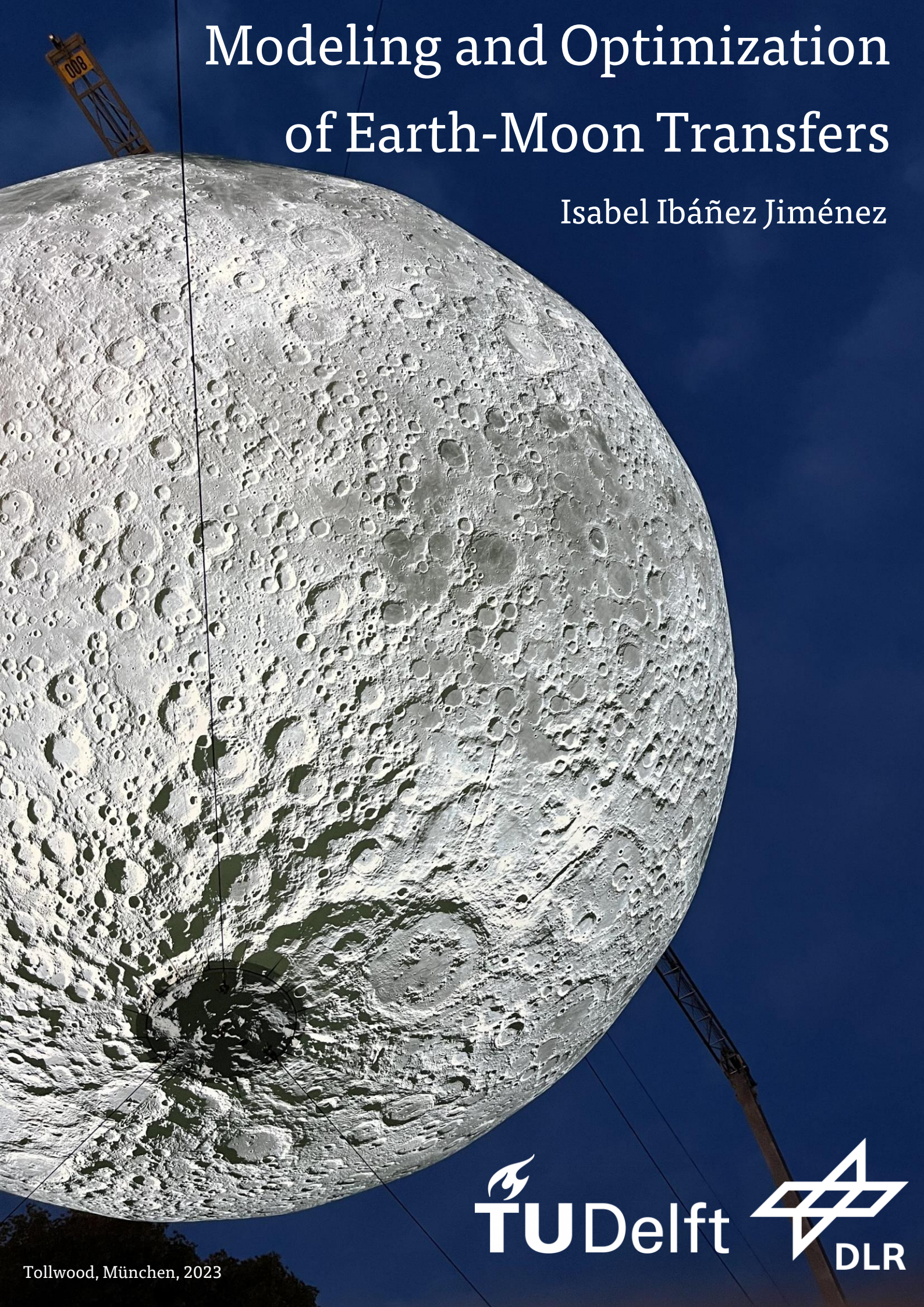
Modeling and Optimization of Earth-Moon Transfers

Isabel Ibáñez Jiménez



TU Delft

DLR

Tollwood, München, 2023

Modeling and Optimization of Earth-Moon Transfers

by

Isabel Ibanez Jimenez

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday September 25^{th}, 2023.

Student number:	5483093	
Project duration:	December 1^{st}, 2022 – August 10^{th}, 2023	
Thesis committee:	Ir. R. Noomen	TU Delft supervisor
	Dr.ir. E.J.O Schrama	Committee Chair
	Dr. I. Akay	Examiner
	Ir. M. Keller	DLR Examiner

TU Delft

Preface

This master thesis represents an academic year where research, growth and a new working experience have been be the primary objectives of my everyday life. The official starting date of this work is the 1^{st} of December 2022, however, due to administrative delays and the Christmas holidays, the actual work started the 9^{th} of January 2023, hence showing a 7-month long project full of enriching challenges.

It was during a 4-month internship in the DLR's (Deutsches Zentrum für Luft- und Raumfahrt) Flight Dynamics team, when I became aware of all the fascinating possibilities that astrodynamics could bring concerning space exploration. Moreover, I became part of a team whose main objective was to use all their knowledge, hard work and eagerness towards every space project that stood in front of them.

I decided that 4 months had not been enough, that I wanted to spend more time learning from the team and working in a project that could be part of a real-life mission one day. Therefore, when I got the opportunity to stay and work on lunar transfers for them, I could not refuse. The Moon, our only natural satellite, the most captivating body when observing it from Earth, and the door to the future of space exploration. What a great way to start a career by learning to optimize the best transfers towards it.

I want to thank my family, my parents, and my sister, who has always been my best role model, for supporting my journey in this amazing path around the world. I want to thank DLR for the trust placed on me, specially Sofya Spiridonova, my DLR mentor, for trusting and helping me through the unexpected challenges. Lastly, I would like to thank TU Delft, who was able to open unthinkable paths for us, who taught me more things in 7 months that I could have ever imagined, and who allowed me to work with the best mentor I could have asked for, Ron Noomen, always there to guide and motivate me in every small step. And specially, I need to thank TU Delft for allowing me to meet the people I admire the most academically and personally, my friends, always part of the Monte Carlo fan club.

Contents

List of Figures

List of Tables

Nomenclature

a	Semi-major axis	
ACC	Accuracy	
A_{CR}	Cross-sectional area for solar radiation pressure	
$\vec{a}_g$	Gravitational acceleration	m s^{-2}
$\vec{a}_{ng}$	Non-gravitational acceleration	m s^{-2}
Az_{dim}	Halo orbit vertical extension	km
C_D	Drag coefficient	
CMA-ES	Covariance Matrix Adaptation Evolution Strategy	
CPU	Computational	
C_R	Coefficient of reflectivity	
Deepest	Deep Space Orbit Determination and Parameter Estimation Tool	
DDPP	Directional derivative of the perturbation parameters	
DE	Differential Evolution	
DLR	Deutsches Zentrum für Luft- und Raumfahrt	
CR3BP	Circular Restricted Three-Body Problem	
E	Eccentric anomaly	rad
E, P_1	Earth	
e	Eccentricity	-
ECI	Earth Centered Inertial reference frame	
EME2000	Earth's Mean Equator and Equinox of year 2000	
ERP	Earth's Rotation Parameters	
$et0$	Initial epoch	
$f(\vec{x})$	Objective fitness function	
G	Gravitational constant	6.67 10^{-11} m^3 $kg^{-1}s^{-2}$
GEO	Geostationary orbit	
GSOC	German Space Operations Center	
GTO	Geostationary transfer orbit	
i	Inclination	-
JPL	Jet Propulsion Laboratory	
L2	Second Lagrange point	
LEO	Low Earth orbit	
LLO	Low lunar orbit	
LTO	Lunar transfer orbit	
M	Mean anomaly	rad
M, P_2	Moon	
m_1	Mass of the central gravitational body	
m_2	Mass of the secondary gravitational body	
m_P, m_3	Mass of the satellite	
N	Population number in PSO algorithm	
NASA	National Aeronautics and Space Administration	

NRHO	near-rectilinear halo orbits	
O	Center of the reference frame	
ODE	Ordinary Differential Equations	
OPT-EM	Earth-Moon transfers optimization software	
P, P_3	Satellite	
PSO	Particle Swarm Optimizer	
P_{SR}	Solar pressure	$4.560 \cdot 10^{-6}$ N/m^2
PQW	Perifocal Coordinate System	
PyGMO	Python Parallel Global Multi-objective Optimizer library	
QSW	"QSW" local orbital frame	
Rp_{dim}	NRHO periseline radius	
SC	Scaling factor for the constraints	
SCF	Scaling factor for the fitness (cost) function	
SEMpy	Sun-Earth-Moon system in Python	
SH	Spherical harmonics	
SLLSQP	Sequential Linear Least SQuares Programming	
SP	Scaling factor for the parameters of the decision vector	
SQL	Sequential Quadratic Programming	
SRP	Solar radiation pressure	
STM, $\Phi(t, t_0)$	State Transition Matrix	
t_i	Time in first orbit	s
T_i	Orbital period of orbit around body i	s
TA, θ	True anomaly	rad
TAI	International Atomic Time	
TBD	Barycentric Dynamical Time	
TT	Terrestrial Time	
ToF	Time of flight	
UTC	Coordinated Universal Time	
WSB	Weak Stability Boundary	

Δ	Allowed deviation for the constraints	
ΔV	Total velocity increment	km/s
$\Delta\vec{V}_1$		km/s
$\Delta\vec{V}_2$		km/s
η_1	In PSO, Social learning factor	
η_2	In PSO, Cognitive learning factor	
μ_i	Standard gravitational parameter of body i	km^3 s^{-2}
μ	Mass parameter of the CR3BP system	-
τ_1	Fraction of the orbital period of orbit around Earth	-
τ_2	Fraction of the orbital period of orbit around Moon	-
ω	In PSO tuning parameters, Inertia weight	
ω	in Keplerian orbital parameters, Argument of perigee	rad
Ω	Right Ascension of the Ascending Node	rad

Abstract

This thesis aims to calculate optimal trajectories from a user-defined Earth-bounded orbit to a user-defined Moon-bounded orbit using a bi-impulse direct transfer ultimately under the influence of a full dynamical model with perturbations, hence reflecting the actual physical environment.

Two tools are developed to achieve this goal. The first tool employs a global optimization algorithm, in particular a Particle Swarm Optimizer (PSO), to find an initial guess within a simplified dynamics model, exploring the user-defined search space. The second tool employs a gradient-based Sequential Linear Least SQuares Programming (SLLSQP) optimizer to refine the initial guess and include the relevant perturbations that act in real life. Additionally, the tools are supported by methods for evaluating the results, providing plotting and analysis tools to make the most out of the obtained solutions.

For the initial guess calculation, the dynamics model includes the point-mass gravity field of Earth and the Moon. The output provides the required ΔV for the transfer and the epochs at which each maneuver should be performed. The SLLSQP optimizer subsequently corrects the initial guess considering the user-specified perturbations, optimizing the time in the first orbit, the different components of both maneuvers, and the time of flight to reach the required orbit in an optimal way.

The capabilities of the tools are demonstrated through several test cases. The first test involves transferring from a circular low Earth orbit (LEO) to a circular near-polar low lunar orbit (LLO), resulting in a total ΔV of 4716.62 m/s. A second and a third test case involving transfers from a LEO or a geostationary transfer orbit (GTO) to an eccentric lunar orbit are also conducted, obtaining a ΔV of 3859.81 m/s when transferring from the LEO and of 1512.95 m/s when doing so from a GTO, corresponding to a decrease of around 60%. The solution obtained from the transfer from the GTO leads to a 4.5% improvement compared to preliminary results found in literature. The forth test comprises transfers from another circular LEO orbit to a high-altitude lunar polar orbit, requiring a ΔV of 3996.44 m/s, being 4.6% higher than the solution found in literature.

These test cases validate the functionality of the code and showcase its versatility in handling various scenarios. In conclusion, the developed tools provide efficient and robust solutions for optimizing direct transfers from Earth to the Moon under the influence of real-life perturbations.

Part I

Introduction to research problem

Chapter 1

Introduction

Optimizing Earth-Moon transfers including a complete dynamics model has been a subject of research since the beginning of the Moon race in 1959. However, it continues to be a topic of interest due to the ongoing discussion within the space sector surrounding the concept of an inhabited lunar orbital station as a first step toward a manned exploration of the Solar System. Moreover, the technological and theoretical advancements give many opportunities for new research projects. Throughout this chapter, the importance of the Moon as the target in past and current space missions will be addressed in Section 1.1 as a background to the problem. Moreover, the main research objectives for this specific project will be explained in Section 1.2.

For a better understanding of the reasons behind the research of this thesis, the general formulation of the research problem will be presented in Chapter 2, including the outline followed in this paper. Chapter 3, where some theoretical background used throughout the project is presented, will then be the end of this introductory Part I.

1.1 Research background

The Moon has been an object of study since scientists such as Thomas Harriot finished the first sketch of the Moon's surface with the help of a telescope in 1609, or since Galileo discovered that the Moon had mountains and valleys in that same year, as reported by Chapman (2009). It is therefore not surprising that the Moon was the first target of exploration during the space race. Biesbroek et al. (2000) reports that the Soviet satellite Luna-1 marked the beginning of the lunar exploration, when it flew past the Moon in February 1959, followed by Luna-2, which became the first spacecraft to impact another natural satellite. It was not until Luna-9 (1966) that a spacecraft was able to land softly on the lunar surface, and Luna-16 (1970) was the first robotic sample return mission, able to bring scientifically important samples from areas never visited by Apollo. From the USA side, Apollo-8 became the first manned mission to orbit another world in 1968. In 1969, Apollo-11 became the first manned lunar landing mission.

Even though the last human walked on the Moon in 1972, the scientific interest to return humans to the Moon has been increasing in recent years, becoming a scientific as well as strategic objective for many countries, as it can be used as a testing environment for new technologies, and as a launch platform to planets that are further away than Earth. This *second Moon race* includes programs such as Artemis, and in particular the Gateway station. From this point of view, many different missions will be required not only concerning the manned module that will take the crew to the Moon, but also missions focusing on the control, communications, refueling and many other objectives which include engineering and scientific satellites. Some of the new scientific objectives, as explained by Russell et al. (2011), are to increase our understanding of the complex interaction of the Moon's surface with its space environment, as in-situ measurements from the past are inadequate. In addition, Russell explains that lunar orbits are ideal to

observe and study the pristine solar wind or showers of particles that impact the Earth and interact with its magnetosphere, because it spends most of its time upstream of Earth's bow shock. The study of their spatio-temporal ambiguities, three-dimensional characteristics and dynamical evolution is required, as the solar wind buffeting the surface never stays constant.

The economical resources and technological capabilities have changed significantly since the first lunar race, increasing the complexity of the missions due to the need of reducing the costs and collecting a greater scientific return using less propellant, hence requiring to make the most out of all new available resources and recent research.

All of this links directly to the research objectives of this master thesis, which will be presented in the following section, and summarized as a research question to be answered.

As a further background to the project, it needs to be highlighted that this research will be conduced in the Deutsches Zentrum für Luft- und Raumfahrt (DLR), Germany's research centre for aeronautics and space. In particular, it will be done in the Flight Dynamics team inside the German Space Operations Center (GSOC), whose main activities include mission analysis, operations support, and software development.

1.2 Research objectives

The objective of this project is to develop a versatile tool capable of optimizing trajectories from an arbitrary Earth-bounded orbit to an arbitrary Moon-bounded orbit using a dynamics model that includes user-selected perturbations. The tool aims to provide an initial guess for the propellant required, maneuvers to be performed, and optimal transfer times while considering a dynamics model suitable for mission operations, representing what is called, the physical truth.

Given the diverse possibilities in addressing this problem, an extensive literature study was conducted to explore different approaches and identify an innovative and useful procedure to optimize Earth-Moon transfers, including the transfer possibilities and the propulsion system to be used.

Many studies existing in literature rely on models like the Circular Restricted Three-Body Problem (CR3BP), which offers interesting capabilities but lack the ability to include perturbations like Earth's spherical harmonics or the solar radiation pressure. Moreover, the research often focuses on specific trajectory solutions rather than providing a user-friendly tool to analyze and find optimal transfers based on user-defined initial and final orbits. For these reasons, the main research question addressed in this thesis is:

What is the most effective approach to find the optimal Earth-Moon direct two-impulse transfers, from a user-defined Earth-bounded orbit to a user-defined Moon-bounded orbit including a user-selected dynamics model?

To answer this question, several sub-questions are proposed:

1. Which parameters to optimize play a significant role in an auxiliary Earth-Moon transfer?
2. Is it feasible to directly use a gradient-based optimizer to optimize direct transfers to the Moon?
3. How can an accurate initial guess be obtained to enhance the optimization process?
4. If a global-search algorithm is necessary for obtaining a good initial guess, can a gradient-based optimizer effectively refine the solution?
5. Are there any advantages in calculating the initial guess using the CR3BP? Are there any limitations?
6. Is tuning the optimizers necessary? How can it be done effectively?

7. Can a gradient-based optimizer reach a solution with a full dynamics model, even if the initial guess was calculated without considering these perturbations?

By answering these questions, this thesis aims to provide valuable insight and a reliable tool for optimizing auxiliary Earth-Moon transfers. Chapter 2 will introduce the research problem, and Section 2.3 will present the thesis outline related to it.

Chapter 2

Research problem

This chapter will introduce the main problem definition and the decisions made, as well as the resources that will be used, in order to help the reader gain a clearer understanding of the problem formulation as well as the main path that will be followed during the research.

2.1 General formulation

The primary objective of this project, as mentioned earlier, is to optimize direct transfers from Earth to the Moon. Given the broad scope of this topic, an extensive literature review was necessary before getting into the research and methodology of the thesis. The review covered transfer possibilities, propulsion systems, and potential initial and final orbits. This section will provide a summary of the main options and decisions made, along with the reasons behind them. The final conclusions reached will be put together in Subsection 2.1.4.

2.1.1 Transfer possibilities

Direct transfers, found in Figure 2.3, are the quickest and simplest transfers to the Moon, and hence one of the most used transfers in past and current missions. They have been extensively studied, and first-order approximations are available. Their time of flight (ToF) is short, offering benefits when looking at radiation shielding and the time for mission control and communications. Moreover, the spacecraft stays within close proximity to either Earth or the Moon, hence reducing the effect of the Sun's gravity field, helpful for the modelling of first approximations, and becoming easier to handle in emergency situations. Nevertheless, its main drawback is the ΔV required to reach the Moon, which is larger than for other options such as low-energy transfers that take advantage of the gravity of other celestial bodies to increase the orbital energy of the satellite, as explained by Biesbroek et al. (2000). The ΔV relates directly to the total cost of the mission.

Other types of transfers also show some advantages. Bi-elliptic transfers, found in Figure 2.1, offer a decrease in ΔV, as the potential inclination change is performed further away from the main body, but it comes with longer transfer times.

Weak Stability Boundary (WSB) transfers, depicted in Figure 2.2, specially when using ballistic capture, have shown substantial improvements in terms of ΔV performance, as explained by Miller et al. (1991). In addition, low-energy transfers have access to a much broader range of lunar orbits for a particular arrival date than direct transfers. It has to be highlighted, as explained by Villac et al. (2003), that a ballistic transfer offers higher benefits when transferring to periodic three-body orbits rather than to a Keplerian lunar orbit, specially if the objective orbit is close to circular, as the extra ΔV required to decrease the

eccentricity will also need to be considered, and the propellant needed is considerably increased. Its ToF is also increased to a range from 3 to 5 months when transferring from a low earth orbit (LEO) to a low lunar orbit (LLO), increasing the costs of the overall mission due to its longer duration.

Manifold transfers have also been studied as an interesting low-energy option to reach the Moon; they exploit the dynamics of the CR3BP and allow for a decrease in ΔV, without, in general, a large increase in ToF, as is the case for other low-energy options. Its main advantage comes when transferring to a periodic or quasi-periodic orbit through their related stable manifolds. Their main disadvantage is that the CR3BP formulation is required, as manifolds rotate in an inertial frame.

Even though other types of low-energy transfers could help reduce the amount of propellant required to reach a target orbit around the Moon, they will not be the main focus of this research. Direct two-impulse transfers are greatly used by the industry for initial studies, and they are reliable. The main goal of this research will then be to optimize simpler direct two-impulsive transfers but using a more complicated dynamics model.

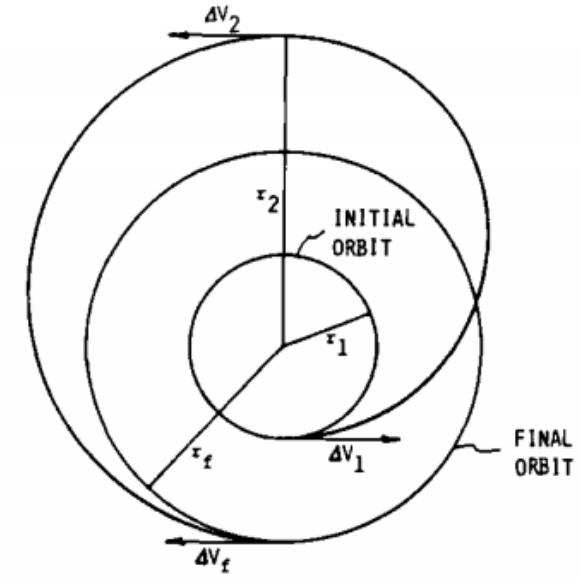


Figure 2.1: Geometry of a bi-elliptic transfer. Retrieved from Chobotov (2002).

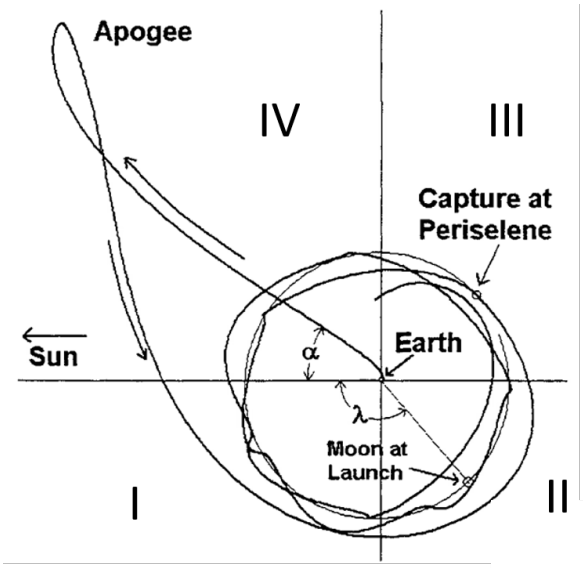


Figure 2.2: Quadrant II WSB transfer in Sun-Earth rotating frame. Modified from Belbruno et al. (2000).

2.1.2 Propulsion systems

This project will focus on high-thrust two-impulse transfers to the Moon, as spacecraft using solar electric propulsion require different trajectories consisting on a high number of spirals with increasing apogee obtained with continuous low-thrust arcs, until the satellite is at a sufficient distance to become captured by the Moon's gravity field. This will therefore require an optimal control problem solution instead of ΔV minimization, and a significantly greater transfer time.

2.1.3 Initial and final orbits

To optimize a trajectory, an initial state is required. The user may know the exact state consisting of position, velocity, and time, but in general, and as a tool to obtain a first approximation of the optimal trajectory, the user has requirements on the initial orbital parameters, but not on the exact state. Therefore, this tool will have as input the Keplerian elements of an auxiliary Earth-bounded orbit, which is the general way of defining an orbit. The exact position in that orbit where the satellite will start

is defined by an additional variable that needs to be analyzed. This decision leaves open the possibility of having other orbital parameters as free variables.

For the final state, the same conclusion can be reached. Nevertheless, two possibilities can be included as the final orbits, which are Keplerian orbits around the Moon and quasi-periodic orbits such as halo orbits or near-rectilinear halo orbits (NRHO).

Keplerian orbits can be modelled both in the CR3BP and in an ephemeris model (explained in Section 3.1). They are simple, and have been used in many missions, as they offer solutions for many scientific and engineering interests, such as for communication with the lunar south pole. However, they are not static in an inertial reference frame centered in Earth, which may complicate the calculation of the final injection.

On the other hand, periodic and quasi-periodic orbits offer the opportunity to exploit the dynamics of the CR3BP, and are static in this frame. They include aspects that can be beneficial for missions and operations to the Moon, which is the reason why they have been greatly studied and chosen as target orbits in recent years. Nevertheless, they have a big drawback, which is that they cannot be modelled directly, as is the case for Keplerian orbits. To use them as a final state, they need to be generated beforehand in a discretized manner, and the state for each individual point needs to be calculated and stored. In addition, they require the use of the CR3BP, as they cannot be generated in an inertial frame.

Keplerian orbits will be the main focus of study throughout this project due to the advantages stated above, and because they can be modelled in an inertial frame. Moreover, including quasi-periodic orbits induces the need of additional computations which may become very time consuming, and are not part of the main requirements of the software. Nevertheless, these orbits do offer advantages, and hence are an interesting topic for further study.

Since the trajectory for the optimization process starts from an initial orbit, it will not include the ΔV required for the launcher to reach that initial orbit, as it falls outside the scope of this project. Nonetheless, it is important to take this into account when evaluating the overall mission cost.

2.1.4 General problem formulation

The main conclusion reached for the general formulation of the problem, is that the aim will be to optimize bi-impulsive direct transfers from a Keplerian Earth orbit to a Keplerian Moon orbit.

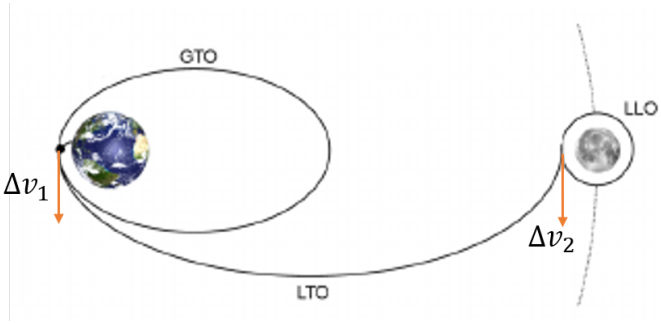


Figure 2.3: Bi-impulsive direct transfer diagram from a GTO to a LLO, including two impulsive maneuvers parallel to the velocity in the perigee and apogee of the LTO. Figure modified from Lange et al. (2008).

Throughout this project, the initial orbits around Earth will be referred to as E, the orbits around the Moon as M, and the lunar transfer orbit as LTO. The velocity difference from E to LTO will be denoted as $\Delta\vec{V}_1$, and the velocity difference from LTO to M as $\Delta\vec{V}_2$. The sum of the magnitudes of both velocities will be expressed as ΔV.

2.2 Optimization method

Once the general transfer problem formulation has been selected, the actual optimization method needs to be designed.

The main classification of the available optimizes is explained in detail in Section 3.2.2, where it is stated that there are two types of optimizers: gradient-based optimizers, such as the Sequential Linear Least SQuares Programming (SLLSQP), which have advantages but require a good initial guess, and global optimizers, such as Genetic Algorithms or the Particle Swarm Optimizer (PSO), which can handle a full design space and provide a solution to the complex Earth-Moon transfer problem. To exploit the benefits of both approaches, a hybrid method is chosen. Initially, a PSO is employed to perform a global search and find either the optimal solution or a set of local optima. Moreover, the global search optimizer will also help to understand the characteristics of the problem and the design space. Subsequently, a gradient-based optimizer is used to refine and fine-tune the solution.

However, using PSO comes with a major drawback - it requires numerous function evaluations, leading to significant computational expense. To calculate the ΔV of the required maneuvers efficiently, a combined method of Lambert solver and a multiple-shooting algorithm, as explained in Section 6.2, is employed. Including all perturbations in this part of the project would be excessively complicated and time-consuming.

Therefore, the SLLSQP optimizer plays a dual role. It will not only refine the solution, but will also correct for the additional perturbations that could not be included for the initial guess calculation. These processes are independent, except for the fact that the initial guess for SLLSQP is derived from the solution obtained with the more general search. In theory, this initial guess could be obtained through other means as well.

For both cases, the optimization will focus on a single objective which will be explained later on, rather than in conducting a multi-objective analysis.

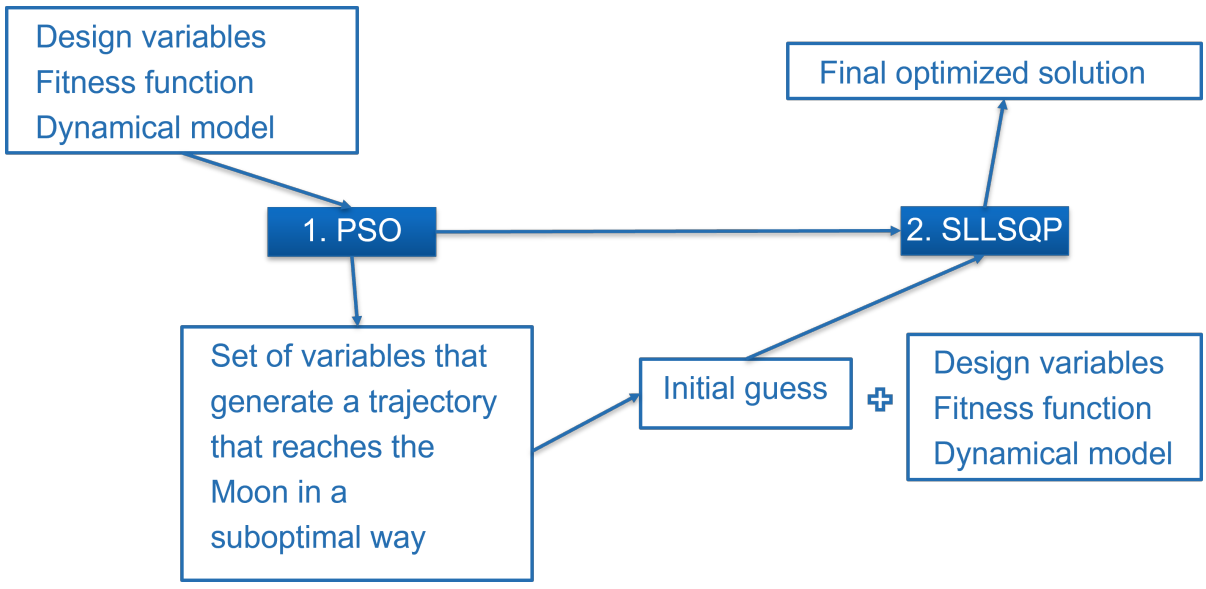


Figure 2.4: General diagram of the optimization method.

The methodology employed follows the diagram depicted in Figure 2.4. It includes two main parts: a global optimization phase using the PSO optimizer (described in Part II), and a subsequent local optimization phase using the SLLSQP optimizer (explained in Part III).

To initiate the PSO optimizer, the problem is formulated as outlined in Chapter 4. As an output, the

optimizer finds a set of variables, referred to as a *particle*, which yields a trajectory to the Moon with the chosen dynamics model in a nearly optimal manner.

The obtained set of variables serves as the initial guess for the gradient-based optimizer. However, to use the SLLSQP optimizer, the problem formulation requires a reevaluation, even if it encompasses the same essential aspects. This formulation is detailed in Chapter 9.

Throughout the SLLSQP optimization phase, the solution is fine-tuned while also accounting for additional perturbations. As a result, a final optimized solution is obtained. Do note that the design variables, fitness function and dynamics model for the two steps can be different.

2.2.1 Available Software

For these two main parts, two different software tools will be used. The PSO will be carried out using a Python environment named SEMpy, which will be introduced and explained in Chapter 5. On the other hand, the optimization with SLLSQP will be performed using Fortran 90, and it will be based on a DLR-internal software called Deepest, which stands for Deep Space Orbit Determination and Parameter Estimation Tool. This software will be called OPT-EM

Additionally, for validation purposes and to facilitate visualization, FreeFlyer, a commercial software for space mission design, analysis, and operations, will also be employed.

2.3 Research outline

The main outline of this research is as follows:

Part I includes the introduction to the research problem previously explained, and the general theoretical background required to understand some of the important theory in Chapter 3, including the reference frames.

Part II focuses on the optimization using a global optimization algorithm in SEMpy. This section presents a self-contained solution, starting with the problem formulation in Chapter 4, and an introduction to the software used in Chapter 5, highlighting its capabilities. The solution generation is explained in Chapter 6, followed by the optimizer tuning in Chapter 7.

Next, Part III focuses on the optimization in OPT-EM using a gradient-based algorithm. This part uses the solution obtained from the previous part as an initial guess, but it is otherwise entirely independent. It includes the problem formulation in Chapter 9, the propagation and solution generation in Chapter 10, and the tuning of the optimizer in Chapter 11.

Lastly, Part IV includes the results and conclusions. Chapter 12 includes the definition of four different test cases, where solutions and many findings are obtained in Chapters 13, 14 and 15 respectively. The conclusions and recommendations for future work are found in Chapter 16.

Chapter 3

Theoretical background

The theoretical background includes the dynamics models explained in Section 3.1, the possible optimizers, found in Section 3.2 and the reference frames to use, explained in Section 3.3.

3.1 Dynamical models

3.1.1 Ephemeris model

If a high-fidelity model with high accuracy is needed, the so-called Ephemeris model is required. In it, the motion of a spacecraft under the influence of the gravitational attraction of any or all of the planets and the Moon is approximated using accurate ephemerides that model the motion of these planets relative to the Sun, and are measured in an inertial frame. There are various ephemeris models, and, as explained by Parker and Anderson (2013), the DE421 Planetary and Lunar Ephemerides developed by the Jet Propulsion Laboratory (JPL) and the California Institute of Technology is the most accurate model of the Solar System. In it, the lunar orbit is accurate to within a meter, and the orbits of Earth, Mars, and Venus are accurate to within a kilometer.

In DLR's functions, an ephemeris model is also included based on the equations developed later on and Chebyshev coefficients to provide interpolations.

Within this model, additional perturbations such as the spherical harmonics of the bodies, the solar radiation pressure and the drag due to the presence of an atmosphere can also be included.

Equations of motion

To express the equations of motion of a satellite it must be noted that all the bodies in the system interact with each other through Newton's law of gravitation. These equations can be written as a summation of gravitational (g) and non-gravitational (ng) accelerations as found in Equation 3.1.

$$\vec{a} = \vec{a}_g + \vec{a}_{ng} \tag{3.1}$$

In general, it is assumed that the mass of the spacecraft is negligible compared to that of the gravitational bodies.

Two-body problem

This formulation takes only two bodies into account: the satellite, whose mass is denoted as m_P, and the central gravitational body, with a mass denoted as m_1. The rest of the gravitational and other non-gravitational accelerations are ignored. It is a useful model because the motion of the spacecraft can be approximated by conic sections, which makes a state (position, velocity) of the spacecraft very well predictable and quick to calculate. Therefore, Kepler's equations can be used to relate geometric

properties to the orbits that a body follows when it is subject to a central force, as explained in Chobotov (2002). The force acting on the spacecraft is defined in Equation 3.2.

$$\vec{F_g} = m_P \cdot \vec{a_g} \tag{3.2}$$

where $\vec{a_g}$ exerted by the body of mass m_1 to P is defined in Equation 3.3, having $\vec{r}_{1P}$ as the position vector from the center of mass of object 1 to P.

$$\vec{a_g} = -G\, m_1\, \frac{\vec{r}_{1P}}{|\vec{r}_{1P}|^3} \tag{3.3}$$

n-body problem

In reality, as found in Figure 10.7, the motion of a spacecraft is actually influenced by multiple gravitational bodies. As defined by Kemble (2006), the equation that can be used to model this n-body problem following the sketch in Figure 3.1 is Equation 3.4.

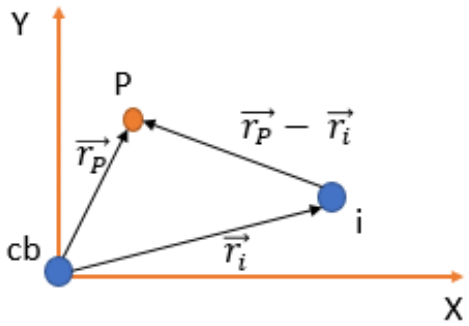


Figure 3.1: Sketch of vectors in the n-body problem.

$$\vec{a}_g = -\mu_{cb}\frac{\vec{r}_P}{|\vec{r}_P|^3} + \sum_{i=1}^{n}\left(-\mu_i\,\frac{\vec{r}_P - \vec{r}_i}{|\vec{r}_P - \vec{r}_i|^3} - \mu_i\frac{\vec{r}_i}{|\vec{r}_i|^3}\right) \tag{3.4}$$

where index $_{cb}$ denotes the central body and index $_i$ the secondary body. $\vec{r}_P$ and $\vec{r}_i$ are the position vectors of the satellite and the secondary body w.r.t. the central body respectively.

It is sometimes complicated to define a dominating body, as it depends on the distance to the third bodies and their perturbations, which can change significantly during the trajectory. The decision is then influenced by the position of the center of the reference frame, as the position vectors of the gravitational bodies w.r.t. it are required. For instance, if the center of the reference frame is placed in the barycenter of the system, denoted as B, $\vec{r}_P$ will need to be redefined as $\vec{r}_{BP} - \vec{r}_{Bcb}$, with $\vec{r}_{Bi}$ being the position vector from the barycenter to body i.

Therefore, if multiple gravitational bodies need to be included in the propagation, their positions have to be obtained. This can be done in multiple ways. For instance, in the Solar System, the trajectories of the planets could be propagated following a two-body approximation with the Sun as the central body, and then use these positions to model the trajectory of the satellite using the n-body approximation. Nevertheless, this is an approximation, as the motion of each planet is also influenced by the gravity field of the rest of the bodies. More accurate models considering these interactions exist, as explained in Section 3.1.1.

3.1.2 Circular Restricted Three-Body Problem

The CR3BP uses a non-inertial synodic reference frame. A sketch of the model can be found in Figure 3.2.

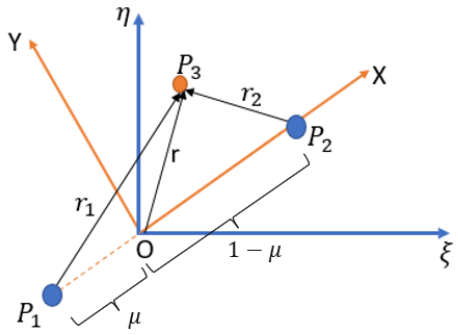


Figure 3.2: CR3BP elements and reference frame. The x- and y-components of this rotating coordinate frame can be found in orange. The blue axes are inertially fixed.

This model, as explained in Cowan (2021), describes the motion of a spacecraft P_3, with mass m_3, under the gravitational influence of two larger bodies, typically called primaries: P_1 with mass m_1, and P_2 with mass m_2. Throughout this paper, and since $m_1 > m_2$, P_1 will be designated as primary, and P_2 as secondary. As this problem is not planar, the spacecraft is allowed to move through the entire physical space. The main assumptions of this model are:

- The mass of the satellite m_3 is much smaller than m_2, which is smaller than m_1. This is: $m_3 \ll m_2 < m_1$. The problem is then *restricted*, as the spacecraft does not influence the motion of the primaries.
- P_1 and P_2 move in circular orbits around their barycenter, which is their mutual center of mass, and is set as the center of the reference frame O. This is the reason why the problem is called *circular*, and the primaries move in their orbits with a constant angular velocity. Hence, P_1 and P_2 move in a single plane, and their motion is delimited by the conservation of energy and angular momentum, but it is not the case for P_3.
- All bodies' gravitational fields can be modelled as point masses, assuming the bodies are purely spherical.
- The x-axis is defined from P_1 to P_2.
- The z-axis is parallel to the angular momentum direction of the primaries.
- The y-axis is computed using the right-hand rule.
- The distance from O to P_1 is set as μ, which is the so-called mass parameter of the system defined as:

$$\mu = \frac{m_2}{m_1 + m_2} \tag{3.5}$$

- As the distance between both bodies is non-dimensionalized to 1, so the distance from O to P_2 is set as $1 - \mu$.

To further define this coordinate system, the magnitudes are non-dimensionalized with the distance between the primaries. In addition, $m_1 + m_2$ is chosen as the unit of mass, and $1/\omega$ is the unit of

time, assuming the angular velocity of the relative motion of the primaries to be one. The equations of motion, as found in Gómez et al. (2001), will then be:

$$\begin{aligned} \ddot{x} - 2\dot{y} &= \Omega_x \\ \ddot{y} + 2\dot{x} &= \Omega_y \\ \ddot{z} &= \Omega_z \end{aligned} \tag{3.6}$$

with the subscripts $_x$, $_y$, $_z$ denoting the partial derivatives of the potential Ω completely defined in Equation 3.7.

$$\Omega(x, y, z) = \frac{1}{2}(x^2 + y^2) + \frac{1-\mu}{r_1} + \frac{\mu}{r_2} + \frac{1}{2}\mu(1-\mu) \tag{3.7}$$

with:

$$\begin{aligned} \vec{r_1} = [\mu + x,\ y,\ z]^T \rightarrow r_1 = \sqrt{(\mu + x)^2 + y^2 + z^2} \\ \vec{r_2} = [x - 1 + \mu,\ y,\ z]^T \rightarrow r_2 = \sqrt{(x - 1 + \mu)^2 + y^2 + z^2} \end{aligned} \tag{3.8}$$

This potential accounts for gravitational and centrifugal accelerations but not for the Coriolis term. It produces a non-central force and it is conservative, as it is not time-dependent. It can be noted that as μ approaches zero, the dynamics go towards that of the two-body problem but represented in a rotating frame.

3.2 Optimization

Before making a decision on the choice of optimizer it is important to understand that the perfect optimizer for all problems does not exist, and that many aspects need to be considered.

An optimization consists on minimizing or maximizing a given objective by changing the decision variables under some constraints. A mathematical description of the problem is given by Chong et al. (2013):

$$\begin{aligned} &min/max\ f(\vec{x}) \\ &x_i^{(L)} \leq x_i \leq x_i^{(U)} \qquad i = 1, 2, ...I \\ &\text{subject to:} \\ &\qquad h_j(\vec{x}) = 0 \qquad j = 1, 2, ...J \\ &\qquad g_k(\vec{x}) \geq 0 \qquad k = 1, 2, ...K \end{aligned} \tag{3.9}$$

where f represents the objective fitness function, $\vec{x}$ is the vector of the design variables, h the equality constraints, and g the inequality constraints. J should be lower than I, but K may be larger. In an optimization, there could be multiple local minima and maxima, but only one global optimum, which is the objective result. It is also important to understand that the global optimum could be located in the limits of the design region, which can decrease the chances of the optimizer to find a solution, decreasing its robustness. In addition, not all problems in the design region have continuous derivatives, which may be a problem for numerical methods.

An optimization has three important parts, as found in Equation 3.9, which are the optimization objectives, the decision variables that will be used including their limit values, and the constraints. These objectives are included in the function to optimized, which is called the fitness function, and may include a system of penalties based on the constraints. Identifying the most optimum and relevant design variables and constraints will be part of the master thesis project.

3.2.1 Optimization objectives

Many optimization objectives can be considered when computing a lunar transfer. The main ones to examine are the time of flight (ToF) and the propellant consumption, which is typically expressed as ΔV so that the capabilities of the actual rocket engine are not needed. Reducing the time of flight is important for specific missions. Manned missions are limited by radiation exposure, and time may be crucial in emergency situations. Moreover, the required ΔV is directly related to the launch and spacecraft cost, as the required propellant needs to be carried within the satellite, and hence it is always an objective to consider.

Nevertheless, there are also other possible objectives which could be useful for some applications such as the number of maneuvers, eclipse time or ground station coverage.

In the case of multiple optimization objective, there are essentially two different ways to deal with these. The first one is to focus on only one objective, becoming a so-called single-objective optimization, and leaving aside other possible considerations or including them in a combined objective function f, found in Equation 3.9, using weight factors. This may also include penalties for not satisfying problem constraints. Its main advantage is that a single solution is obtained as output, which is typically easier to deal with.

On the other hand, the problem can be studied considering different objectives, and hence a multi-objective optimization will be required. These objectives can be opposing ones. For example, a decrease in the propellant consumption leads to an increase in ToF in most cases. Therefore, the output is not a single solution, but a so-called Pareto front of optimal solutions from which to choose. The Pareto front is easy to visualize for two objectives, but can become very complicated to identify and understand when considering three or more objectives.

As stated in Section 2.2, this project will focus on single-objective optimization, as the aim is to obtain a single best solution, rather than a set of solutions that would need to be analyzed further by the user. The fitness function that will be used throughout the project is defined in Section 4.1.

3.2.2 Optimization algorithms

There are many ways of categorizing all the possible optimizers that can be used for the problem. For instance, an optimizer can be global or local. Local optimizers can typically only handle a small search space within which the function $f(\vec{x})$ is to be optimized. Nevertheless, this function can have many optima, but only one will be the global optimum. For these local optimizers to obtain the global optimum, a good initial guess is required. On the other hand, global algorithms try to deal with this problem in different manners, however, even with these global methods, there is no guarantee of convergence to the global minimum.

This distinction is very important to understand the different optimizers, but the main way to categorize them throughout this report is in gradient-based and evolutionary/genetic/PSO algorithms, respectively. The latter three follow different approaches but can be explained together. A last approach combining both options, a so-called a hybrid algorithm, will also be explained.

Gradient-based optimization

These optimizers, as explained by Snyman et al. (2005), rely on the derivatives of the objective function $f(\vec{x})$ calculated either analytically or numerically to find the best combination of decision variables $\vec{x}$ that lead to the most optimal result. These derivatives can be expressed with the gradient, the Jacobian or the Hessian matrices depending on the order of the method used and the number of design variables. They are mainly used in convex functions, and can get increasingly complicated when dealing with non-linear functions.

One of the main advantages of the gradient-based optimization is that, given a good initial guess, the

method will converge to the global optimum. Furthermore, it is easy to establish the search direction, hence the convergence is quicker than evolutionary and genetic algorithms, as they take advantage of the previous knowledge of the problem. Moreover, they have been intensely studied and are very well understood.

Nevertheless, they also have some disadvantages. For instance, if the initial guess is not close enough to the solution, the global optimum may not be found, specially when the optimized functions have a large number of local minima (multi-modal functions). Moreover, it has difficulties dealing with functions with discontinuities.

Evolutionary programming and genetic algorithms

The evolutionary and genetic optimizers, as explained by Vikhar (2016), are global search algorithms inspired by the biological process of evolution by means of natural selection, also called *survival of the fittest*, hence they are both evolutionary algorithms. Mooij et al. (2022) explains that it strategically searches the whole design space by creating a population with N individuals of guesses of design variables. Each of them are evaluated and a fitness (a measure of the goodness of fit) is assigned to each of them. Then, a subsection of the initial population, called *parents* (in general, the particles with the highest fitness values) is selected and recombined to create a new population (*offspring*) also with N individuals which are again evaluated. This procedure is repeated until a termination condition is reached. Through the fitness value of each individual, the algorithm will focus more in the most promising areas of the design space across each iteration, following, in general, natural processes. such as the flocking of birds searching for food (Particle Swarm Optimization - PSO), or the pheromone trails used by ants to find the shortest path between their nest and food (Ant Colony Optimization). PSO is a heuristic search method similar to genetic or evolutionary algorithms, but requiring less computational bookkeeping, as explained by Boeringer et al. (2004).

The main advantage of these algorithms, as found by Goldberg (1989), is that they are not limited by the objective function or the design space characteristics. This is because, as they are not gradient-based, they can handle discontinuities, non-linearities and multi-modal functions. Moreover, these algorithms will search throughout the whole space, and hence the initial guess is not determinant to reach the global optimum.

Nevertheless, they also have some drawbacks. For instance, as explained before, and even though these methods are global algorithms, there is no guarantee that they will converge to the global optimum, as they depend on random heuristics and because the solution is "better" only in comparison with the other solutions. Also, there is a large increase in computational time compared to gradient-based methods. This is because the model will need to be evaluated many times, computing new individuals and their associated fitness value for every parameter set. If this evaluation is computationally expensive, this approach can become impractical.

Hybrid algorithms

Hybrid algorithms consist of the combination of a global search by an evolutionary algorithm and a local optimization by Sequential Quadratic Programming (SQP). Ceriotti et al. (2006) explains that solutions can be generated with a genetic algorithm able to search the whole space and find a feasible solution close to the global optimum, and then refine the solution by using it as an initial guess for a gradient-based algorithm.

As highlighted in Section 2.2, a hybrid method will be used in this project to exploit the benefits of both approaches.

3.3 Reference frames

This section will include the different reference frames that can be used throughout the optimization procedure, as a clear understanding of them has proven to be crucial to obtain a good solution.

3.3.1 Earth-Centered Inertial reference frame

The most common inertial frame used for Earth-orbiting satellites is the J2000 frame. It was explained by Smith et al. (1989) after it became the main convention established in the International Astronomical Union assembly of 1977. It is an Earth Centered Inertial (ECI) frame with its origin located in the center of mass of Earth. It is defined using Earth's orbital plane and the orientation of its rotation axis, using as a basis Earth's Mean Equator and Equinox (EME) at the beginning of year 2000, so its also denoted as EME 2000. Its main axes $[X_{J2000}, Y_{J2000}, Z_{J2000}]$ can be defined as follows:

- X_{J2000}: Aligned with the vernal direction at 12:00 on January 1st 2000.
- Z_{J2000}: Aligned with the Earth rotation axis on that same date.
- Y_{J2000}: Completes the frame using the right-hand rule.

3.3.2 Perifocal Coordinate System (PQW)

This frame is centered at the focus of the orbit, which is the center of mass of the central body. As found in Figure 3.3:

- $\hat{P}$ lies in the plane of the orbit and is defected towards the periapsis of the orbit
- $\hat{Q}$ lies in the orbital plane and is perpendicular to $\hat{P}$
- $\hat{W}$ is the angular momentum vector and it is directed orthogonal to the orbital plane such that:

$$\hat{W} = \hat{P} \times \hat{Q} \tag{3.10}$$

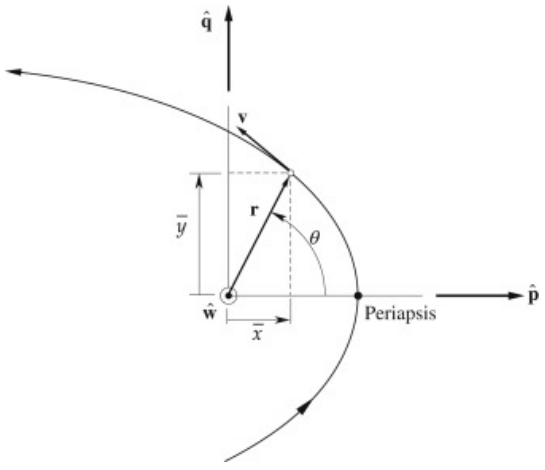


Figure 3.3: "PQW" reference frame. Retrieved from Conte (2014).

3.3.3 CR3BP

This frame co-rotates with the motion of two massive bodies about their barycenter, and its characteristics have been explained in detail in Section 3.1.2.

3.3.4 The "QSW" local orbital frame

- X-axis: unit vector in the direction of the satellite's radial vector.
- Z-axis: unit vector in the same direction as the orbit's angular momentum vector, that is, perpendicular to the orbital plane.
- Y-axis: unit vector chosen so that the (X,Y,Z) trihedral is following the right-hand rule

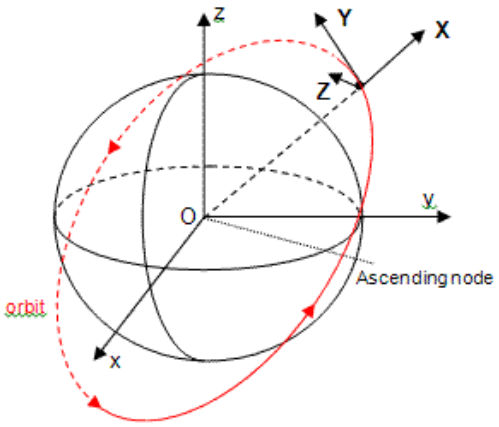


Figure 3.4: "QSW" reference frame. Retrieved from ISAE-SUPAERO (2023).

3.3.5 Coordinate systems in OPT-EM

The DLR libraries used offer many different options for both Earth-centerd and Moon-centered reference frames. The main ones have already been discussed, but having an idea of all the possibilities and their differences is important to avoid future errors.

Earth-centered reference frames

- EARTH_MEAN_EQUEX_OF_J2000: EME2000.
- EARTH_MEAN_ECLEQX_OF_J2000 - ECL2000: Mean ecliptic of J2000 system, which considers the mean obliquity.
- EARTH_MEAN_EQUINOX_OF_DATE: Earth mean equator and equinox of date system. It includes the rotation due to the precession matrix, based upon the IAU 1976 system of astronomical constants.
- EARTH_TRUE_EQUINOX_OF_DATE: Earth true equator and equinox of date system. It accounts for nutation and precession of Earth's rotational axis.
- EARTH_BODY_FIXED

Moon reference frames

- MOON_TRUE_EQUIAU_OF_DATE - IAU 1994
- MOON_TRUE_EQUIAU_OF_DATE_JPLDE - JPL ephemerides

- MOON_BODY_FIXED - IAU94
- MOON_BODY_FIXED_JPLDE - JPL-DExxx libration

To get the lunar positions, Earth's nutation and the lunar libration is accounted for. The first two reference frames are non-rotating and centered at the Moon. The IAU1994 model is based on the IAU 1994 rotational elements from Davies et al. (1995), and JPLDE takes the data from the JPL Development Ephemerides, Standish Jr et al. (1976).

3.4 Time systems

When talking about propagation, not only the reference frame and coordinate systems are important, but also the time systems. For instance, a small difference in the maneuver timing can have a significant effect.

TAI: International Atomic Time, defined in SI seconds. It is the most precise time-keeping standard and it is based on the combination of 400 atomic clocks.

TT: Terrestrial Time. Defined to be consistent with SI second (from TAI) and the General Theory of Relativity. It is mainly used by astronomers to measure the planetary positions in relation to the Earth's center.

UTC: Coordinated Universal Time. It is the main time standard that regulates the world's daily-life clocks and times. Leap seconds are added to compensate for the accumulated difference between TAI and the time measured by Earth´s rotation. These leap seconds are tabulated, and need to be given as inputs in the software.

TDB:. Barycentric Dynamical Time (from the French Temps Dynamique Barycentrique) is a relativistic coordinate time scale that considers time dilatation when calculating orbits and astronomical ephemerides of planets, asteroids, comets and interplanetary spacecraft in the Solar System. (Brumberg et al. (2001)).

To change from TT to UTC, the Earth's Rotation Parameters (ERP) table is needed, which contains the pole axis coordinates. Every time a leap second is added to UTC the separation between UTC and TT grows further.

Part II

Global optimization

Chapter 4

Problem formulation

The primary objective of the global optimization is to obtain an initial guess for a transfer orbit from a chosen Earth-bounded orbit to a designated Moon-bounded orbit without having an initial guess beforehand, hence dealing with a big search space.

The initial step to address this objective is to formulate the problem. This chapter provides a detailed explanation of the formulation, including the procedures to be followed, encountered issues, and the solutions obtained.

The main decision to be made includes three key aspects: identifying the fitness function to minimize, which will be examined in Section 4.1, selecting the design variables to optimize, as discussed in Section 4.2, and determining the algorithm to use, outlined in Section 4.3.

Throughout this chapter, a "particle" denotes a specific set of design variables. Therefore, the goal is to identify the optimal set of design variables, thus obtaining the optimal particle.

4.1 Fitness function

The fitness value is a measure used to evaluate the performance of a particular particle on the problem being addressed, and hence reflects how close the particle is to the optimal one. This value is calculated through a fitness function, which is also known as an objective function, cost function, or evaluation function. It takes a particular particle as input and gives this value as output. The goal of the optimization algorithm is to find the particle that has the highest (or lowest) fitness value, hence maximizing or minimizing the fitness function.

The fitness function plays a crucial role in the optimization algorithm, and needs to be well-designed for the problem being solved to ensure that the algorithm can converge to the optimal solution efficiently and effectively.

For the problem at hand, the fitness will be a combination of the magnitude of the two maneuvers needed to go from an Earth-bounded initial orbit to a Moon-bounded final orbit in km/s, as expressed in Equation 4.1. This value can then be directly related to the required propellant once the characteristics of the rocket engine are known. The calculation of these ΔVs is not trivial, and in this case will be calculated using a combined method between the solution to Lambert's problem and a multiple-shooting algorithm, explained in detail in Section 6.2.

$$\text{Fitness} = |\Delta\vec{V}_1| + |\Delta\vec{V}_2| \tag{4.1}$$

Although the fitness function is based on the sum of ΔVs, two adjustments were made during the

optimization process. Firstly, it was discovered that some solutions brought the satellite too close to either Earth or the Moon due to the dynamics model's limitations, which only considers them as point masses. To address this issue, the trajectory is propagated once the ΔVs are obtained, and if the satellite comes closer than the predetermined threshold (100 km above Earth's surface and 50 km above Moon's surface), the fitness value is increased with 100 km/s. A value over 100 km/s implies that the satellite intersects with one of the bodies, and over 200 km/s means both. This allows the optimizer to stay away from such solutions. Secondly, some difficulties in convergence were encountered while computing the ΔVs, and when a solution cannot be calculated or takes too long, a fitness value of 300 km/s is assigned.

Therefore, three sets of solutions are obtained: valid fitness values below 100 km/s, values between 100 and 300 km/s for solutions that do not meet the constraints, and 300 km/s for infeasible solutions that are impossible or time-consuming to compute. After some computations, it was observed that some particles got a fitness value higher than 300 km/s. These particles refer to viable options with a very low time of flight but requiring a very high ΔV, making them not physically interesting, hence being removed in later iterations.

This fitness function corresponds to a single-objective optimization. As discussed in Section 3.2.1, the ToF could also be an interesting objective to include as a weighted sum. Nevertheless, this option will be left for further analysis in future work.

4.2 Design variables

In an optimization problem, the design variables are the parameters that can be adjusted until the optimum solution is found, and they are usually defined within certain bounds.

The choice of design variables is critical in optimization problems, as it affects the size and complexity of the search space, the time required to evaluate the fitness function, and the time needed to reach the optimum. Therefore, the selection of design variables must balance the level of detail needed to accurately represent the problem, with the computational efficiency required for the optimization algorithm to converge to an optimal solution.

In order to tackle the problem at hand, numerous design variables could be selected. To simplify the decision-making process, they have been categorized into two sets: primary design variables and secondary design variables. Primary design variables refer to those that are necessary to solve the fitness function, while secondary design variables may have a significant impact on the outcome but are not essential.

- Primary design variables: they consist of the departure state, the arrival state and the ToF, as they are used as inputs for Lambert's solver to determine the required ΔVs. Because the departure and arrival state are part of a known orbit, a single parameter is required to characterize every point in it. In this case, these parameters are τ_1, τ_2, which correspond to the time elapsed since a reference point, divided by the orbital period, depicted in Figure 4.1. This parameter spans from 0 to 1 when representing a full orbit, and is has proven to be a very interesting way of characterizing points within an orbit, as it can be also used for all kinds of orbits, including periodic orbits.
- Secondary design variables: they include variables such as the initial epoch ($et0$), the Right Ascension of the Ascending Node (Ω), and other orbital parameters for both the initial and objective orbits. After careful consideration, it has been determined that the initial epoch has the greatest impact on the optimization process, and hence will also be included. By optimizing the initial epoch, the optimal Earth-Moon configuration and overall system geometry is found. Parameters τ_1 and τ_2 can have values larger than 1, showing that the satellite stays on orbit longer than a full orbital period. Nevertheless, it is not required, as the combination between their value and the initial epoch acts as a means for phasing.

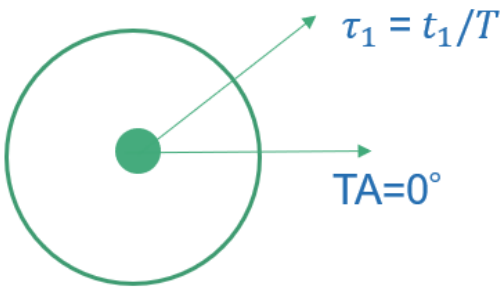


Figure 4.1: Sketch of the design variable τ_1.

4.3 Optimizer

This phase of the research will focus on using a global search algorithm to obtain an optimal transfer orbit. For this purpose, PyGMO will be employed. PyGMO is a Python Parallel Global Multi-objective Optimizer library that offers a wide range of global search algorithms suitable for different types of optimization problems, both for single- and multi-objective problems..

When dealing with problems that have a large search space and many solutions with high fitness values, population-based optimization algorithms, such as PSO, Differential Evolution (DE), and Covariance Matrix Adaptation Evolution Strategy (CMA-ES), all of which are included in PyGMO, can prove useful. These algorithms can efficiently explore the search space by maintaining a population of candidate solutions and iteratively updating them based on the best positions of either themselves or their neighbors. The basic functioning of these algorithms has been explained in Section 3.2.2.

The choice of the most suitable algorithm will depend on the specific requirements and characteristics of the optimization problem, such as the presence of constraints and available computational resources. While all three possibilities will be analyzed, PSO has been selected due to its ease of use, scalability to large-scale optimization problems, and robustness in noisy and dynamic environments. Consequently, PSO will be explained in more detail for further consideration.

4.3.1 Differential Evolution

DE, as described by Storn et al. (1997), is a population-based optimization algorithm that efficiently explores high-dimensional spaces with non-linear and non-convex search landscapes by means of a mutation-crossover-selection strategy.

One of its primary advantages is its simplicity and ease of implementation, as it does not require extensive parameter tuning or knowledge of the problem's gradient. Additionally, DE can be easily parallelized, enabling it to handle larger-scale optimization problems. However, DE is sensitive to the selection of its population size and mutation factor. It can be prone to stagnation or premature convergence, and maintaining diversity in the population can be challenging, potentially leading to sub-optimal solutions.

4.3.2 Covariance Matrix Adaptation Evolution Strategy

CMA-ES is a stochastic and evolutionary optimization algorithm introduced by Hansen and Ostermeier (2001) as a variant of the Evolution Strategies algorithm. It is powerful and efficient, and it was designed for continuous optimization problems with noisy or computationally expensive objective functions. It updates the covariance matrix based on the performance of the sampled solutions to adapt its search

distribution to the problem at hand, allowing it to effectively explore the search space and converge to the global optimum.

Hansen (2007) explains that it is a derivative-free algorithm with a high convergence rate, typically requiring fewer function evaluations to find a good solution. However, it may not be the best choice for problems with very high-dimensional search spaces or limited computational resources, as it has a high memory usage, because it requires storing the mean and covariance matrix of the Gaussian distribution, which can be computationally expensive for high-dimensional problems. Moreover, it was originally designed to perform well with small populations, and hence it has limited scalability.

4.3.3 Particle Swarm Optimization

PSO is a population-based optimization algorithm that is inspired by the social behavior of birds flocking or fish schooling. In PSO, a group of particles move through the search space to find the optimal solution to a given problem. Each particle represents a potential solution to the problem, made out of a combination of design variables.

Zhang et al. (2005) explains that the particles in the swarm move through the search space following the velocity update rule in Equation 4.2.

$$x_i(t+1) = x_i(t) + v_i(t+1) \tag{4.2}$$

Their velocity is adjusted based on their own previous position, the position of the best solution found so far (also known as the global best), and the position of the best solution found by the actual particle (also known as the local best), as denoted in Equation 4.3.

$$v_i(t+1) = \omega \cdot v_i(t) + \eta_1 \cdot rand() \cdot (p_g - x_i(t)) + \eta_2 \cdot rand() \cdot (p_i - x_i(t)) \tag{4.3}$$

where $x_i(t)$ and $v_i(t)$ are the position and velocity of particle i at time t, p_i and p_g are the best positions found by particle i or by any particle (g) so far, and ω is the inertia weight, controlling the effect of the particle's previous velocity. η_1 and η_2 are the social and cognitive learning factors determining the influence of the particle's own best position and the global best position, respectively, and rand() is a random number generator, introducing stochasticity to the optimization process.

This process of updating the velocity and position of each particle is repeated until a stopping criterion is met. In the case of the PSO algorithm in PyGMO, the stopping criterion is the maximum number of iterations, given by the population size and the number of generations.

One of the main advantages of PSO is its simplicity and ease of implementation. It also has the ability to converge quicker than other global search algorithms, especially when dealing with high-dimensional search spaces. PSO is also robust to noise and able to handle multimodal objective functions (functions with multiple optimal solutions).

The main disadvantage of PSO is that it may converge prematurely to a sub-optimal solution. This can be mitigated by carefully selecting the parameters of the algorithm (such as the learning factors and the swarm topology), but finding the optimal parameters can be challenging.

Compared to other global search algorithms, PSO is often faster and more efficient for high-dimensional search spaces. It also has a lower computational cost compared to other population-based algorithms, such as genetic algorithms. For these reasons, PSO was the optimization algorithm selected7 for this problem.

Chapter 5

Introduction to SEMpy

SEMpy, the name of which comes from Sun-Earth-Moon system in Python, is an open-source Python library developed within ISAE-SUPAERO. It is a powerful tool designed specifically for non-Keplerian orbital mechanics. With a primary focus on the CR3BP and its solutions, such as Halo and Near-Rectilinear Halo orbits (NRHO), SEMpy provides a feature-rich environment that can be used in many space mission scenarios. In addition, another of the library's key features is the support for an ephemeris model, enabling precise planetary and celestial-body positions.

The functionalities in SEMpy include the possibility to work with different reference frames, perform differential corrections, explore dynamics within the CR3BP, and use ephemeris models with or without spherical harmonics. Additionally, the library provides essential tools like Lambert solvers, CR3BP orbit simulations, low-thrust maneuvers, and diverse propagators to analyze and predict spacecraft trajectories accurately.

SEMpy has been the environment of choice for this project due to its capabilities, well-documented examples and ease of use. Therefore, some of its main capabilities will be explained throughout this chapter.

5.1 Ephemeris dynamics

SEMpy offers the possibility to propagate an initial state using the ephemeris force model, as described in Section 3.1.1. This force model involves a set of first-order Ordinary Differential Equations (ODEs) that govern the motion of a particle under the gravitational attraction of different celestial bodies in the Solar System, expressed in an inertial reference frame. To perform the propagation, relative positions of the desired celestial bodies with respect to the origin are required, which are obtained from the ephemeris data from the SPICE routine *spkgps*.

The gravity force exerted by the celestial bodies can be modeled either as the result of point masses or by incorporating the spherical harmonics. In the latter case, a file containing the necessary data is needed. For Earth's spherical harmonics, the EGM2008 model is used. This model was released by the National Geospatial-Intelligence Agency (NGA) EGM Development Team and uses data from the GRACE satellite mission, providing a high-resolution global gravity model, as explained by Pavlis et al. (2012).

Within SEMpy, the ephemeris force model can propagate the state, requiring 6 ODEs. Additionally, if needed, it can also propagate the 6x6 State Transition Matrix (STM), thus involving 42 first-order ODEs. Furthermore, there is an option to propagate both the STM and the 6x1 partial derivatives of the state with respect to the epoch time, resulting in a total of 48 equations.

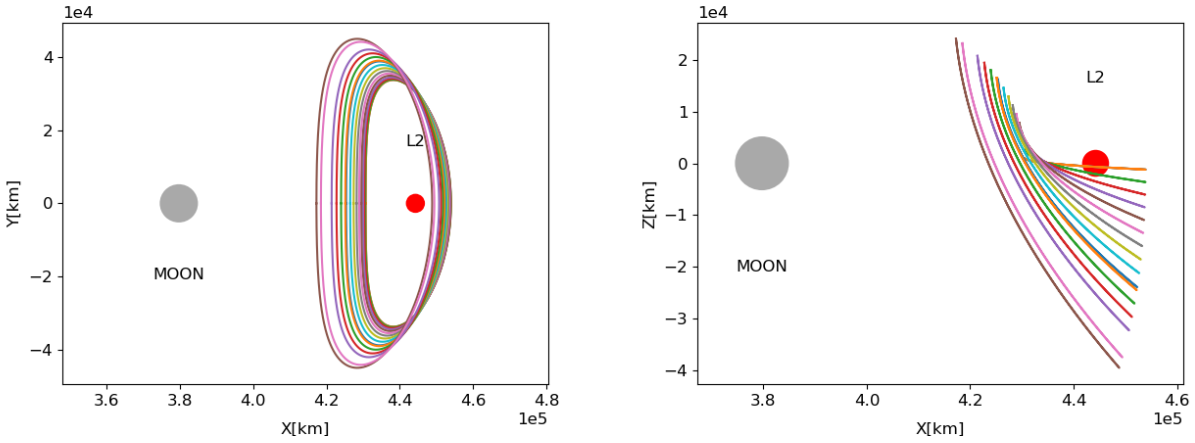
5.2 CR3BP dynamics

As mentioned before, a key advantage of SEMpy is working within the CR3BP in a direct manner, and the possibility to convert the solutions into an inertial frame. The main assumptions of the CR3BP have already been explained in Section 3.1.2. Therefore, the focuse now is to explain the main solutions that can be obtained with the help of the given libraries.

5.2.1 Periodic orbits

Periodic orbits are solutions within the CR3BP that represent stable configurations where the gravitational forces from the two main bodies balance out, resulting in a repetitive and predictable three-dimensional trajectory. These orbits can take various shapes, such as Lissajous orbits, Halo orbits, and NRHOs. Obtaining and calculating periodic orbits is usually a complicated tasks. This is because, in many of the cases, analytical solutions do not exist, and numerical iterative techniques must be employed to find approximate solutions. However, these difficulties have already been solved in SEMpy, and hence families of solutions can be obtained directly.

To generate a Halo orbit, SEMpy requires the selection of the proper CR3BP to be used, the libration point about which the orbit is placed, the family required to be created, and/or the orbit vertical extension (Az_{dim}), the Jacobi constant or the orbit's period. Figure 5.1 shows a southern family of Halo orbits around L2 with a vertical extension from 1000 to 31000 km every 2000 km.

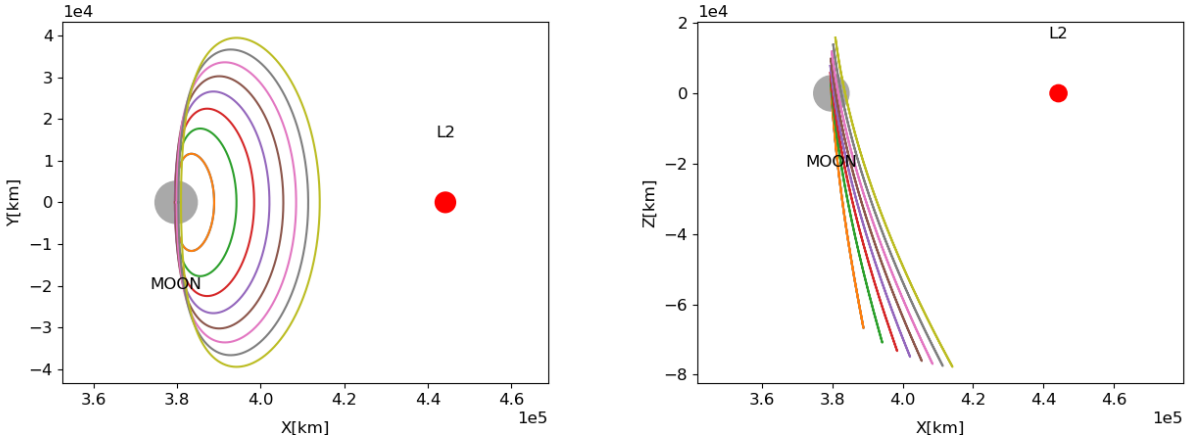


(a) X-Y plane

(b) X-Z plane

Figure 5.1: Earth-Moon CR3BP system, Halo family about L2. Nominal Az_{dim} from 1000 km to 31000 km, step=2000 km.

NRHOs are part of the family of Halo orbits, but with bounded stability indices. They are nearly polar and offer favorable stability that allows satellites to maintain their motion with a very low propellant consumption, which is the reason why it is the planned orbit for the Gateway station in the direct vicinity of the Moon. Figure 5.2 shows a southern family of NRHO with a periseline radius (Rp_{dim}) from 1800 to 17000 km.



(a) X-Y plane (b) X-Z plane

Figure 5.2: Earth-Moon CR3BP system, NRHO family about L2. Nominal Rp_{dim} from 1800 km to 17000 km, step=2000 km.

5.2.2 Optimization of transfers between Halo orbits

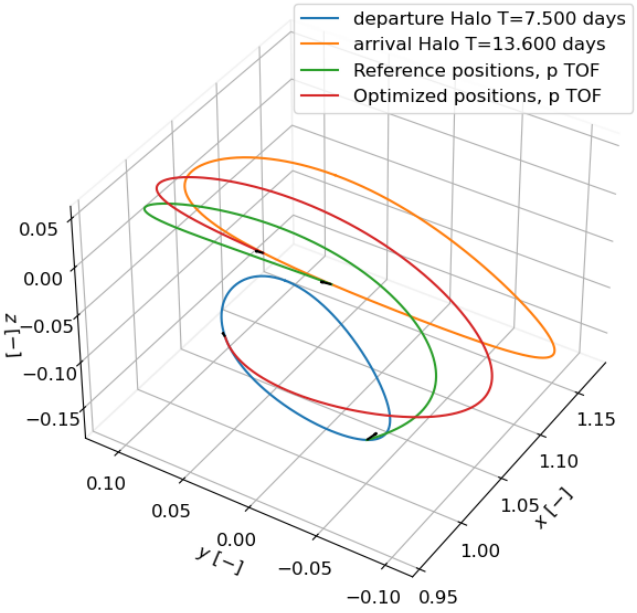


Figure 5.3: Reference (red) and optimized (green) transfer between two L2 southern Halo orbits.

SEMpy also offers an important and interesting feature that allows users to calculate transfers between two periodic orbits. This capability is achieved through a combination of a Lambert solver and a multiple-shooting algorithm. Throughout the project, this method will play a crucial role and is thoroughly explained in Section 6.2. Additionally, SEMpy provides an example demonstrating the optimization of these transfers using a local optimizer. Its solution is found in Figure 5.3. The blue and orange orbits represent the departure and arrival Halo orbits, respectively. The green curve indicates the reference

transfer, which serves as the initial guess. After optimization, the final transfer is depicted in red. The total ΔV required for the reference transfer is 172 m/s, but through optimization, this value is reduced to 117 m/s.

5.2.3 Transition from CR3BP into an ephemeris model

Lastly, the SEMpy software is able to correct the periodic orbits found in the CR3BP for a high-fidelity N-body ephemeris model. Figure 5.4 shows the correction for three revolutions. This feature is interesting in order to include the periodic orbits with a high-fidelity model. However, their computation and corrections take around 10 s, making it impractical for real-time use in optimization, as these orbits and corrections are dependent on the actual epoch, and they would have to be recalculated for each particle.

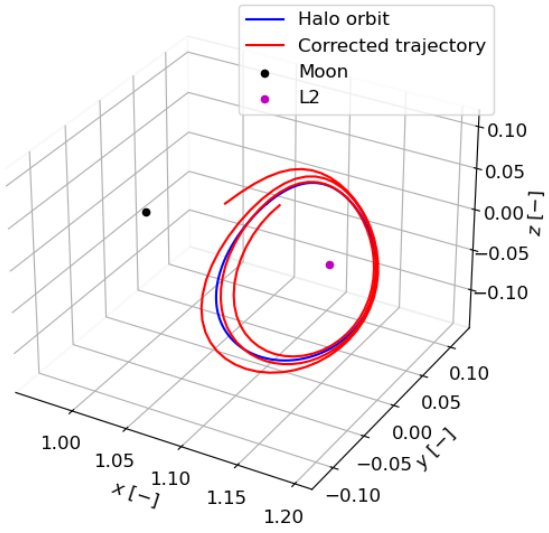


Figure 5.4: Halo orbits and its correction for a higher-fidelity N-body ephemeris model.

5.3 Change of reference frames

SEMpy also offers a very useful set of changes of reference frames, but they need to be understood and tested in order to make use of them with certainty.

5.3.1 Synodic to inertial

For this transformation, the inertial reference frame is defined such that its x, y, and z-axes are aligned with the x, y, and z axes of the synodic frame for t=0, where t is the time expressed in normalized units. As a consequence, the z direction of the two frames coincides at every time, while the x and y directions of the synodic frame rotate w.r.t. the inertial ones at a constant rate (n = 1).

The origin of the inertial frame is selected to be either the first or second primary associated to the input CR3BP structure.

5.3.2 Synodic to EME2000

This module implements the change of coordinates from the synodic reference frame defined for a CR3BP system, to the instantaneous EME2000 (J2000) inertial frame centered at the first primary. To build the instantaneous rotation matrices it is required to know the states for both primaries and the system's barycenter, as well as their relatives distances. Hence, their values are obtained from ephemeris data from the Python library *spkgeo*.

The main theory behind this transformation, based on the sketch in Figure 5.5, is the following :

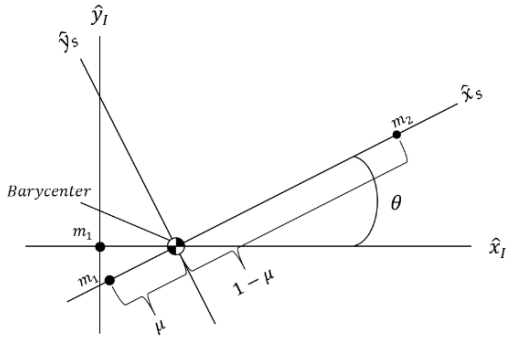


Figure 5.5: Synodic and EME2000 reference frames. Retrieved from Dahlke (2018).

$\vec{R}_S$ and $\vec{V}_S$ represent the position and velocity in the synodic frame, and $\vec{R}_I$ and $\vec{V}_I$ in the EME2000 frame. Their $\hat{z}$-axis coincides, but there is an offset because the CR3BP frame is centered at the barycenter and the EME2000 at Earth's center. An intermediate position only accounting for this translation ($\vec{R}_{S'}$) is calculated as:

$$\vec{R}_{S'} = [x_s + \mu, y_s, z_s] \tag{5.1}$$

Then, the required rotation for the position and velocity is:

$$\vec{R}_I = Q^{IS}\vec{R}_{S'} \tag{5.2}$$

$$\vec{V}_I = Q^{IS}\vec{V}_S + \dot{Q}^{IS}\vec{R}_{S'} \tag{5.3}$$

where:

$$Q^{IS} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.4}$$

and its derivative is:

$$\dot{Q}^{IS} = \dot{\theta}\begin{bmatrix} -\sin(\theta) & -\cos(\theta) & 0 \\ \cos(\theta) & -\sin(\theta) & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.5}$$

5.3.3 Orbital to inertial

From the Keplerian elements, the state is calculated in the perifocal orbital reference frame (o), as found in the following chapter, in Equation 6.2. To convert to the EME2000 (i) the transformation, as explained

by Standish et al., 2006, goes as follows:

$$\begin{bmatrix} r_{xi} & r_{yi} & r_{zi} \end{bmatrix} = \begin{pmatrix} \cos\omega\cos\Omega - \sin\omega\cos i\sin\Omega & -\sin\omega\cos\Omega - \cos\omega\cos i\sin\Omega \\ \cos\omega\sin\Omega + \sin\omega\cos i\cos\Omega & \cos\omega\cos i\cos\Omega - \sin\omega\sin\Omega \\ \sin\omega\sin i & \cos\omega\sin i \end{pmatrix} \cdot \begin{bmatrix} r_{xo} \\ r_{yo} \end{bmatrix} \tag{5.6}$$

As the position and velocity in the $\hat{z}$-direction in the perifocal frame, normal to the orbital plane, is zero by definition.

5.4 Testing propagation and transformation from CR3BP to ephemeris model

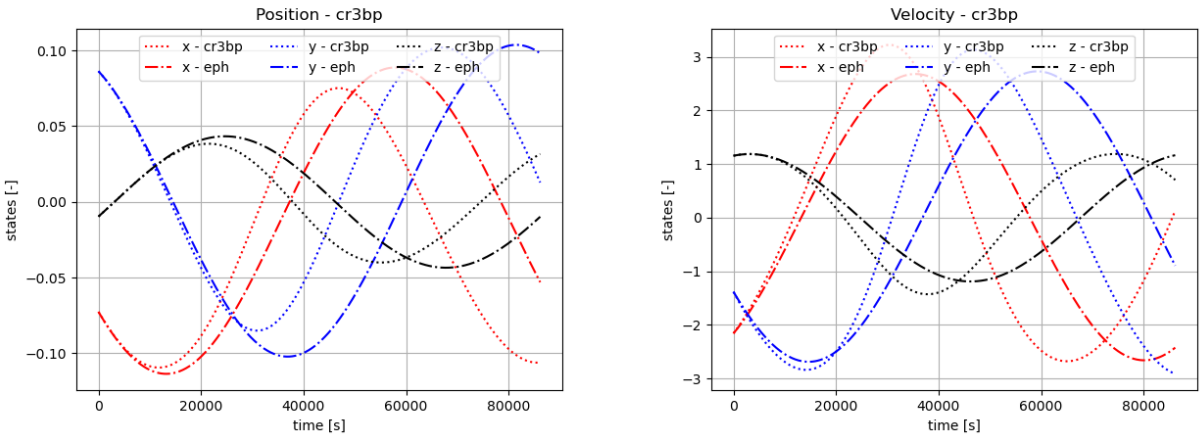
If the propagation in the CR3BP is to be used, it is important to test the transformation between this reference frame and the EME2000 frame, as it is a necessary step for further analyses.

The test involves taking an initial state in the EME2000 frame, translating it to CR3BP, propagating it both in the CR3BP and the ephemeris model, and then converting each point from the ephemeris propagation into the CR3BP reference frame in order to compare both outcomes. To make both models comparable, the ephemeris propagation only included Earth's and the Moon's point-mass gravity fields.

During the testing process, significant differences were observed between both propagations, and these differences were found to be dependent on the initial epoch, as seen in Figures 5.6, 5.7, and 5.8.

After thorough analysis, it was identified that the problem lies in the method used to generate the CR3BP and to switch between the reference frames. One of the main assumptions in the CR3BP is that the Moon moves in a circular orbit around their common barycenter, hence with a constant angular velocity, but this is not the case in reality.

2020 JUN 18 12:00:00.000

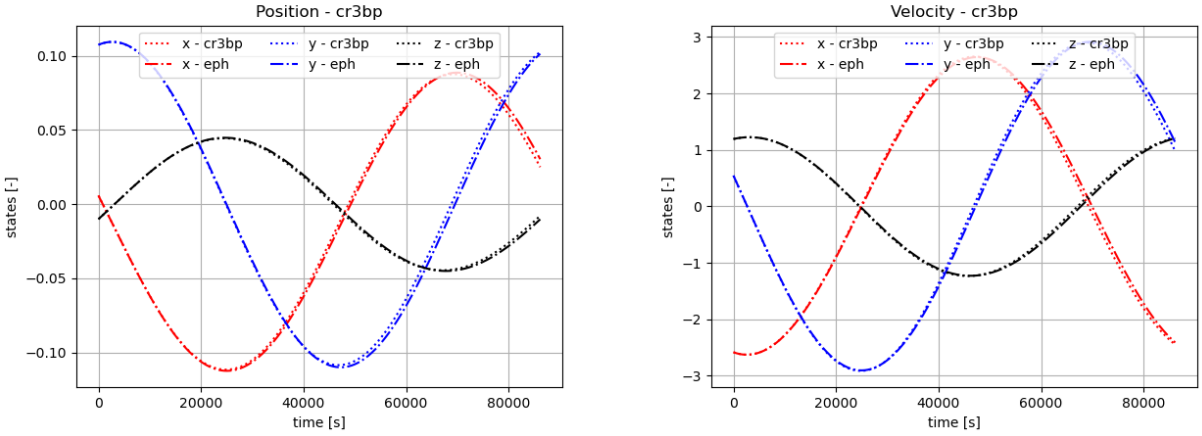


(a) Position

(b) Velocity

Figure 5.6: x-y-z position and velocity evolution during one orbital period in the CR3BP for a GEO the 18^{th} June 2020 at 12pm.

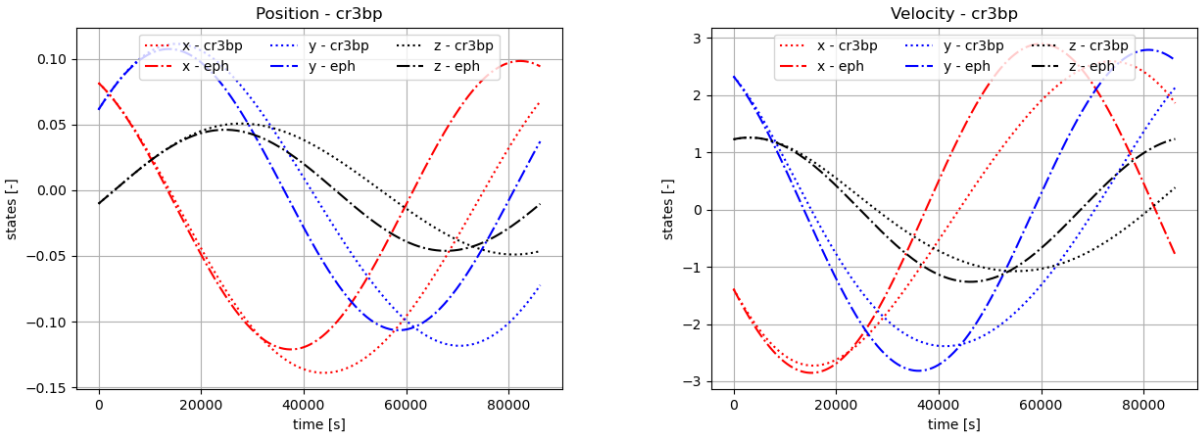
2020 JUN 22 00:00:00.000



(a) Position (b) Velocity

Figure 5.7: x-y-z position and velocity evolution during one orbital period in the CR3BP for a GEO the 22^{nd} June 2020 at 12am.

2020 JUN 25 12:00:00.000



(a) Position (b) Velocity

Figure 5.8: x-y-z position and velocity evolution during one orbital period in the CR3BP for a GEO the 25^{th} June 2020 at 12pm.

SEMpy builds the CR3BP by taking the instantaneous position and velocity of the Moon at the time given, and seting its direction as the direction of the $\hat{x}$-axis. Nevertheless, the model takes as the nondimensionalization parameters the mean Moon semi-major axis, which is 384400.0 km, and its period using Kepler's law, as found in NASA (2021). Therefore, the transformation only matches both outcomes when the position and velocity of the Moon are the same as the ones taken as constant, used when initializing the model. Nevertheless, when they do not, the discrepancies grow large.

While the CR3BP offers an interesting and efficient way to analyze transfers, the difficulty in validating the change of reference frame and the errors associated with it made it unsuitable for the subsequent parts of the project. As a result, the decision was made to exclusively use the ephemeris model for the optimization.

Additionally, one of the main advantages of CR3BP, which is the inclusion of periodic orbits, will not be used, as Kepler orbits around the Moon were selected as the target final orbits.

Chapter 6

Solution generation

This chapter focuses on the calculation of the trajectory and the required maneuvers from the input design variables, which are the time of flight, the initial epoch, and parameters τ_1 and τ_2. Apart from them, the main inputs include the desired initial orbit around Earth and the final orbit around the Moon. The most convenient and user-friendly representation of an orbit is through its Kepler elements, hence it is the way the inputs are defined.

This chapter consists of two main sections. Section 6.1 covers the computation of the initial and final orbits based on the given input Kepler elements. Section 6.2 includes the calculation of the required ΔV using a combination of a Lambert solver and a multiple-shooting method.

6.1 Initial and final orbits

To determine the actual state for the first and last maneuver, considering τ_1 and τ_2, the Keplerian elements provided as inputs need to be converted into position and velocity in the EME2000 reference frame.

The required Keplerian elements in all cases are: the semi-major axis (which can also be specified using the altitude, given the known mean radii of Earth and the Moon), the eccentricity (e), the inclination (i), the argument of perigee (ω), and the right ascension of the ascending node (RAAN, Ω). Additionally, the initial epoch in ephemeris seconds is necessary for the propagation in the ephemeris model, as it requires the positions of Earth and the Moon at the specified time. Two options are analyzed: calculating the state from the mean anomaly (M) or from the true anomaly (TA, θ). A sketch for both angles can be found in Figure 6.1.

The mean anomaly represents the fraction of an elliptical orbital period that has elapsed since the orbiting body passed periapsis, expressed as an angle in radians. For this option, the initial mean anomaly and time are required. However, the software also provides the option to calculate the state from TA, which is the angle between the direction of periapsis and the current position of the body. This alternative can be used to avoid propagating from the initial state to the state of maneuver firing, using τ.

In both cases, a class called *KeplerOrbit* will be initialized, encompassing the main parameters of the orbits.

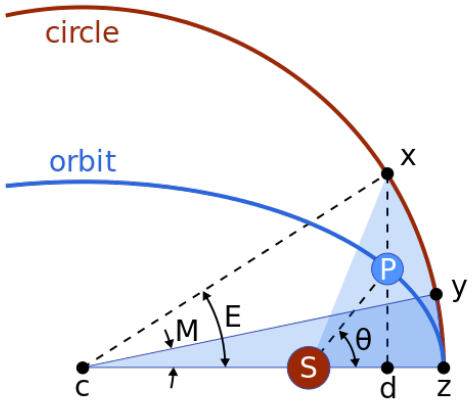


Figure 6.1: Mean (M), true (TA, θ) and eccentric (E) anomaly. Retrieved from Wikimedia Commons, 2008.

Inertial states from true anomaly

The first step is to calculate the eccentric anomaly, which is the third angular parameter that defines the position of a body that is moving along an elliptic Kepler orbit.

$$E(t) = 2 \cdot \tan^{-1}(y(t)/x(t)) \qquad x(t) = \sqrt{1+e} \cdot \cos(\theta(t)/2) \qquad y(t) = \sqrt{1-e} \cdot \sin(\theta(t)/2) \tag{6.1}$$

Knowing the eccentric anomaly, the position and velocity of the satellite can be calculated as:

$$r_{orb}(t) = r_c(t) \begin{pmatrix} \cos\,\theta(t) \\ \sin\,\theta(t) \\ 0 \end{pmatrix}, \qquad v_{orb}(t) = \frac{\sqrt{\mu a}}{r_c(t)} \begin{pmatrix} -\sin\,E(t) \\ \sqrt{1-e^2}\,\cos\,E(t) \\ 0 \end{pmatrix} \tag{6.2}$$

where the distance to the central body is:

$$r_c(t) = a\,(1 - e\,\cos\,E(t)) \tag{6.3}$$

To convert from r_{orb} and v_{orb} to the position and velocity in inertial frame, the rotation matrix found in Section 5.3.3 is used.

Calculating inertial states from the mean anomaly

The first step is to get the actual mean anomaly from the actual time, the initial time (t_0) and the mean anomaly at the initial time (M_0) as:

$$M(t) = M_0 + \Delta t \sqrt{\frac{\mu}{a^3}} \qquad \text{with} \quad \Delta t = 86400(t - t_0) \;\; [\text{s}] \tag{6.4}$$

Then, $M(t)$ is normalized to be in $[0,\, 2\pi)$. Afterwards, Kepler's equation is solved to obtain the eccentric anomaly.

$$M(t) = E(t) - e\,\sin\,E(t) \tag{6.5}$$

The true anomaly can then be obtained from equation:

$$\tan\left(\frac{\theta}{2}\right) = \left(\frac{1+e}{1-e}\right)^{1/2} \tan\left(\frac{E}{2}\right) \tag{6.6}$$

There is no difference between both calculations apart from the fact that the mean anomaly approach will obtain points that are equally spaced in time, and the true anomaly will not. Moreover, τ_1 and τ_2 can be easily translated to angles that represent the true anomaly.

6.1.1 Testing of transformation from Kepler elements to inertial states

Verifying the function that takes the Keplerian elements as inputs and generates the corresponding state in EME2000 as output is crucial due to its importance in the performance of the code. The test has been conducted by converting back and forth between Keplerian elements and states. The obtained results were then compared with those from an independent function provided in SEMpy, which was written separately by another author. Both the initial orbit around Earth and the final orbit around the Moon were considered during this process.

The obtained state, defined by its position and velocity,matches the one obtained with the independent function. Furthermore, bidirectional conversion also yields the exact results, both verifying this part of the code.

An important observation made during testing is the significance of unit consistency. To make the software more user-friendly, the input values for angular parameters are set in degrees, as it is easier for visualization. However, the underlying functions operate in radians. Therefore, it is required to convert from degrees to radians before the start of the propagation.

Lastly, it needs to be noted that for consistency, the true anomaly has been set to zero for the calculation of the initial state. Moreover, Ω and ω should also be set to zero even if the orbit is equatorial or circular respectively.

6.1.2 Inclination of Moon-bounded orbit

In SEMpy, the Keplerian elements are calculated in the EME2000 frame, but the Moon can also be set as the central body, so the calculations of the states around the Moon can be conducted in the same way concerning the semi-major axis and the eccentricity. However, the inclination is calculated w.r.t. the equator of Earth, and not w.r.t. the equator of the Moon, which will be the input inclination given by the user. To solve this problem, it is important to analyze the geometry of the Earth-Moon system.

Figure 6.2 shows that the Moon orbits Earth near the ecliptic, which is the plane of Earth's orbit around the Sun, with a fixed angle between its own equator and this plane of 1.543°. Moreover, the figure shows that the rotation axis of Earth is tilted with respect to the ecliptic, and this angle is known as the obliquity. Hohenkerk et al. (1992) explains that the obliquity of Earth suffers from both long- and short-term variations, and its mean value is 23.44°, which remains nearly constant throughout the cycles of axial precession, defined as the slow, and continuous change in the orientation of the body. Earth's cycle lasts approximately 26000 years, hence this value can be taken as constant for the time scale of a mission to the Moon.

Therefore, the actual inclination used by the software needs to be calculated from the input inclination plus 23.44° minus 1.543°.

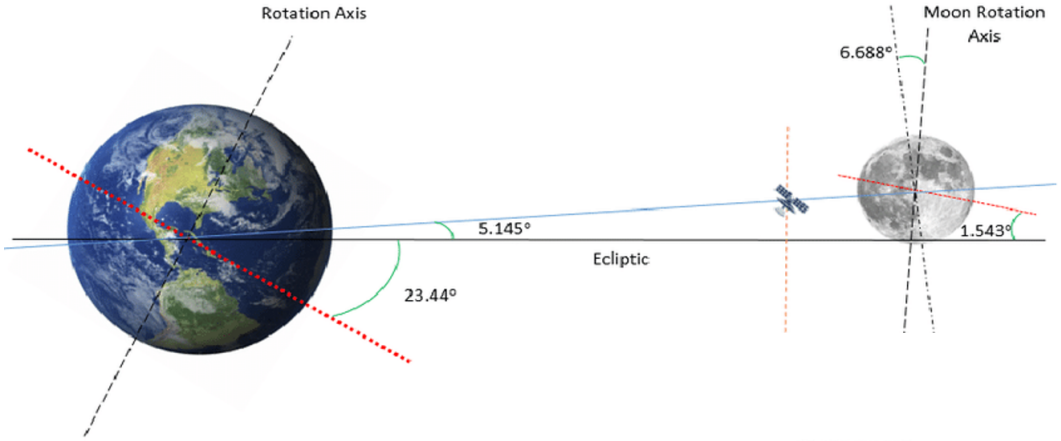


Figure 6.2: Earth-Moon system geometry, not in scale. The red dotted lines correspond to the bodies' equatorial planes. Figure modified from Kara et al. (2015).

6.1.3 Initial state for Lambert's problem

The actual initial state required for Lambert's solver is calculated by using the transformation explained above, which gets the state in the initial input orbit at a true anomaly θ of 0°. This value is independent of the design variables. Afterwards, this state is propagated to a time $\tau_1 \cdot T_E$ or to a time $\tau_2 \cdot T_M$, to find the state where the maneuver should be conducted.

As an initial hypothesis, this propagation will be conducted using a point-mass gravity field of Earth and the Moon, as it is also the model used for Lambert's solver. The main assumption is that this model would lead to a solution that is close enough to the trajectory under additional perturbations, so that the following part including the refinement and correction of the initial guess found here can be carried out. Nevertheless, if this were not the case, the spherical harmonics of Earth could be included as perturbations for the propagation in the initial orbit to reduce the error of the initial state and hence of the initial guess obtained with Lambert's problem.

6.2 Lambert's solver

Lambert's problem explained here has two main parts: the nominal Lambert's approach based on a two-body approximation explained in Subsection 6.2.1, and the multiple-shooting method used to correct the initial guess calculated with the nominal Lambert's approach within a three-body formulation, explained in Subsection 6.2.2. Furthermore, some issues encountered will be addressed in Subsection 6.2.4.

6.2.1 Lambert's Approach

As followed by Chobotov (2002), Lambert's problem is a two-body analytical method used to calculate transfer orbits between two given position vectors in a specified transfer time. It allows for out-of-plane transfers and a variation of the time of flight. It has been subject of extensive research and is greatly used to obtain first approximations for more complex models. Figure 6.3 shows the main parameters of a Lambert transfer between two points. According to Lambert's theorem, the transfer time ToF to go from P_1 to P_2 is only dependent on the sum of the magnitude of the position vectors $(r_1 + r_2)$, the semi-major axis (a), and the length of the chord joining P_1 and P_2 (c). Therefore, it is independent of the orbit's eccentricity.

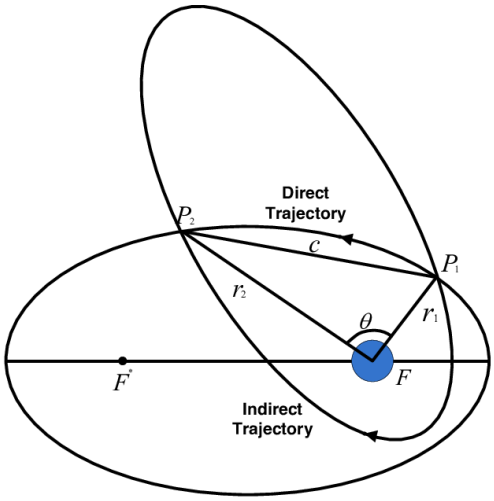


Figure 6.3: Transfer orbit geometry for Lambert's problem. Retrieved from Hosseini (2011).

Knowing that:

$$\begin{aligned} c &= ||\vec{r_2} - \vec{r_1}|| = \sqrt{r_1^2 + r_2^2 - 2r_1 r_2 \cos(\theta)} \\ s &= \frac{c + r_1 + r_2}{2} \end{aligned} \tag{6.7}$$

with c being the chord length and s the semi-perimeter. It can be obtained that the ToF is:

$$\text{ToF} = \sqrt{\frac{a^3}{\mu}} \left(\alpha - \beta - (\sin\alpha - \sin\beta)\right) \tag{6.8}$$

where:

$$\begin{aligned} \sin\left(\frac{\alpha}{2}\right) &= \sqrt{\frac{s}{2a}} \\ \sin\left(\frac{\beta}{2}\right) &= \sqrt{\frac{s-c}{2a}} \end{aligned} \tag{6.9}$$

One very interesting conclusion to reach from the analysis of these equations, as explained by Peet (2018), is that there exists a minimal semi-major axis length $a_{min} = (r_1 + r_2 + c)/4$ for which it is possible to have both points in the same orbit. To ensure that a solution exists, the minimal ToF corresponding to a parabolic case can be calculated as well as the maximal ToF related to a_{min}. To solve Lambert's equation, several options can be used including Newton iteration, series expansion or a bisection method. The quickest method, as explained by De La Torre et al. (2018) and the one that will be chosen, is to use a first-order Taylor's expansion.

As can be seen, Lambert's problem has an analytical solution in the two-body problem. Nevertheless, in more complex dynamical models such as the three-body problem or the CR3BP, an analytical approach does not exist and an iterative procedure is required, becoming a time-fixed single-shooting problem with thirteen variables. Four fixed variables (x_0, y_0, z_0, ToF), three free variables ($\dot{x}_f$, $\dot{y}_f$, $\dot{z}_f$), three constraints (x_f, y_f, z_f) and three unknowns ($\dot{x}_0$, $\dot{y}_0$, $\dot{z}_0$). Here, subscript $_0$ refers to the initial point and subscript $_f$ to the final point.

6.2.2 Single and multiple-shooting methods

As explained before, differential corrections are required when using Lambert's approach in models that include forces apart from the central body's point-mass gravity field. For this purpose, single-shooting and multiple-shooting methods are used. As explained by Gupta (2020), the goal of the problem using a single shooting method is to determine the $\Delta\bar{v}_0$ required to reach a desired final position $\bar{r}_f = [x_f\ y_f\ z_f]^T$ from a given initial position $\bar{r}_0 = [x_0\ y_0\ z_0]^T$ and time t_0 with an associated initial velocity $\bar{v}_0 = [\dot{x}_0\ \dot{y}_0\ \dot{z}_0]^T$ in a time Δt. At the end of each propagation, the state will be $\bar{r}_F = \bar{r}(\bar{r}_0, t_f)$, with t_f being the final time.

The problem now consists of a design variable vector ($\bar{X}$) containing the initial velocity and time, and the constraint vector $\bar{F}(\bar{X})$ representing the difference between $\bar{r}_F$ and $\bar{r}_f$. The objective is that $\bar{F}(\bar{X}) = \bar{0}$. The solution of this problem is based on the difference between the obtained final state and the target one in order to estimate the required adjustment in the initial state vector, using the Jacobian matrix as $D\bar{F}(\bar{X}) = \delta\bar{F}/\delta\bar{X}$ or the state transition matrix (STM) as $\Phi(t, t_0) = \delta\bar{x}(t)/\delta\bar{x}(t_0)$, with $\bar{x}$ represented as $\bar{x} = [x\ y\ z\ \dot{x}\ \dot{y}\ \dot{z}]^T$. Firstly, a first-order Taylor series expansion is used:

$$\bar{F}(\bar{X}) \approx \bar{F}(\bar{X}_0) + \bar{D}F(\bar{X}_0) \cdot (\bar{X} - \bar{X}_0) \tag{6.10}$$

with $\bar{X}_0$ being the initial guess. Then, Equation 6.11 is iteratively solved for $\bar{X}^{j+1}$ until a certain tolerance ϵ is met so $||\bar{F}(\bar{X}^{j+1})|| \leq \epsilon$.

$$\bar{F}(\bar{X}^j) + \bar{D}F(\bar{X}^j) \cdot (\bar{X}^{j+1} - \bar{X}^j) = \bar{0} \tag{6.11}$$

If the number of constraints and the number of free variables is the same, $\bar{D}F(\bar{X}^j)$ is a square matrix and it is invertible. Hence, the solution at each iteration can be solved as:

$$\bar{X}^{j+1} = \bar{X}^j - \bar{D}F(\bar{X}^j)^{-1}\bar{F}(\bar{X}^j) \tag{6.12}$$

This correction is based on linearized equations of motion, which means that the adjustment found is approximate, but it will converge in many cases after some iterations if the error in the initial guess is not too high. However, this convergence is difficult to achieve in the CR3BP using a single-shooting strategy due to its chaotic characteristics, especially close to the primaries, which make the final position highly sensitive to the initial state. Moreover, round-off errors can become too large, and may create large-scale deviations in the spacecraft's trajectory, as explained by Parker and Anderson (2013). Therefore, another option is to use a so-called multiple-shooting strategy. This strategy is based on the discretization of the trajectory in n patched points so that the propagation is done over shorter periods of time, giving

better results when using the linearization approximation. Parker and Anderson (2013) explains that these segments can be patched together with very small maneuvers, smaller than those required for station-keeping, to counteract the round-off error built-up in each segment. And as they are small, they do not have to be included as deterministic maneuvers for the mission.

As explained by Parker and Anderson (2013), the multiple-shooting method is an iterative process with two levels where the states of the patch points are adjusted simultaneously. Level 1 consists on adjusting the velocities at the patch points using a single-shooting differential corrector to achieve a position at the end of each section that is the same as the one of the next patch point ($\delta r_f = 0$). Therefore, after this level, a continuous trajectory is obtained, but a ΔV would be required at each patch point. For this level, the following equations are used:

$$\delta \bar{x}_f = \Phi(t_f, t_0) \cdot \Delta \bar{x}_0 \tag{6.13}$$

with $\bar{x} = [\bar{r}\ \bar{v}]$. Hence:

$$\begin{bmatrix} \delta r_f \\ \delta v_f \end{bmatrix} = \begin{bmatrix} \Phi_{rr} & \Phi_{rv} \\ \Phi_{vr} & \Phi_{vv} \end{bmatrix} \begin{bmatrix} \Delta r_0 \\ \Delta v_0 \end{bmatrix} \tag{6.14}$$

As $\Delta r_0 = 0$, and δv_f is in general unconstrained, the equation simplifies as:

$$\delta r_f = \Phi_{rv} \cdot \delta v_0 \tag{6.15}$$

Some iterations are required at this level because $\Phi(t, t_0)$ is propagated with linearized equations.

In Level 2, the positions and epochs of the patch points are adjusted with a least-squares method to reduce the ΔV cost of the trajectory. At the end of this level, the trajectory will not be continuous but it will require less ΔV. The main equation to be used in this level is the following:

$$[\delta \Delta \bar{V}_2] = \begin{bmatrix} \frac{\delta \Delta \bar{V}_2}{\delta \bar{r}_1} & \frac{\delta \Delta \bar{V}_2}{\delta t_1} & \frac{\delta \Delta \bar{V}_2}{\delta \bar{r}_2} & \frac{\delta \Delta \bar{V}_2}{\delta t_2} & \dots & \frac{\delta \Delta \bar{V}_2}{\delta \bar{r}_n} & \frac{\delta \Delta \bar{V}_2}{\delta t_n} \end{bmatrix} \begin{bmatrix} \delta \bar{r}_1 \\ \delta t_1 \\ \delta \bar{r}_2 \\ \delta t_2 \\ \dots \\ \delta \bar{r}_n \\ \delta t_n \end{bmatrix} \tag{6.16}$$

with $\Delta \bar{V}_2$ being the velocity deviations at each of the n patch points. The matrix composed of the partial derivatives will be written as M for simplicity. The system in Equation 6.16 is undetermined, and hence, the smallest Euclidean norm is commonly used to obtain a good solution.

$$\begin{bmatrix} \delta \bar{r}_1 \\ \delta t_1 \\ \delta \bar{r}_2 \\ \delta t_2 \\ \dots \\ \delta \bar{r}_n \\ \delta t_n \end{bmatrix} = M^T M M^{T^{-1}} [\delta \Delta \bar{V}_2] \tag{6.17}$$

As explained by Parker and Anderson (2013), each velocity discontinuity is only dependent on the positions and epochs of the three nearest patch points, and hence the majority of the M matrix will be filled with zeros. Moreover, Parker explains that other constraints useful for practical spacecraft missions can be

included in the differential corrector, for instance, limiting the maximum change in position of each patch point during each iteration of Level 2.

As explained before, Level 2 will lead to a trajectory that requires lower ΔVs in the patch points bur that will not be continuous. Hence, several iterations over both levels are required until the discontinuity in position and velocity at each patch point is below a given tolerance.

A sketch of the method can be found in Figure 6.4.

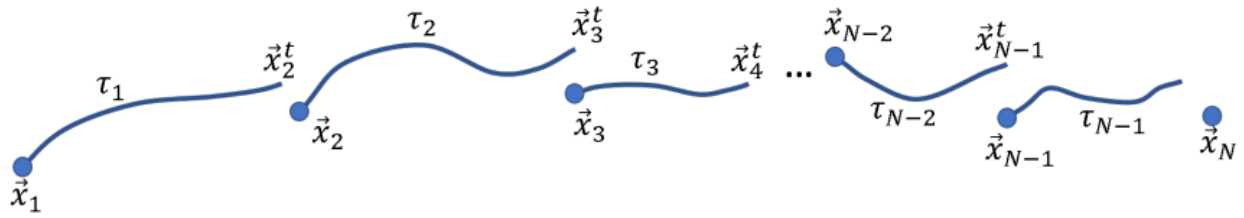


Figure 6.4: Trajectory arcs in a multiple shooting scheme. Retrieved from McCarthy (2019).

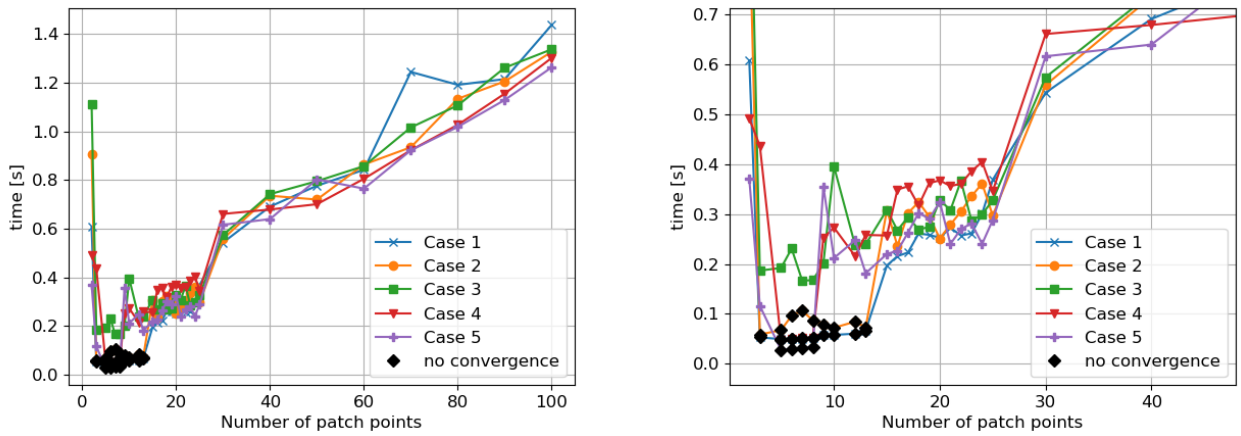
6.2.3 Number of patch points

When working with the multiple-shooting method in a Lambert solver, it is important to analyze the number of patch points to include in the algorithm. This decision depends on various factors, including the desired accuracy level, the available computational resources and the likelihood of convergence. Generally, a larger number of patch points results in a slower convergence, but in a higher possibility of convergence. On the other hand, insufficient patch points can hinder the method's ability to obtain a solution. Therefore, a study has been conducted where the impact of different numbers of patch points, ranging from 2 to 100, has been examined across five distinct cases. These cases deffer from one another by having various starting and ending conditions, as well as different ToFs.

Firstly, it is worth noting that the minimum number of patch points required in the multiple-shooting method is two: the initial and final points, equivalent to a single-shooting procedure. Moreover, it has to be stated that, for this analysis, the precision used in position was 10 m.

Figure 6.5 (original left, zoomed right) showcases four distinct regions. Firstly, using two patch points allows the method to convergence, but at the cost of high computational (CPU) time, taking over 1 s for Case 4. Subsequently, for approximately three to 13 patch points, while the computational time decreases, the solution is not reached in many cases. The following range, spanning from approximately 15 to 25 patch points, shows convergence in all cases taking in between 0.2 to 0.4 s to reach a solution. Lastly, for a higher number of patch points, the computational time required for a solution increases.

Based on the observations from Figure 6.5, it can be concluded that the optimal number of patch points for this problem lies between 15 and 25. Although a slight increasing trend is observed in the mean across all cases, the optimal value varies depending on the specific case. Therefore, it has been determined that 20 is the most appropriate number, finding a balance between avoiding the limit where convergence may not be attained (below 14 points) and maintaining a CPU time within an acceptable range.



(a) (b)

Figure 6.5: Computational time it takes to obtain a Lambert + multiple-shooting solution for a range of patch points.

6.2.4 Analysis of Lambert's solver

One of the main issues with this method is that the initial guess calculated with Lambert's approach is calculated only with the two-body problem, neglecting the influence of the Moon. Consequently, when the initial guess is too far away from the actual trajectory, the multiple-shooting method struggles to find a solution. However, an even bigger problem arises when the initial guess gets too close to the center of the Moon. In this case, the multiple-shooting method encounters difficulties in handling the propagation of the STM, leading to the need of a significant CPU time. Additionally, even if this time is spent, the resulting solution often yields a cost value of 300 km/s for such particle, indicating either the absence of a solution or the failure to satisfy one of the hard constraints, such as the requirement for the final transfer to never go beneath 50 km above the Moon's surface. Therefore, including these particles in the optimization process becomes unnecessary, and a solution should be found to eliminate them before the calculation begins.

The initial idea was to terminate the execution if it were to exceed a certain time limit. However, this approach proved ineffective since there is no practical way to stop a process in Python once it has started, particularly when it is implemented from an external library. Upon careful analysis of the problem, it became apparent that if the satellite approaches a distance to the Moon below a specific threshold, the propagation of the STM requires an excessive amount of time, and the process has a high likelihood of not attaining a useful solution. Therefore, and knowing that propagating only the states can be done rapidly, this threshold has been used to avoid entering in the propagation of the STM in these cases.

The main challenge then is to determine the optimal threshold value. Initially, the radius of the Moon (1737.4 km) was chosen as the threshold, providing satisfactory and quick results during the algorithm's tuning phase. However, subsequent tests were conducted using lower thresholds, for instance 1000, 500, and 100 km, while keeping the optimization parameters fixed. As discussed throughout the tuning process, a crucial aspect to consider is the trade-off between CPU time and achieving better results.

Using the Moon's radius as the threshold results in an initial population being generated in 6.6 seconds, with only one useful particle among them. This useful particle met the criterion of a fitness below 300 km/s, being 248.511 km/s. A threshold of 100 km was also tested, but it required significantly more time. It is important to note that the population size for this test was 75. Within the first 15 individuals, it already found three useful particles, with the lowest fitness being 14.28 km/s. However, it took 136 seconds, which exceeds the acceptable CPU time for a genetic algorithm. Consequently, the optimization

process was terminated.

In the case of a threshold of 1000 km, the initial population yielded 10 useful particles, with the lowest fitness being 7.65 km/s. It took 8.9 seconds to obtain the entire initial population. For the threshold of 500 km, the initial population required 22 seconds to calculate, indicating a significant increase in time. Nevertheless, this duration is still considered acceptable until further analysis. It resulted in 19 useful particles, with the one having the lowest fitness being 5.939 km/s. Both options will be further evaluated by fully running them with a population size of 75 and for 100 generations. The results from these runs will be compared to determine the best choice.

To compare the two options, two optimization processes were conducted, one for each threshold, and the results are presented in Table 6.1. The first line represents the final cost reached by both optimizations, including the CPU time and iteration number N. When using the 500 km threshold, a solution with savings of approximately 154 m/s was achieved. However, it took four times longer to reach that solution compared to the 1000 km threshold.

Nevertheless, before reaching a conclusion, it is important to consider other factors as well, as the optimization parameters including the number of generations and population size have not been tuned yet. Therefore, Table 6.1 also displays the CPU time and the iteration number of the first particle that attains a fitness value starting from the number shown in the fitness column. The 500 km threshold allows the solver to propagate solutions that would otherwise be discarded, resulting in a quicker convergence at the beginning, finding a particle with a fitness lower than 6 km/s 19 times faster. However, as the optimization continues, the time taken to achieve a new best solution is consistently longer, even when found with a lower number of generations. For instance, it took 365 s to find a particle with a fitness lower than 5.6 km/s for a threshold of 1000 km, being the particle number 1555, and 586 s with a threshold of 500 km, being the particle number 1074.

Considering the significant constraint of CPU time, the 1000 km threshold is selected for the rest of the optimization.

500 km			**1000 km**		
fitness [km/s]	time [s]	N	fitness [km/s]	time [s]	N
5.280	12952	7575	5.434	3262	7575
5.9...	10	44	5.9...	190	957
5.7...	291	657	5.7...	216	1056
5.6...	793	1257	5.6...	285	1299
5.5...	586	1050	5.5...	365	1555
5.4...	617	1074	5.4...	562	2124

Table 6.1: Cost of the particle at different CPU times and number of iterations N.

Chapter 7

Optimizer tuning

The PSO algorithm includes various parameters that can be adjusted to achieve its optimal performance for a given problem. In Section 7.1, the different approaches for evaluating its performance are discussed. Additionally, Section 7.2 includes a small study of these parameters, their meaning, and their influence on the performance of the optimizer. Lastly, the key findings and conclusions of the tuning are presented in Section 7.3.

7.1 Performance of an optimization algorithm

Determining the best configuration for a PSO algorithm involves evaluating the performance of the algorithm on the problem at hand. Some common criteria to assess its performance are:

- Robustness. The robustness of the algorithm refers to its ability to find a good and similar solution across multiple runs with different initial conditions. It also include the possibility to distinguish between the global optimum and sub-optima. To evaluate it, the algorithm can be run multiple times with different random seed values, and compare the results obtained using different parameter configurations.
- Fitness values. The fitness value of the optimum found measures of how well the PSO algorithm has optimized the problem being evaluated. Hence, lower values indicate better performance.
- Convergence rates. The convergence rate measures how quickly the algorithm converges to the optimal solution. Faster convergence is generally preferred. To evaluate it, the fitness values can be plotted against the number of iterations for different parameter configurations, or the actual run time can be evaluated.
- Scalability. The scalability of the algorithm refers to how well it performs on larger problem sizes. The algorithm can be tested on larger instances of the problem, and the performance for different parameter configurations can be compared.

Ultimately, determining the best configuration for a PSO algorithm involves balancing the trade-off between performance and computational resources. The best configuration is the one that gives the best performance while using a reasonable amount of computational resources. It is also important to consider the specific requirements and constraints of the problem at hand.

7.2 PSO parameters

There are numerous parameters to adjust in PSO. However, the focus will be primarily on the population size, the number of generations, the inertia weight or construction factor (ω), the social component (η_1) and the cognitive component (η_2).

7.2.1 Population size

The population size is an important parameter in PSO as it affects the performance and convergence of the algorithm. In general, increasing the population size can lead to better global exploration, but at the cost of increased computational resources and longer convergence times. Similarly, reducing the population size can lead to faster convergence but with a risk of getting stuck in local optima.

An appropriate population size can be determined by studying the problem at hand and evaluating the convergence rate and the computational cost. Some guidelines suggest that the population size should be proportional to the dimensionality of the problem, while others recommend using a population size between 20 and 100 particles, as explained by Bratton et al. (2007).

It is important to note that the optimal population size may vary depending on the specific problem being solved, and it may require some trial and error to find the best population size for a given problem.

7.2.2 Number of generations

The number of generations is another important parameter in evolutionary algorithms in general, and in PSO in particular. The number of generations refers to the number of iterations of the algorithm. In PSO, each generation involves updating the positions and velocities of the particles based on their previous positions and velocities, as well as the ones from their neighbors.

The number of generations affects the performance of the algorithm, as a larger number of generations may lead to better convergence and a higher likelihood of finding the global optimum, but at the cost of increased computational resources and longer run times.

The optimal number of generations depends on the complexity of the problem, the population size, and the convergence criteria. In general, the number of generations should be sufficient to allow the algorithm to explore the search space adequately, but not too large to prevent unnecessary computations.

Determining the optimal number of generations may require experimentation and fine-tuning. One common approach is to monitor the convergence of the algorithm and stop the algorithm when the convergence criteria is met or when the improvement in the fitness value is below a certain threshold. Nevertheless, these options are not included in the PSO algorithm in PyGMO, and hence tuning is required.

7.2.3 Inertia weight, social and cognitive components

This section will analyze the other three key parameters in PSO: the inertia weight, and the social and cognitive components.

In PSO, each particle moves around in a high-dimensional search space to find the optimal solution. Their movement is influenced by two factors: their current velocity and their current position. The parameters stated above allow the particles to communicate and share information about the search space through the velocity update equation, found in Section 4.3.3. To understand this equation, it is important to note that each particle maintains a personal best position (Pbest), which is the best solution it has found so far, and a global best position (Gbest), which is the best solution found by any particle in the swarm, explained

by Zhang et al. (2005). The velocity equation involves three terms: the particle's previous velocity, its distance to its personal best position, and its distance to the swarm's global best position.

The inertia weight ω determines the weight of the particle's previous velocity term, hence controlling how much the particle's velocity is affected by its past behavior, explained by Innocente et al. (2011).

A high inertia weight allows the particle to maintain a high level of momentum and explore the search space more globally, while a low inertia weight allows the particle to focus more on the local search. Therefore, the inertia weight plays a crucial role in balancing global and local search and determining the convergence speed and performance of the algorithm.

Sengupta et al. (2018) explains that an inertia weight equal or greater than one implies that the swarm velocity will increase over time towards the maximum velocity (V_{max}). If this happens, the particles will not be able to change their heading to fall back towards the promising regions, and the swarm will eventually diverge. On the other hand, an inertia weight less than one reduces the velocity of the swarm until it eventually becomes a function of only the acceleration factors. For these reasons, ω, in PyGMO, can only range from 0 to 1.

The optimal value of the inertia weight depends on the problem being solved and the other parameters of the algorithm. The most commonly used values for the inertia weight, as given by Eberhart et al. (2000), are between 0.4 and 0.9. Moreover, the value of the inertia weight does not have to be kept constant, and it is typically decreased over time to reduce the exploration rate and focus more on local search. Nevertheless, this is not an option in PyGMO.

The social component η_1 refers to the shared experience of the swarm, and hence is responsible for updating the term related to the influence of the swarm's global best position on the particle's behavior.

It can be implemented in different ways depending on the topology of the swarm, which determines the way the particles are connected and communicate with each other, essentially describing a subset of particles with whom a particle can initiate information exchange. The choice of the topology affects the degree of information sharing between the particles, and the speed of convergence of the algorithm. Some common topologies used in PSO, as explained and defined by Kennedy et al. (2002), include the ring topology (lbest), the fully connected topology (gbest), the Von Neumann neighborhood structure, and the adaptive random. They are all available as options in PyGMO, but the default one and the one used is lbest, which allows each individual to be influenced by some smaller number of adjacent members of the population array. Sengupta et al. (2018) explains that this structure leads to multiple best particles, one in each neighborhood, and consequently the velocity update equation of the PSO has multiple social attractors. This sometimes leads to slower convergence but significantly increases the chance of finding global optima. The topology also depends on the topology parameter, which defines the number of neighbours to consider when updating its position and velocity. The value chosen in PyGMO is 4.

A large η_1 can result in a stronger exploitation of the current best solution, which can be useful when the algorithm is close to convergence and is searching for fine-tuned solutions. Moreover, it can be useful in problems where the search space is well-behaved and the optimal solution is likely to be found near the current best solution. Nevertheless, it can lead to premature convergence, where the algorithm gets stuck in a sub-optimal solution due to an over-reliance on the global best position so far. On the other hand, a smaller social learning parameter value can be useful in problems where the search space is complex and the optimal solution is likely to be found far from the current best solution. In such cases, a smaller value allows the particles to explore a wider range of the search space and potentially find better solutions.

The cognitive component η_2 allows each particle to update its position based on its own historical information, hence permitting it to explore the search space around its best known solution.

A large η_2 value means that the particle is more strongly influenced by its personal best position, which can be useful in problems where the particle's individual experience is informative. Nevertheless, a large η_2 value can also lead to a slower convergence rate, as the particle may be reluctant to explore the search

space beyond its own best position. On the other hand, a smaller η_2 value can lead to a faster convergence rate, but may also result in premature convergence, where the particle gets stuck in a sub-optimal solution. A smaller cognitive learning parameter can be useful in problems where the search space is complex and requires a more explorative approach.

As explained by Sengupta et al. (2018), by combining the cognitive component with the social component, PSO is able to balance exploration and exploitation of the search space. They are typically kept constant, and it was empirically found that the optimum pair seemed to be 2.05 for each, which coincides with the default value in PyGMO.

7.2.4 Additional parameters

The PSO algorithm in PyGMO involves additional parameters that could also be tuned. Nevertheless, they have a lower influence on the performance of the optimizer and hence their values will be kept as the default ones from PyGMO. These parameters are:

- **The maximum velocity:** It determines the range of movement for each particle in the search space. A large V_{max} introduces the possibility of global exploration, hence allowing particles to move faster towards the optimal solution. Nevertheless, if the value is too high, the particles may overshoot the optimal solution and oscillate around it without converging. On the other hand, a small value implies a local, intensive search, preventing premature convergence. However, if the value is too low, the particles may not be able to move far enough to explore the search space effectively. Its value can range from 0 to 1, as it is expressed as a fraction of the search space size, and the default one is set to 0.5.
- **The algorithmic variant:** It refers to the specific PSO algorithm used to solve a particular problem. There are six options available in PyGMO: Canonical, Same social and cognitive rand, Same rand for all components, Only one rand, Canonical (with constriction factor), and Fully Informed (FIPS). The one that will be used is the Canonical variant with constriction factor, explained by Freitas et al. (2020) and initially introduced by Clerc (1999).

7.3 Tuning

Finding the best values for the optimization parameters in PSO involves a process known as parameter tuning or hyperparameter optimization, which involves searching the space of possible parameter values to find the combination that produces the best performance for the problem at hand.

There are many methods available to tune an algorithm. For instance, using a grid search, which involves specifying a set of discrete values of the different optimization parameters, and evaluating the performance of the algorithm for each combination of values. There is also the possibility to use more complicated methods, such as using Bayesian optimization, explained by Frazier (2018), which involves applying a probabilistic model to approximate the relationship between the parameters, or using another metaheuristic algorithm, such as simulated annealing or tabu search, explained by Abdel-Basset et al. (2018), to search the space of possible parameter values. Additionally, automated tuning methods such as F-Race, Revac, ParamILS and SPO exist, which are explained and compared by Montero et al. (2014).

As previously mentioned, five parameters will be tuned for this project, and the process to follow will be explained hereunder. Firstly, each parameter will be studied separately, and the performance of the algorithm will be evaluated for discrete values assigned to each parameter. The evaluation will involve analyzing the fitness value associated with each configuration. In particular, to ensure robustness and reliability, each configuration will be executed using five different seeds, which relate to the initial

population generated. The seeds selected for this purpose are: 10, 20, 30, 40 and 50, and the constant values for the rest of the parameters are:

- Population size: 50
- Number of generations: 50
- Inertia weight: 0.7298
- Social and cognitive component: 1.5

7.3.1 Inertia weight, social and cognitive component

During the tuning process, four different options of **inertia weight** around the default value have been examined. The values considered are: 0.5, 0.6, 0.7298, and 0.85.

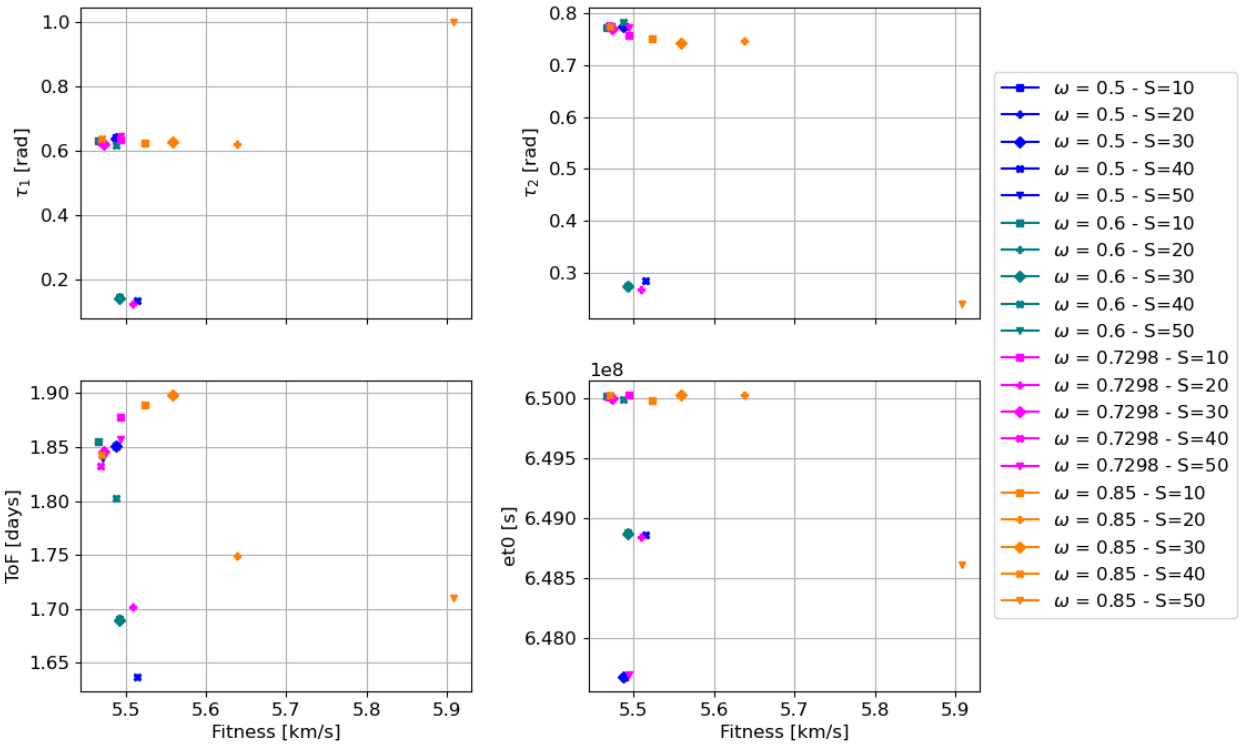


Figure 7.1: Optimized values of the design vector with respect to the fitness value obtained for each seed and various inertia weights.

The data displayed in Figure 7.1 indicate that an inertia weight value of 0.7298 not only yields consistent results across different seed numbers but also corresponds to the solutions with the lowest fitness values. This makes it the optimal parameter value in terms of both the fitness obtained and the robustness of the algorithm. To study the convergence rates, the CPU time spent for each run has been plotted in Figure 7.4a. It can be observed that a value of ω of 0.6 may result in longer computational times, whereas an ω value of 0.85 in lower times, but with the drawback of more scattered solutions. For the remaining values, the time duration ranges from approximately 15 minutes to 20 minutes, varying depending on the seed,

finding no clear advantage among them. Therefore, it has been determined that an inertia weight value of 0.7298 will be used for all optimization cases in this problem.

Figure 7.2 shows a similar analysis but for the **social component**. The examined values in this case are 1.5, 2.05, and 2.5, which includes PyGMO's default value and two values around it.

In this problem, there is no clear pattern observed when considering larger or smaller values of η_1. Surprisingly, the highest fitness values are obtained when η_1 is set to 2.05, but it is also the value that yields the lowest fitness.

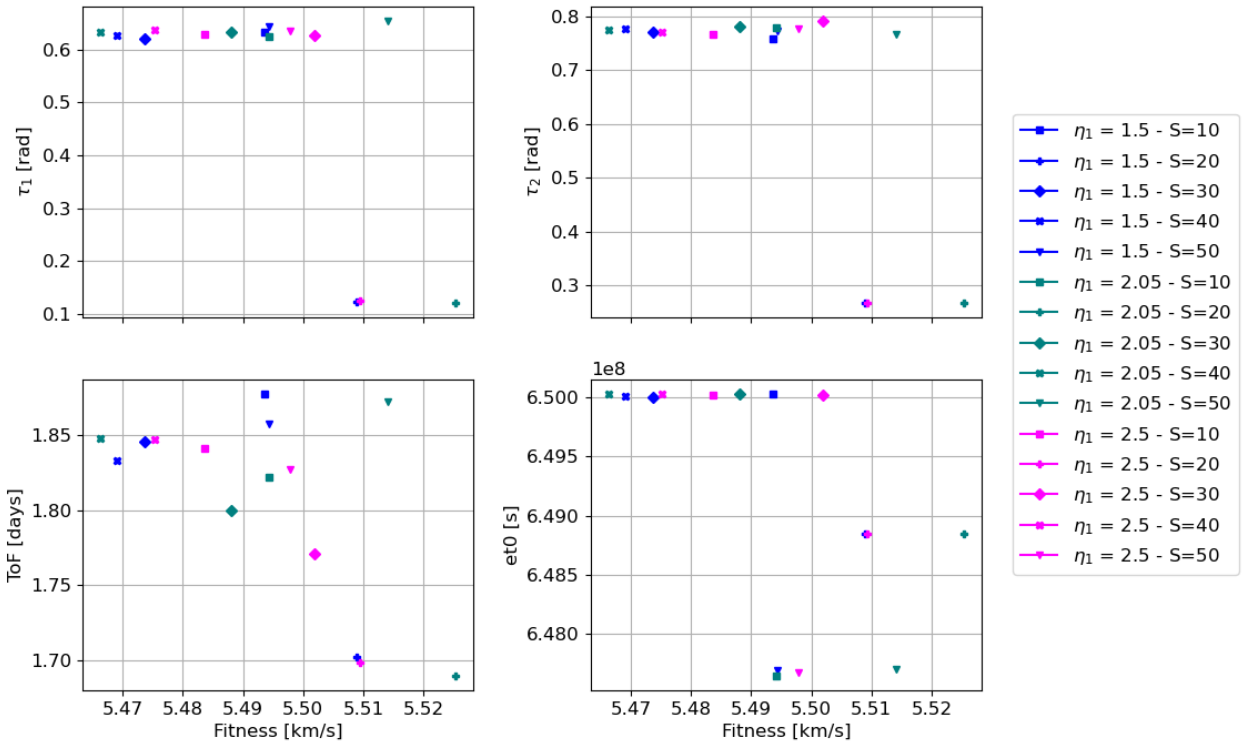


Figure 7.2: Optimized values of the design vector with respect to the fitness value obtained for each seed and various social components.

Analyzing the impact of a parameter on CPU time from Figure 7.4 is also important in this case, but no clear advantage is found for any value. As a result, it has been decided that a value of 1.5 will be set as the social component for this optimization.

Lastly, the parameter related to the **cognitive component** of the PSO has also been analyzed, considering values of 1.5, 2.05 and 2.5. In this case, it can be seen in Figure 7.3 that a larger value of η_2 leads to more scattered solutions in the design variables and their corresponding fitness values. Similar to the social component, a lower value of η_2 gives better results, adding robustness and reaching solutions with lower fitness values. Analyzing the CPU time required, Figure 7.4c shows similar conclusions as for η_1: a higher value of η_2 results in a faster execution time. However, the maximum difference is 3 minutes, which is not considered significant. The conclusion is that a cognitive component value of 1.5 will be used as the default setting for the optimization process.

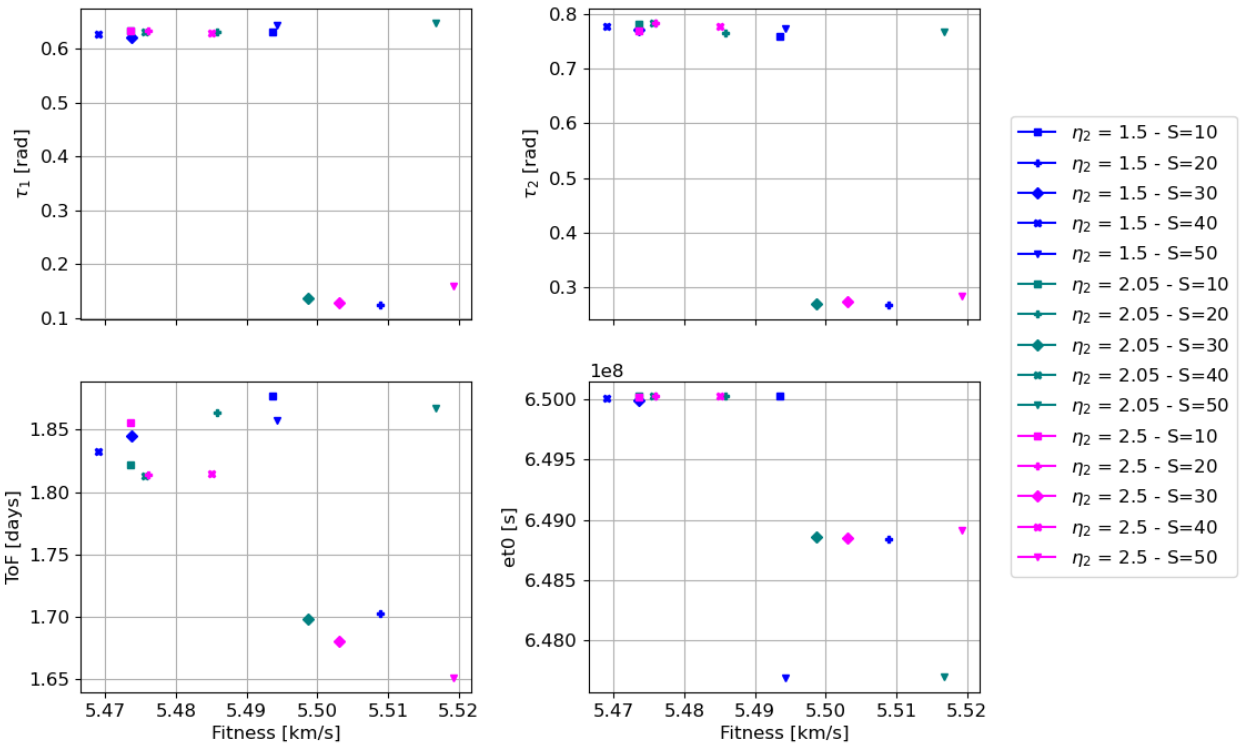


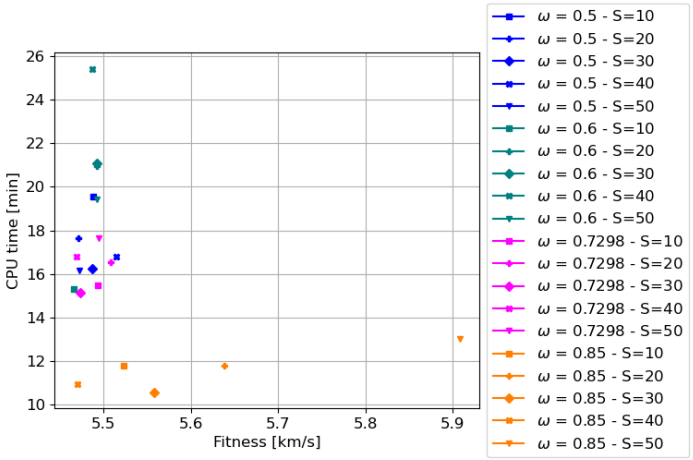
Figure 7.3: Optimized values of the design vector with respect to the fitness value obtained for each seed and various cognitive components.

Lastly, it is important to highlight that conducting a full grid search to explore the combination of all three parameters would be desirable. Nevertheless, such analysis would be time-consuming and the insight obtained from the current individual search is sufficient to reach a conclusion. Moreover, this analysis allows to understand which parameter has a greater impact on the final solution and its fitness value. The chosen value of the inertia weight has a higher effect on the solution obtained. This is seen because, for all four design variables, the range of the solutions for different seeds is larger for the inertia weight compared to the social and cognitive components.

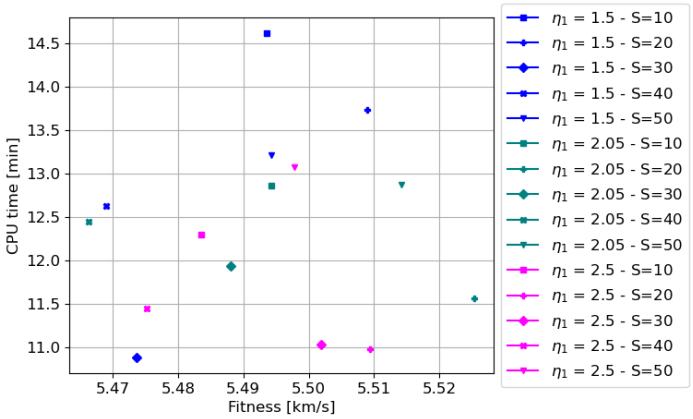
7.3.2 Population size and number of generations

During the individual analysis of the adequate population size and number of generations, it was determined that their combination plays a crucial role on the quality of the results, the fitness values obtained, and the computational time for each run. Consequently, a comprehensive analysis was conducted considering all possible combinations. For this study, ω, η_1 and η_2 have been set as the optimal ones, which are 0.7298, 1.5 and 1.5 respectively.

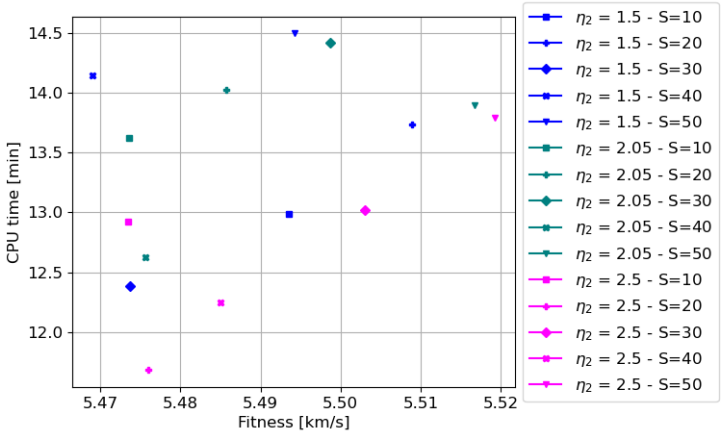
The population size values and the number of generations values considered were 25, 50, 75 and 100. This results in a total of 80 different options, as 5 seeds are being considered for each combination.



(a)



(b)



(c)

Figure 7.4: Computational time with respect to the fitness value obtained for each seed and various PSO parameters: (a) Inertia weight, (b) Social component, (c) Cognitive component

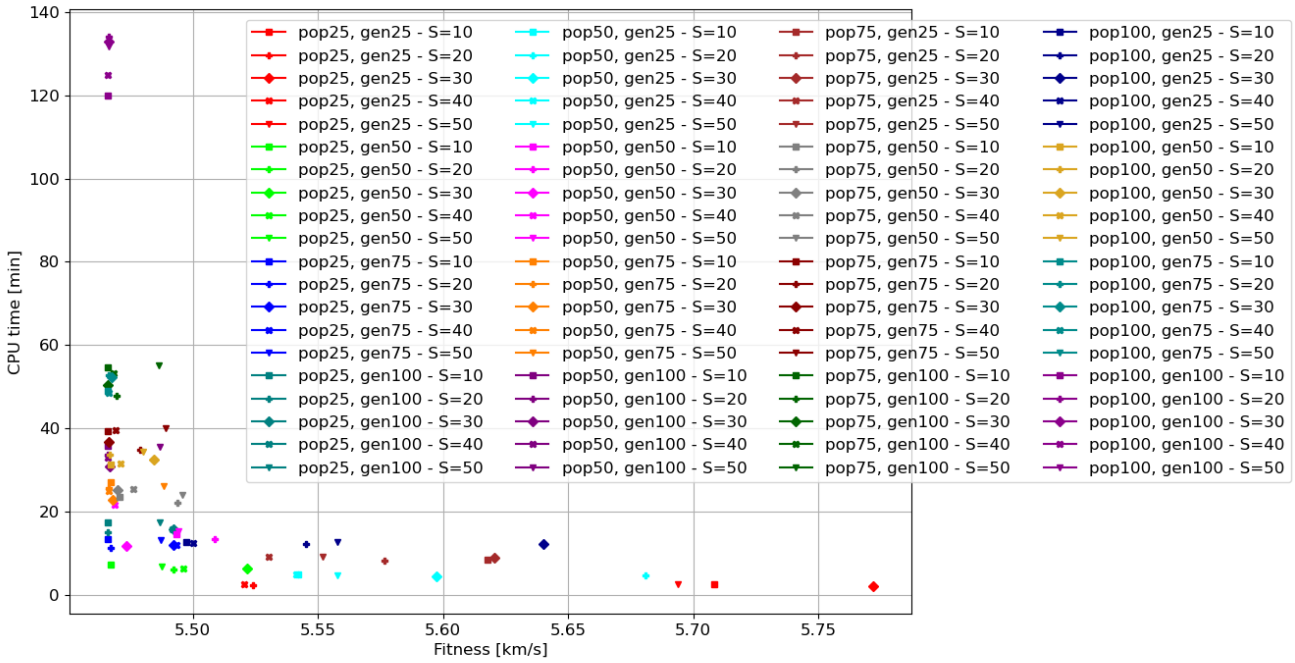


Figure 7.5: Computational time with respect to the fitness value obtained for each seed and various population sizes and number of generations.

Figure 7.5 shows the computational time required for the propagation of each parameter combination, along with the corresponding fitness value obtained. Based on the results, several categories can be found.

The first category includes combinations that take less than 15 minutes to execute, but their solutions are spread out in between 5.466 and 5.773 km/s. It can be highlighted that these values physically correspond to the sum of the ΔVs required to go from an Earth-bounded orbit to a Moon-bounded orbit, hence obtaining a difference of 300 m/s. Although these combinations offer faster solutions, the robustness of the solver is compromised as the results are more scattered. The second category includes combinations that require a CPU time in between 15 and 25 minutes. These solutions show less dispersion, but still give differences between results of around 35 m/s. The third category achieves CPU times between 25 and 60 minutes. In this range, the solutions correspond to fitness values between 5.466 and 5.485 km/s. Even though the third category shows a spread of 19 m/s, which is around 15 m/s lower than the second category, the CPU time required triples. Moreover, there is no clear benefit in terms of the best solution found. These combinations give a good balance between robustness and computational time, particularly the ones with a population size of 25 and a number of generations of 200 (represented in teal) and a population size of 50 with 75 generations (orange), which yield to good solutions within approximately 20 minutes.

The fourth and final category includes combinations with CPU times higher than 120 minutes. This combination gives the best results, with all seeds consistently achieving low fitness values. However, it is worth noting that the run time for combinations with a population size of 100 and a number of generations of 100 is double that of combinations with a population size of 100 and 75 generations. The first gives solutions between 5.4660 and 5.4666 km/s, and the latter in between 5.4662 and 5.4667 km/s. The difference in the best result is less than 3 m/s, and for the worst it is 1 m/s.

The main conclusion obtained from this study is that, even though a larger population and more generations yield slightly better solutions in fitness and robustness, the gains obtained are minimal compared to the

substantial increase in computational cost involved.

To understand the solutions generated by various combinations within the third category, which has been selected as the most appropriate, the parameters of the optimal solution obtained for each seed using five different options are presented in Figure 7.6. Within this context, two families of solutions that are close to optimal are observed when considering τ_1, τ_2, and the time of flight, while three solutions are identified for the start epoch (et0). Although seed 50 shows a different solution for all possibilities, this figure clearly illustrates that the solution achieved with a population size of 50 and a number of generations of 75 (highlighted in orange) yields results with less variability.

As a result, a population size of 50 with 75 generations has been determined to be the optimal compromise between the quality of solutions obtained and the computational time required.

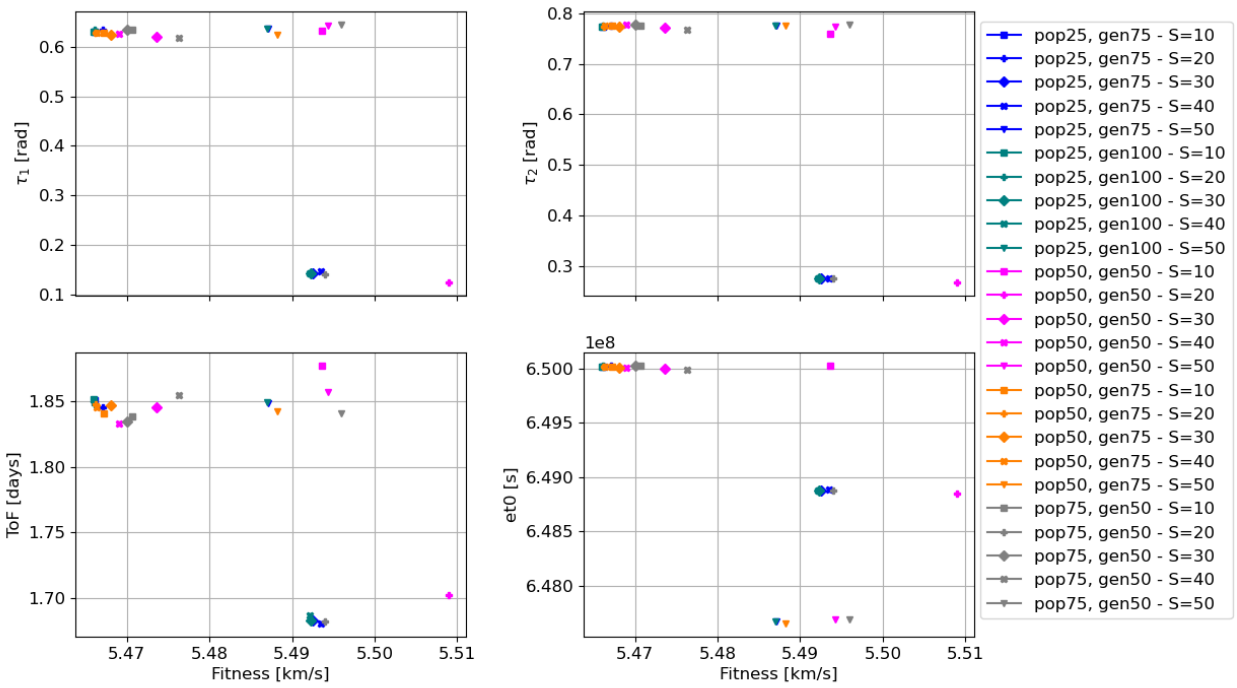


Figure 7.6: Design parameters with respect to the fitness value obtained for each seed and various population sizes and number of generations.

7.3.3 Final configuration for the PSO algorithm

After the analysis conduced before, the parameters for the optimization will be set to:

- Population size: 50
- Number of generations: 75
- Inertia weight: 0.7298
- Social component: 1.5
- Cognitive component: 1.5
- Maximum normalized velocity: 0.5

- Algorithm variant: Canonical with constriction factor (5)
- Neighbour type: lbest (2)
- Topology parameter: 4

Chapter 8

Output files

After formulating the problem and creating solutions for each particle, as well as fine-tuning the optimizer, the PSO algorithm becomes ready for application. The resulting solution, including maneuver epochs and components, can serve as an initial guess for the following optimization using a gradient-based method, detailed in Part III, or as a final solution with the given dynamics model.

The software generates two output files. The first file, shown in Figure 8.1, contains general optimization data in its header, followed by 7 columns with the parameters of each particle: τ_1 (tau1), τ_2 (tau2), ToF in seconds, et0 in ephemeris seconds, fitness value in km/s, runtime for Lambert's + multiple-shooting method in seconds, and cumulative runtime. With a population size of 50 and 75 generations, this file contains a total of 3800 particles. It concludes with a notification that the optimization is complete, details of the champion found, and information about the optimization's phase durations, shown in Figure 8.2.

```
CREATION_DATE        = 2023-07-26T03:24:50.525109

et0 = 775203315.0557911    which is: 2024 JUL 23 00:00:00.000
Earth orbit: a = 6621.008366666666  e = 0.0  i = 28.4  w = 0.0  RAAN = 0.0
Moon orbit: a = 1837.4000000000003  e = 0.0  i = 85  w = 0.0  RAAN = 0.0

Spherical Harmonics = True

Boundaries of the design space:
tof = [8640.0, 518400.0]
tau1 = [0.0, 1.0]
tau2 = [0.0, 1.0]
et0 = [773668869.1834997, 776260869.1834997]

Optimization parameters:
For population generator: pop_size = 50  seed = 20
For optimizer: gen = 75  omega = 0.7298  eta1 = 1.5  eta2 = 1.5  max_vel = 0.5  variant = 5  neighb_type = 2  neighb_param = 4  memory = False  seed = 20

nb_pts = 15  nb_revs = 0

tau1                    tau2                    ToF                          et0                          Fitness                  time      run time

0.9544122740388181 2030  0.9522779497531080 8834  461006.99068448640173301101  775279768.42225062847137451172  300.00000000000000000000  0.159  0.159
0.9504516503521608 3727  0.6728656567583514 1587  335067.63626330881379544735  773775590.45226526260375976562  300.00000000000000000000  0.132  0.291
0.8165542975053773 3558  0.7825114124622932 7653  437640.83291784796165302396  775574682.11646938323974609375  300.00000000000000000000  0.162  0.453
0.8662023185403848 7462  0.3155242143606871 7187  251993.15416211818228475749  776018605.18073678016662597656  300.00000000000000000000  0.140  0.593
0.5754899856986666 0624  0.4745243846530137 1995  9338.86694937148058670573    774374026.05528521537780761719  182.17848300321375631938  0.132  0.725
0.9871143894153132 6474  0.9297162216533177 9808  145458.92974698523175902665  774148188.98326921463012695312  300.00000000000000000000  0.137  0.862
0.4473428305919238 1238  0.2529740938058246 4022  239787.61310800994397141039  775934787.17538368701934814453  300.00000000000000000000  0.260  1.122
0.3194599488046812 0414  0.5759382433442890 0258  155890.13519454267225228250  775763299.76625931262969970703  117.15596094122547299321  1.467  2.589
0.8349116810288570 4450  0.7471319452792977 1560  131739.28082221676595509052  774812744.41206085681915283203  15.333946516595888880932  0.611  3.200
0.3833492476285065 2868  0.4180541136915862 3769  54034.993544699682388454 68  775651677.46262764930725097656  121.05330711057301584788  0.404  3.604
0.0490421477140509 0989  0.6740416619737119 5857  110223.96014188062690664083  773720243.50194275379180908203  9.00072247826638793811  0.333  3.937
```

Figure 8.1: Example of header and initial data of the first output file from PSO.

```
0.72476553868534832059  0.71070610001114487009  148830.29456514390767551959     775861984.77317810058593750000  110.13264397395087712539    0.442      1429.241
0.63824711857263738413  0.78844087217145641944  143773.11031492211623117328     775199790.09273552894592285156  4.80236364741183763272      0.681      1429.922
0.97431550922411669369  0.39389747197676111545  144705.84362877396051771939     775375744.99690902233123779297  300.00000000000000000000    0.103      1430.025
0.63213048547783490250  0.78910136437567479639  144646.85979240760207176208     775209354.34715974330902099609  4.78314923689249305028      0.847      1430.872
0.63232706396349258604  0.78956478503993032803  142989.15971967074437998235     775209882.56900823116302490234  4.80570405190700000730      0.797      1431.669

Optimization finished, champion found

0.62110670951902113845      0.77553371907452073319      146850.74397220811806619167      775186487.97173213958740234375      4.77657056021551618130

time for initial population generation [s]: 10.919011116027832
time for algorithm generation [s]: 0.0
time for algorithm evolution [s]: 1420.757426738739
total time [s]: 1431.6764378547668
which is [min]: 23.861273964246113
```

Figure 8.2: Example of result and final data of the first output file from PSO.

The data from this file represents the champion particle, used as an initial guess for the gradient-based algorithm. However, these values lack straightforward physical interpretation. Therefore, another output file is generated upon optimization completion, as seen in Figure 8.3. This file also contains optimization parameters in its header, along with champion particle data. Furthermore, it provides the important times of the optimization in both ephemeris seconds and UTC time.

In addition, Figure 8.4 displays state values at various epochs in the EME2000 frame in m and m/s, while also presenting the required maneuvers in EME2000 coordinates.

```
CREATION_DATE        = 2023-07-26T03_48_42

Earth orbit: a = 6621.008366666666  e = 0.0  i = 28.4  w = 0.0  RAAN = 0.0
Moon orbit: a = 1837.4000000000003  e = 0.0  i = 85.0  w = 0.0  RAAN = 0.0

Spherical Harmonics = True
n_max = 10         m_max = 10

nb_pts = 15  nb_revs = 0  precision = 1e-08

Lambert initial parameters:
tau1 = 0.6211067095190211  tau2 = 0.7755337190745207  et0 = 775186487.9717321  tof = 146850.74397220812


Important times:

Initial time [s]: 775186487.9717321
which is: 2024 JUL 25 13:33:38.788300

Time in the first orbit [s]: 3330.1470390022973

Time of first maneuver [s]: 775189818.1187712
which is: 2024 JUL 25 14:29:08.935341

Time of flight [s]: 146850.74397220812

Time of second maneuver [s]: 775336668.8627434
which is 2024 JUL 27 07:16:39.679357

Time in the second orbit[s]: 5481.053356472648

Last time [s]: 775336668.8627434
which is 2024 JUL 27 07:16:39.679357
```

Figure 8.3: Example of header and important times of the second output file from PSO.

```
States:

Initial state w.r.t Earth (state0) [m, m/s]:
6621008.36666666623204946517944335937 5      0.000000000000000000000000000000      0.000000000000000000000000000000
      0.000000000000000000000000000000   6825.210001244500745087862014770508   3690.377281010701153718400746583939

Start state of LTO without maneuver (state1 prim) [m, m/s]:
-4730533.180221087299287319183349609375   -4051808.840837975032627582550048828125   -2200892.750008819159120321273803710938
    5422.803593095291034842375665903091      -4907.852273786759724316652864217758      -2638.555348762242374505149200558662

 Start state of LTO (state1) [m, m/s]:
-4730533.180221087299287319183349609375   -4051808.840837975032627582550048828125   -2200892.750008819159120321273803710938
   7617.011836001038318499922752380371      -6705.971501017125774524174630641937      -4350.660150019835782586596906185150

End state of LTO (state2) [m, m/s]:
338611879.431401371955871582031250000000   133521524.579299271106719970703125000000   65358357.495668187737464904785156250000
     2480.862354483994295151205733418465         503.882868713771983948390698060393       1400.035233888016136916121467947960

End state of LTO without maneuver (state2 prim) [m, m/s]:
338611879.431401371955871582031250000000   133521524.579299271106719970703125000000   65358357.495668187737464904785156250000
     1214.802318305517019325634464621544         784.986302762645209440961480140686        722.735285577486592956120148301125


Maneuvers:

First maneuver [km/s]:
2.194208242905746786277632054407    -1.798119227230365879677265184000    -1.712104801257593678087687294465

Second maneuver [km/s]:
-1.266060036178477288260069144599    0.281103434048873279671454383788    -0.677299948310529642547805906361
```

Figure 8.4: Example of states and maneuvers of the second output file from PSO.

In Part IV, various test cases will illustrate and present the solutions derived through this PSO process. These files should be taken as examples, but they are actual output files from TC1, seed 20.

Part III

Gradient-based optimization

Chapter 9

Problem formulation

The second part of the project involves employing a gradient-based algorithm in Fortran called OPT-EM to refine the results obtained from the genetic algorithm (the results will be provided in Part IV). Additionally, this phase of the project will introduce new perturbations to get a final solution that defines a trajectory under a full dynamics model. The formulation of the problem for gradient-based optimization will be explained throughout this chapter.

The main difference compared to the previous optimization is that the initial guess is now known, and the local optimizer will refine the solution based on it. In general terms, the method will involve propagating the obtained results, which will be explained in Chapter 10, to determine the actual orbit around the Moon that the satellite reaches under the influence of the new dynamics model, which may include additional perturbations. By using the desired final Keplerian elements as constraints, the software will correct for discrepancies and seek a solution that not only reduces the needed ΔV as given by the PSO but also ensures the satellite reaches the desired orbit under a complete dynamics model. The chosen local optimizer is explained in Section 9.3, and details on how the constraints will be handled are presented in Section 9.4.

It is important to note that this part of the project, apart from using the initial guess, is entirely independent of the previous part. Therefore, it can be employed even if the initial guess is obtained through alternative methods.

9.1 Fitness function

The fitness value for this section will be the same as the one used in the PSO algorithm, as described in Section 4.1. It is calculated as the sum of the magnitudes of the maneuvers required to reach the desired Moon-bounded orbit.

$$\text{Fitness} = |\Delta\vec{V}_1| + |\Delta\vec{V}_2| \tag{9.1}$$

As mentioned in the PSO optimization, the ToF could also be considered including it with a weight factor. However, it has not been included because the refinement process does not heavily impact the parameters, and consequently, the change in the time of flight is minimal compared to the reduction in ΔV.

9.2 Design variables

The optimization process involves propagating the initial state including maneuvers, hence the design variables previously formulated for PSO need to be modified. To achieve this, an initial state associated

with a specific time is necessary. In this optimization, the actual initial epoch will be treated as a constant rather than a design parameter. From it, the the first maneuver will be fired after a time t_1, and the second after a ToF.

Consequently, there will be a total of eight parameters to be optimized:

- Time in the first orbit: t_1
- Three velocity components of the first maneuver in an EME2000 reference frame: ΔV_{1x}, ΔV_{1y}, and ΔV_{1z}
- Time of flight: ToF
- Three velocity components of the second maneuver in an EME2000 reference frame: ΔV_{2x}, ΔV_{2y}, and ΔV_{2z}

Past experience and knowledge have demonstrated that the optimizer is capable in effectively optimizing this number of parameters.

9.2.1 Including the initial epoch as a design variable

Initially, the initial epoch will be maintained as a constant in the formulation. However, there is an option to include it as an additional design variable to determine the optimal date for the maneuver, considering the new dynamical model. This option has been implemented using a flag. Nevertheless, it requires careful examination due to potential issues it may introduce.

9.3 Gradient-based optimizer

For the local gradient-based optimization, the SLLSQP optimizer has been chosen.

SLLSQP is a Sequential Linear Least SQuares Programming algorithm designed to solve general non-linear optimization problems. The optimizer approximates the non-linear function around the current guess using linearized least squares, and then finds the direction of greatest descent with the calculated gradients and a system of jumps, dividing the problem in many sub-problems, as explained by Horn et al. (1983).

This optimizer (in Fortran) has been greatly used by the DLR team, and therefore is well known. In addition, it has proven to work correctly with similar problems, which is the reason why it will be the optimizer chosen for this part of the project. Nevertheless, is important to take into account that the function to optimize and the constraints need to be continuous, and that it is a local optimizer, meaning that it may lead to a local minimum if the initial guess is not chosen correctly. This optimizer has been used before by the author in a previous project.

A diagram of the way the optimization works is given in Figure 9.1. Firstly, the initial guess for the design variables (also called decision vector) P is required. From there, FUNDER is called, where the scaled decision vector (P_S), the fitness (V), and the constraint vector (G) are calculated, and are taken as inputs for the optimizer.

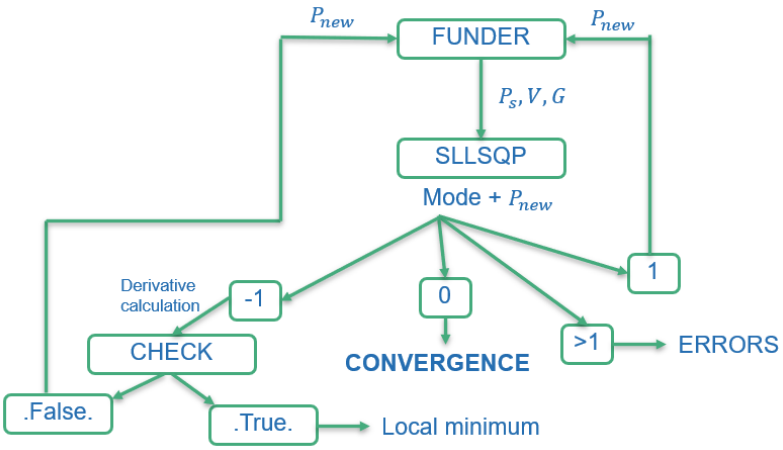


Figure 9.1: SLLSQP optimization diagram.

The SLLSQP optimizer will give the new decision vector (P_{new}) and the so-called mode. The mode decides the next steps. With mode 1, the given P_{new} will be used for the new calculations and as inputs for SLLSQP. With mode -1, the derivatives will be calculated again for a gradient evaluation where checks are performed to see if the change is big enough to continue the optimization or if a local minimum has been found. If not, the process starts again. If the mode is larger than 1, the optimization will be stopped because an error has occurred. The most common errors are: mode = 6, meaning that a positive directional derivative has been found, and mode = 8, meaning that the iteration count has been exceeded. If one of the error modes appears, it is very useful to check the optimization parameters, like the accuracy or the weights. Lastly, the optimization will be finished when the output mode is 0, meaning that convergence has been reached.

9.4 Constraints

The SLLSQP optimizer can handle equality and inequality constraints. The inequality constraints have as a general formulation: $F(\bar{z}) \leq \bar{b}$. Nevertheless, when adding them as inputs to the optimizer, they need to be expressed as $\bar{b} - F(\bar{z}) \geq 0$. For the case of the equality constraints, the input should be expressed as $\bar{b} - F(\bar{z}) = 0$.

The main goal of this optimization is to reach a desired orbit, and this critical requirement will be treated as a constraint. Constraints can be classified into two types: equality and inequality constraints. However, to avoid imposing overly strict limitations, all constraints will be considered as inequality constraints, and their bounds can be defined by the user.

For the comparison with the objective orbit, we need to consider the Keplerian elements such as the semi-major axis a, eccentricity e, inclination i, right ascension of the ascending node Ω, only applicable to inclined orbits, and argument of pericenter ω, only applicable to non-circular orbits.

The user will specify the exact value P_0 he/she aims to achieve in the input file, along with the acceptable limit for solutions. The limit will be expressed as a Δ value, defining upper and lower bounds so that a parameter P must satisfy $P_0 - \Delta < P < P_0 + \Delta$. Therefore, two constraints will be included for each Keplerian element under consideration.

This general approach of writing the constraints requires some adjustments because, in certain cases, the defined limits may not align with physical requirements.

Semi-major axis

To ensure the satellite remains at a safe distance from the Moon's surface, the lower bound for the desired semi-major axis is at 50 km above the Moon's surface, whose value is taken as 1738.0 km from DLR's global library. So the lower limit for the semi-major axis is the value of 1778.0 km.

Eccentricity

If the lower bound of the eccentricity is negative, it will be set to zero, as a negative eccentricity does not have a physical meaning.

Inclination, Ω, ω

For the angular parameters, coterminal angles need to be accounted for, for instance the fact that 360° and 0°, 10° and 370°, and -10° and 350° represent the same physical angle but have different numerical values. To address this, there are three possible cases based on the objective value A, the objective value plus or minus the permitted variance Δ, and the actual value obtained from the propagation X. These cases can be visualized in Figure 9.2.

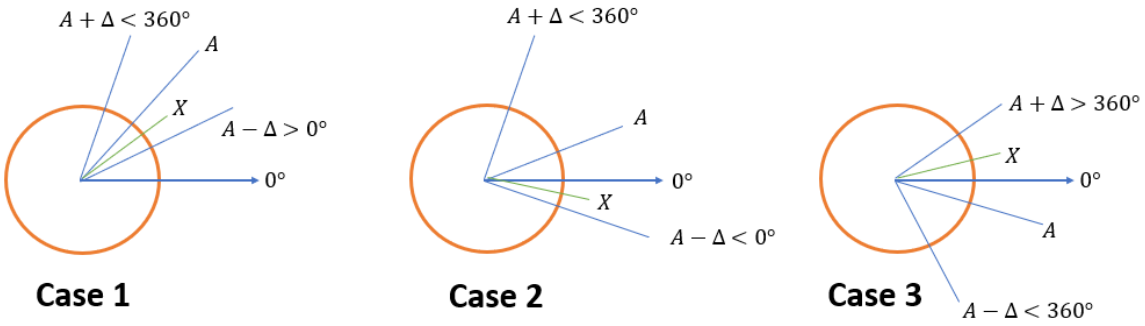


Figure 9.2: Cases for angular constraints in the SLLSQP optimizer. A is the objective value, Δ is the tolerance and X is the actual value.

The general inequalities are as follows:

$$\begin{aligned} X &< A + \Delta \\ X &> A - \Delta \end{aligned} \tag{9.2}$$

In Case 2 and Case 3, some adjustments need to be made to the actual value X. In Case 2, the lower limit is a negative value. Therefore, if X is larger than the positive equivalent of the lower limit $(360° + (A - \Delta))$, then:

$$X = X - 360° \tag{9.3}$$

This makes X negative now. Similarly, in Case 3, the upper limit is over 360°. Hence, if X is lower than the negative equivalent of the upper limit $((A - \Delta) - 360°)$, then:

$$X = X + 360° \tag{9.4}$$

With these adjustments, Equation 9.2 can be used directly.

For the inclination case, the limit angle ranges from 0° to 180° instead of to 360°, this consideration has been taken into account.

Chapter 10

Propagation in OPT-EM

Having explained the gradient-based optimization formulation in the previous chapter, the subsequent focus is to define the approach for conducting the actual trajectory propagation.

Therefore, this chapter focuses on the trajectory propagation of a satellite during its journey from Earth to the Moon. The propagation is conducted in Fortran 90, using available DLR functions that encompass various force and perturbation models. These perturbations include the gravity field of celestial bodies within the Solar System, Earth's atmospheric drag, the solar radiation pressure, and other factors that will be elaborated upon later in the discussion.

A code capable of propagating the initial state of the satellite including these perturbations has been developed and will be explained throughout this chapter. Furthermore, the trajectory propagation will be compared with that of SEMpy (in Python) and also with the propagation conducted in FreeFlyer. This comparison will provide valuable insight into the accuracy and performance of the developed code.

Firstly, the data required in the input file will be explained in Section 10.1. Then, the way the propagation is conducted using a prediction subroutine is presented in Section 10.2. The propagation will be tested, where an issue was identified and solved in Section 10.3.

10.1 Input file

The input file required for the propagation includes several sections, such as Satellite, Force model, Satellite trajectory, Prediciton and Optimization.

1. SATELLITE. This section contains the general data of the satellite, including its name, ID, and additional information required for force calculations, such as the initial mass, the drag area, the absorbing area, the drag coefficient C_D and the coefficient of reflectivity C_R.
2. FORCE MODEL. The force model includes data required for the integration, such as the relative and absolute errors and the central body used for the propagation. Subsequently, the gravity model is defined by specifying the relevant celestial bodies involved, and the degree and order of the spherical harmonics considered for each body. If a point mass gravity field is used, the value of its gravitational parameter needs to be given. For bodies modeled with spherical harmonics, an input file containing these harmonics and the reference system must be included. Additionally, this section also incorporates flags for the other perturbations that may or may not be included in the propagation, namely, the solar radiation pressure and the atmospheric drag of Earth, both explained in Section 10.5.
3. SATELLITE TRAJECTORY. This section includes the information regarding the initial state vector, the initial epoch, and the maneuvers to be considered. For the initial state vector (expressed in m and

m/s), the epoch time system (usually given in UTC) and the coordinate system including the central body and the reference frame are required. The available coordinate system and time systems have been explained in Sections 3.3.5 and 3.4 respectively. Regarding the maneuvers, several parameters are needed. Firstly, the time when the maneuver will take place (in the specified time system). Next, the duration of the maneuver and the mass flow rate. For an impulsive maneuver, these values can be set to 0, and the duration will be automatically set to 1 s, to model the impulsive shot. Additionally, the components of the ΔV in the EME2000 frame must be provided.

4. PREDICTION. This section includes the output options for the propagation results. It begins with defining the time system and coordinate system to be used for the output data. Subsequently, the initial and final epochs of the output and the time step are specified. If these epochs are the same, the resulting output will include only the state at that specific time. Moreover, it is important to consider that using a very small time step generates a significant amount of data, which may create challenges when reading and evaluating it. Lastly, the output options are presented, offering choices such as the Cartesian state, the Keplerian elements, and the acceleration.

10.2 Prediction

A subroutine has been created to propagate an auxiliary s/c state from an initial epoch to a desired epoch. This subroutine can handle both forward and backward propagation. However, when incorporating maneuvers during backward propagation, extra caution is needed, as the maneuvers must be accounted for in the opposite direction.

It is important to note that the propagation code requires all input epochs to be in TT, so a conversion is necessary from the input time system. The output values, which include the state in EME2000 and the time in TT, are then saved.

This code is capable of including maneuvers by turning on the required thrust throughout the maneuver's duration. The correct formulation of maneuvers is crucial for reaching the desired Moon orbit, and they play a key role in the final cost.

The subroutine calls the DE function from the Fortran DLR libraries, which integrates an n-dimensional system of first-order differential equations of the form from Equation 10.1, where $\bar{Y}(I)$ is evaluated at time T. The DE function uses a modified divided difference form of the Adams PECE formulas and local extrapolation to control the local error during integration. It requires as input an absolute and relative error. From its documentation, it is found that the relative error should be set to 10^{-14}, and the absolute error to 10^{-10}, to ensure an accuracy of 1 cm for an integration of a transfer orbit over a period of 4 days.

$$\frac{d\bar{Y}(I)}{dT} = F(T, \bar{Y}(1), ..., \bar{Y}(N)) \tag{10.1}$$

Additionally, the subroutine calls the ACCEL function, which computes the acceleration and the partial derivatives of the position, velocity, and model parameters.

Within this ACCEL subroutine, the spacecraft's mass is recalculated in case of maneuver propellant consumption. Moreover, the gravitational accelerations are then computed, starting with the central body's acceleration, where spherical harmonics can be used. The acceleration due to the gravity fields of perturbing bodies is also calculated, along with the effects of solar radiation pressure, atmospheric drag, and thrust. The thrust direction plays a critical role, and is determined conducting a change of reference frame from the input one to EME2000.

Throughout the calculations, various data inputs are required to compute the gravitational and other perturbations accurately. These include a file containing the ephemeris position of the Moon from JPL, the gravitational data from the Moon and Earth, and the flux tables from the Sun.

10.3 Testing propagation

The propagation conducted using the new, developed OPT-EM code needs to be tested and verified to ensure the correct performance of the optimization algorithm. This verification process will involve a series of tests, starting from the basic case where only the point-mass gravity of Earth and the Moon are considered. Then, the effect of additional perturbations will be analyzed.

In addition to testing, this section will address the problems, difficulties and lessons learned that have arisen during the implementation. These issues will be examined and discussed, along with the corresponding solutions and any modifications made to improve the code's performance, which offers valuable insight to prevent future mistakes and increase the accuracy and reliability of the software.

The initial analysis focuses on propagating the satellite's trajectory using the same dynamics model as employed in SEMpy. This model only accounts for the point-mass gravity fields of Earth and the Moon. The propagation results obtained from SEMpy, FreeFyler, and OPT-EM should ideally be identical since the input parameters provided for each software are the same. Moreover, the propagation has also been conducted in Deepest, a DLR software for deep-space propagation that has been used as a basis for the author's code.

It needs to be highlighted that, when transferring data from Python to the other software, there is a possibility of losing some significant digits. To minimize this error, the number of significant digits has been set to 20. In a propagation scenario like this, even a small error in the initial state or the values of the maneuvers can accumulate over time and lead to significant discrepancies.

During this testing phase, several problems were encountered, as the comparison between SEMpy and OPT-EM led to very different orbits around the Moon, as seen in Figure 10.1. While both trajectories do result in orbits around the Moon, the final orbits differ by approximately 1000 km in terms of semi-major axis. On the other hand, FreeFlyer's propagation showed similar, though not identical, results compared to SEMpy. The Deepest software, which served as the basis for OPT-EM, produced the same outcome as OPT-EM.

After studying the small but noticeable difference between SEMpy's and FreeFlyer's propagation, it was determined that the problem comes with the way the maneuvers are included in FreeFlyer and the time step chosen, as it will not fire the maneuvers at the exact time selected, but at the next time step, clearly something important to consider when using the software.

Moreover, the outcome obtained from OPT-EM and Deepest suggests that the issue of the difference does not lie within the basic functions, and is more likely to come from the code implementation itself or the input parameters given.

To investigate the reason behind these discrepancies, a thorough analysis was performed. However, it became clear that finding the root cause of the difference would not be an easy task. Therefore, three test cases were designed to explore the potential sources of error in the problem.

- T1: Comparison of the propagation between SEMpy, FreeFlyer and OPT-EM without maneuvers. Hence starting from State 1 (depicted in Figure 10.2), right after the first maneuver.
- T2: Test of the effect of including the first maneuver. The propagation now starts from State 0, continues for a time $\tau_1 \cdot T_E$, then the maneuver is done, and the state is propagated until some time after the Moon is reached.
- T3: Testing the implementation of both maneuvers, starting as in T2, but firing the second maneuver after ToF. The results from this test case are shown in Figure 10.1.

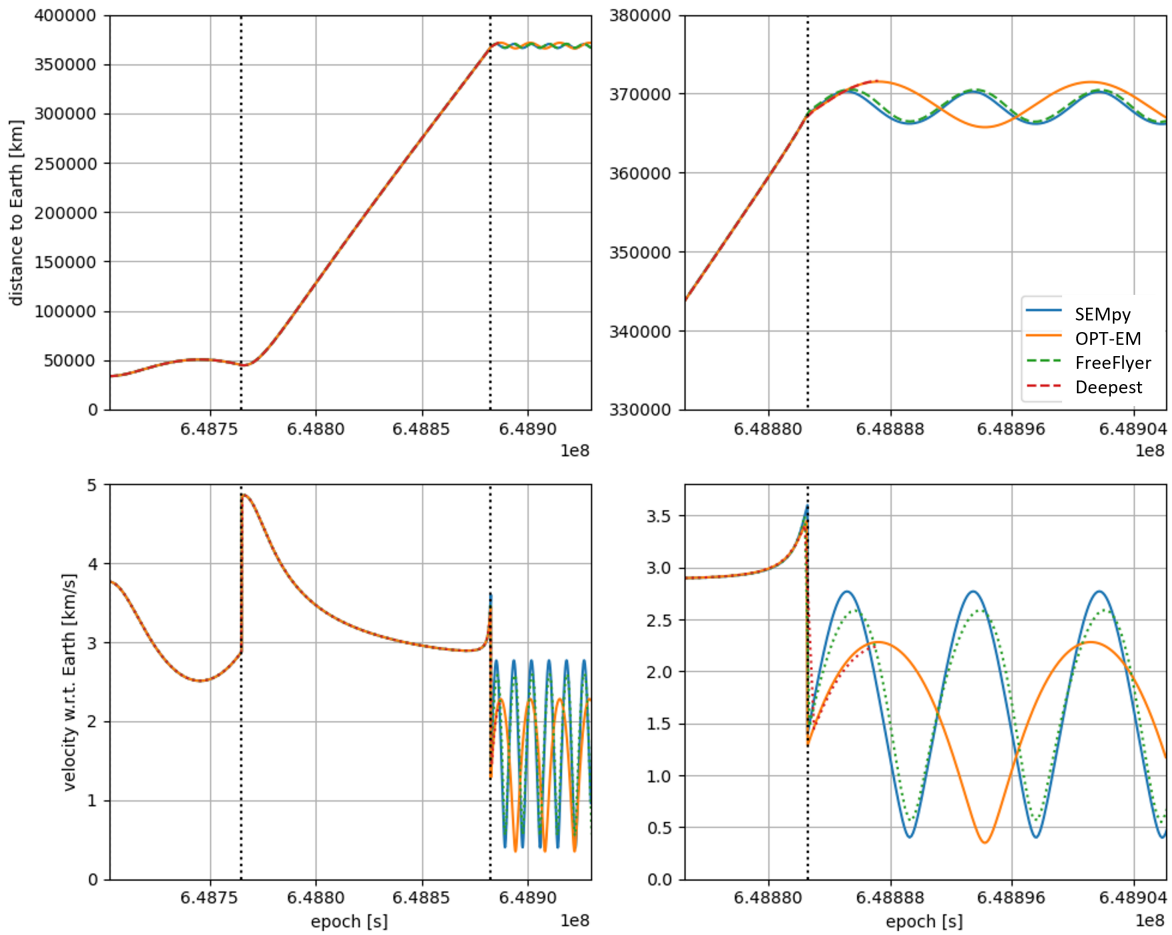


Figure 10.1: Velocity magnitude and distance to Earth for testing SEMpy, FreeFlyer, OPT-EM and Deepest. The dashed vertical lines indicate the time of each maneuver. Left: original, right: zoomed around the second maneuver.

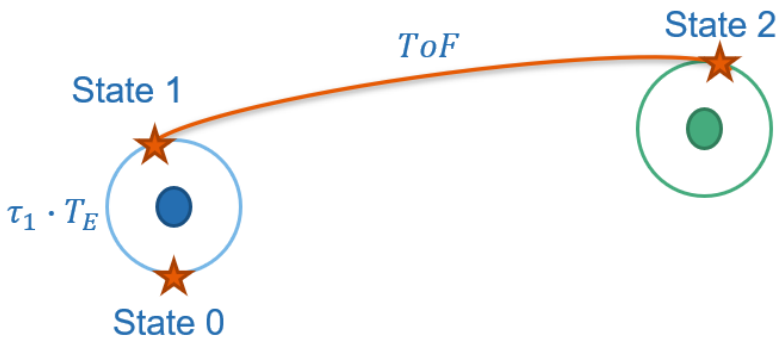
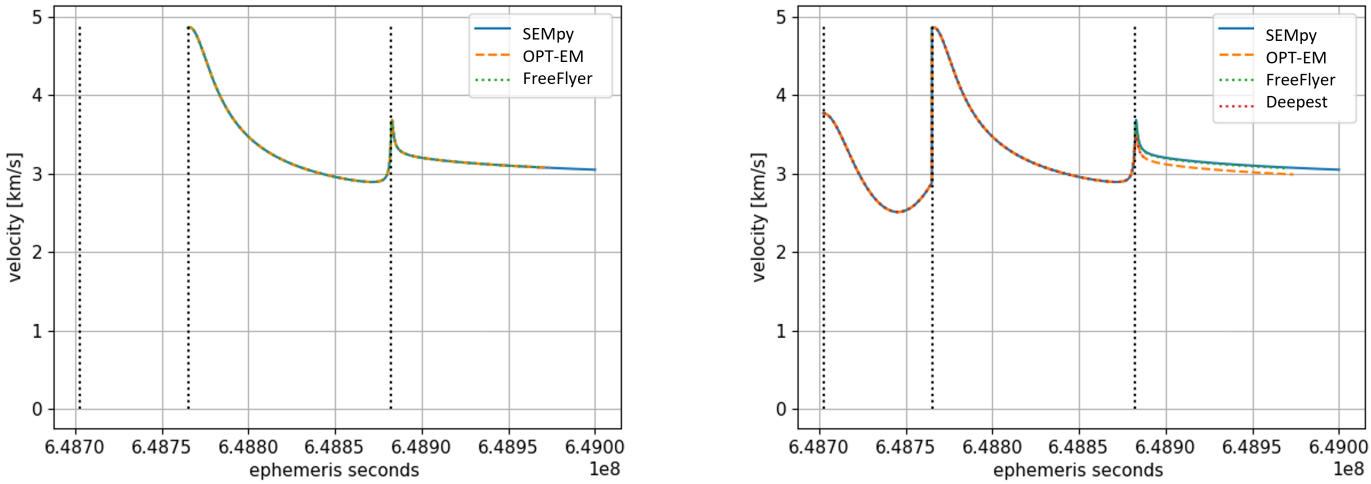


Figure 10.2: Diagram of the states around Earth (blue) and the Moon (green). Where τ_1 is the fraction of time in orbit, and T_E is the period of the orbit around Earth.

Figure 10.3 shows the velocity magnitude for tests 1 and 2. An absolute match is found between all propagations for T1, where no maneuvers are included. However, in T2, both propagations follow a similar trajectory as in T1 until their approach to the Moon, where a significant divergence is observed. The initial analysis focuses on examining the position of the Moon and ensuring that the input used in both OPT-EM and SEMpy is consistent. This test is detailed in Subsection 10.3.1. After confirming that the Moon's position is not the source of error and noting that T1 shows a perfect match of the trajectories,

it became evident that the problem arises when incorporating the maneuvers.



(a) (b)

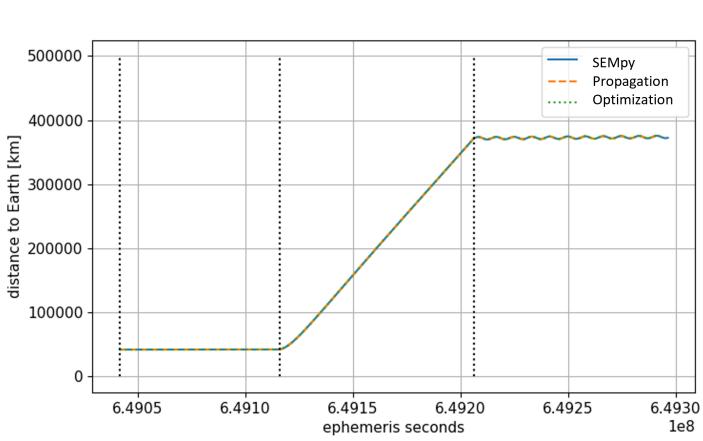
Figure 10.3: Velocity magnitude for the trajectory from Earth to the Moon for tests T1 (a) and T2 (b). The dashed vertical lines indicate the time of each maneuver.

To gain a deeper understanding of the maneuvers, their velocity components have been compared in both the inertial X-Y-Z coordinate frame (as it is the input frame) and the QSW frame as in Section 3.3.4. However, no insight to understand the core of the problem was obtained from it, as the same minor discrepancies are observed for both frames directly after the maneuver jumps, but these discrepancies build up over time.

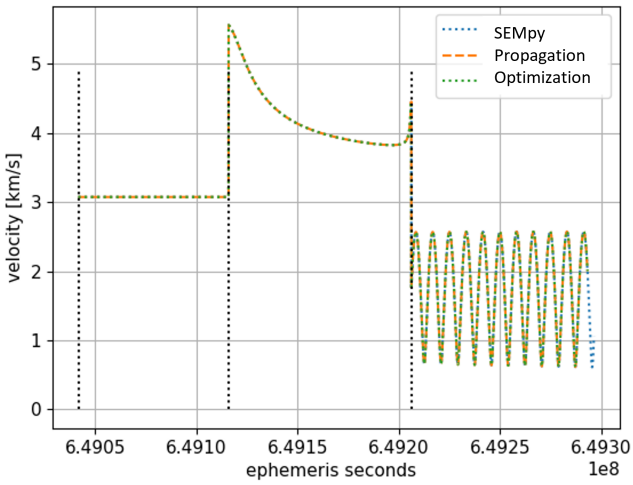
To further investigate, all maneuver parameters were examined, with particular focus on how impulsive maneuvers were handled in OPT-EM, as a direct comparison between impulsive and non-impulsive maneuvers cannot be made. The code allowed for inputs of maneuver duration and exhaust mass flow rate, and it was discovered that setting the duration to either 0 seconds or a small value produced the same outcome. The same method of modeling impulsive maneuvers had been used successfully for Deepest, suggesting that it is not the issue.

Thereafter, the coordinate system used to input the maneuvers in OPT-EM was analyzed, and it was found to be given in an inertial frame. However, there are multiple possible formulations for an inertial frame. After an extensive study of all the reference frames in the code, the source of the error was identified. The inertial reference frame used to obtain the maneuvers from SEMpy was EME2000, which was also the required frame for OPT-EM. The discrepancy arose because the time of the maneuvers input was in UTC, while OPT-EM required it to be in TT. The code attempted to convert the time, but it was erroneously considering an additional change in the coordinate reference frame to account for precession and nutation in the maneuver's values, which was unnecessary. Once this error was resolved, a perfect match between the two coordinate frames was achieved, as seen in Figure 10.4.

This figure includes two OPT-EM possibilities: stand-alone propagation and propagation within optimization. The distinction is conducted to ensure that, once the error had been found, the propagation inside the optimization module gave the same result as the propagation done independently.



(a)



(b)

Figure 10.4: Velocity magnitude and distance to Earth for the trajectory to the Moon after the precession and nutation errors have been solved for SEMpy, and two OPT-EM options: stand-alone propagation and propagation within optimization. The dashed vertical lines indicate the time of each maneuver.

10.3.1 Moon's position

To ensure the consistency of the results from all software used, it is important to test the data that is being used for the propagation. The Moon's gravity has a big effect on the dynamics of the problem, and the actual position of it is required. Therefore, the Moon's position in the inertial frame has been taken from FreeFlyer and from OPT-EM.

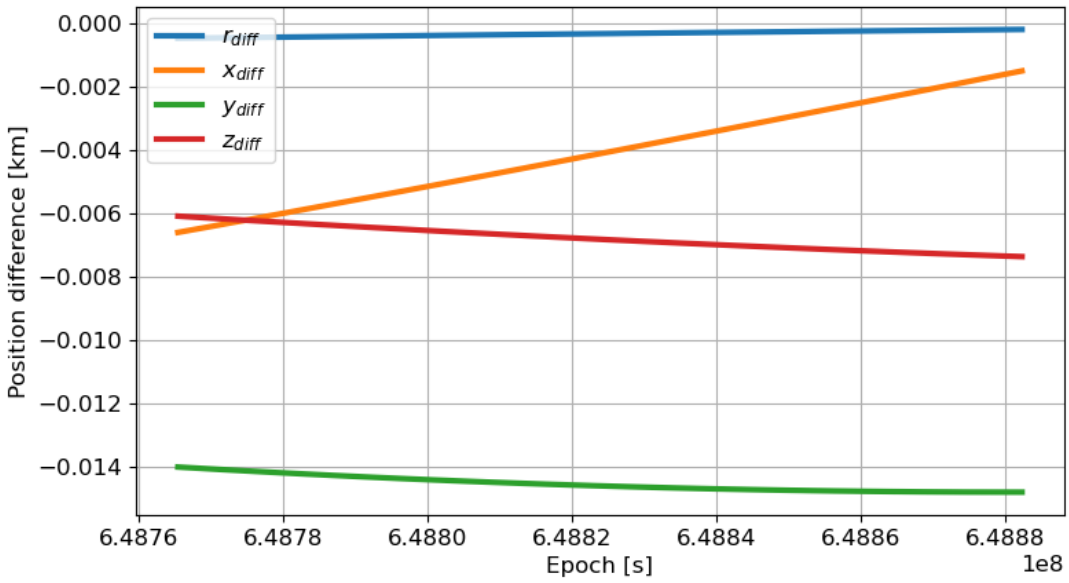


Figure 10.5: Difference in position of the Moon from FreeFlyer and OPT-EM.

Figure 10.5 shows the difference in position of the Moon from both softwares for two days. The difference in distance to the Earth is less than 1 m. The largest difference is in the y direction, in the EME2000

reference frame, and it is 14 m. This difference does not play a big role is the dynamics of a satellite flying over the Moon's surface, at a distance of more than 1778 km from the center of the Moon.

When plotting these data, a significant difference was initially observed in the distance components within the EME2000 frame. This divergence came from the reference frame of the epoch that had been set, in particular the difference between UTC and TT. This once again emphasizes the criticality of handling reference frames with great care, as they can introduce substantial differences in astronomical dimensions.

Another important aspect to consider is that, initially, the differences were computed using an interpolator. However, when the actual differences involve orders of meters while the values themselves are on the order of 300000 km, interpolation errors can become significant. To mitigate these errors, the positions were obtained at the same epochs, eliminating the need for interpolation and ensuring more accurate results.

10.4 Keplerian Elements

An important aspect of the propagation in OPT-EM for optimization is the accurate calculation of the Keplerian elements, which should match those of the final orbit. To ensure the reliability of the results, a comparison was made between the values obtained in OPT-EM and those obtained from SEMpy. Both functions are independent, and hence their result being the same can be taken as an assurance that there are no errors in the codes.

The states of the satellite around the Moon, derived from the OPT-EM propagation, were converted to Keplerian elements using SEMpy (by reading the file and then converting) and directly within OPT-EM. A comparison was made between these two sets of solutions.

In Figure 10.6, the errors in the elements for one day in the final orbit around the Moon for Test Case 1 are depicted. The maximal difference in the semi-major axis was found to be 1 m, the difference in the eccentricity was 10^{-7}, and the difference in the inclination was 10^{-6} degrees. These discrepancies fall within acceptable error margins, confirming the correctness of the conversion for both cases. The periodicity of the error can also be noted, showing 12 cycles for the case of the eccentricity and 24 cycles for the case of the inclination. The lunar orbit has a period of around 120 minutes, hence the satellites goes around the Moon a total of 12 times for a day, coinciding with the number of cycles of the eccentricity and being half of those of the inclination. The inclination was calculated in OPT-EM with the *EARTH_MEAN_EQUEQX_OF_J2000* frame. To obtain the actual inclination w.r.t. the Moon's equator, the *MOON_TRUE_EQUIAU_OF_DATE_JPLDE* is used, and the conversion described in Subsection 6.1.2 is required.

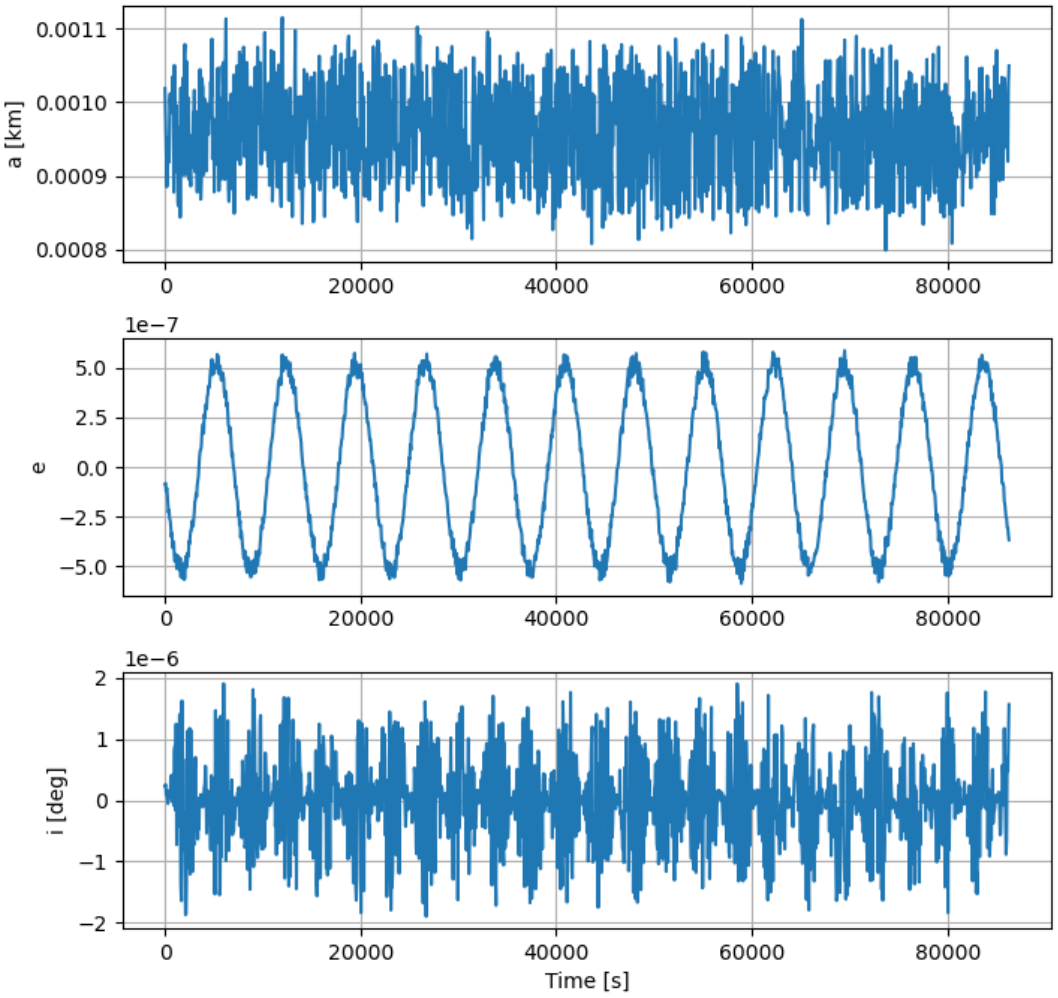


Figure 10.6: Difference in Keplerian elements in SEMpy (after conversion) and in OPT-EM (directly).

10.5 Propagation with additional perturbations

One of the goals of this project is to evaluate whether the optimizer can find an optimized solution that satisfies the constraints when additional perturbations are introduced to the propagation model. It is important to note that, until now, the model used has been a simplified three-body point-mass gravity model. To understand the impact of these perturbations, they will be individually explained and incorporated into the propagation process. The order of magnitude of the accelerations they exert on a satellite depending on its altitude w.r.t. Earth can be found in Figure 10.7.

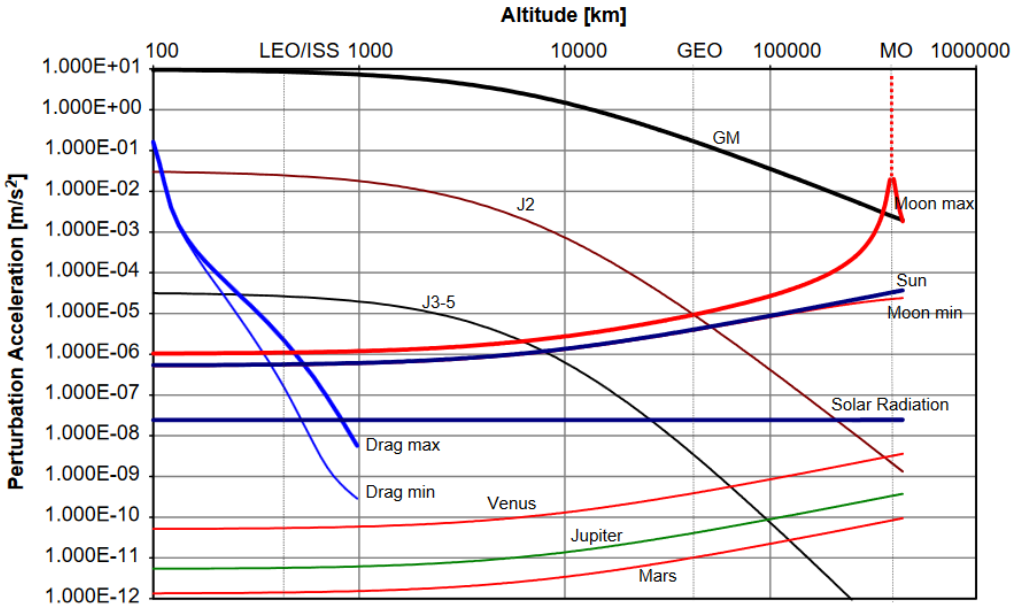


Figure 10.7: Acceleration levels of various perturbation sources for a spacecraft in a geocentric orbit to the Moon. Retrieved from Yazdi et al. (2004).

The following perturbations are considered:

- **Gravitational model with spherical harmonics (SH):** The spherical harmonics represent the spatial variations of the body's gravitational field, offering a systematic and efficient approach to describe the complex gravitational potential and acceleration. To analyze the impact of Earth and the Moon's spherical harmonics on the trajectory, an input file containing the respective harmonic values for the desired degree and order is necessary.

- **Solar radiation pressure (SRP):** The solar radiation pressure is a force produced by the impact of photons from the Sun on the surface of the spacecraft. To calculate its acceleration, Equation 10.2 is used, as explained by Parker (2018). The satellite reflectivity C_R and cross-sectional area A_{CR} need to be given in the input file. P_{SR} is the solar pressure, taken as $4.560 \cdot 10^{-6}$ N/m^2, the initial mass m_P is given as input and the actual mass of the satellite is updated with each maneuver, and $\vec{r_S}$ is the unit vector from the satellite to the Sun. All these calculations assume that the spacecraft surface is normal to the Sun's direction. In addition, the presence of Earth as an occulting body is considered in *illumination*. Figure 10.7 shows that its value is approximately constant and its effect on the acceleration is of the order of 10^{-7} m/s^2.

$$\begin{aligned} \vec{a}_{SRP} &= -\frac{C_R \cdot A_{CR} \cdot P_{SR}}{m_P} \cdot \frac{\vec{r_S}}{|\vec{r_S}|} \\ \vec{a}_{SRP} &= illumination \cdot \vec{a}_{SRP} \end{aligned} \tag{10.2}$$

- **Drag:** Drag is the force experienced by a spacecraft moving through a medium, such as the Earth's atmosphere. It has a noticeable effect on satellites in LEO, but its effect decreases considerably with altitude, reaching a value lower than that of the SRP at altitudes higher than 1000 km. The drag coefficient needs to be given in the input file, and the atmospheric model selected in the software is Jacchia-Gill, based on a smooth bipolynomial fit of density value (Jacchia (1977)).

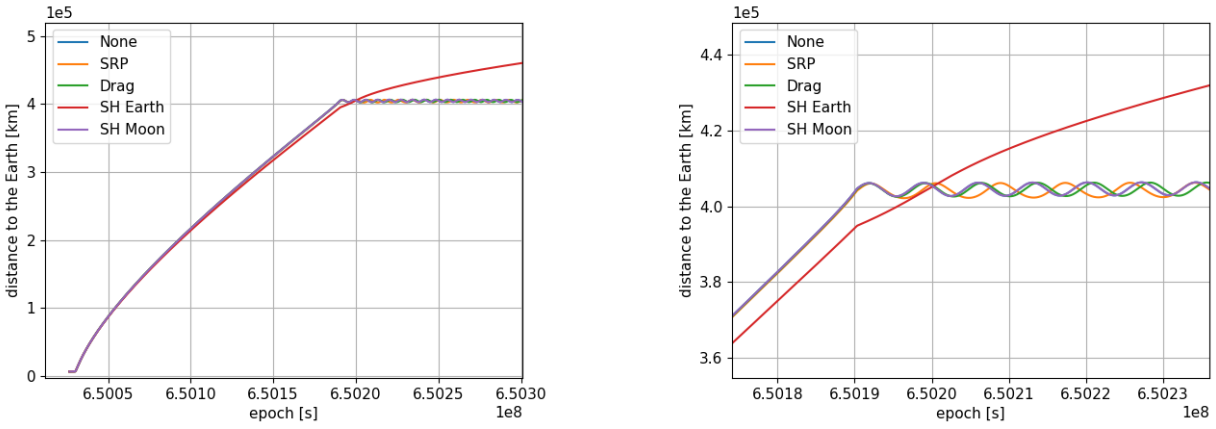
Using the same initial state, the individual perturbations are incorporated, and the resulting trajectories are plotted in Figure 10.8. The spherical harmonics for Earth have a degree and order of 100, while for

the Moon, it is set to 10. The plot reveals that the trajectories under all perturbations, except for Earth's SH, lead to an orbit around the Moon. However, for a clearer understanding of the results, Table 10.1 presents the Kepler elements of the final orbit around the Moon. Do note that the results are based on a propagation of one specific state.

Both the table and the figure highlight that the highest effect is caused by Earth's SH. Even though its impact is more pronounced closer to Earth, over time, the initial difference builds up throughout the trajectory. The second-most influential perturbation is SRP, resulting in an orbit approximately 200 km higher and with an eccentricity nearly 0.2 larger. Additionally, the inclination increases by 2 degrees. Drag also induces an effect. Even though it is only significant during the satellite's time in LEO, its effect is sufficient to alter the state where ignition of the first maneuver occurs. Lastly, the Moon's spherical harmonics have a minimal effect, which becomes more noticeable as the satellite spends more time in orbit around the Moon.

Case	a [km]	e	i [deg]
None	1836.9	0.0005	83.56
SRP	2063.4	0.1612	88.25
Drag	1875.1	0.0564	85.16
SH Moon	1837.8	0.0004	83.56
SH Earth	-	-	-

Table 10.1: Final orbit's Keplerian elements from a specific initial state with inclusion of different perturbations.



(a) (b)

Figure 10.8: Satellite's trajectory from a specific initial state with inclusion of different perturbations. Figure (a) is zoomed in Figure (b).

10.5.1 Optimization with additional perturbations

While testing the optimization process, it was discovered that the optimizer can effectively correct differences caused by various perturbations when the initial guess already results in an orbit around the Moon. Based on this observation, all perturbations except for the SH could be included in the analysis. However, this approach would not be appropriate since the SH perturbation has the highest impact on the trajectory, and neglecting it would eliminate the need of correcting for other perturbations, as the error will already be defined by it.

To find a better initial guess, the hypothesis from Section 6.1.3 regarding the inclusion of the Earth's SH

perturbation in the satellite's orbit propagation around Earth was tested. This approach proved effective, providing an initial state that, although leading to an orbit different from the objective one, does reach an orbit that can be corrected by the gradient-based optimizer. These tests will be presented in the test cases of Part IV.

A reference input file for the optimization in OPT-EM for TC1 and seed 20 has been included at the end of the Appendix A.1.

Chapter 11

Tuning in OPT-EM

As previously mentioned, the tuning process plays an important role, as it directly influences the performance of the optimizer. When it comes to the SLLSQP optimizer, there are multiple parameters that can be modified.

- Scaling factor for the parameters of the decision vector (SP)
- Scaling factor for the constraints (SC)
- Scaling factor for the fitness (cost) function (SCF)
- Accuracy (ACC)
- Directional derivative of the perturbation parameters (DDPP)

The initial test has as an objective to evaluate the impact of each parameter individually. To achieve this, all the scaling factors (SP, SC, SCF) and the accuracy (ACC) will be set to 1, and the DDPP will be set to 10^{-5} for all parameters. The tuning has been conducted without any additional perturbations, and the constraints have been set to ´True' in all cases, hence all solutions that lead to a result satisfy the constraints.

For a better understating of the decisions of the values to test, the exact values used as the initial design vector can be found in Table 11.1.

$\mathbf{t_1}$ [s]	$\mathbf{V_{1x}}$ [m/s]	$\mathbf{V_{1y}}$ [m/s]	$\mathbf{V_{1z}}$ [m/s]	ToF [s]	$\mathbf{V_{2x}}$ [m/s]	$\mathbf{V_{2y}}$ [m/s]	$\mathbf{V_{2z}}$ [m/s]
3391.249	1311.049	-1464.162	-3471.679	162210.930	-1299.070	525.849	-546.386

Table 11.1: Initial design vector.

The cost of this initial guess is 5493.590 m/s. The main objective orbital parameters are stated in Table 11.2.

	a [km]	e [deg]	i [deg]	Ω
Objective orbit	1837.4	0	85	0
Allowed deviations	1	0.001	1	1

Table 11.2: Objective orbit parameters and their allowed deviations.

The initial orbit use3d for tuning is a circular orbit at an altitude of 250 km, an inclination of 28.4°, and with Ω and ω set to 0°.

11.1 Tuning scaling factors of parameters

There are eight parameters in the decision vector: t_1, V_{1x}, V_{1y}, V_{1z}, ToF, V_{2x}, V_{2y} and V_{2z}, which implies that there are eight scaling factors that can be adjusted. However, for the sake of simplicity, the scaling factors for all components of V_1 and V_2 are kept the same. Therefore, there are four SP to be tuned, specifically those associated with t_1, V_1, ToF and V_2. Several options have been explored and are presented in Table 11.3. If only one value is written in the table, it means that all SP are set to the same number. To facilitate this possibility, two options have been provided in the input file. The first option is to input "ONE", which indicates that only one SP value needs to be specified, and it will be used as the scale for all parameters. The second option is to input "ALL" followed by the four required scaling factors.

Cases 1, 4, 5 and 6 correspond to a constant SP for all parameters, while cases 2, 3, 7 and 8 show scenarios where the parameters have been scaled to have a similar order of magnitude, driven by the values in Table 11.1. In Case 2, all scaled parameters are of the order of 10^{-1}, in Case 3 they are of the order of 10^4, in Case 7 of 10^6, and in Case 8 of 10^0.

	Case	**1**	**2**	**3**	**4**
SP	**Value** t_1, V_1, ToF, V_2	1	10^{-4}, 10^{-4}, 10^{-6}, 10^{-3}	100, 100, 1, 1000	1000
	Fitness [m/s]	5492.570	ERROR 6	5493.655	5493.664
	Case	**5**	**6**	**7**	**8**
	Value t_1, V_1, ToF, V_2	100	0.01	10^4, 10^4, 10^2, 10^5	10^{-3}, 10^{-3}, 10^{-5}, 10^{-2}
	Fitness [m/s]	5493.647	ERROR 6	5493.592	ERROR 6

Table 11.3: Tuning of the scaling factor of the design vector's parameters.

In Table 11.3, ERROR 6 indicates that the optimizer did not find a solution and terminated due to a positive directional derivative. Moreover, the base case has been highlighted in blue, and the results that improve the default case would be highlighted in green, hence showing that none of the cases lead to better results than Case 1.

Conclusion

The analysis of the results presented in Table 11.3 leads to an important conclusion, as it is found that having SP lower than 1 complicates the optimizer's ability to find a solution, resulting in a positive directional derivative. This may be related to the values of the default DDPP, but the relationship between these parameters will be further studied later on.

Moreover, it was discovered that scaling the parameters so that they have a similar order of magnitude does not have a large effect in the outcome of the optimization, as differences observed are less than 1 m/s. Starting from the base case, none of the new cases lead to a decrease in the final fitness value. It is worth noting that determining the appropriate scaling values that align the order of magnitude of the parameters requires the analysis of the actual values of the initial guess. For these tests, the values used are outlined in Table 11.1.

The main conclusion drawn from this analysis is that the SP values have a very small influence on the performance of the optimizer. Additionally, the results show that errors are more likely to appear when the design parameters are scaled to low values. As a result, leaving SP as 1 for all parameters will be the default choice for this project.

11.2 Tuning scaling factors of constraints

To simplify the process of inputting SC with the possibility of enabling or disabling the constraints, the options "ONE" and "ALL" will also be included in the input file.

It has to be noted that, in the case of using the option "ALL", the user needs to consider which constraints are active (TRUE), and the dependencies between certain constraints. For instance, if the inclination is set to 0°, the constraint related to the value of Ω will always be considered inactive (FALSE). Similarly, if the eccentricity is set to 0, the constraint related to the value of ω will be inactive (FALSE). The number of scaling factors required is the same as the number of active constraints, and their order is: a, e, i, Ω, ω.

In this Test Case, all the constraints are active, except for the ones related to ω, as the objective orbit is circular. Their values can be found in Table 11.2. It has to be noted that the input values for the semi-major axis (a) are in km, but the code works in meters, and hence it is the order of magnitude that needs to be considered for the constraints.

	Case	**1**	**2**	**3**	**4**	**5**
SC	**Value** a, e, i, Ω	1	1000	0.001	10^5	1, 10^4, 10^2, 10^2
	Fitness [m/s]	5492.570	5492.570	5492.570	5492.570	5492.570

Table 11.4: Tuning of the scaling factor of the constraints.

Conclusion

Based on the data presented in Table 11.4, it is found that SC has no impact on the solution. This is primarily because the initial guess leads to an orbit very close to the solution, where the constraints are already satisfied.

11.3 Tuning scaling factor of fitness function

The SCF is able to adjust the relative importance of the fitness function in the optimization process, as it allows to balance its contribution with respect to the constraints.

	Case	**1**	**2**	**3**	**4**	**5**
SCF	**Value** t_1, V_1, ToF, V_2	1	1000	100	0.01	500
	Fitness [m/s]	5492.570	ERROR 6	5425.673	5493.590	5422.629

Table 11.5: Tuning of the scaling factor of the fitness function.

Conclusion

The analysis of the SCF proves that the optimizer puts more emphasis on decreasing the fitness value when its value is higher. However, it is important to be careful because setting SCF too high can lead to errors if the initial solution is not close enough. To address this, previous projects using this software have used a loop that gradually increases the scaling factor of the fitness function. This iterative process takes the solution from the previous optimization as input and gradually raises the scale value, helping the optimizer to reach the solution with lowest fitness value.

11.4 Tuning accuracy of constraints

The accuracy of the constraints has also been tuned. Nevertheless, as all the constraints for the problem are inequality constraints, where a variance Δ has already been included, this parameters makes less sense.

	Case	**1**	**2**	**3**	**4**	**5**
ACC	**Value** t_1, V_1, ToF, V_2	1	100	0.001	100000	0.00001
	Fitness [m/s]	5492.570	5493.590	5492.570	5493.590	5492.570

Table 11.6: Tuning of the accuracy of the constraints.

Conclusion

The main conclusion obtained from the data presented in Table 11.6 is that using a low ACC yields the same result as using a value of 1. However, having a high ACC leads to the same solution as the initial guess. This is due to the fact that the initial guess leads to an orbit that is very close to the desired one. Therefore, when a low accuracy requirement (high ACC value) is specified, all constraints are already satisfied.

11.5 Tuning parameters' derivatives

DDPP is the parameter used as the perturbation for the derivative calculations, and it is defined for all four design parameters, again assuming a constant value for all the components in $\vec{V}_1$ and in $\vec{V}_2$.

	Case	**1**	**3**	**5**	**7**
DDPP	**Value** t_1, V_1, ToF, V_2	10^{-5}	10^{-2}	10^{-7}, 10^{-7}, 10^{-5}, 10^{-7}	10^{-5}, 10^{-5}, 10^{-8}, 10^{-5}
	Fitness [m/s]	5492.570	5492.785	5492.598	5492.570
	Case	**2**	**4**	**6**	**8**
	Value t_1, V_1, ToF, V_2	10^{-8}	10^{-7}	10^{-3}, 10^{-3}, 10^{-5}, 10^{-3}	10^{-2}, 10^{-5}, 10^{-2}, 10^{-5}
	Fitness [m/s]	ERROR 6	ERROR 6	5472.511	5472.493

Table 11.7: Tuning of the parameters' derivatives.

Conclusion

Several conclusions can be drawn from this analysis. Firstly, if all of the DDPPs are too low, an error can occur due to a positive directional derivative. Additionally, it was discovered that there is a dependence on this value independently, as well as on the combination of them. Moreover, using these values with different cases revealed that the best combination found here was not always the best for other initial guesses. As a result, the value of 10^{-5} for all parameters, which has proven to effectively work for all the tested initial guesses, will be set as the default. Nevertheless, this underscores the need for users to adjust this parameter for each specific case if they wish to seek improved results.

11.6 Additional tuning considerations

Based on the analysis conducted earlier, it was found that SCF and DDPP have a significant impact on the optimizer's outcome. Several tests were performed with different combinations of these parameters. One important conclusion is that a high SCF can lead to certain DDPP values reaching the error related

to positive directional derivatives. After numerous attempts, it was observed that using the default values of 10^{-5} for all parameters provides the highest level of robustness.

To enhance flexibility, the implementation allows for multiple loops where the SCF values can be changed. In this case, the solution obtained from the previous loop becomes the initial solution for the next one. Although increasing the scale in each iteration is typically preferable based on previous knowledge from other projects, after various trials, it was found that using a series of values like 500, 10, 300, 10, 500 yields the lowest final cost. However, setting the second scale to a higher number would result in an error in that iteration. Still, even if an iteration for the different SCF values results in a positive directional derivative, the final decision vector obtained can be used for the next iteration and, in many cases, if the next SCF is not too high, the optimizer is able to find a solution from this initial guess.

This analysis indicates that adjusting scaling values requires careful consideration from the user. Nevertheless, the default value of SCF of 100 will be retained, as it has proven effective across all cases.

In conclusion, it is advisable to experiment with different SCF and DDPP values, as their effects depend on the specific case. Each run takes less than a minute, allowing the user to explore various options without having to spend a large amount of time.

11.7 Tuning with initial epoch as design variable

The hypothesis that including the initial epoch as a design variable is very promising, as this parameter has a large effect on the outcome, as found during the PSO. However, during the tuning process, it was discovered that it introduces challenges for the optimizer, particularly concerning its DDPP value. When the DDPP value exceeds 10^{-11} with a scaling factor of 1, it leads to drastic changes in the solution, causing the optimizer to take an extended amount of time for each iteration, and in some cases, fails to find a solution at all. On the contrary, setting the DDPP value below the threshold has no discernible impact on the solution, as it stays as the original value.

The author has contemplated two potential sources of the problem. It could be due to the accuracy of the floating-point numbers and significant digits used in OPT-EM, or it may require a different problem formulation to mitigate the drastic impact of small changes in the initial epoch on the optimizer's outcomes.

Given the potential benefits and opportunities for achieving even better results, this possibility will be left for further investigation. However, it has been determined that excluding this parameter still yields a satisfactory solution, and therefore, it will not be included in the evaluation of the test cases.

Part IV

Results and conclusions

Chapter 12

Test cases

Throughout this chapter, various test cases will be examined. These test cases serve three important purposes. Firstly, they ensure that the code operates effectively for different configurations, increasing confidence in the validity of the assumptions made and their impact on the code's robustness. Secondly, they provide insightful ways to analyze the obtained results. Thirdly, these test cases can be used as a basis for comparison with existing solutions from research papers.

To do so, different scenarios of interesting and useful initial (Section 12.1) and final (Section 12.2) orbits need to be developed.

12.1 Earth orbits

Three different Earth orbits will be analyzed. E1 and E2 refer to general LEOs, whereas E3 is a GTO. Refer to Table 12.1 for the Keplerian elements defining the different orbits.

E1

The first initial orbit chosen is a circular LEO parking orbit around Earth with an altitude of 250 km, hence with a semi-major axis of around 6621 km. Its inclination is set in relation to the latitude of Cape Canaveral, which is 28.40°.

A parking orbit is a temporary orbit used during the initial phases of a mission. The launch vehicle propels the spacecraft into the parking orbit, where it remains for a relatively short period before igniting its engines again to enter the final intended trajectory.

The use of a parking orbit offers several advantages. Firstly, it allows the spacecraft to be launched within a specific time frame known as the launch window, which is crucial for reaching the Moon or a planet at a desired time, as launch delays are possible. Hall (1977) explains that the launch window can range from seconds or minutes to several hours due to the use of these parking orbits. The longer the launch delay, the shorter the spacecraft would stay in the parking orbit, but the epoch for the critical trans-lunar injection maneuver is kept the same.

Secondly, as found in NASA (2023), for crewed lunar missions like in the Apollo program, a parking orbit serves an additional purpose. In the event of a malfunction, it is safer and easier for astronauts to return from an Earth orbit than from a high-speed trajectory headed towards the Moon. Moreover, the parking orbit provides a chance for a thorough spacecraft checkout while still being relatively close to Earth before committing to the longer lunar journey.

E2

The second Earth orbit is chosen based on the optimization conducted by Tselousova et al. (2019). This paper shows an optimization of direct two-impulse transfers from a LEO to two high circular polar orbits around the Moon. This orbit has been chosen not only because it is a viable option but also to validate the method used in this report comparing the results obtained to the ones found in the paper.

E3

The last Earth orbit used in these test cases is a GTO. As found in Biesbroek et al. (2000), GTOs are very interesting starting points for lunar missions due to the possibility of reducing launch costs by sharing an Ariane-5 launcher with a commercial flight to GEO.

Lagier (2016) explains that Ariane has launched more than half of the communications satellites into GEO, benefiting of the unique location of the Kourou Europe Spaceport due to its low latitude (5°3'), which helps to minimize the propellant needed to reach the equatorial plane. The orbital parameters of the GTO are also found in Lagier (2016). It has a perigee altitude of 250 km, and its apogee is at the GEO altitude of 35786 km.

12.2 Lunar orbits

Three orbits around the Moon will be considered. M1 is a low lunar circular polar orbit (LLO), M2 is an orbit studied by Ely et al. (2006) as part of constellations designed for continuous polar coverage. M3 is a polar circular orbit with a higher semi-major axis. Refer to Table 12.2 for the Keplerian elements defining the different orbits.

M1

The first Moon orbit to be analyzed is the same orbit used for tuning in previous sections. It is a LLO with an altitude of 100 km (semi-major axis of 1837.4 km). This orbit is circular and nearly polar, facilitating the study of the lunar poles, a primary objective in lunar exploration. Moreover, it can serve as a potential parking orbit before attempting a lunar landing.

The altitude of this orbit closely aligns with the standard lunar orbit used for Apollo missions, which was planned as a nominal 60-nautical-mile (110 km) circular orbit above the Moon's surface, as explained by Berry (1970).

M2

Ely et al. (2006) conducted a study on a polar coverage constellation of satellites to investigate permanently shadowed craters near the Moon's poles. The primary focus was on locating frozen volatiles, especially around the Moon's south pole. Although this study was conducted in 2006, significant developments have occurred since then. For instance, a map of the molecular water located near the lunar south pole has been generated by Honniball et al. (2022) using the SOFIA Telescope spectrometer, making it an even more interesting orbit to study.

To ensure continuous coverage, Ely et al. (2006) proposed a constellation of three satellites. Nevertheless, only one of the orbits will be considered for this test case.

M3

Similarly to E2, M3 corresponds to one of the orbits used for the optimization conducted by Tselousova et al. (2019).

12.3 Summary of test cases

Earth orbit	h [km]	e [-]	i [deg]	Ω [deg]	ω [deg]
E1	250	0	28.4	0	0
E2	200	0	51.6	8.26	0
E3	24389	0.7285	6	0	178

Table 12.1: Earth orbits.

Moon orbit	a [km]	e [-]	i [deg]	Ω [deg]	ω [deg]
M1	1837.4	0	85	0	0
M2	6541.1	0.6	56.2	0	90
M3	10000	0	90	0	0

Table 12.2: Moon orbits.

Test Case	Earth orbit	Lunar orbit
1	E1	M1
3	E1	M2
2	E3	M2
4	E2	M3

Table 12.3: Summary of the test cases.

Test case 1 explores a transfer from a LEO to a LLO, and it has been established as the basic case for all the tuning procedures (Chapter 13). Test case 2 and 3 allow to estimate the potential propellant savings for a satellite departing from a GTO compared to a LEO (Chapter 14). Lastly, test case 4 will serve as a mean to compare the calculated results with those obtained by Tselousova et al. (2019) (Chapter 15).

Furthermore, these four test cases will serve to validate the assumption that only using spherical harmonics during the trajectory of the satellite in its orbit around Earth for the calculation of the initial guess is a sufficiently accurate approximation, because the OPT-EM code has the capability to make necessary corrections.

12.4 General data

The common input data provided for all the test cases for the optimization with the PSO is as follows:

- The initial epoch around which the optimization is conducted: 23^{rd} July 2024.
- Spherical harmonics for the Earth up to degree and order 10.
- Boundaries for the time of flight: 0.1 days and 6 days.
- Boundaries for the initial epoch: 15 days before and after the initial epoch given.
- Boundaries for τ_1 and τ_2: from 0 to 1.

For test case 3, and to ensure consistency with the study made by Tselousova et al. (2019), the initial epoch for the optimization will be set to the 15^{th} of January 2028.

The general input data provided for all the test cases for the optimization with SLLSQP is as follows:

- Mass of the satellite: 1500 kg.
- Drag area: 3.5 m^2.
- C_D: 2.3.
- A_{CR}: 10 m^2.
- C_R: 1.3.
- Allowed deviations for a: 1 km.
- Allowed deviations for e: 0.001.
- Allowed deviations for i: 1°.
- Allowed deviations for Ω: 10°, only available if inclination is not 0°.
- Allowed deviations for ω: 10°, only available if eccentricity is not 0°.

Chapter 13

Test case 1

This first test case, which involves a transfer from LEO E1 to LLO M1, will be thoroughly studied to show the results that can be achieved using the procedure developed in this master thesis. Additionally, it will demonstrate the available tests, plots, and their significance for the user.

As outlined in the report, the optimization process consists of two major steps. The first step involves a global search optimization performed in Python using the PSO algorithm. In this step, the dynamics model considers the point-mass gravity field of Earth and the Moon, with the inclusion of Earth's spherical harmonics when the spacecraft is in its orbit around Earth. The second step involves taking the results from this global optimization and conducting a local search using SLLSQP in OPT-EM. In this local search, a full dynamics model is employed, not only to further optimize the trajectory but also to correct for deviations resulting from the reduced dynamics model used in the global optimization.

13.1 Optimization with PSO

The initial step in this optimization involves running the first test case with five different seeds: 10, 20, 30, 40, and 50. The objective is to compare the optimum obtained from each seed and identify the best result.

Figure 13.1 illustrates the results, showing that the solution obtained from seed 30 yields the lowest fitness value (4773.57 m/s), closely matching the result from seed 10. Additionally, seed 40 reaches a similar solution but with a fitness value 6 m/s higher. However, seed 50 leads to a distinct result compared to the best solutions found, as indicated by the significant difference in τ_1 and τ_2, which is of 0.5, corresponding to opposite sides of both orbits, hence showcasing a sub-optimum solution.

The lower-left figure displays the starting epoch in ephemeris seconds. The difference in epochs between seed 50 and the value for the other seeds is approximately 13.47 days, nearly half of the Moon's orbital period (27.3 days). Additionally, the time of flight is 4 hours longer for seed 50 compared to seed 30.

To understand the reason behind these variations and better interpret the findings, further analysis is required. To this end, the evolution of the entire population from seed 30 has been plotted in Figure 13.2. Each particle is represented by a different color according to its fitness value, with darker colors indicating lower values, which are the more desirable outcomes. The plot reveals that the particles are following two distinct paths, leading to two different (local) optima.

A new plot only including particles with a fitness value lower than 4.8 km/s is also generated to gain further insights into the solutions. This selective visualization will provide a clearer understanding of the more promising solutions within the population.

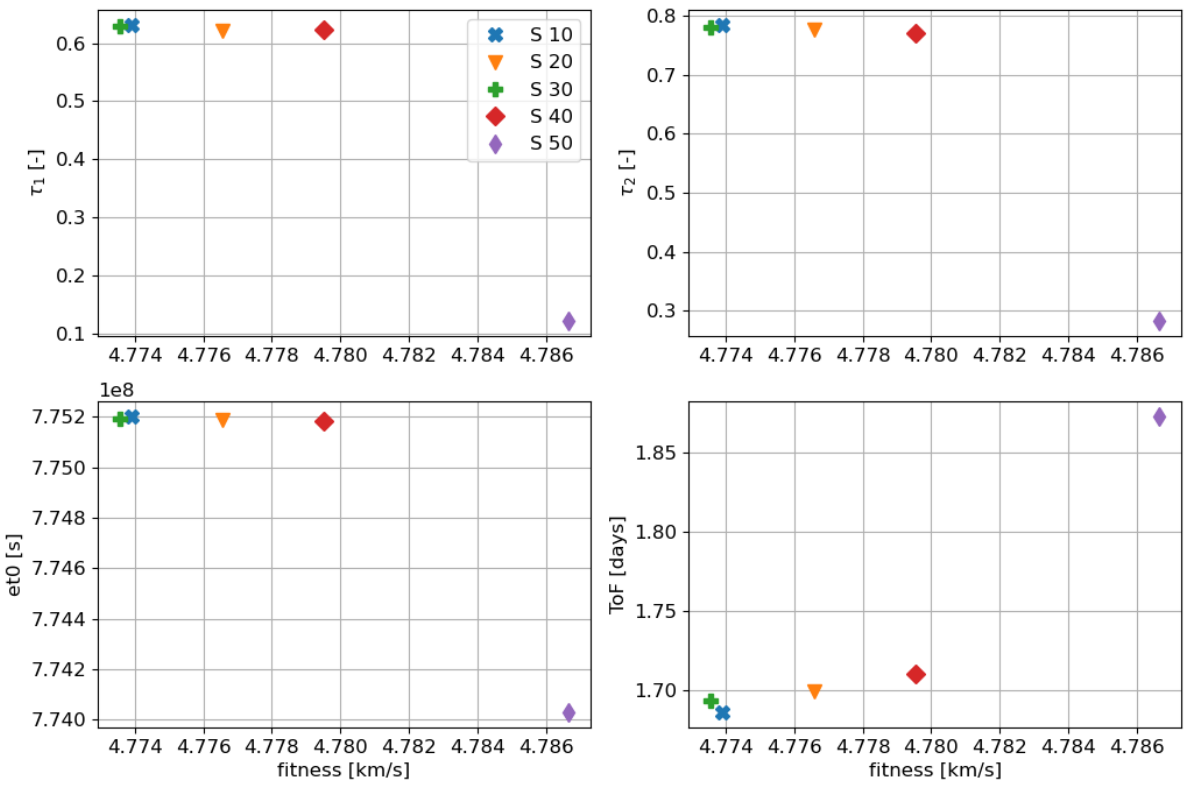


Figure 13.1: Design parameters for the optimal particles found for each five seeds in test case 1.

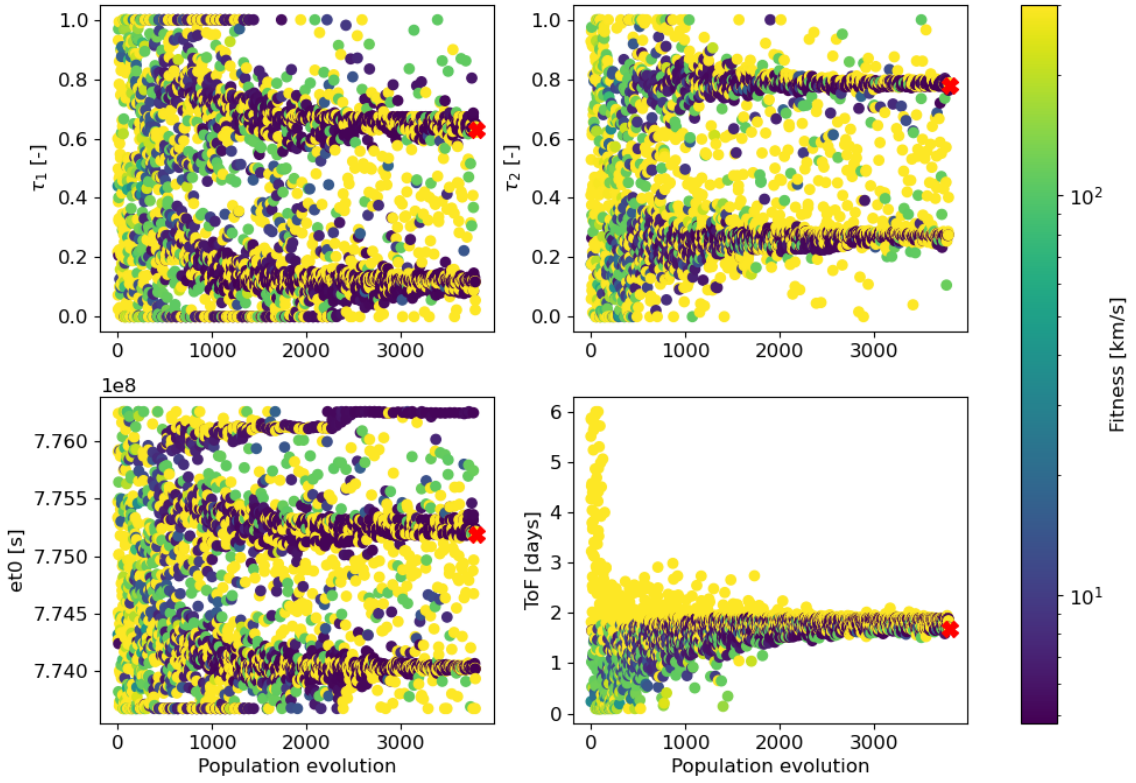


Figure 13.2: Evolution of the whole population for test case 1 and seed 30.

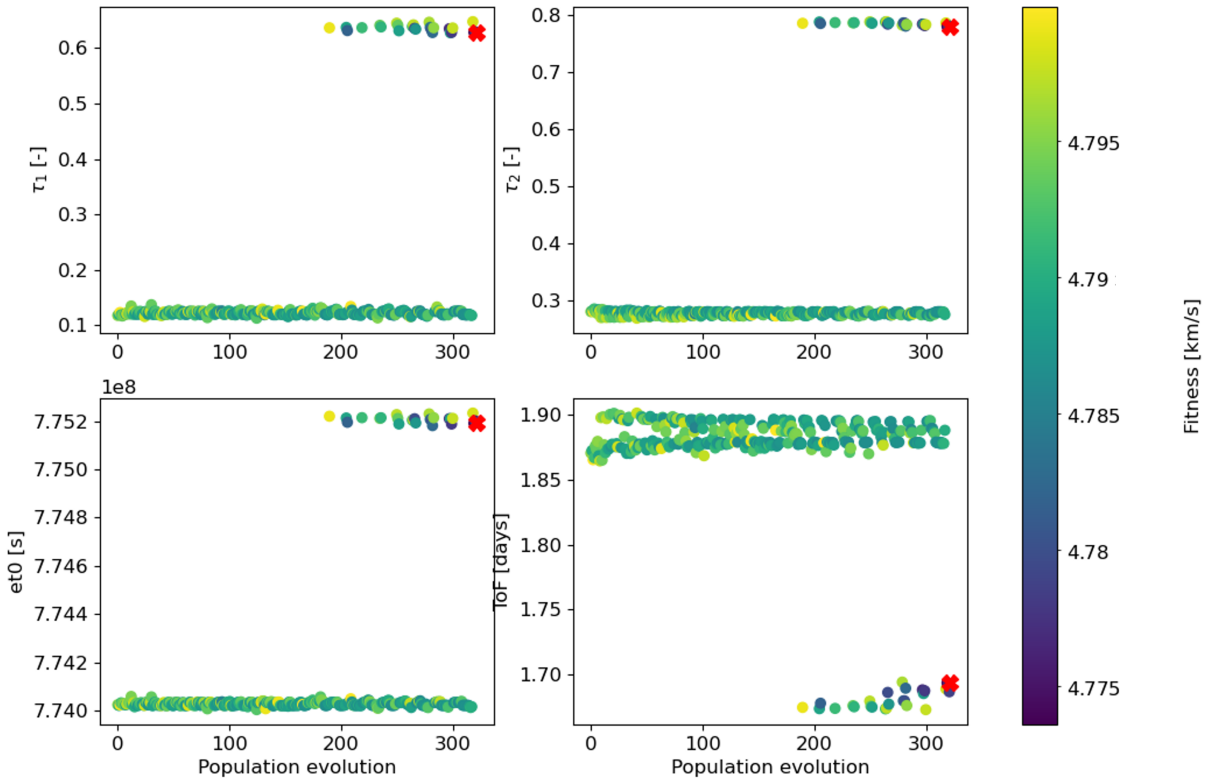


Figure 13.3: Evolution of the population with a fitness value lower than 4.8 km/s for test case 1 and seed 30.

Figure 13.3 shows that these two local optima give solutions with similar fitness values, differing by approximately 10 m/s. Upon analyzing the plot, it becomes evident that the particles closer to the global optimum appear later in the optimization process compared to those for the local optimum ($\tau_1 \approx 0.12$). This observation suggests that having more generations in the optimization process could potentially be beneficial, but it also implies that the fitness values for the two options are so close that the optimizer might face challenges in converging to a single optimal solution. Nevertheless, four out of five seeds were able to find this solution, with two of them producing the best and very similar outcomes, hence indicating that the optimizer has successfully obtained a good initial guess. Moreover, the fact that the optimizer has been able to identify a global and a local optima can be advantageous for mission analysis, as it provides additional launch options. Having such closely performing solutions opens up more choices and flexibility in planning the mission, potentially accommodating various operational requirements and constraints.

Similar plots have been created for the results from seed 10. To maintain clarity in the report, these plots can be found in Appendix A as Figures A.1 and A.2, respectively. Interestingly, these plots display an opposite trend compared to those from seed 30, with particles in close proximity to the global optimum observed in earlier populations. This reiterates the significance of using different seeds in the optimization process, as the initial population has a significant influence on the final outcome. Despite the different evolution observed among different seeds, they converge to a highly similar optimum, demonstrating the robustness of the optimization procedure and its ability to identify favorable solutions despite different initial conditions.

To visualize the physical meaning of the two minima obtained, all optimal solutions found have been plotted in FreeFlyer, as depicted in Figure 13.4. This figure explains the reason why seed 50 gives a

solution which is half a lunar period earlier. Moreover, it reveals that even though seed 10 (blue) and seed 30 (green) produce similar results, the actual trajectory they follow is different, emphasizing the sensitivity of the outcomes to small parameter changes. This sensitivity is a topic on its own and should be carefully studied as a future work.

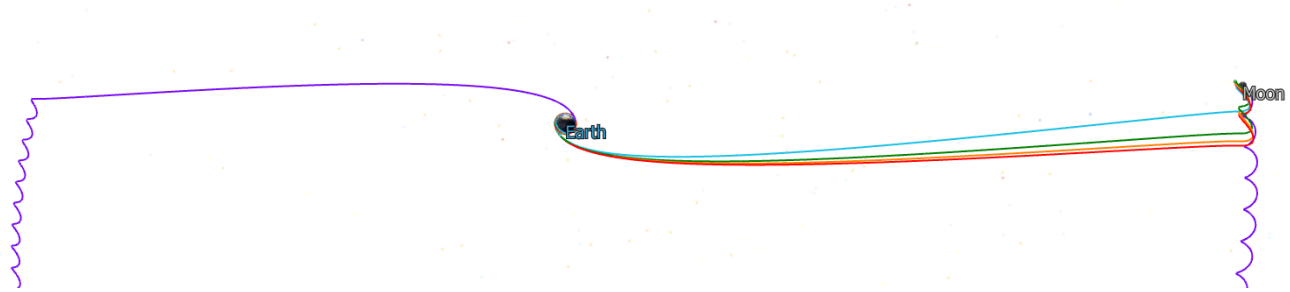


Figure 13.4: Trajectories obtained from PSO optimization for test case 1. The seeds corresponding to each color are: blue for seed 10, orange for seed 20, green for seed 30, red for seed 40 and purple seed 50.

A last significant conclusion drawn from this test case is the importance of visualizing the results rather than solely relying on numerical data of the the best solution provided by the optimizer. To facilitate this visualization process, a user-friendly code has been developed, enabling easy generation of these plots. The code only requires the output files from the optimizer to be given as input. This visualization aids in making informed decisions during mission planning and analysis.

13.2 Optimization with SLLSQP

After the initial optimization is completed, the subsequent step involves refining the most promising initial guess in OPT-EM using the SLLSQP method. This optimization process will be applied to the solution from all seeds in this test case, enabling a thorough comparison of the results obtained and an assessment of the code's robustness. The results of the optimization are found in Table 13.1. It is important to emphasize that the initial epoch will not undergo optimization; therefore, each optimization is performed independently, and an exact match for all cases is not expected. This table shows two columns for each seed, one including the Keplerian elements directly obtained by propagating the results from the PSO "First", and the other showing their values after optimization "Last". The results obtained in "First" do not match the expected ones due to the added perturbations.

The values from the table demonstrate that the solutions after the optimization "Last" all fall within the selected limits, being apparent that the optimizer tends to align closely with the established limits. Another noteworthy observation is that, despite seed 40 providing the highest cost among its options during the PSO, the SLLSQP optimizer, incorporating the perturbations, managed to find the lowest fitness value among all four cases. This highlights the effectiveness of the local optimization step in refining the trajectories and improving the overall solution.

While obtaining the results from Table 13.1, the significant impact of the scaling factor of the fitness value (SCF) on the optimization outcomes has been observed. This finding confirms the conclusion drawn from the optimizer tuning process. The SCF can lead to differences in the solution by approximately 10 m/s, depending on the selected seed. Therefore, it is recommended for the user to experiment with different SCF values during the optimization to achieve the most suitable results.

The results found in the table are informative, however, visualizing their meaning directly can be quite complex, so the final trajectories have been plotted in FreeFlyer.

	Objective	Limits	First S10	Last S10	First S20	Last S20
a [km]	1837.40	1836.40 to 1838.40	3315.13	1836.40	4593.87	1836.40
e [-]	0.000	0.000 to 0.001	0.149	0.001	0.349	0.001
i [deg]	85	84 to 86	66.33	84.00	71.57	84.00
Ω [deg]	0	-10 to 10	-0.87	1.18	-0.93	-1.19
Fitness [m/s]			4773.91	4737.16	4776.57	4725.61
	First S30	Last S30	First S40	Last S40	First S50	Last S50
a [km]	3147.99	1837.16	3222.62	1836.40	3146.94	1836.40
e [-]	0.480	0.001	0.445	0.001	0.480	0.001
i [deg]	88.28	84.00	88.46	84.09	88.27	84.00
Ω [deg]	-0.83	1.16	-0.86	-1.28	-1.33	-0.17
Fitness [m/s]	4773.57	4735.90	4779.53	4716.62	4786.64	4760.08

Table 13.1: Value of the Keplerian elements for the final lunar orbit before (First) and after (Last) optimization for the best solution of all seeds for test case 1. The green colour highlights the best solution found.

Figure 13.5 illustrates all final orbits around the Moon, showing that all satellites successfully reach an orbit around the Moon with the specified conditions after the SLLSQP optimization (a). A noteworthy aspect is observed in Figure 13.6, where only the trajectories for seed 30 and seed 40 are depicted. The bright red and dark green curves represent the trajectories before SLLSQP, while the darker red and light green curves represent the optimized solutions. The left part of the figure demonstrates how, close to Earth, both trajectories from each seed closely follow a similar path. However, as they approach the Moon, it can be observed how the paths from before the optimization (b) converge right before firing, and how the same thing happens for the paths from after SLLSQP.

In conclusion, this test case shows that the two-step optimization approach can successfully guide a satellite from a specified Earth-bounded orbit to a specified Moon-bounded orbit, considering the most significant perturbations.

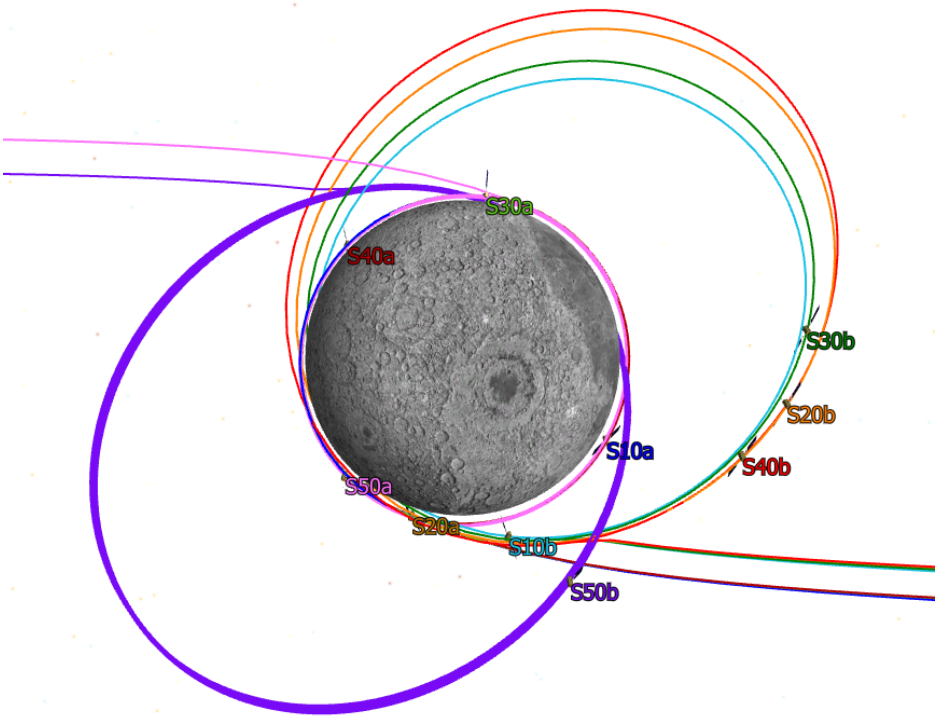


Figure 13.5: Final orbits around the Moon before (b) and after (a) the SLSQP optimization.

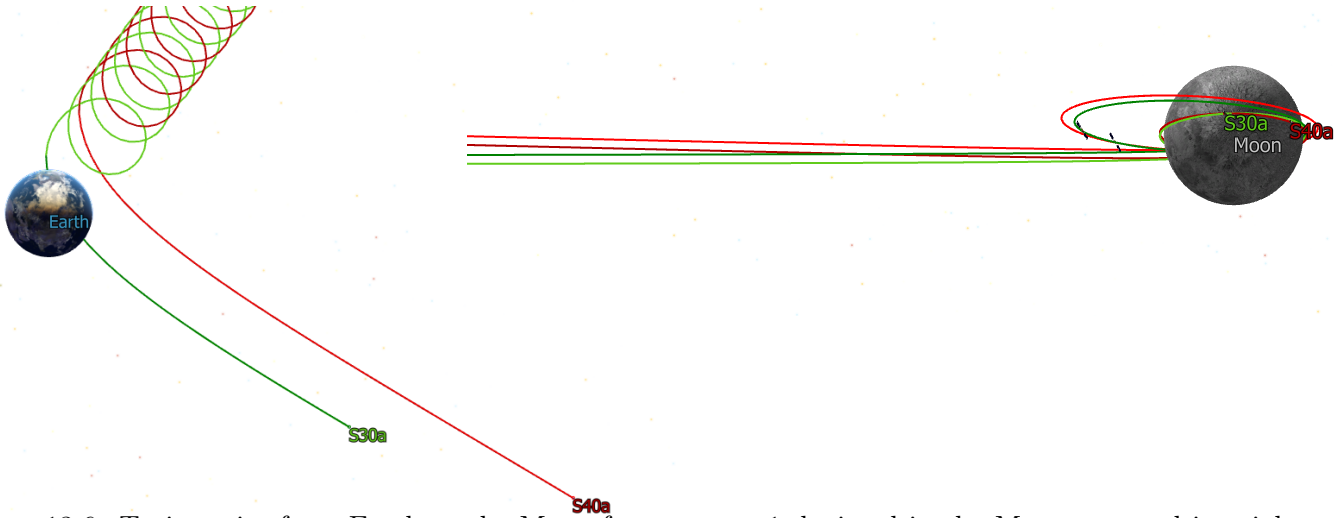


Figure 13.6: Trajectories from Earth to the Moon for test case 1 depicted in the Moon-centered inertial frame. Solutions for seeds 30 (green) and 40 (red).

Chapter 14

Test case 2 and test case 3

Test cases 2 and 3 examine two different transfers to a lunar orbit designed for polar coverage. Test case 2 includes the transfer from a LEO and test case 3 from a GTO, both aiming at M2, an eccentric inclined lunar orbit. Comparing both test cases can give a good estimate of the propellant that can be saved by including the satellite as piggibacking to a mission to GEO.

14.1 Optimization with PSO

The optimal particles found for TC2 after the PSO optimization can be found in Figure 14.1. Multiple options are presented, with solutions from seed 20 and seed 10 demonstrating quite similar outcomes. However, the solutions obtained from the other seeds result in various combinations of parameters, leading to distinct trajectories. To understand the reason of this divergence, the evolution of the population for seed 10 has been plotted in Figure 14.2. This figure shows that convergence has not yet been found, as a trend is still being followed, and no clear option was selected. A similar situation is observed for seed 40, found in Figure A.3. Therefore, the propagation has been conducted again with the same population size but with 100 generations instead of 75. The resulting solutions are presented in Figure A.4, showing that the best solution obtained now has a fitness value 100 m/s lower. However, some level of divergence in solutions still exists. To address this, a final optimization was performed using a population size of 75 and 100 generations. The solutions are illustrated in Figure 14.3, revealing that the optimum particles for each seed are much closer to each other, with the exception of the solution for seed 50, which still exhibits some divergence. This emphasizes the importance of the optimization parameters in certain cases. Ultimately, the lowest fitness was obtained for seed 30, with a value of 3859.38 m/s.

The comparison of optimal particles in TC3 obtained with each seed is shown in Figure 14.4. Unlike test case 1, this set of initial and final orbits leads to a unique global optimum, and no local optimums are found. The seed that diverges the most from the others has a fitness lower than 2.5 m/s higher. This difference is attributed to the initial population and its evolution. Seed 30 reaches the lowest fitness value, with a ΔV of 1523.16 m/s. When plotting the particles that go under 1.6 km/s for both seed 30 and seed 40, it can be observed that seed 40 has fewer particles, suggesting that its initial population was further away from the solution (Figures A.6 and A.8). Nonetheless, when considering the scale of the values, the difference is not substantial, but its effect should be further analyzed in a future sensitivity analysis. The total evolution of seed 30 appears smooth, while the evolution of seed 40 is not as smooth. However, seed 40 still reaches a solution close to the global optimum, and the other four seeds obtain a solution very close to it (Figures A.5 and A.7).

By comparing the initial guesses for both tests, it is evident that launching from a GTO instead of a LEO results in a decrease of 2336.22 m/s in the required ΔV. This represents a reduction of approximately 60% in the ΔV requirement.

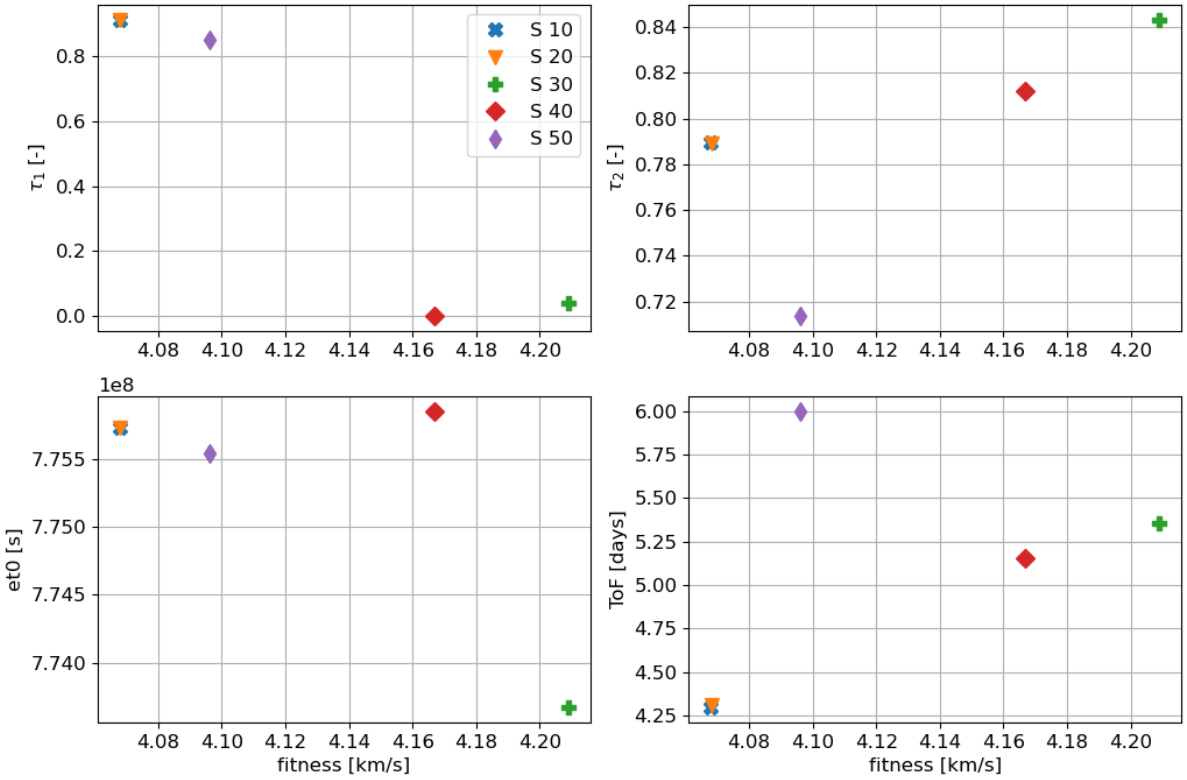


Figure 14.1: Design parameters for the optimum particles found for each five seeds in test case 2 with default optimization parameters.

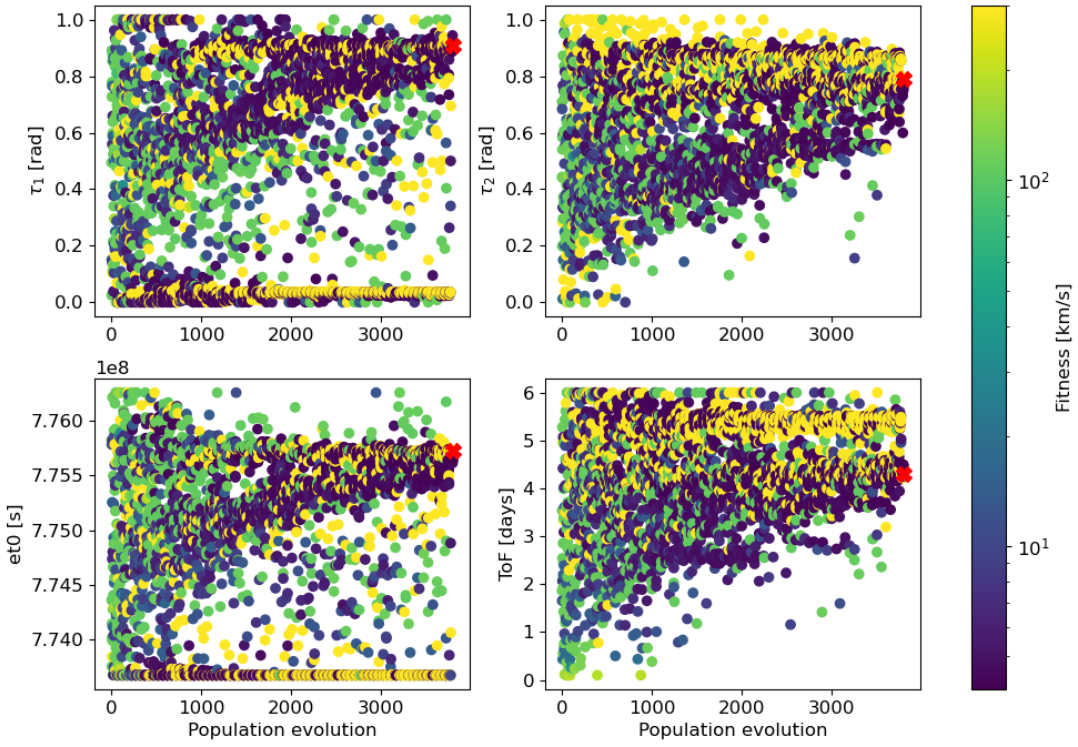


Figure 14.2: Evolution of the whole population for test case 2 and seed 10.

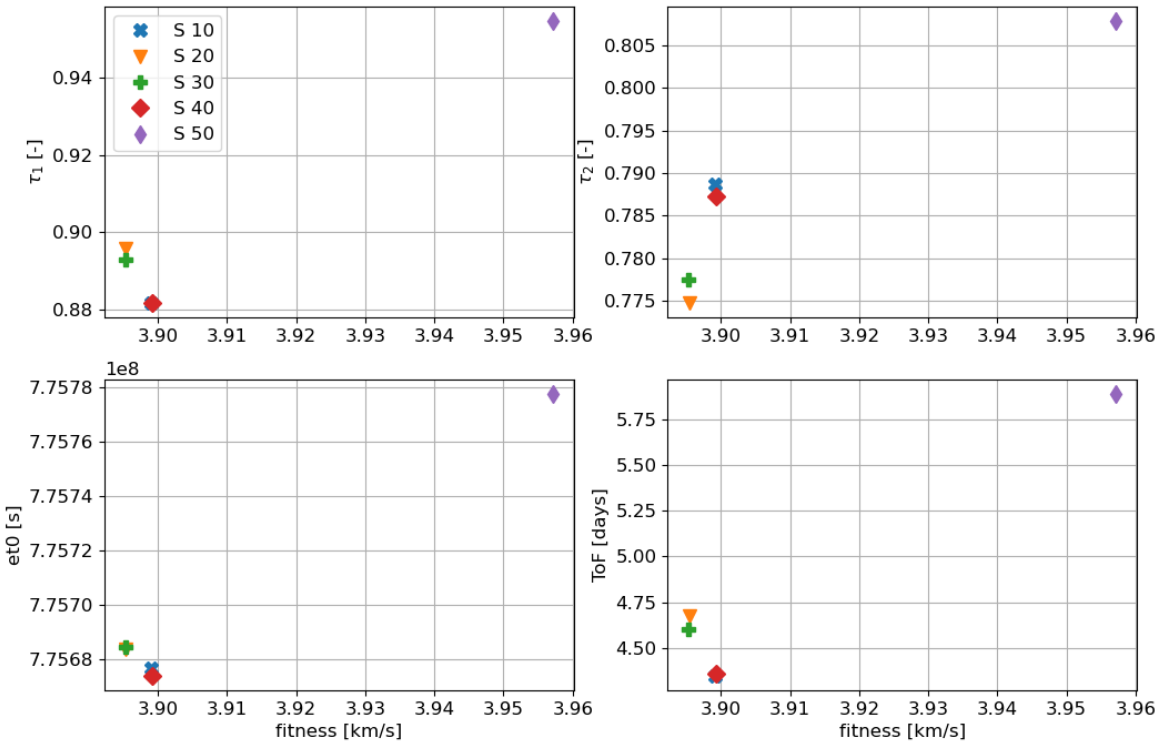


Figure 14.3: Design parameters for the optimum particles found for each five seeds in test case 2. Population size: 75, number of generations: 100.

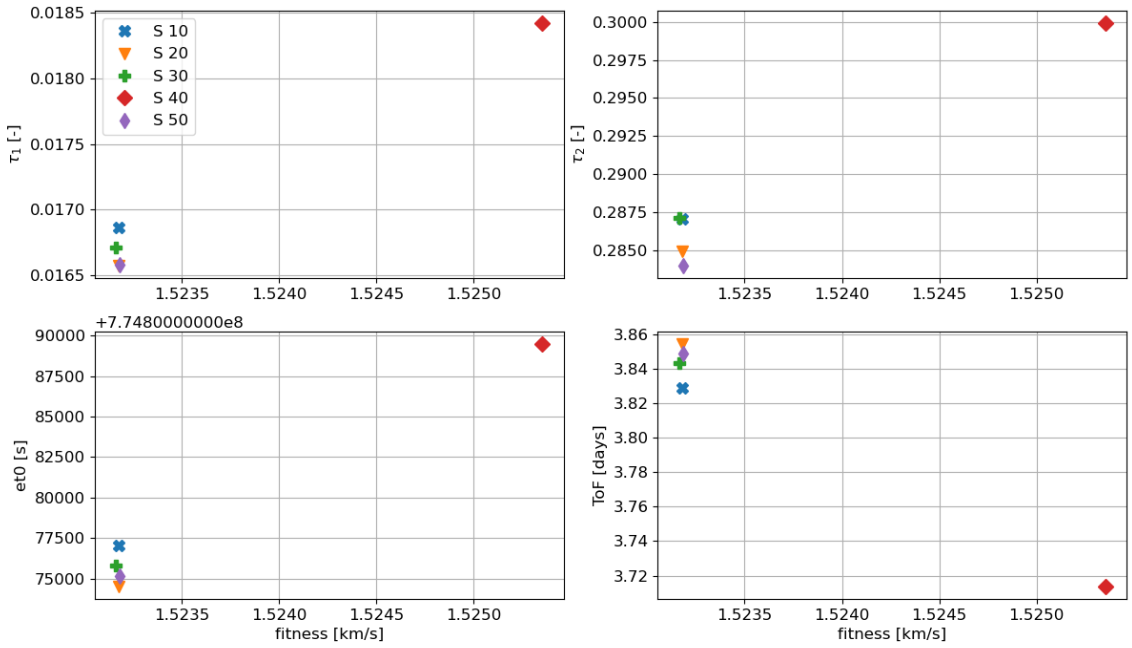


Figure 14.4: Design parameters for the optimum particles found for each five seeds in test case 3.

14.2 Optimization with SLLSQP

Due to the fact that all solutions give similar results for both test cases, only the solutions obtained from seed 30 will be further optimized with the full dynamics model.

	Objective	Limits	First TC2	Last TC2	First TC3	Last TC3
a [km]	6541.10	6540.10 to 6542.10	12489.56	6542.09	8140.76	6540.10
e [-]	0.6	0.599 to 0.601	0.462	0.599	0.540	0.599
i [deg]	40.00	41.00 to 39.00	38.05	39.00	46.75	39.00
Ω [deg]	0	-10 to 10	327.95	10.00	-3.00	-0.48
ω [deg]	90	80 to 100	121.72	80.00	91.38	91.62
Fitness [m/s]			3895.37	3859.81	1523.16	1512.95

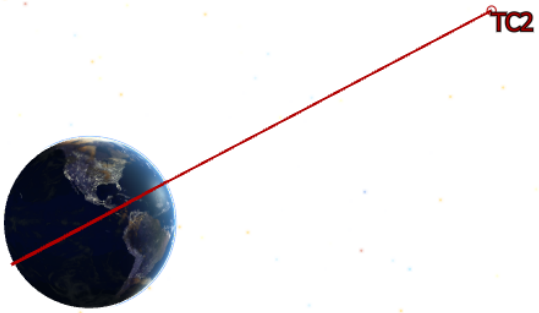
Table 14.1: Value of the Keplerian elements for the final lunar orbit before (First) and after (Last) optimization for the best solution of all seeds for test case 2.

For this test case, the SLLSQP optimizer is also able to correct for the additional perturbations and to refine the solution even further, reaching a total ΔV of 3859.81 m/s for a transfer from a LEO and 1512.95 m/s for a transfer from a GTO. Biesbroek et al. (2000) states that the Moon Orbiting Observatory (MORO) would have used a ΔV of 1580 m/s to get into a lunar polar orbit from Ariane's GTO, hence obtaining a solution requiring a ΔV 4.5% lower.

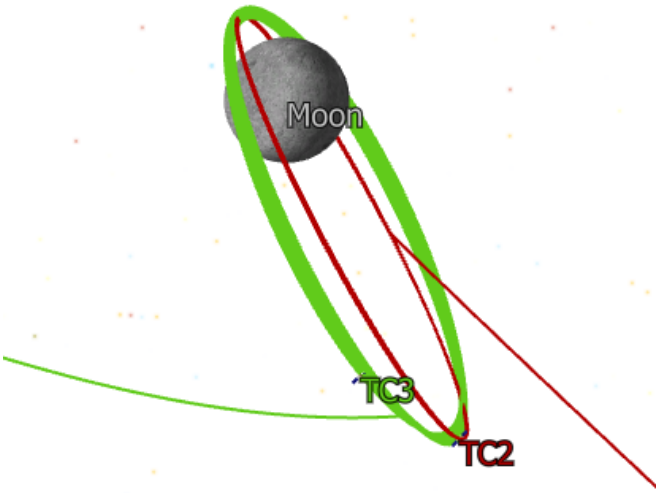
Another interesting conclusion drawn from these optimizations is that, when considering a LEO, the lunar orbit achieved from the initial guess obtained through PSO optimization is significantly different than the desired lunar orbit, particularly concerning the semi-major axis, whose value nearly doubles the intended one. This outcome is explained by the highest influence of Earth's spherical harmonics when the satellite is close to its surface.

In addition, Figure 14.5 illustrates the trajectories that these satellites would follow. It is interesting to observe that the optimum trajectories are characterized by minimal or zero inclination changes from their initial Earth orbits to the LTO. The required inclination change should also be minimized for the lunar injection maneuver. However, it is found that it does not influence the total ΔV as much as the inclination change required to get into the LTO, due to the larger effect of the gravity field of Earth.

Analyzing the trajectory from TC3, it is determined that the maneuver is executed near the perigee of the GTO, which makes sense since it is where the satellite achieves its highest velocity. Consequently, it has been found that for orbits with eccentricity and inclination, the geometry of the initial orbit around Earth with respect to the objective lunar orbit is a crucial aspect to minimize the ΔV, hence optimizing ω and Ω of the initial orbit around Earth could lead to further improvements in the results.



(a) Satellite from TC2 leaving its orbit around Earth.



(b) Satellites from TC2 and TC3 reaching their orbit around the Moon.

(c) Satellite from TC3 leaving its orbit around Earth.

Figure 14.5: Trajectories from satellites from TC2 and TC3 around Earth and the Moon.

Chapter 15

Test case 4

Test case 4 represents the case defined by Tselousova et al. (2019), and will be used for comparison with it. The procedure that will be followed is the same as for the previous test cases.

15.1 Optimization with PSO

Figure 15.1 shows the best solution obtained for each seed. It is noticeable how the values for τ_1 and τ_2 are quite similar across the seeds. For τ_1, the divergence is approximately 0.004 of the orbital period, which in this case is around 88.35 minutes, resulting in a difference of 21 seconds. For τ_2, the difference is 0.1, but since the period of the orbit around the Moon is 24.92 hours, the difference amounts to 15 minutes.

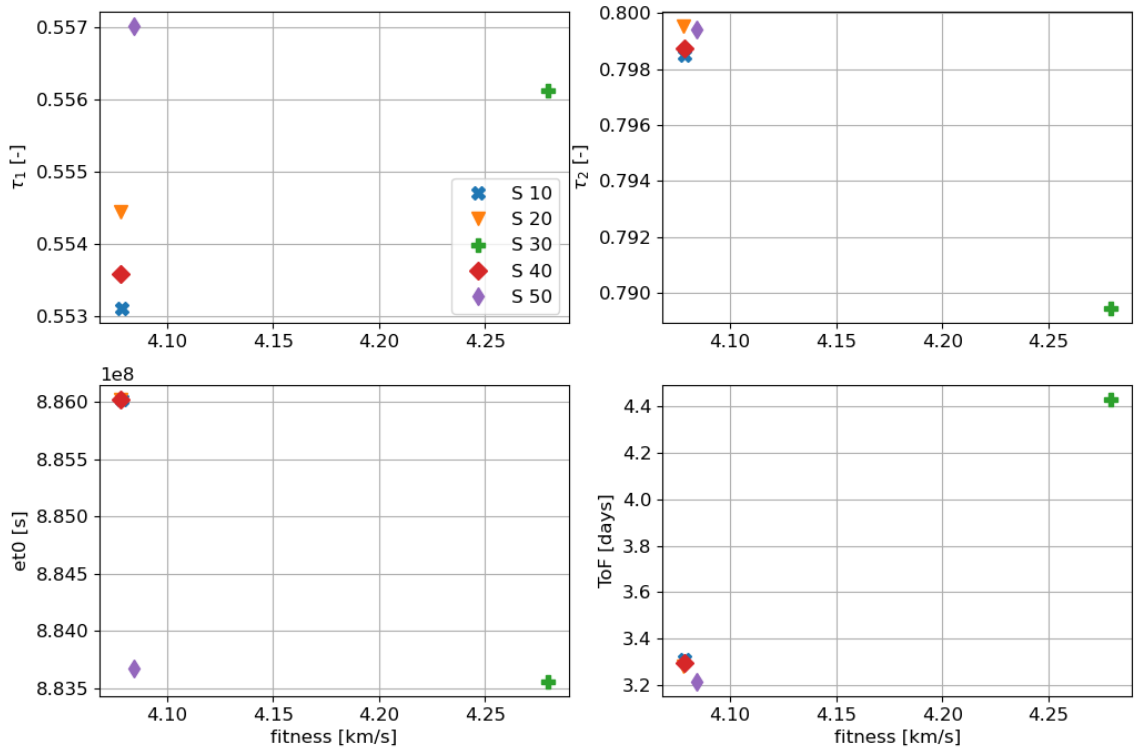
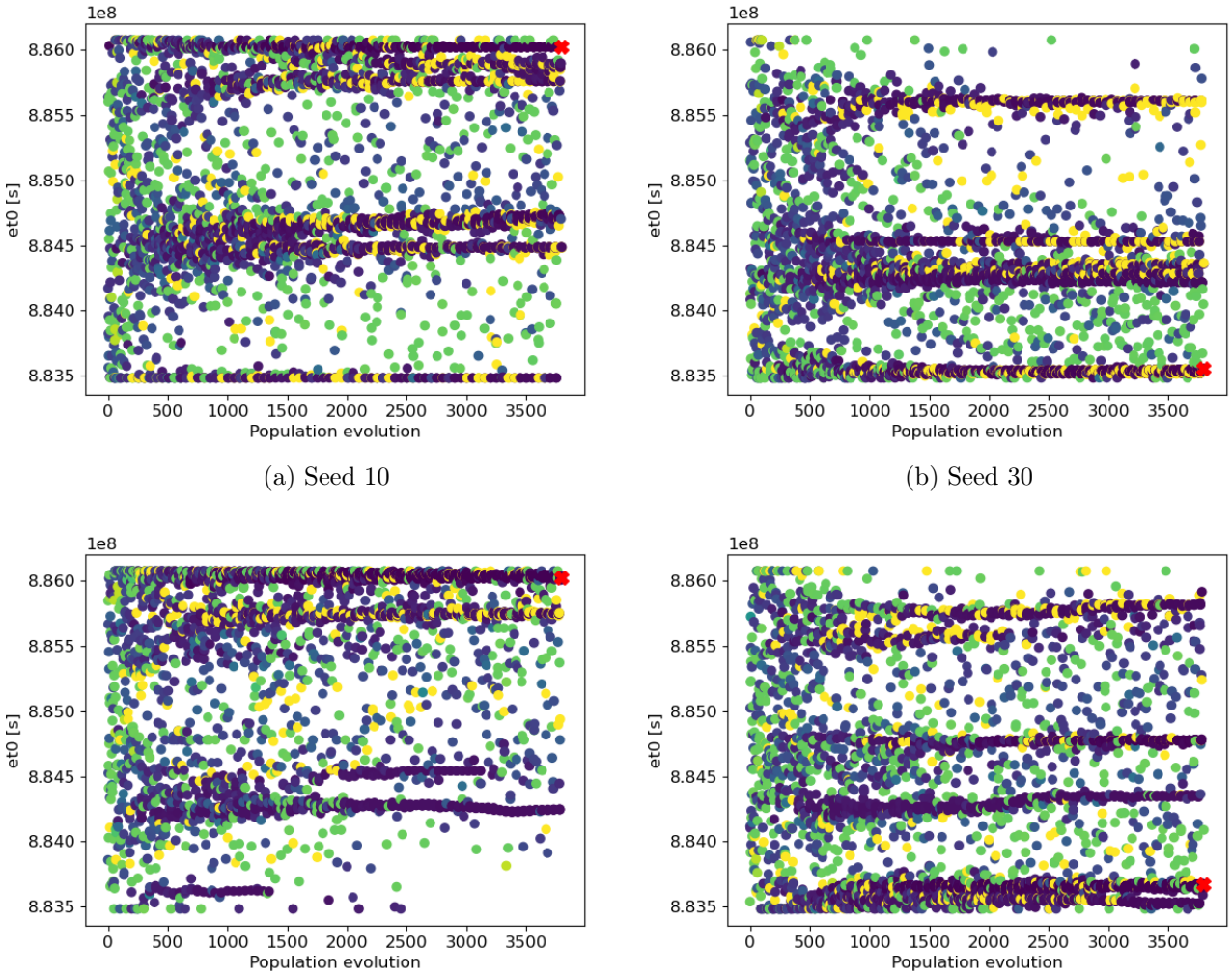


Figure 15.1: Design parameters for the optimum particles found for each five seeds in test case 4.

The figure shows that seed 30 leads to a solution with a fitness value 200 m/s higher than the rest of the particles. In many cases, this indicates that convergence has not been achieved yet. This can be further observed in Figure A.9, where the evolution of the optimization process is plotted, demonstrating that it is still undergoing changes and improvements, and hence indicating that for this seed, a larger number of generations would be required.



(a) Seed 10

(b) Seed 30

(c) Seed 40

(d) Seed 50

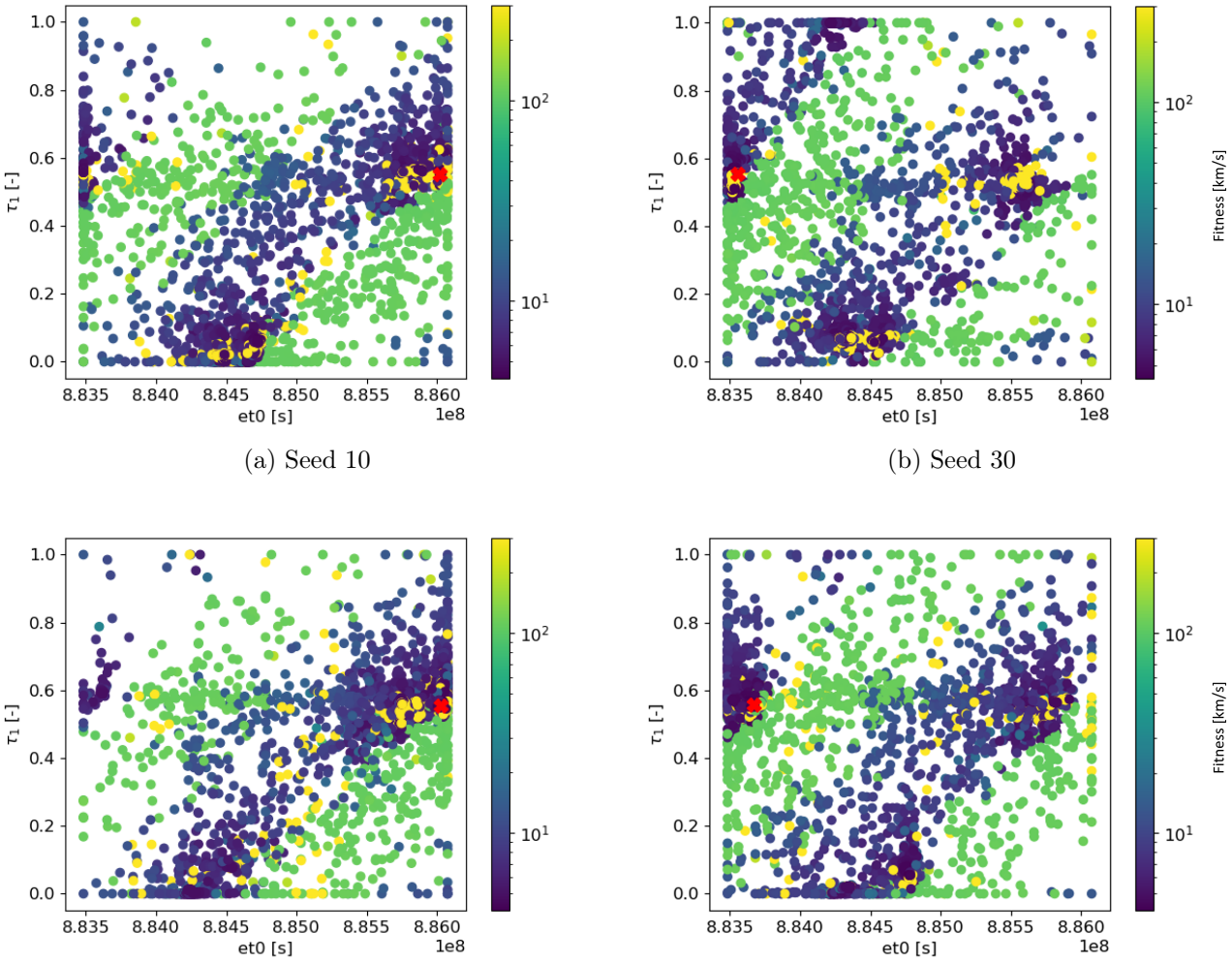
Figure 15.2: Evolution of et0 for the whole population for test case 4. The darker coloured particles represent lower fitness values.

It is worth noting that for seeds 30 and 50, the initial epoch obtained is approximately 27.21 days earlier than for the other three seeds, which corresponds to a full lunar period. Since the total search space for the initial epoch is 30 days, this means that the optimization process found the global and the local optimal solutions very close to the boundary limits, which can hinder convergence.

Figure 15.2 clearly illustrates the problem of convergence with optimal solutions close to the boundaries. For seeds 30 and 40, the initial populations did not contain particles with an initial epoch close to the found global optimum, which resulted in the evolution not exploring those areas further. Seed 10 shows a trend for both optima, but seed 40 does not include particles within the local optima. Interestingly, this figure also reveals the existence of another promising solution at an intermediate epoch. This suggests that exploring different initial epochs can lead to diverse and potentially valuable solutions. However, these particles did not reach fitness values close to the optimal one.

Analyzing the relationship between τ_1 and the initial epoch in Figure 15.3 provides useful insight into the geometry of the solutions. For all seeds, there exists a section of particles (shown in blue) that are viable, surrounded by particles that result in fitness values between 100 and 200, which represent trajectories that go through Earth or the Moon. This observation makes sense since the initial state in the inertial reference frame is fixed. Consequently, as the Moon goes around its orbit, the satellite must also spend more time in its orbit to find a path capable of reaching the Moon without intersecting Earth.

It is also noteworthy how this region appears to repeat periodically with the lunar period, which is the reason why two seeds found optimal solutions a month (lunar period) earlier. Thus, if three months were being analyzed, three blue stripes would be present, with green stripes in between them.



(a) Seed 10

(b) Seed 30

(c) Seed 40

(d) Seed 50

Figure 15.3: τ_1 vs the initial epoch for seeds 10, 30, 40 and 50 for test case 4.

15.2 Optimization with SLLSQP

The initial guesses provided by the PSO optimizer resulted in fitness values that were higher than those reported in the research conducted by Tselousova et al. (2019). The lowest achieved fitness value was 4078.19 m/s, whereas the paper reported a value of 3800 m/s. However, to obtain comparable values, a gradient-based optimization must be conducted to incorporate the perturbations considered in the research. These perturbations include the spherical harmonics of Earth up to degree and order 100, the spherical harmonics of the Moon up to degree and order 8, and the influence of solar radiation pressure. Notably, the initial paper did not impose any constraints on Ω, so it will remain unrestricted for this optimization. The results for all seeds are presented in Table 15.1.

	Objective	Limits	First S10	Last S10	First S20	Last S20
a [km]	10000	9999 - 10001	10822.03	9999.00	10679.07	9999.00
e	0.000	0.000 to 0.001	0.316	0.001	0.311	0.001
i [deg]	90	89 - 91	92.76	90.99	92.47	91.00
Fitness [m/s]			4078.52	3996.44	4078.19	3997.20
	First S30	Last S30	First S40	Last S40	First S50	Last S50
a [km]	18788.26	9999.01	10763.11	9999.00	10367.35	9999.00
e	0.568	0.001	0.313	0.001	0.295	0.001
i [deg]	96.76	89.00	92.65	91.00	92.12	91.00
Fitness [m/s]	4279.34	4162.77	4078.27	3996.62	4084.18	4005.41

Table 15.1: Value of the Keplerian elements for the final lunar orbit before (First) and after (Last) optimization for the best solution of all seeds in TC4. The green colour highlights the best solution found.

Although this optimization successfully decreases the required ΔV while correcting for perturbations, it achieves a reduction of around 100 m/s, leading to a fitness value larger than the one reported in the paper. It should be noted that the epoch found for the lowest value is the 29th of January 2028, which, according to the paper, yields a ΔV of 3.8 km/s, resulting in a difference of 4.6%. One hypothesis for this disparity is that the inclination objective for the polar orbit in the paper is calculated with respect to the selenographic coordinate system called Mean-Earth/Mean-Rotation System (MER) from Archinal et al. (2011), which is different from the frame used in OPT-EM. Moreover, the paper states that the model of motion considers gravitational perturbations from the Sun and the Solar System planets, SRP, and the non-spherical gravitational field of the Moon up to degree and order 8, but nothing is mentioned regarding the gravitational field of Earth. Nevertheless, it has been assumed that including the Earth's gravitational field is important since, as found before, it has the highest effect among the perturbations, and not including it would limit the accuracy of the results.

Despite the difference in the results, it is crucial to note that a solution has been found for all seeds, correcting for the perturbations, demonstrating the robustness of the methodology used in finding a solution able to reach the wanted orbit under a complete dynamical model.

Furthermore, Tselousova et al. (2019) performed an analysis of the required ΔV for each day in January 2028, as depicted in Figure 15.4. This analysis explores the effect of the initial epoch on the mission. Therefore, a PSO with the boundaries of the epoch limited to selected dates in January 2028, spaced every five days throughout the month has been conducted. In Figure 15.5, it is observed that the trend for most days is similar between PSO and the results presented in the paper. However, there is an exception on the 20^{th} of January, where PSO yielded the highest fitness, while the paper reported the lowest. Additionally, for the other dates, the trend is comparable but exhibits higher variance compared to the paper's results. It is interesting to note that the reason behind the high ΔV required to reach the orbit around the Moon the 5^{th} of January is because of the inclination change required for the initial orbit compared to the other

three dates, as seen in Figure 15.6 between T1 and T5.

When the optimization was carried out using the SLLSQP method, the optimizer successfully corrected for perturbations at times T1, T5, T15, and T30, but failed to do so for the other three times. The reason for this limitation is that the initial guess did not result in an orbit around the Moon, and hence the optimizer was not able to find a solution. It was found that this issue is more likely to occur when the orbit around the Moon is at a higher altitude. With smaller changes, the satellite can miss the Moon, leading to difficulties in the calculation of the Keplerian elements.

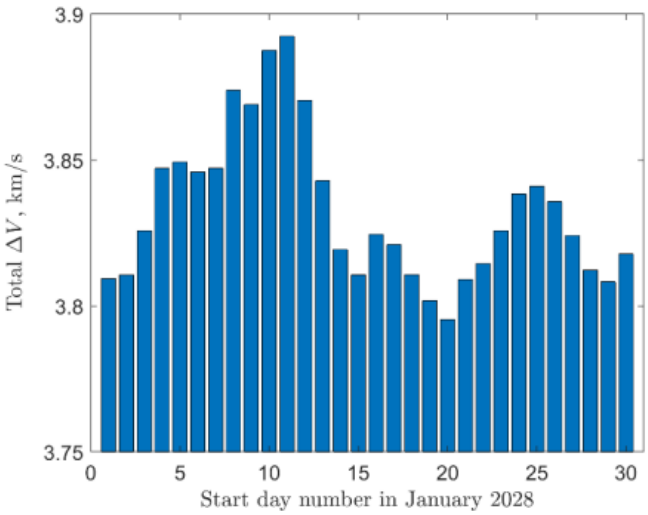


Figure 15.4: The minimum attainable values of the total ΔV for a transfer to the polar lunar orbit with an altitude of 10,000 km in case of the launch on different days of January 2028. Retrieved from Tselousova et al. (2019).

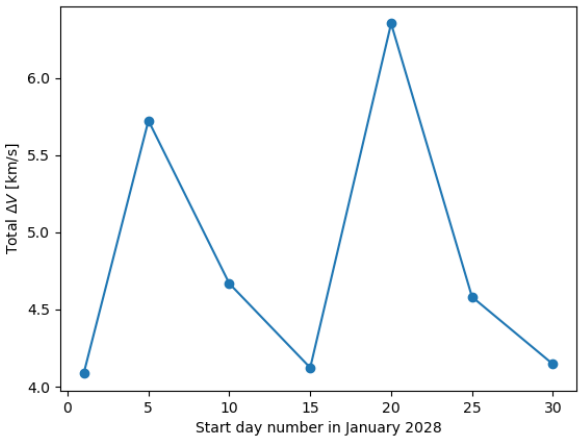


Figure 15.5: The minimum attainable values of the total ΔV for a transfer to the polar lunar orbit with an altitude of 10,000 km in case of the launch on different days of January 2028 calculated with PSO.

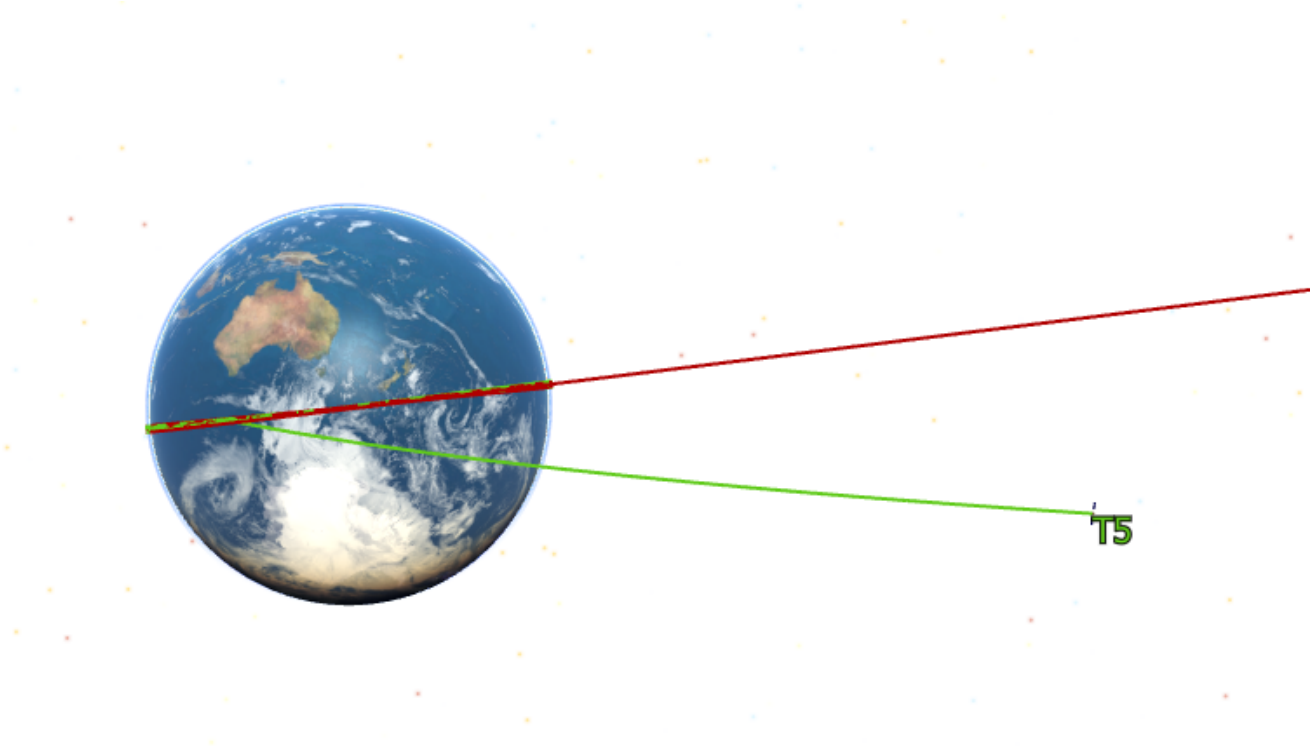


Figure 15.6: Trajectory followed by two satellites form TC5. One departing the 1^{st}. of January 2028 (red) and the other one departing on the 5^{th}.

Chapter 16

Conclusions and future work

This thesis thoroughly explored the methodology to find the best Earth-Moon direct two-impulse transfers, from a user-defined Earth-bounded orbit to a user-defined Moon-bounded orbit, considering a user-selected dynamics model. By systematically addressing the research subquestions proposed, an effective approach has been developed and verified across different test cases.

The main research question is the following:

What is the most effective approach to find the optimal Earth-Moon direct two-impulse transfers, from a user-defined Earth-bounded orbit to a user-defined Moon-bounded orbit including a user-selected dynamics model?

And the proposed sub-questions are:

1. Which parameters to optimize play a significant role in an auxiliary Earth-Moon transfer?
2. Is it feasible to directly use a gradient-based optimizer to optimize direct transfers to the Moon?
3. How can an accurate initial guess be obtained to enhance the optimization process?
4. If a global-search algorithm is necessary for obtaining a good initial guess, can a gradient-based optimizer effectively refine the solution?
5. Are there any advantages in calculating the initial guess using the CR3BP? Are there any limitations?
6. Is tuning the optimizers necessary? How can it be done effectively?
7. Can a gradient-based optimizer reach a solution with a full dynamics model, even if the initial guess was calculated without considering these perturbations?

Firstly, it was found that the most crucial parameters to optimize the total magnitude of the required maneuvers for these transfers are the satellite's position in the specified orbit, the time of flight, and the epoch of the maneuvers. To ensure user-friendliness, the initial and final states were computed from the Keplerian elements of the desired orbit provided by the user. During the test cases, it was observed that orbital parameters such as the right ascension of the ascending node and the argument of periapsis, can also have an impact on the required propellant, even when their effect is lower than the selected parameters.

After analyzing various optimizers, a hybrid optimization approach was chosen, using a global Particle Swarm Optimization algorithm to obtain an initial guess, and a gradient-based algorithm (SLLSQP) to refine and correct the solutions under newly introduced perturbations. Moreover, it was found that both of the optimizers needed to be tuned to ensure robustness. Using a PSO algorithm to search through the large defined space of solutions has proven to be the most effective and direct way of generating the initial

guess. This first optimization and its required calculations were conducted using the SEMpy environment (in Python). To obtain the required maneuvers for each particle, Lambert's problem was solved, and a multiple-shooting method was employed to account for the Moon's point-mass gravity field. Additionally, it was found that including Earth's spherical harmonics during the propagation of the satellite's trajectory in the initial orbit was necessary to obtain a suitable initial guess that allowed the SLLSQP optimizer to correct the trajectory under the new chosen dynamics model.

The gradient-based optimizer proved highly useful not only in refining the solutions obtained by PSO using the same dynamics model, but specially in correcting for additional user-specified perturbations that could not be directly accounted for during the generation of solutions in PSO.

The optimization was conducted in the EME2000 reference frame for both parts of the project, because, during the research, an unexpected challenge arose regarding modeling the Circular Restricted Three-Body Problem and transforming it into an inertial reference system, leading to divergences in the propagations in some cases.

Lastly, it has to be highlighted that both parts of the optimization procedure are independent, making it a versatile and valuable tool for primary optimization, both with and without perturbations. Overall, this thesis makes a significant contribution to the field of astrodynamics and space mission planning, offering valuable insight into optimal Earth-Moon transfers.

During the analysis of the test cases, several important findings emerged, and they will be presented as valuable recommendations for using the optimizers in this specific problem.

16.1 Recommendation on the optimizers usage

From the user's perspective, there are several important conclusions and lessons to highlight.

Regarding the PSO, it was found that visualizing the results is crucial, as PSO can identify multiple local minima, which can be valuable for mission planning. Additionally, the user-defined boundaries have a significant impact on the solution if the optimum is close to them, as particles might not explore those limits if no suitable matches are found in the initial generations, hence the need to increase the search space. Concerning the search space, some of the orbital configurations showed interesting trends that could be used to decrease its limits, as the optimual particles are found within specific sections of it.

Although the tuning process was done carefully to ensure a generic tool, it was essential to consider CPU time as an important factor, which affected the decision of the optimization parameters. However, in some specific cases, the default parameters selected, particularly the population size and the number of generations, were not high enough to reach the optimal solution. Therefore, some complex orbit configurations might require increasing these values.

Furthermore, it should be noted that the propagation can be performed either without spherical harmonics, considering the point-mass gravity field of Earth and the Moon, or with spherical harmonics. However, the inclusion of spherical harmonics will only affect the satellite's path around Earth and not the trajectory towards the Moon. As a result, the model is not consistent throughout the entire trajectory.

Lastly, the user should be aware of the software's limitations due to certain assumptions made. Firstly, the Lambert + multiple-shooting method may struggle to find a solution when the initial guess from the Lambert technique is too distant from the trajectory that incorporates the Moon's point-mass gravity field, particularly when the satellite approaches the Moon closely. Consequently, these particles had to be excluded. An analysis was conducted to determine when to eliminate them to minimize the chances of such particles being optimal solutions, but it remains an important consideration.

Additional conclusions have also been drawn from the optimization using the SLLSQP optimizer. It is noteworthy to observe that the optimizer successfully finds a solution when the initial guess results in an

orbit around the Moon. However, if the initial guess does not lead to such an orbit, the optimizer fails to find a suitable solution during the optimization process, which happens more frequently where the desired orbit around the Moon has a large semi-major axis. Furthermore, once the optimizer encounters a solution where this happens, it is unable to correct the situation and get back on track. To address this issue, adjustments to the optimizer's parameters are necessary. As with previous optimizations, the parameters have been fine-tuned, but further adjustments might be required to improve performance for specific cases. For the SLLSQP optimizer, the scale of the fitness function plays a crucial role, and it is the parameter that should be studied first if no convergence is reached and an error is given by the optimizer. If the initial guess results in an orbit around the Moon but the optimization leads to a solution that flies away from it, the DDPP parameter should be adjusted, in general decreasing it.

16.2 Future work

Several aspects have been identified as valuable continuations of the conducted work.

Firstly, it would be highly beneficial to further study the transformation between the CR3BP and the ephemeris model to enable the inclusion of CR3BP periodic orbits as potential objective orbits in the tool. Although not in a direct way, they could be incorporated with careful consideration and by using SEMpy's Halo and NRHO generator and corrector into a higher-fidelity N-body ephemeris model.

Another interesting addition would be to include spherical harmonics in the Lambert + multiple-shooting method, which will lead to a more accurate initial guess, but at the cost of a higher computational time.

Furthermore, as mentioned earlier, a decision had to be made regarding the selection of parameters to optimize. However, it was discovered that certain other parameters, despite having a lower degree of influence, still play a role in the required ΔV for the trajectory. Therefore, it would be highly beneficial to consider including these parameters as additional options for optimization in the PSO.

In addition, the study revealed challenges in simultaneously correcting all constraints during the SLLSQP optimization, particularly for the Ω. A useful approach has been to first correct without considering the constraint in Ω and then subsequently address the Ω correction. However, currently, there is no direct method to achieve this, hence it is required to conduct two separate optimizations. It would be highly valuable to incorporate an option in the optimization process that allows to include sequential optimizations with the solution form the previous one.

Moreover, considering the initial epoch as a design parameter in the gradient-based optimization could be very useful, as the epoch is currently being optimized for a trajectory without perturbations. Although it is already available with a flag, it does not work as expected, hence requiring further analysis. This analysis could involve a change in the optimization formulation. For example, if the propagation in this optimization were done backward, the likelihood of the satellite reaching an orbit around Earth would be higher than around the Moon, thereby increasing the chances of finding a solution.

Lastly, after generating these tools, the effect of the optimized parameters on the final orbits reached and the effect of the initial guesses on the performance of the optimizer should be analyzed following a thorough sensitivity study.

Appendix A

Plots from the test cases

A.0.1 Test case 1

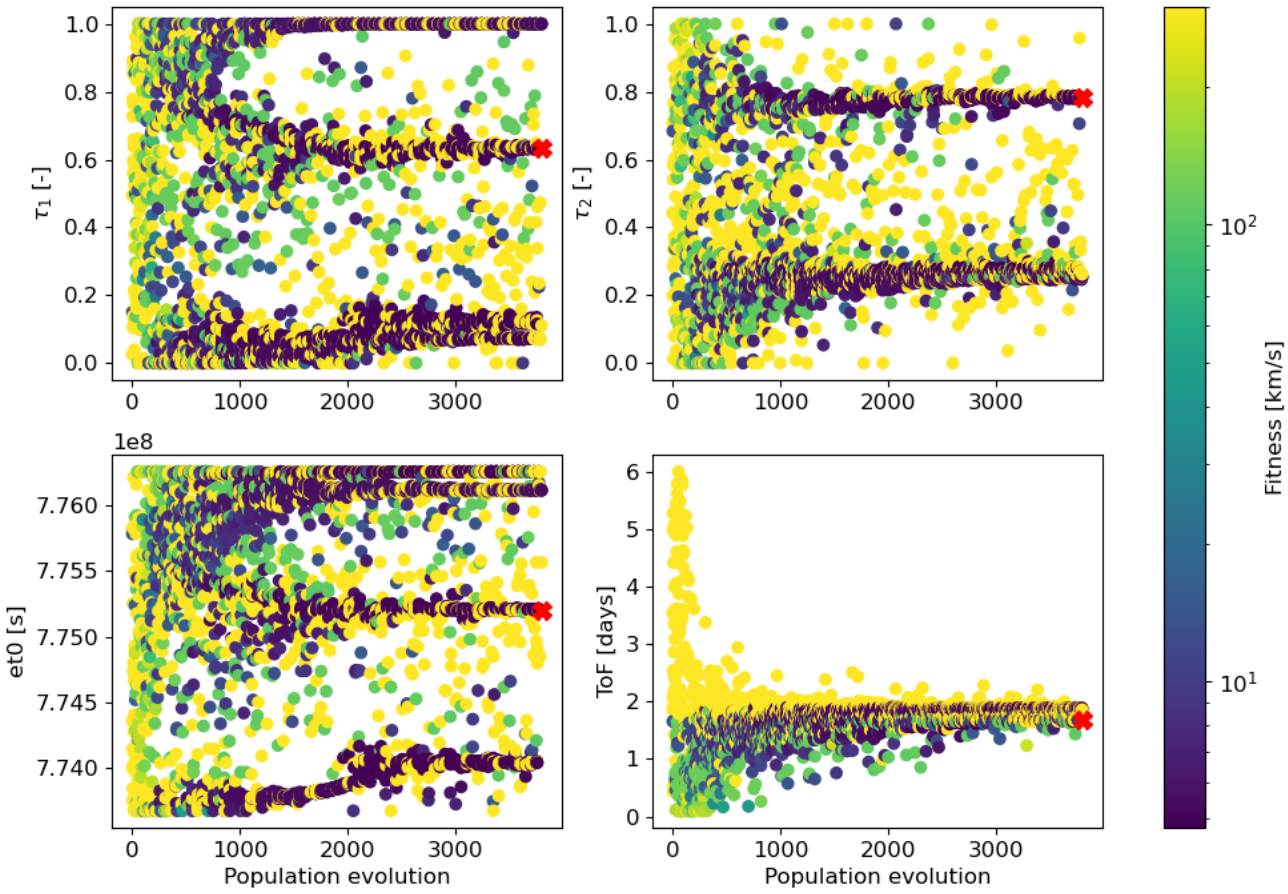


Figure A.1: Evolution of the whole population for test case 1 and seed 10.

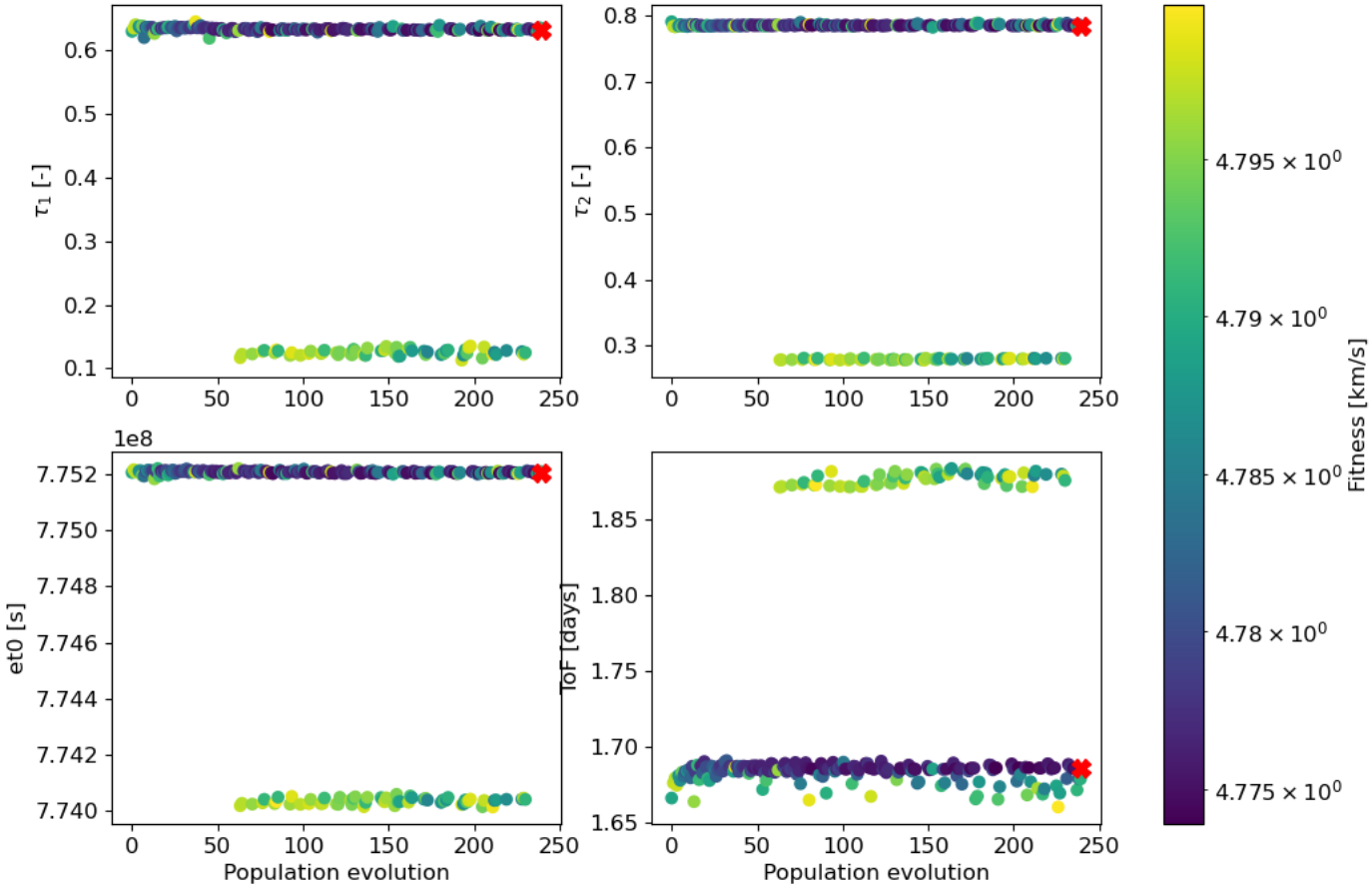


Figure A.2: Evolution of the population with a fitness value lower than 4.8 km/s for test case 1 and seed 10.

A.0.2 Test case 2

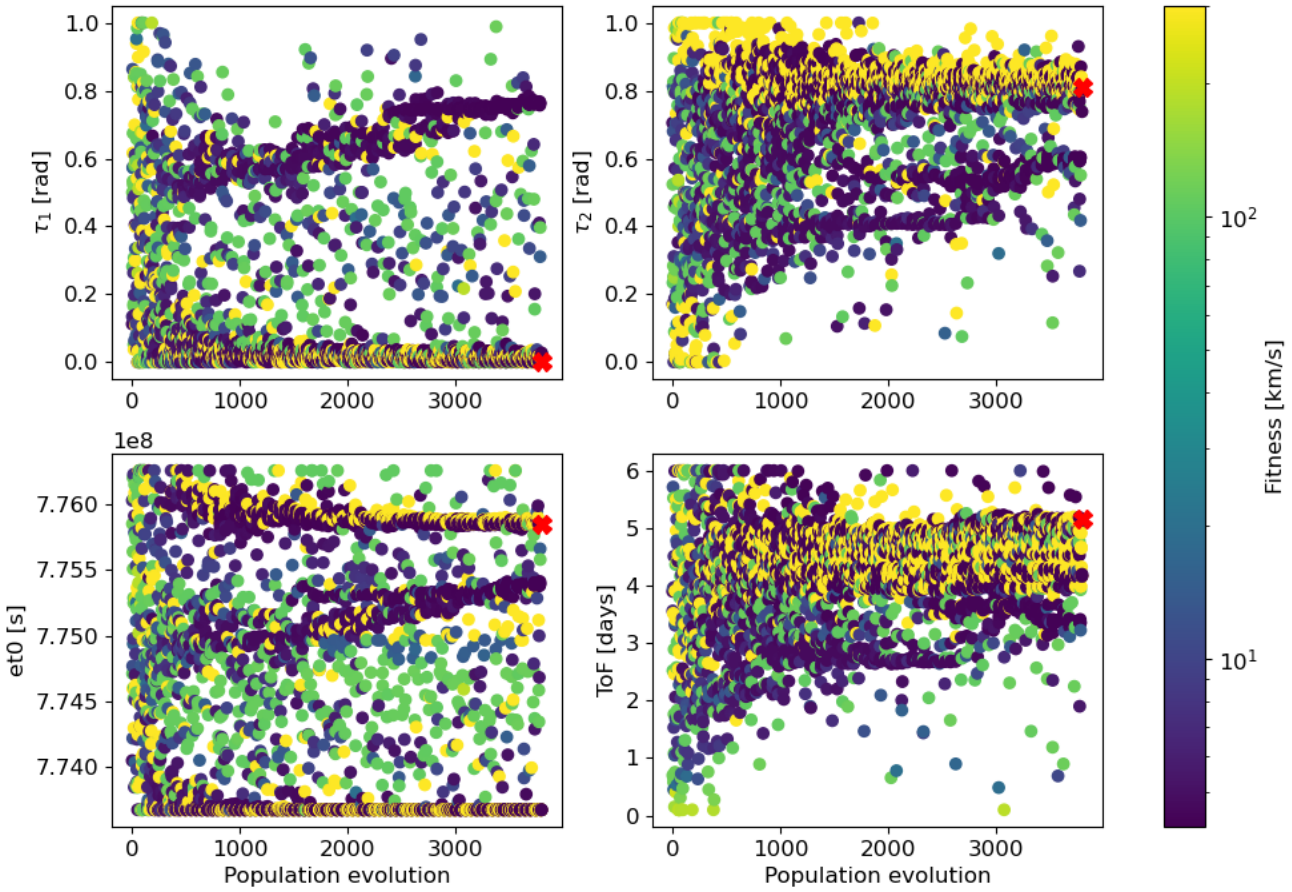


Figure A.3: Evolution of the whole population for test case 2 and seed 40.

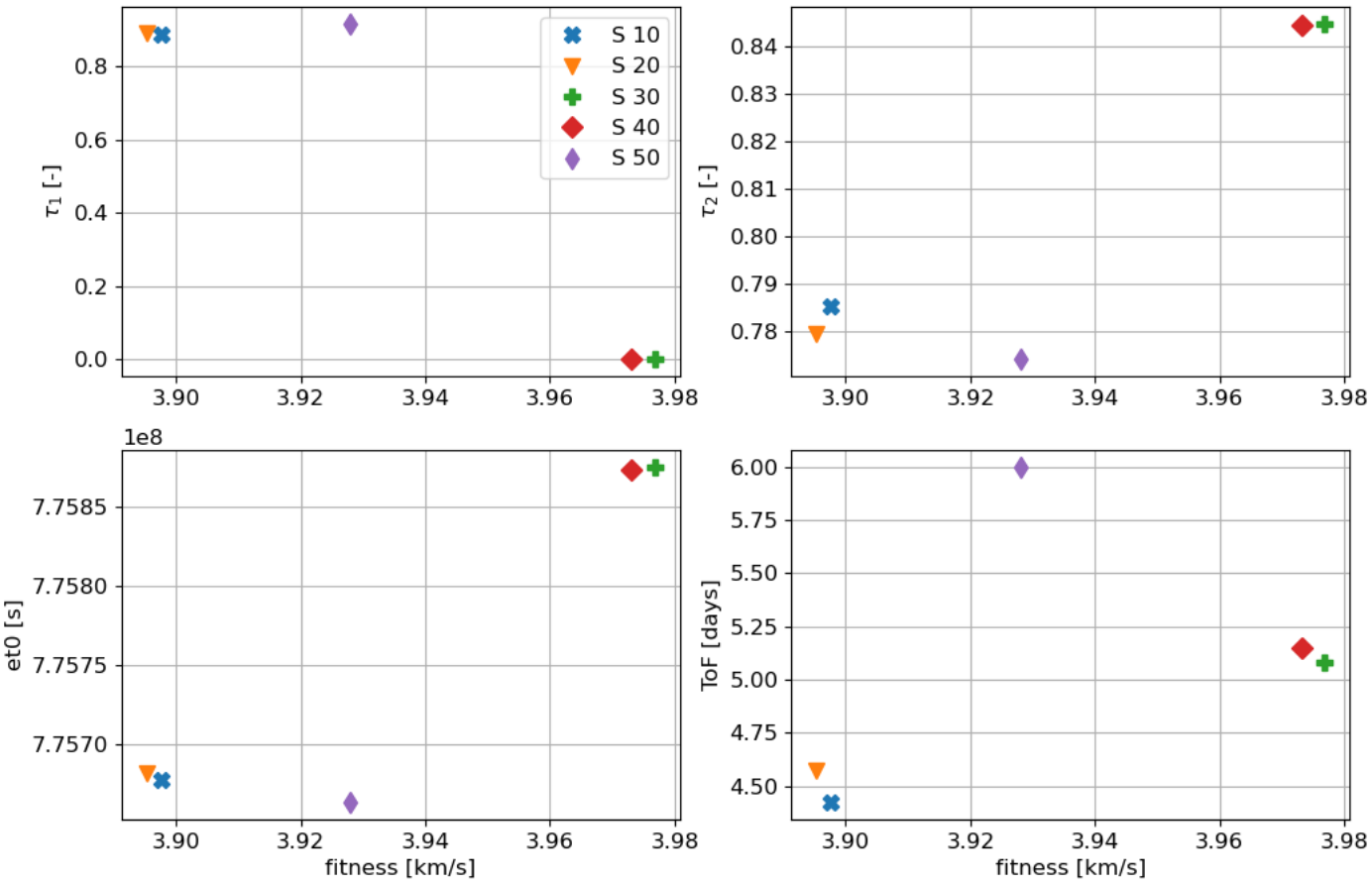


Figure A.4: Design parameters for the optimum particles found for each five seeds in test case 2. Population size: 50, number of generations: 100.

A.0.3 Test case 3

Seed 30

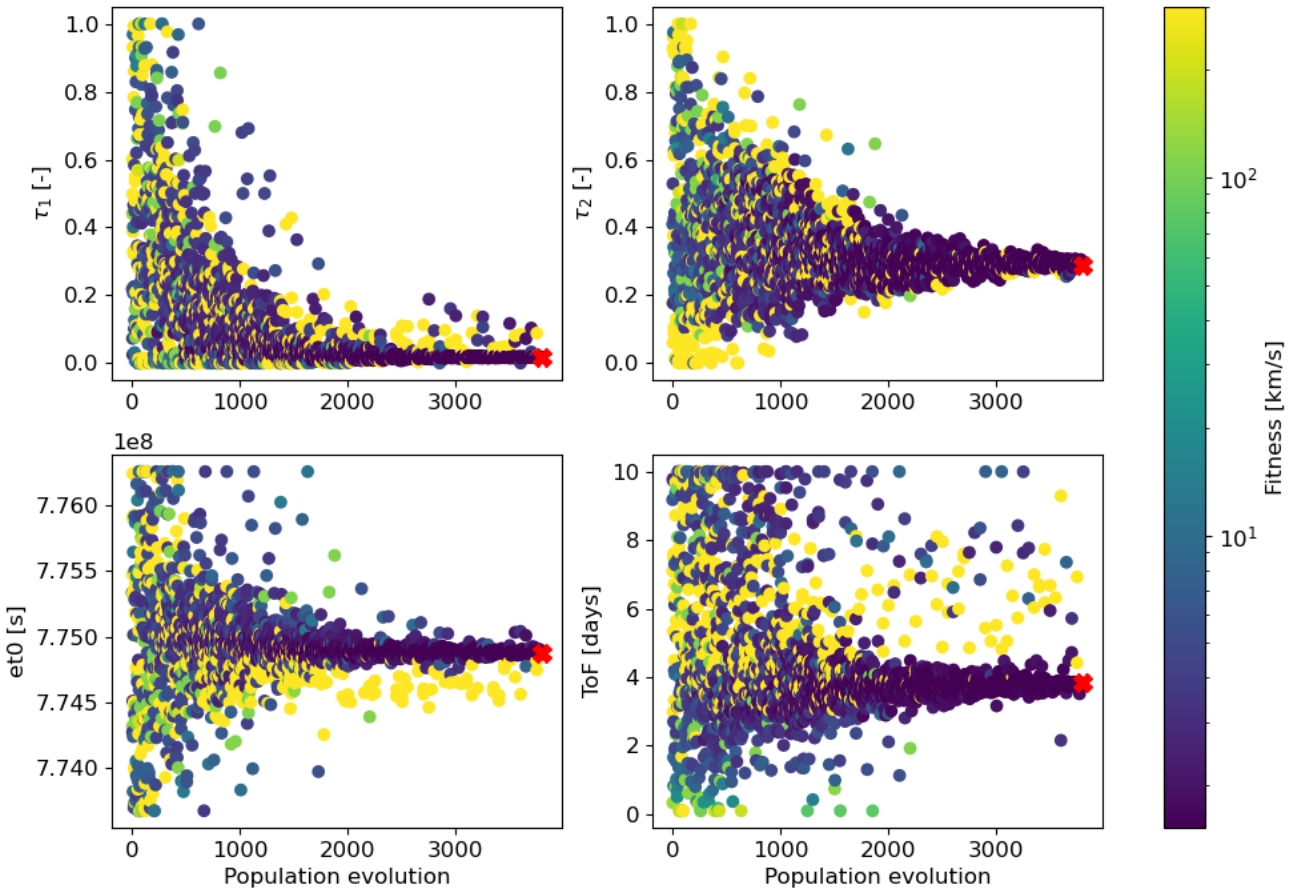


Figure A.5: Evolution of the whole population for test case 3 and seed 30.

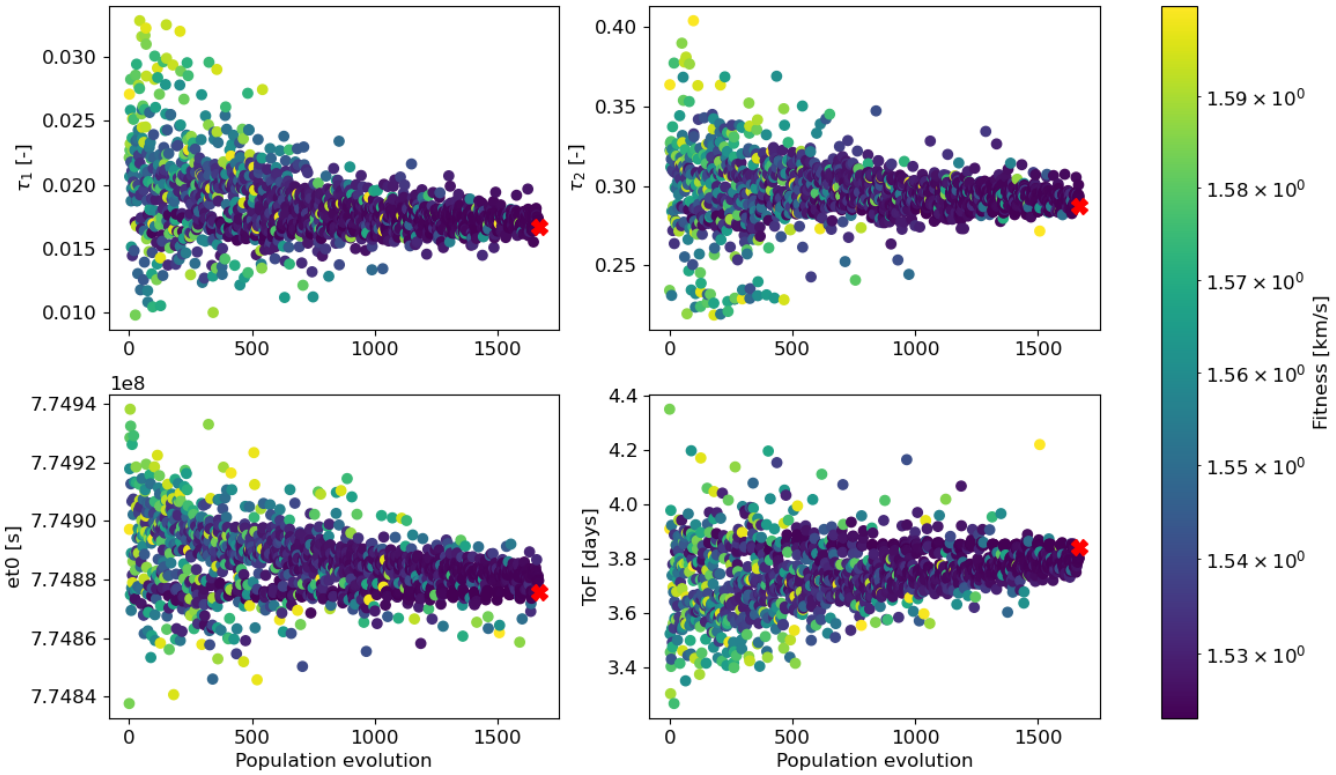


Figure A.6: Evolution of the population with a fitness value lower than 4.8 km/s for test case 3 and seed 30.

Seed 40

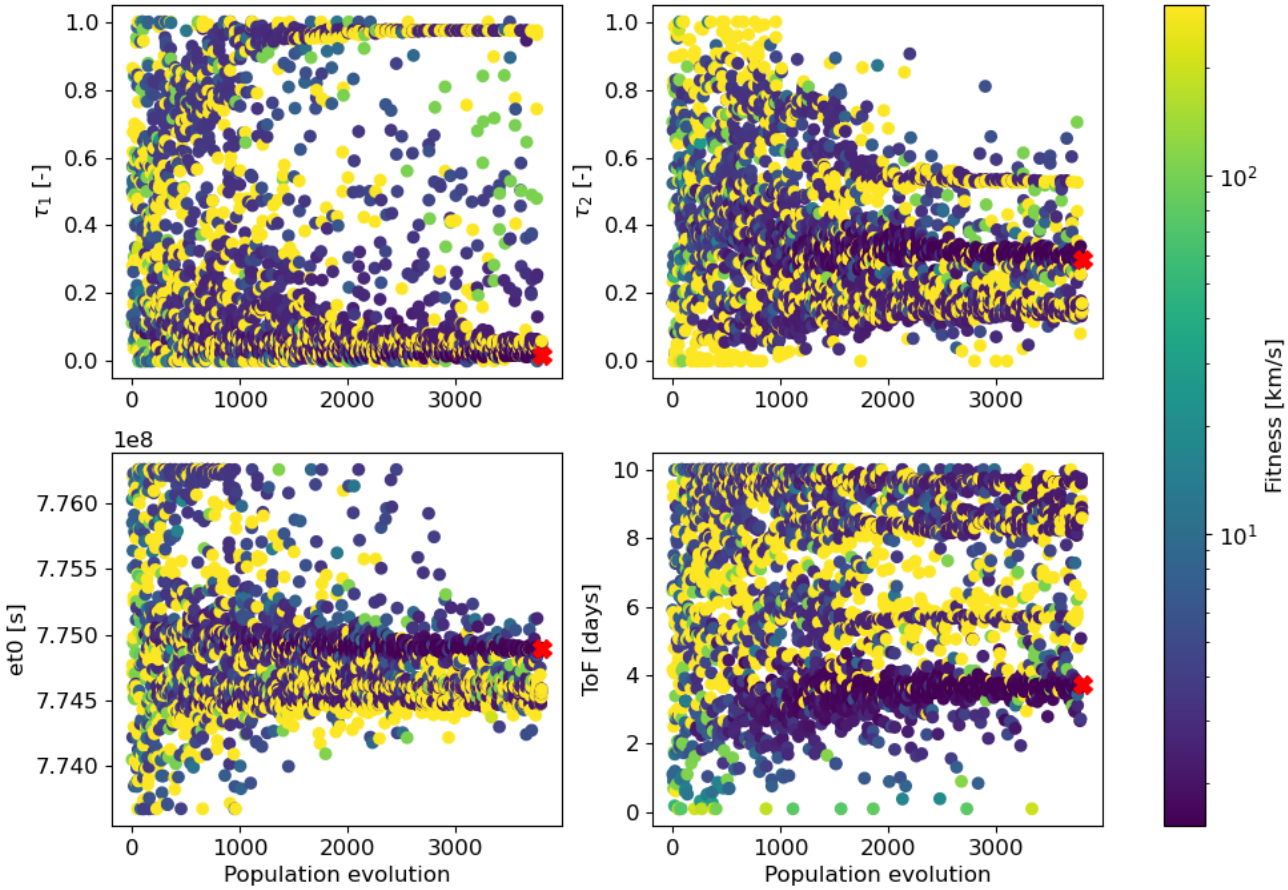


Figure A.7: Evolution of the whole population for test case 3 and seed 40.

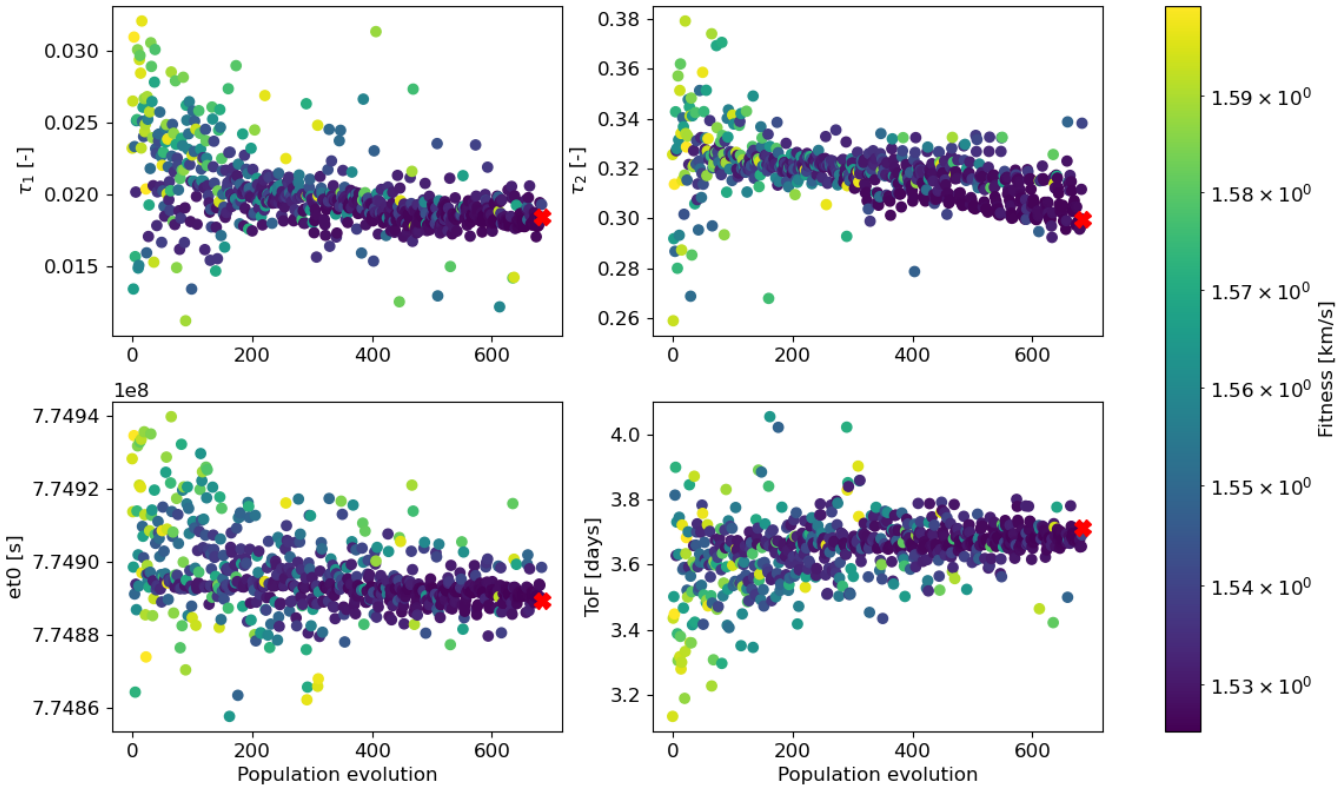


Figure A.8: Evolution of the population with a fitness value lower than 4.8 km/s for test case 3 and seed 40.

A.0.4 Test case 4

Seed 30

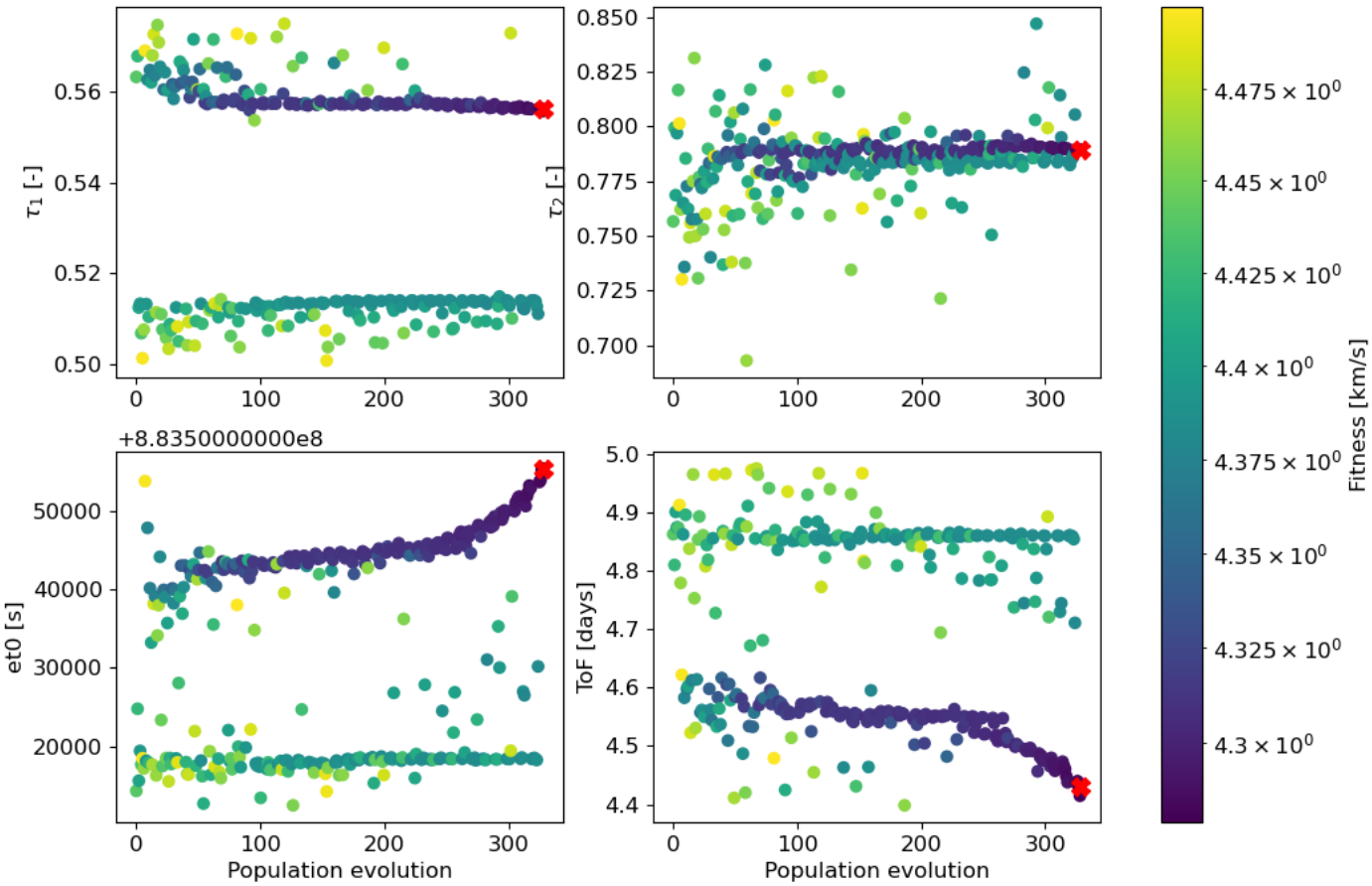


Figure A.9: Evolution of the population with a fitness value lower than 4.5 km/s for test case 4 and seed 30.

A.1 OPT-EM input file

```
OPT_EM_SETUP
          TITLE     'TEST'
%-------------------------------------------------------------------------------

SATELLITE
          NAME      LRO
          SAT_ID    9800101
          MASS      1500   % kg
          AREA_CD   3.5    % m^2
          AREA_CR   10.0   % m^2
          CD        2.3    SIGMA  0.10
          CR        1.3    SIGMA  0.10
END_SATELLITE

%-------------------------------------------------------------------------------

FORCE_MODEL
 INTEGRATION
          REL_ERROR   1.0D-14
          ABS_ERROR   1.0D-10
          CENTER    EARTH   % The center body needs an ephemeris file (in the .bat) and a reference frame
 GRAVITY
          EARTH   ORDER  10 10
                  REFSYS  EARTH_BODY_FIXED
          MOON    ORDER  8   8
                  REFSYS  MOON_BODY_FIXED_JPLDE
 END

 INDIRECT_ACCEL          TRUE
 SOLAR_RAD_PRES         TRUE
 ATMOSPHERIC_DRAG   TRUE     % only possible for Earth as center body

END_FORCE_MODEL

%-------------------------------------------------------------------------------

SATELLITE_TRAJECTORY

 EPOCH_TIME_SYSTEM    UTC
 COORDINATE_SYSTEM    EARTH     EARTH_MEAN_EQUEQX_OF_J2000    % EME2000

 EPOCH    2024/07/25   13:33:38.788300

 STATE_VECTOR   % m, m/s

 6621008.366666666232049465179443359375   0.00000000000000000000000000000000
 0.00000000000000000000000000000000
 0.00000000000000000000000000000000   6825.21000124450074508786201477050 8
 3690.37728101070115371840074658393 9
```

```
END_SATELLITE_TRAJECTORY

%-------------------------------------------------------------------------------

PREDICTION

 TIME_SYSTEM        UTC
 COORDINATE_SYSTEM   MOON   MOON_TRUE_EQUIAU_OF_DATE

 FROM   2020/07/09    11:52:18.315912
 TO     2020/07/12    08:46:25.744557

 TIME_STEP   60.0   % [s]

 OPTIONS
          CARTESIAN
          KEPLERIAN
          TRACKING_DATA
 END_OPTIONS

 END_PREDICTION

 %-------------------------------------------------------------------------------

OPTIMIZATION

TIME_SYSTEM                 UTC
COORDINATE_SYSTEM       MOON   MOON_TRUE_EQUIAU_OF_DATE_JPLDE

OPTIONS
 CARTESIAN      % Important to get Y_last
 KEPLERIAN      % Important to get Kep_last
 WRITE_OPT      % Write optimization file
END_OPTIONS


 MANEUVERS
          DURATION                0.    % [sec]
          MASS_FLOW               0.    % [kg/s]
          ESTIMATE                NO
          SIGMA                   0.05
          REFERENCE_SYSTEM        INERTIAL


INITIAL_GUESS
          NOP     8               % Number of parameters to optimize
          TIME1   3330.1470390022973   % s
          V1      2194.208242905746786277632054407    -1798.119227230365879677265184000
                  -1712.104801257593678087687294465   % m/s
          TOF     146850.74397220812   % s
```

```
    V2    -1266.060036178477288260069144599   0281.10343404887327967145438378
          -0677.29994831052964254780590636l    % m/s
    EPOCH  FALSE   2024/07/25   13:33:38.788300   % UTC

LIMITS  % +/-
        TIME1   1000   % [s]
        V1      100    % [m/s]
        TOF     1000   % [s]
        V2      100    % [m/s]
        EPOCH   0.000001     % [days]

OBJECTIVE_ORBIT

        SMA   1837.4000000000003    1     TRUE  % [km] Objective semimajor axis around MOON, Allowed
Delta
        ECC     0.              0.001  TRUE
        INCL    85.0            1      TRUE  % [deg]
        RAAN    0.              10     TRUE  % [deg]
        OMEGA   0.              10     TRUE  % [deg]  Argument of perigee

PARAMETERS
        T_PROP      600                % time of propagation after second maneuver [s]
        LOOP_IN     10000              % length of inner loop
        ACC         0.000001           % Accuracy for equality and inequality - the lower the number,
                                       % the more it will look for convergence
        ITMAX       20000              % maximum number of iterations
        DDPP        1.0E-5  1.0E-5  1.0E-5 1.0E-5   1.0E-5 % Perturbations for the derivatives (t1, V1, tof, V2)
        WTIT         ALL 7     200  10 100 1 100  500 100        % Cost function scale for outer loop - If ONE,
        % you just give one number, in ALL,  you give first how many options to give, then the options
        PAR_SCALE       ONE 1    % t1, v1, tof, v2
        CON_SCALE       ONE 1     % First option, ONE or ALL. If ONE, only one value needs to be given and it
        % will be set for all cases If ALL is given, a scaling factor  for each Objective orbit parameters set to
        % TRUE is required (if INCL=0, RAAN no, if E=0, OMEGA no)

END_OPT_EM_SETUP
```

Bibliography

Abdel-Basset, Mohamed, Laila Abdel-Fatah, and Arun Kumar Sangaiah (2018). "Metaheuristic algorithms: A comprehensive review". In: *Computational intelligence for multimedia big data on the cloud with engineering applications*, pp. 185–231.

Archinal, Brent Allen et al. (2011). "Report of the IAU working group on cartographic coordinates and rotational elements: 2009". In: *Celestial Mechanics and Dynamical Astronomy* 109, pp. 101–135.

Belbruno, E and J Carrico (2000). "Calculation of weak stability boundary ballistic lunar transfer trajectories". In: *Astrodynamics Specialist Conference*, p. 4142.

Berry, R. (1970). "Launch window and translunar, lunar orbit, and transearth trajectory planning and control for the Apollo 11 lunar landing mission". In: *8th Aerospace Sciences Meeting*, p. 24.

Biesbroek, Robin and Guy Janin (2000). "Ways to the Moon". In: *ESA bulletin* 103, pp. 92–99.

Boeringer, Daniel W. and Douglas H. Werner (2004). "Particle swarm optimization versus genetic algorithms for phased array synthesis". In: *IEEE Transactions on antennas and propagation* 52.3, pp. 771–779.

Bratton, Daniel and James Kennedy (2007). "Defining a standard for particle swarm optimization". In: *2007 IEEE swarm intelligence symposium*. IEEE, pp. 120–127.

Brumberg, V.A. and E. Groten (2001). "IAU resolutions on reference systems and time scales in practice". In: *Astronomy & Astrophysics* 367.3, pp. 1070–1077.

Ceriotti, Matteo, Camilla Colombo, and Massimiliano Vasile (Jan. 2006). "Trajectory design and optimisation for lunar transfer". In: *1st Hellenic-European Student Space Science and Technology Symposium*.

Chapman, Allan (Feb. 2009). "A new perceived reality: Thomas Harriot's Moon maps". In: *Astronomy and Geophysics* 50.1, pp. 1.27–1.33. ISSN: 1366-8781. DOI: 10.1111/j.1468-4004.2009.50127.x.

Chobotov, Vladimir A. (2002). *Orbital mechanics*. 3rd. Reston, VA: AIAA Education Series.

Chong, Edwin K.P. and Stanislaw H. Zak (2013). *An introduction to optimization*. Vol. 75. John Wiley & Sons.

Clerc, Maurice (1999). "The swarm and the queen: towards a deterministic and adaptive particle swarm optimization". In: *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*. Vol. 3. IEEE, pp. 1951–1957.

Conte, Davide (May 2014). "Determination of Optimal Earth-Mars Trajectories to Target the Moons of Mars". PhD thesis. DOI: 10.13140/RG.2.2.22494.74563.

Cowan, Kevin (2021). "Three-body problem". In: *Delft University of Technology*. [Power Point slides].

Dahlke, Jacob A. (2018). "Optimal trajectory generation in a dynamic multi-body environment using a pseudospectral method". In: *Theses and Dissertations*. URL: https://scholar.afit.edu/etd/1764.

Davies, Melvyn E. et al. (1995). "Report of the IAU/IAG/COSPAR working group on cartographic coordinates and rotational elements of the planets and satellites: 1994". In: *Celestial Mechanics and Dynamical Astronomy* 63, pp. 127–148.

De La Torre, D., R. Flores, and E. Fantino (2018). "On the solution of Lambert's problem by regularization". In: *Acta Astronautica* 153, pp. 26–38.

Eberhart, Russ C. and Yuhui Shi (2000). "Comparing inertia weights and constriction factors in particle swarm optimization". In: *Proceedings of the 2000 congress on evolutionary computation. CEC00 (Cat. No. 00TH8512)*. Vol. 1. IEEE, pp. 84–88.

Ely, Todd A. and Erica Lieb (2006). "Constellations of elliptical inclined lunar orbits providing polar and global coverage". In: *the Journal of the Astronautical Sciences* 54.1, pp. 53–67.

Frazier, Peter I. (2018). "Bayesian optimization". In: *Recent advances in optimization and modeling of contemporary problems*. Informs, pp. 255–278.

Freitas, Diogo, Luiz Guerreiro Lopes, and Fernando Morgado-Dias (2020). "Particle swarm optimisation: a historical review up to the current developments". In: *Entropy* 22.3, p. 362.

Goldberg, David E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. New York: Addison-Wesley.

Gómez, G. and J.M. Mondelo (2001). "The dynamics around the collinear equilibrium points of the RTBP". In: *Physica D: Nonlinear Phenomena* 157.4, pp. 283–321. ISSN: 0167-2789. DOI: 10.1016/S0167-2789(01)00312-8.

Gupta, Maaninee (2020). "Finding Order in Chaos: Resonant Orbits and Poincaré Sections". PhD thesis. Purdue University Graduate School.

Hall, R. Cargill (1977). *Lunar impact: A history of Project Ranger*. Vol. 4210. Scientific, Technical Information Office, National Aeronautics, and Space.

Hansen, Nikolaus (June 2007). "The CMA Evolution Strategy: A Comparing Review". In: vol. 192, pp. 75–102. ISBN: 978-3-540-29006-3. DOI: 10.1007/3-540-32494-1_4.

Hansen, Nikolaus and Andreas Ostermeier (2001). "Completely Derandomized Self-Adaptation in Evolution Strategies". In: *Evolutionary Computation* 9.2, pp. 159–195. DOI: 10.1162/106365601750190398.

Hohenkerk, C.Y. et al. (1992). "Celestial reference systems". In: *Explanatory Supplement to the Astronomical Almanac*, pp. 111–114.

Honniball, C. I. et al. (2022). "Regional Map of Molecular Water at High Southern Latitudes on the Moon Using 6 m Data From the Stratospheric Observatory for Infrared Astronomy". In: *Geophysical Research Letters* 49.9. DOI: https://doi.org/10.1029/2022GL097786.

Horn, M.K. and K.H. Well (1983). "Numerical solution of piecewise continuous trajectory optimization problems". In: *IFAC Proceedings Volumes* 16.8, pp. 81–89.

Hosseini, S. (Jan. 2011). "An Exploration Of Fuel Optimal Two-impulse Transfers To Cyclers in the Earth-Moon System". MA thesis. University of California, Irvine.

Innocente, Mauro Sebastián and Johann Sienz (2011). "Particle swarm optimization with inertia weight and constriction factor". In: *Proceedings of the International conference on swarm intelligence (ICSI11)*.

ISAE-SUPAERO (2023). *Local frames*. [Online; accessed July 10, 2023]. URL: https://sourceforge.isae.fr/svn/dcas-soft-espace/support/softs/CelestLab/trunk/help/en_US/scilab_en_US_help/Local%20frames.html.

Jacchia, Luigi Giuseppe (1977). "Thermospheric temperature, density, and composition: New models". In: *SAO Special Report# 375 (1977)* 375.

Kara, Ozan and Çağrı Kılıç (2015). "Comprehensive study of small satellite Moon missions: Architecture design, electric propulsion system optimization and cost analysis". In: DOI: 10.13140/RG.2.1.2770.2487.

Kemble, Stephen (Jan. 2006). *Interplanetary Mission Analysis and Design*. DOI: 10.1007/3-540-37645-3.

Kennedy, James and Rui Mendes (Feb. 2002). "Population structure and particle swarm performance". In: vol. 2, pp. 1671–1676. ISBN: 0-7803-7282-4. DOI: 10.1109/CEC.2002.1004493.

Lagier, Roland (2016). *Ariane 5 User's Manual Issue 5 Revision 2*. Arianespace. URL: https://www.arianespace.com/wp-content/uploads/2011/07/Ariane5_Users-Manual_October2016.pdf.

Lange, T.J. de et al. (2008). "Conceptual Design of a Streamlined Mission to Compete for the Google Lunar X PRIZE". In: *To Moon and Beyond*. Deutsche Geselschaft fur Luft-und Raumfahrt, pp. 1–10.

McCarthy, Brian P. (Jan. 2019). "Characterization of Quasi-Periodic Orbits for Applications in the Sun-Earth and Earth-Moon Systems". In: DOI: 10.25394/PGS.7423658.v1.

Miller, James K. and Edward A. Belbruno (Jan. 1991). "A method for the construction of a lunar transfer trajectory using ballistic capture". In: *Spaceflight Mechanics 1991*, pp. 97–109.

Montero, Elizabeth, Marıa-Cristina Riff, and Bertrand Neveu (2014). "A beginner's guide to tuning methods". In: *Applied Soft Computing* 17, pp. 39–51.

Mooij, Erwin and Dominic Dirkx (2022). "Propagation and Optimisation in Astrodynamics - Global Optimisation". In: *Delft University of Technology*. [Power Point slides].

NASA (2023). *Apollo Expeditions to the Moon*. URL: https://history.nasa.gov/SP-350/toc.html (visited on 07/26/2023).
– (2021). *Moon Fact Sheet*. Accessed: 1 August 2023. Retrieved from https://nssdc.gsfc.nasa.gov/planetary/factsheet/moonfact.html.
Parker, Jeffrey (2018). "ASEN 5050 - Spaceflight dynamics - Solar Radiation Pressure". In: *University of Colorado - Boulder*. [Power Point slides].
Parker, Jeffrey and R. Anderson (2013). *Low-Energy Lunar Trajectory Design*. Vol. 12. DESCANSO, JPL Deep-Space Communications and Navigation Series. Jet Propulsion Laboratory, California Institute of Technology.
Pavlis, Nikolaos K. et al. (2012). "The development and evaluation of the Earth Gravitational Model 2008 (EGM2008)". In: *Journal of geophysical research: solid earth* 117.B4.
Peet, Matthew M. (2018). "Rendezvous and Targeting - Lambert's Problem". In: *Arizona State University*. [Power Point slides]. URL: http://control.asu.edu/Classes/MAE462/462Lecture10.pdf.
Russell, C. and V. Angelopoulos (2011). "The ARTEMIS mission". In: *The ARTEMIS mission*. Springer, pp. 3–25.
Sengupta, Saptarshi, Sanchita Basak, and Richard Alan Peters (2018). "Particle Swarm Optimization: A survey of historical and recent developments with hybridization perspectives". In: *Machine Learning and Knowledge Extraction* 1.1, pp. 157–191.
Smith, C. et al. (1989). "Mean and apparent place computations in the new IAU system. I-The transformation of astrometric catalog systems to the equinox J2000. 0." In: *The Astronomical Journal* 97, pp. 265–279.
Snyman, Jan A., Daniel N. Wilke, et al. (2005). *Practical mathematical optimization*. Springer.
Standish, E. Myles, James G. Williams, et al. (2006). "Orbital ephemerides of the Sun, Moon, and planets". In: *Explanatory supplement to the astronomical almanac*, pp. 279–323.
Standish Jr, E.M., Michael S.W. Keesey, and X.X. Newhall (1976). *JPL development ephemeris number 96*. Tech. rep. NASA-CR-147923.
Storn, Rainer and Kenneth Price (1997). "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces". In: *Journal of global optimization* 11, pp. 341–359.
Tselousova, A., M Shirobokov, and S. Trofimov (2019). "Direct two-impulse transfers from a low-earth orbit to high circular polar orbits around the moon". In: *AIP Conference Proceedings*. Vol. 2171. 1. AIP Publishing.
Vikhar, Pradnya A. (2016). "Evolutionary algorithms: A critical review and its future prospects". In: *2016 International Conference on Global Trends in Signal Processing, Information Computing and Communication (ICGTSPICC)*, pp. 261–265. DOI: 10.1109/ICGTSPICC.2016.7955308.
Villac, B.F. and D.J. Scheeres (2003). "New class of optimal plane change maneuvers". In: *Journal of guidance, control, and dynamics* 26.5, pp. 750–757.
Wikimedia Commons (2008). *Anomalies*. Wikimedia Commons. Retrieved from https://commons.wikimedia.org/wiki/File:Anomalies.PNG.
Yazdi, Kian and Ernst Messerschmid (2004). "Analysis of parking orbits and transfer trajectories for mission design of cis-lunar space stations". In: *Acta Astronautica* 55.3. New Opportunities for Space. Selected Proceedings of the 54th International Astronautical Federation Congress, pp. 759–771. ISSN: 0094-5765.
Zhang, Xuanping et al. (2005). "A modified particle swarm optimizer for tracking dynamic systems". In: *Advances in Natural Computation: First International Conference, ICNC 2005, Changsha, China, August 27-29, 2005, Proceedings, Part III 1*. Springer, pp. 592–601.