figures. Regarding the controls plots, in all the four cases presented the control history is reproduced correctly from the solver and it follows the optimal solution computed on ground. A careful reader may have noted at this point the main difference in comparison with the previous time free Lypaunovs maneuvers. As already anticipated the thrusters are ON all along the maneuver, there is a lack of coasting phases and the spacecraft does not leverage natural path. Furthermore from the classical Two-Body Problem theory we know that a plane change is one of the most expensive maneuver to accomplish. Even though we are dealing with 3BP dynamics this latter consideration remains valid since the maneuver is performed in the vicinity of the Moon. Thus, it is clear that if we keep out Outer orbits from the data collection, we are going to achieve expensive maneuvers to fulfil the constraints.

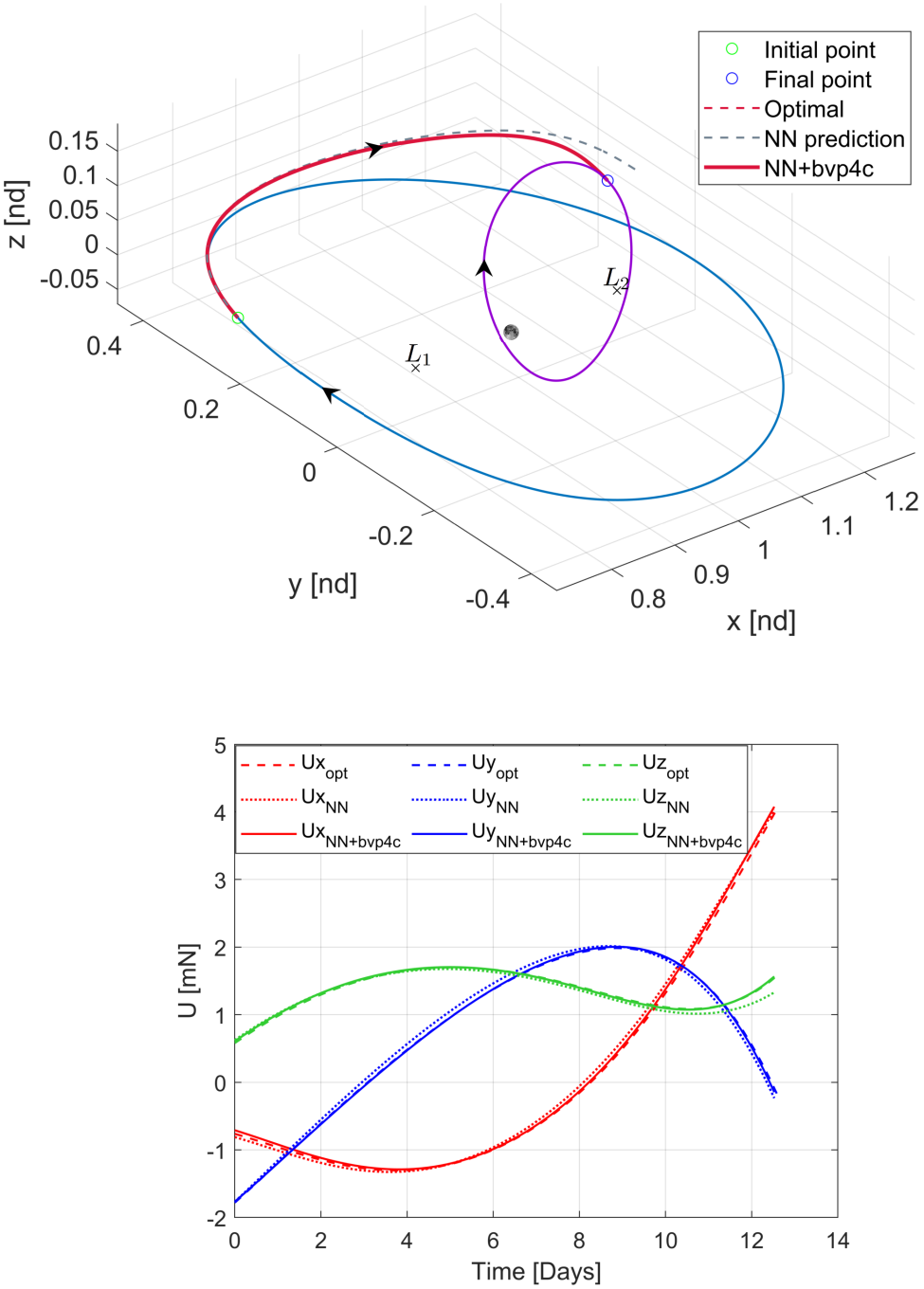


Figure 5.5: 13 days transfer maneuver and control, propagation performed with an error of 0.5% on the NN prediction

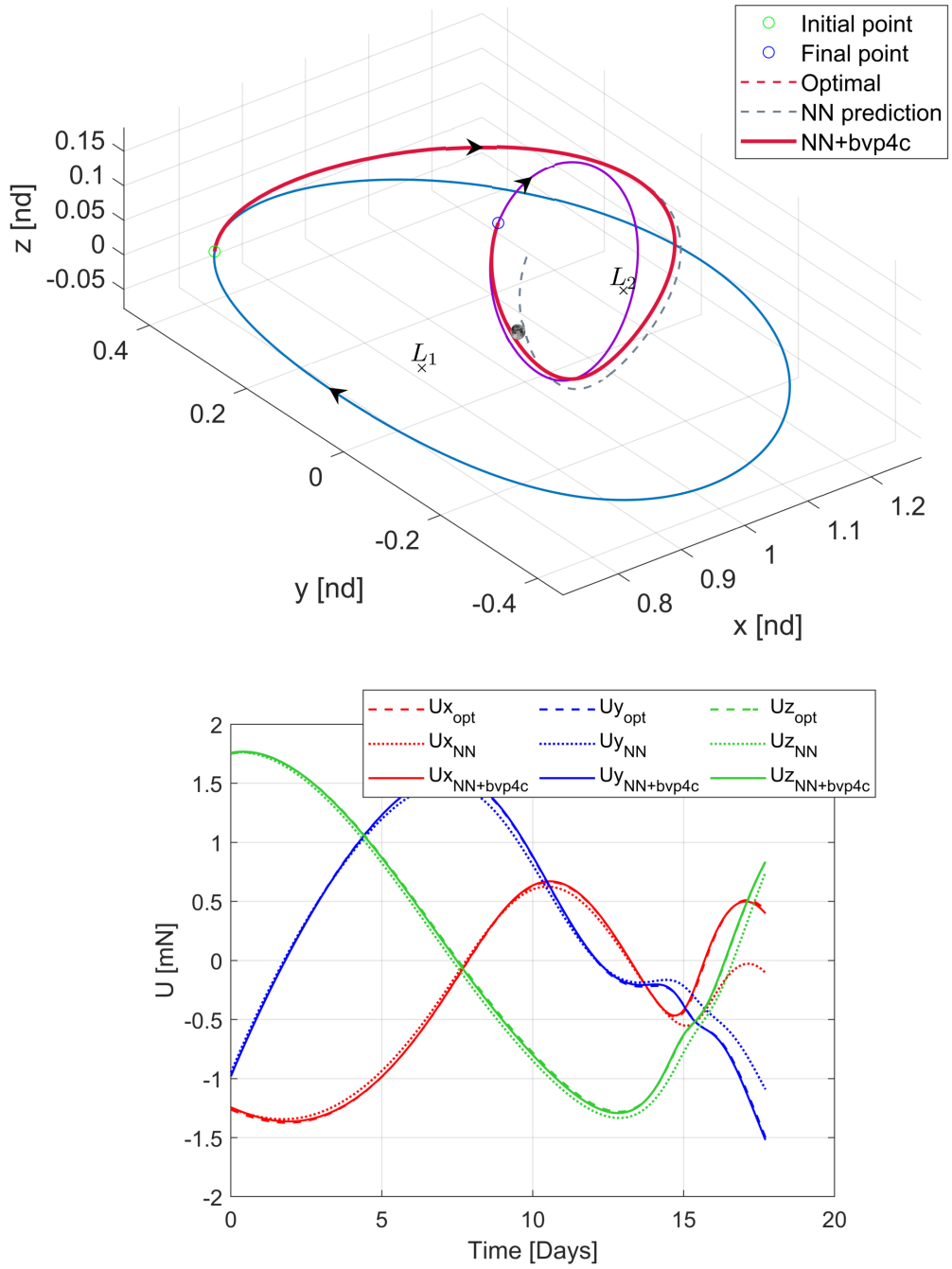


Figure 5.6: 18 days transfer maneuver and control, propagation performed with an error of 0.2% on the NN prediction

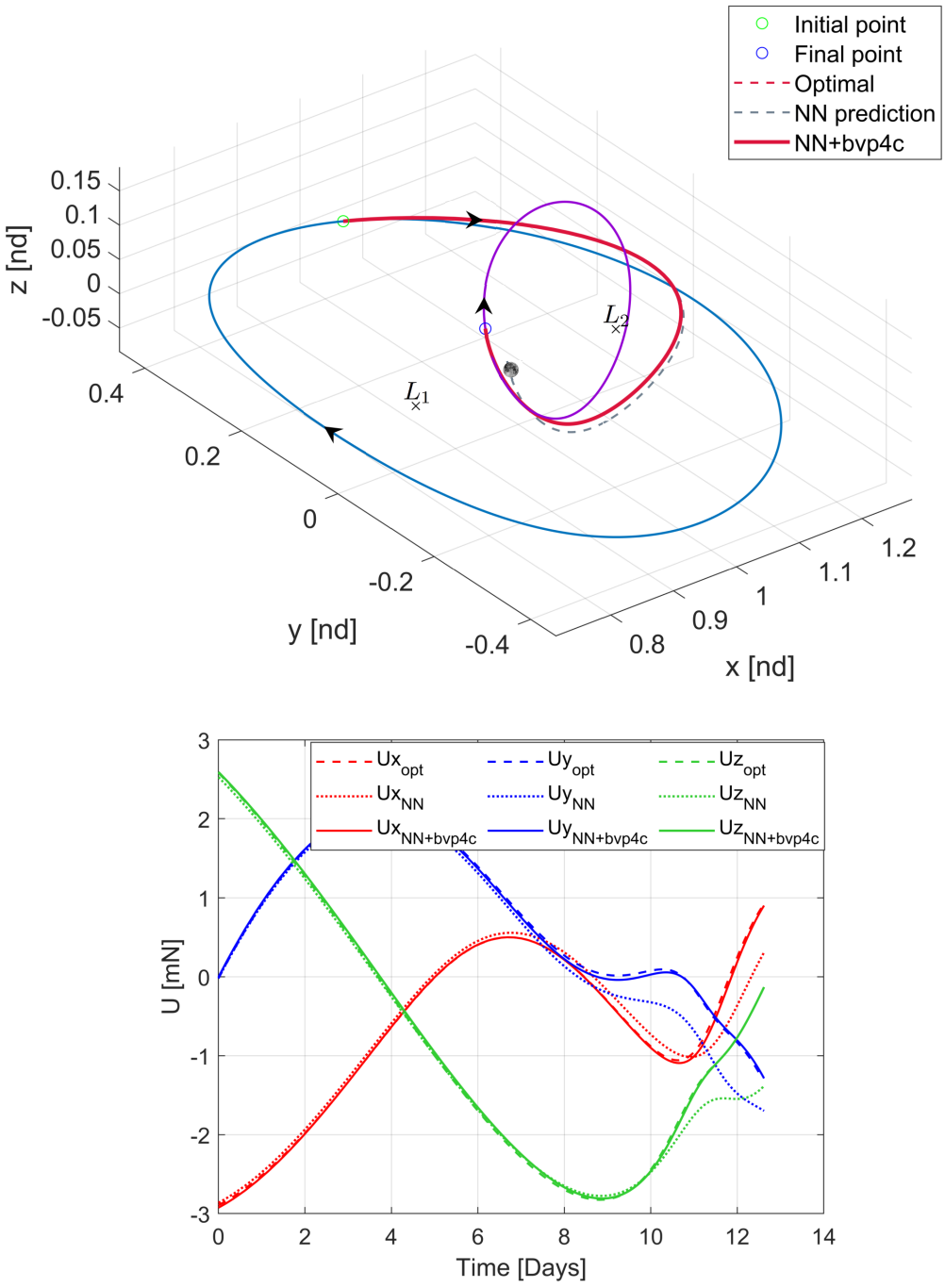


Figure 5.7: 12.6 days transfer maneuver and control, propagation performed with an error of 0.4% on the NN prediction

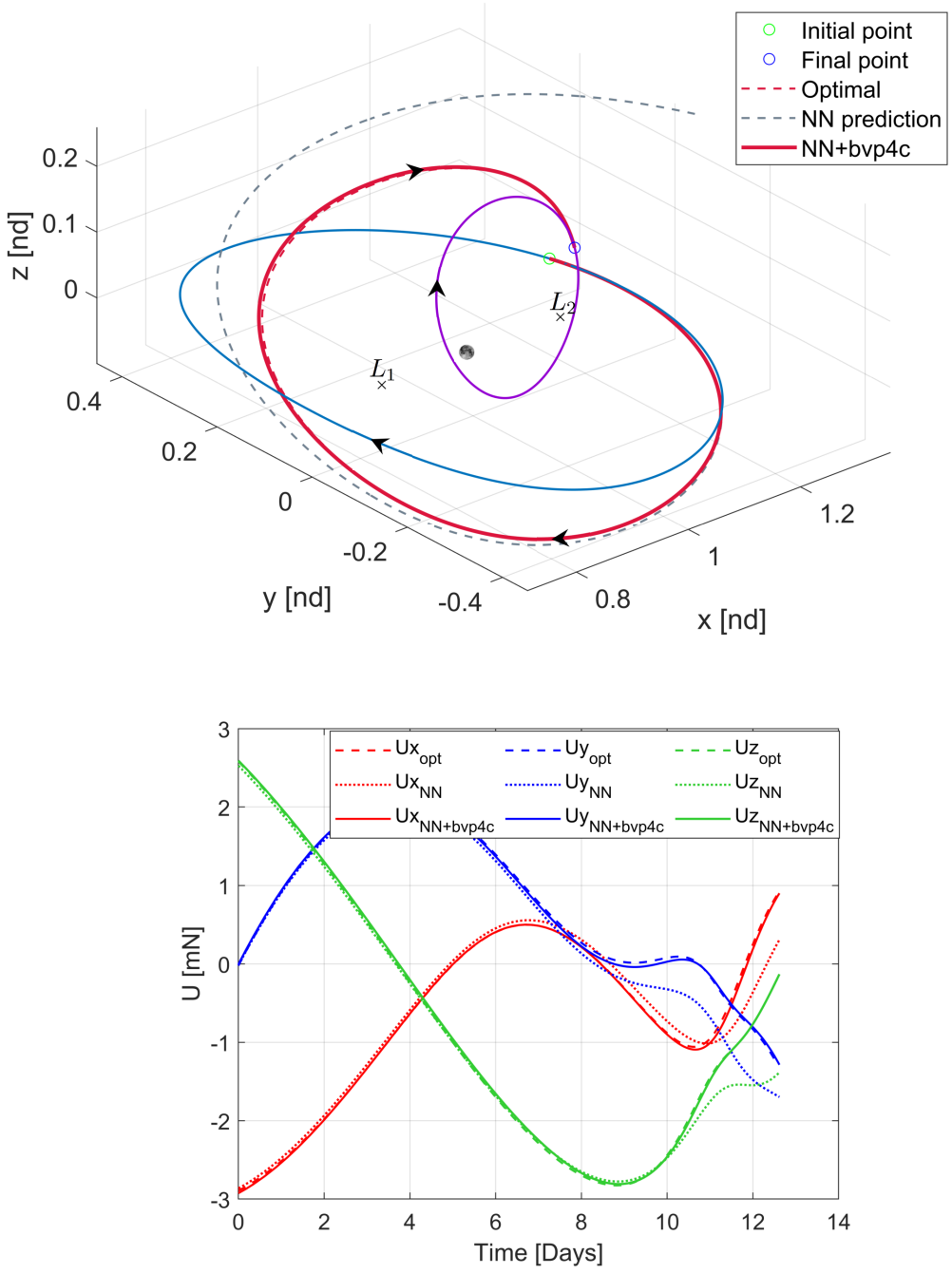


Figure 5.8: 24 days transfer maneuver and control, propagation performed with an error of 0.8% on the NN prediction

5.2 Forward-Backward propagation

As in the previous chapter, certainly in some transfers the error propagation on the NN prediction is remarkable and the TPBVP solver struggles to follow the optimal trajectory. This is the case of Fig.(5.10). As for the Lyapunovs, we tried to implement the forward backward propagation as a possible solutions to improve convergence in such cases. Thus, in this section the neural network's performance and the results are presented.

5.2.1 Neural network performance

A neural network is trained in the same Matlab's environment. The input of the net is a I_{12x1} (i.e. initial and final states) while the output is a O_{13x1} (i.e. initial costates, final costates and transfer time).
The network's structure can be summarized as follows:

- **Architecture**: 3 hidden layers with 15 neurons each;
- **Activation functions**: tanh for the hidden layers, linear for the output layer;
- **Optimizer**: Levenberg-Marquardt;
- **Batch division**: 70% training, 15% validation, 15% test;
- **Results**: MSE $\sim$ 8.820e-04; 95% of the test data has a relative error $\leq 5\%$
- **Training time**: 1h.

Since the net has to map a more difficult relationship the training time is longer. In Fig.(5.9) the performance of the net in terms of mse are reported. As in the previous NN the most important aspect in this graph is the trend between train, validation and test curves. In this case the curves overlay each others, suggesting that the net is validated and can also generalize new data.
All of three trends reach an horizontal asymptote at 8.820e-04. From the author experience, if we seek for better performance of the net we should consider to re-collect new data rather than adjusting the net's parameters and architecture. We can push forward with changing the net's architecture, number of neurons and layers, activation functions and optimizer but at the end we can not improve the net's capabilities significantly. One of the most easiest yet most important concept that this thesis wants to point out is indeed the high correlation between net's performance and the methodology that we use to gather trajectories.

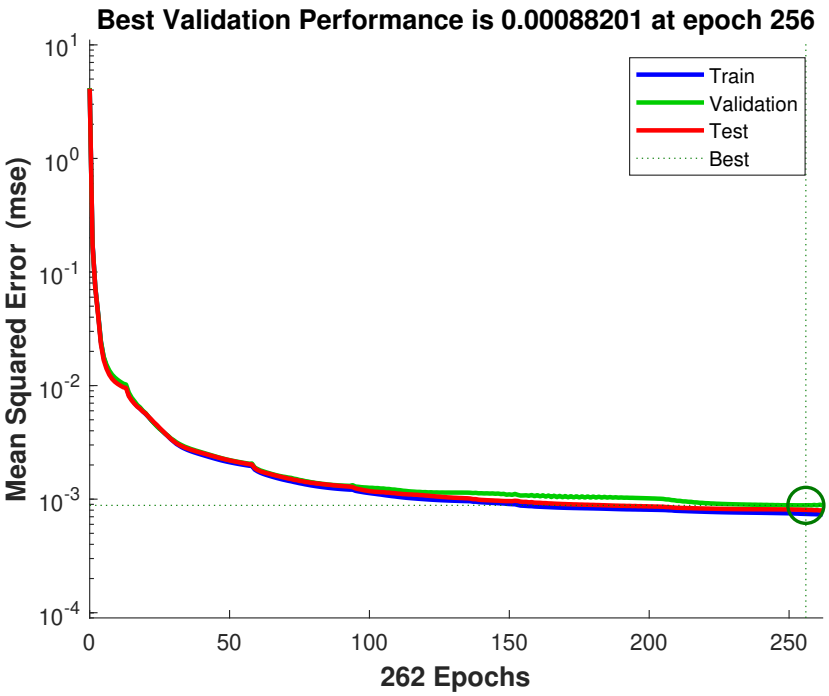


Figure 5.9: NN trained on 30000 transfers beetween a DRO and a Halo

5.2.2 Results

Lets consider now Fig.(5.10) that is obtained from the classical forward propagation. The NN prediction (i.e. initial costates and transfer time) that is illustrated as a grey dashed line, overlays the optimal path (i.e. red dashed line) at the beginning of the transfer and then diverges considerably. The error of the net's prediction is 0.7%, thus it is comparable to the cases of the previous sections, but the error propagation and the sensitivity of that particular TPBVP make the solver to diverge from optimal solution even though the final state is reached.
In Fig.(5.11) the same maneuver obtained through the forward-backward methodology is illustrated. Both the propagations can be noted in the image. The elimination of the discontinuity is not investigated in this thesis since the solver can manage to converge in any case and also because this methodology is presented just as possible solution and should be investigated in future works.
Anyway the results from this method are encouraging since the solver converges to the optimal solution. This can be noted both on the trajectory geometry and in the controls plot where optimal trend is reproduced with a small error.

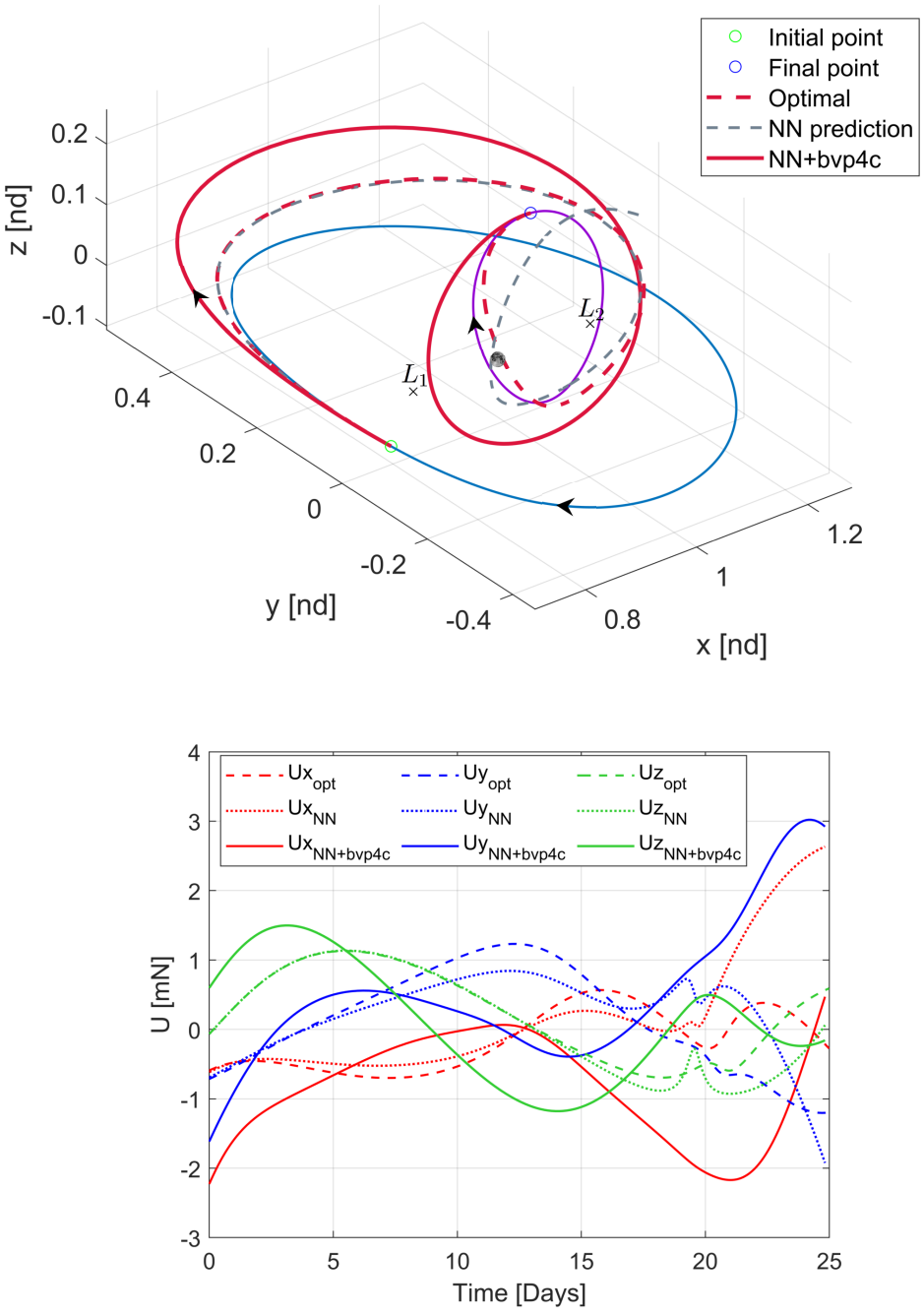


Figure 5.10: 25 days transfer maneuver and control, propagation performed with an error of 0.7% on the NN prediction

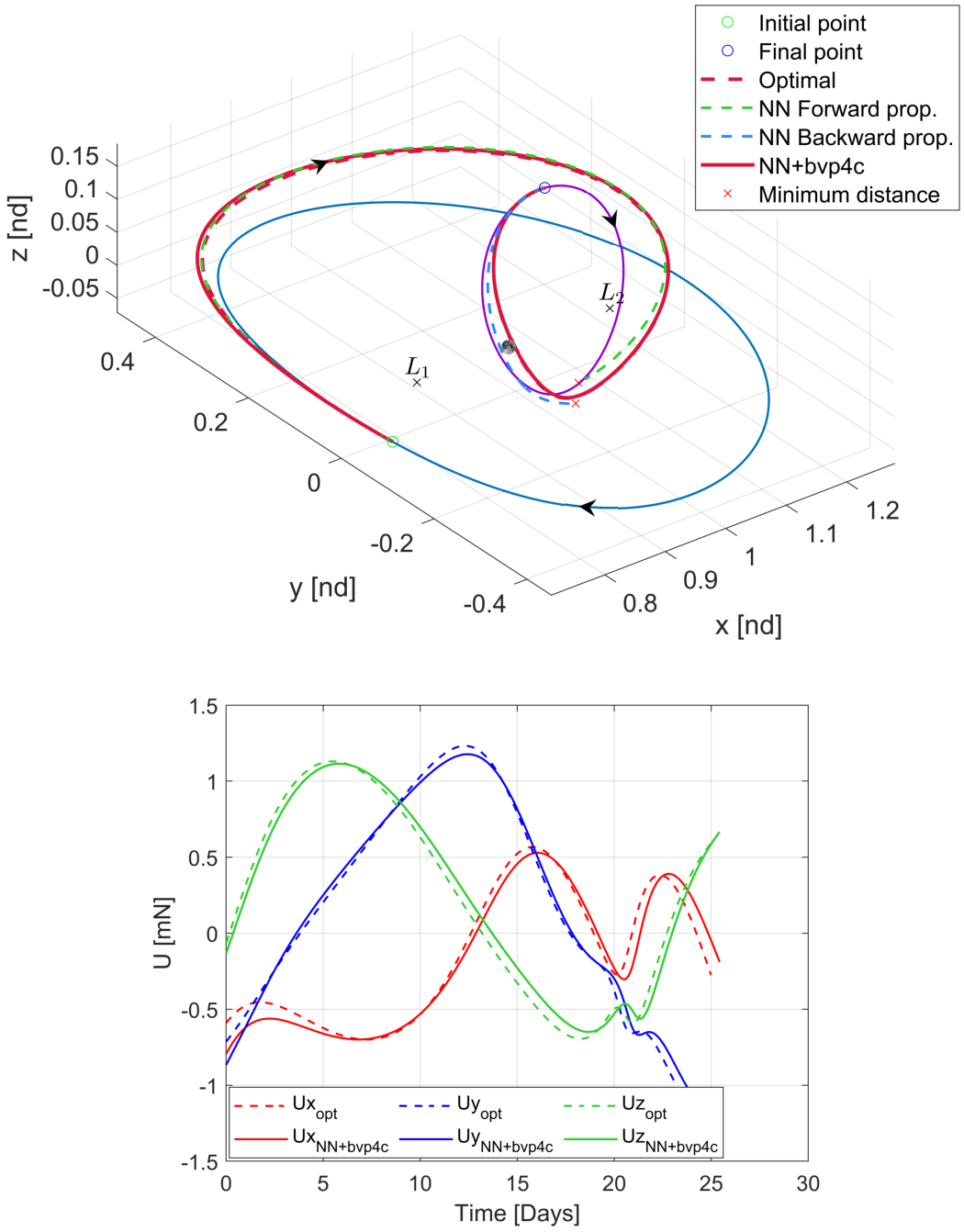


Figure 5.11: 25 days transfer maneuver and control, propagation performed with an error of 1.7% on the NN prediction

Conclusions

Trajectory optimization in Three Body dynamics is highly challenging. Since the neural networks performance depend upon the data collection, the know-how of the engineer is an important skill to encourage the learning phase of the net.
We can divide the results between the ability to gather optimal transfer solution and the performance of the net itself, that are necessarily related.
From the data collection point of view, the method of random initial guesses, to find the first optimal maneuver, is the most common in literature. In addition we must accept the inability to find a global minimum inside this dynamics regime that exhibits high sensitivity to initial conditions and possible chaotic behaviours. Thus, we can just check if the solution represents a feasible local minimum, slightly perturbating the initial guesses that led to convergence.
Moreover, the continuation method implemented to expedite the data collection perfectly suits the Lyapunov L_1 - Lyapunov L_2 transfer. This is due to the planar geometry of the transfer and also to the presence of natural path between the orbits. On the other hand the method struggles in the DRO-Halo transfer which is threedimensional and more challenging. In this latter case, when the continuation method fails, we restart the algorithm with several iterations to find the minimum energy trajectory with random initial guesses. Of course in future works the collection process of this latter case could be further investigated.
Concerning the neural networks performance promising results are achieved for its prediction. In this thesis we did not take into account time series NN because we wanted to enhance the simplicity of the feedforward nets . For this reason, from the initial costates prediction, the optimal maneuver is re-build with the propagations of the dynamics. This approach certainly leads to error propagation even when the error on the net's prediction is apparently negligible. Thus, the TPBVP solver is recalled on board giving as initial guess the entire solution obtained from the propagation of net's prediction. Finally, the neural networks onboard role is to provide a good initial solution to the TPBVP solver in order to decrease the computational time and effort. However, when the error propagation is remarkable this work logic fails. From this came the need to find another possible solution, the forward backward propagation. Although this method has encouraging results,

future works should validate this approach considering also different maneuvers. Finally, as last step the thrusters ON/OFF dynamics should be implemented and a minimum fuel formulation can be studied, using minimum energy solutions as initial guesses.

Bibliography

[1] Lorenzo Federici, Andrea Scorsoglio, Alessandro Zavoli, and Roberto Furfaro. «Autonomous Guidance for Cislunar Orbit Transfers via Reinforcement Learning». In: Aug. 2021 (cit. on p. 1).

[2] Daniel Miller and Richard Linares. «LOW-THRUST OPTIMAL CONTROL VIA REINFORCEMENT LEARNING». In: Jan. 2019 (cit. on p. 1).

[3] Anna-Lisa Paul, Stephen M Elardo, and Robert Ferl. «Plants grown in Apollo lunar regolith present stress-associated transcriptomes that inform prospects for lunar exploration». In: *Communications biology* 5.1 (2022), pp. 1–9 (cit. on p. 6).

[4] Theresa Hitchens. *US Needs New Policies With Move To Cislunar: Aerospace Corp., June, 2020* (cit. on p. 6).

[5] MJ Holzinger, CC Chow, and P Garretson. *A Primer on Cislunar Space*. Air Force Research Laboratory, 2021 (cit. on pp. 6, 7).

[6] RAN Araujo, OC Winter, AFBA Prado, and R Vieira Martins. «Sphere of influence and gravitational capture radius: a dynamical approach». In: *Monthly Notices of the Royal Astronomical Society* 391.2 (2008), pp. 675–684 (cit. on p. 6).

[7] Carolin Frueh, Kathleen Howell, Kyle J DeMars, and Surabhi Bhadauria. «Cislunar Space Situational Awareness». In: *31st AIAA/AAS Space Flight Mechanics Meeting*. 2021 (cit. on pp. 6, 7).

[8] June Barrow-Green. *Poincaré and the three body problem*. 11. American Mathematical Soc., 1997 (cit. on p. 8).

[9] RICHARD Frnka. *The Circular Restricted Three-Body Problem*. 2010 (cit. on p. 8).

[10] V Szebehely. «Theory of Orbits, The Restricted Problem of Three Bodies, Acad». In: *Press, New York* (1967) (cit. on p. 8).

[11] Bilel Daoud. «On the optimal control of the circular restricted three body problem». PhD thesis. Université de Bourgogne, 2011 (cit. on p. 8).

[12] Jeannette Heiligers and Merel Vergaaij. «Time-Optimal Solar Sail Heteroclinic Connections for an Earth-Mars Cycler». In: Oct. 2017 (cit. on p. 10).

[13] Howard Curtis. *Orbital Mechanics for Engineering Students*. Elsevier Butterworth Heinemann, 2005 (cit. on pp. 11, 13, 16).

[14] Gustavo Gargioni, David Alexandre, Marco Peterson, and Kevin Schroeder. «Multiple Asteroid Retrieval Mission from Lunar Orbital Platform-Gateway Using Reusable Spacecrafts». In: Mar. 2019, pp. 1–13 (cit. on p. 12).

[15] CJ Neil. «The Lagrange Points». In: *Cornish Document Created for WMAP Education and Outreach* (1998) (cit. on p. 13).

[16] National Aeronautics and Space Administration. *What is a Lagrange Point?* 2020. URL: https://solarsystem.nasa.gov/resources/754/what-is-a-lagrange-point/ (cit. on p. 13).

[17] Maaninee Gupta, Kathleen C Howell, and Carolin Frueh. «Earth-Moon Multi-Body Orbit to Facilitate Cislunar Surveillance Activities». In: *AIAA/AAS Astrodynamics Specialist Conference*. 2021 (cit. on pp. 17, 18).

[18] Mar Vaquero and Kathleen C Howell. «Leveraging resonant orbit manifolds to design transfers between libration point orbits in multi-body regimes». In: *23rd AAS/AIAA Space Flight Mechanics Meeting, AAS*. Vol. 13334. 2013 (cit. on p. 18).

[19] Emily M Zimovan-Spreen and Kathleen C Howell. «Dynamical structures nearby NRHOS with applications in cislunar space». In: *AAS/AIAA Astrodynamics Specialist Conference, Portland, Maine*. 2019 (cit. on p. 18).

[20] Andrew Jones. *A Chinese spacecraft is testing out a new orbit around the moon*. 2022. URL: https://spacenews.com/a-chinese-spacecraft-is-testing-out-a-new-orbit-around-the-moon/ (cit. on p. 19).

[21] Christopher Adamek. *Gateway system requirements*. Tech. rep. 2019 (cit. on p. 19).

[22] Robert W Farquhar. *Station-keeping in the vicinity of collinear libration points with an application to a lunar communications problem*. Tech. rep. 1966 (cit. on p. 20).

[23] Kathleen Connor Howell. «Three-dimensional, periodic,'halo'orbits». In: *Celestial mechanics* 32.1 (1984), pp. 53–71 (cit. on p. 20).

[24] Christina Zaid. *A Lunar Orbit That's Just Right for the International Gateway*. May 2022. URL: https://www.nasa.gov/feature/a-lunar-orbit-that-s-just-right-for-the-international-gateway (cit. on p. 21).

[25] Joey Roulette. *After Million-Mile Journey, James Webb Telescope Reaches Destination - The telescope's safe arrival is a relief to scientists who plan to spend the next 10 or more years using it to study ancient galaxies*. 2022. URL: https://www.nasa.gov/feature/a-lunar-orbit-that-s-just-right-for-the-international-gateway (cit. on p. 21).

[26] John T Betts. «Survey of numerical methods for trajectory optimization». In: *Journal of guidance, control, and dynamics* 21.2 (1998), pp. 193–207 (cit. on p. 23).

[27] Abolfazl Shirazi, Josu Ceberio, and Jose A Lozano. «Spacecraft trajectory optimization: A review of models, objectives, approaches and solutions». In: *Progress in Aerospace Sciences* 102 (2018), pp. 76–98 (cit. on pp. 23, 24).

[28] Ehsan Taheri, Ella M Atkins, and Ilya Kolmanovsky. «Performance comparison of smoothing functions for indirect optimization of minimum-fuel low-thrust trajectories». In: *2018 Space Flight Mechanics Meeting*. 2018, p. 0214 (cit. on p. 24).

[29] R.W.H. Sargent. «Optimal control». In: *Journal of Computational and Applied Mathematics* 124.1 (2000). Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations, pp. 361–371. ISSN: 0377-0427. DOI: https://doi.org/10.1016/S0377-0427(00)00418-0. URL: https://www.sciencedirect.com/science/article/pii/S0377042700004180 (cit. on p. 25).

[30] Anil V Rao. «A survey of numerical methods for optimal control». In: *Advances in the Astronautical Sciences* 135.1 (2009), pp. 497–528 (cit. on p. 25).

[31] Nathan Luis Olin Parrish. «Low thrust trajectory optimization in cislunar and translunar space». PhD thesis. University of Colorado at Boulder, 2018 (cit. on p. 25).

[32] URL: http://www.matthewpeterkelly.com/tutorials/trajectoryOptimization/canon.html (cit. on pp. 29–31).

[33] Uri M. Ascher and Linda R. Petzold. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. 1st. USA: Society for Industrial and Applied Mathematics, 1998. ISBN: 0898714125 (cit. on p. 31).

[34] Liaquat Lund Baloch, Omar Zurni, Ilyas Khan, Dumitru Baleanu, and Kottakkaran Nisar. «Convective Effect on Magnetohydrodynamic (MHD) Stagnation Point Flow of Casson Fluid over a Vertical Exponentially Stretching/Shrinking Surface: Triple Solutions». In: *Symmetry* 12 (July 2020). DOI: 10.3390/sym12081238 (cit. on p. 33).

[35] Larry Hardesty. «Explained: neural networks». In: *MIT News* 14 (2017) (cit. on p. 35).

[36] Varun Kumar Ojha, Ajith Abraham, and Václav Snášel. «Metaheuristic design of feedforward neural networks: A review of two decades of research». In: *Engineering Applications of Artificial Intelligence* 60 (2017), pp. 97–116 (cit. on pp. 35, 36).

[37] URL: https://towardsdatascience.com/why-neural-nets-can-approximate-any-function-a878768502f0 (cit. on p. 36).

[38] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. «Multilayer feedforward networks are universal approximators». In: *Neural networks* 2.5 (1989), pp. 359–366 (cit. on p. 35).

[39] URL: https://machinelearningmastery.com/difference-test-validation-datasets/ (cit. on p. 37).

[40] URL: https://www.v7labs.com (cit. on pp. 38, 39).

[41] URL: https://towardsdatascience.com/techniques-for-handling-underfitting-and-overfitting-in-machine-learning-348daa2380b9 (cit. on p. 40).

[42] URL: https://blogs.mathworks.com/iot/2018/07/31/create-and-train-a-feedforward-neural-network/ (cit. on p. 42).

[43] Balachandra Kumaraswamy. «6 - Neural networks for data classification». In: *Artificial Intelligence in Data Mining*. Ed. by D. Binu and B.R. Rajakumar. Academic Press, 2021, pp. 109–131. ISBN: 978-0-12-820601-0. DOI: https://doi.org/10.1016/B978-0-12-820601-0.00011-2. URL: https://www.sciencedirect.com/science/article/pii/B9780128206010000112 (cit. on p. 44).

[44] Peter Wilson and H. Alan Mantooth. «Chapter 10 - Model-Based Optimization Techniques». In: *Model-Based Engineering for Complex Electronic Systems*. Ed. by Peter Wilson and H. Alan Mantooth. Oxford: Newnes, 2013, pp. 347–367. ISBN: 978-0-12-385085-0. DOI: https://doi.org/10.1016/B978-0-12-385085-0.00010-5. URL: https://www.sciencedirect.com/science/article/pii/B9780123850850000105 (cit. on p. 44).

[45] URL: https://ssd.jpl.nasa.gov/tools/periodic_orbits.html (cit. on pp. 47, 59, 60, 77, 79).

[46] Jinsung Lee and Jaemyung Ahn. «Homotopic Approach of Free-Final-Time Continuous-Low-Thrust Trajectory Optimization». In: Aug. 2020 (cit. on p. 59).