Global Optimization of Low-Thrust Interplanetary Trajectories Using a Machine Learning Surrogate

Pablo Gómez Pérez

Global Optimization of Low-Thrust Interplanetary Trajectories Using a Machine Learning Surrogate

by

Pablo Gómez Pérez

in partial fulfillment of the requirements for the degree of

Master of Science

in Aerospace Engineering

at the Delft University of Technology,

to be defended publicly on May 29, 2020 at 13:00 hours.

Supervisor:	Ir. K.J. Cowan, MBA,	TU Delft
	Y. Liu,	TU Delft
Thesis committee:	Prof. dr. ir. P.N.A.M. Visser,	TU Delft
	Dr. S. Speretta	TU Delft

The cover image shows an artist's view of ESA's BepiColombo at Mercury.
Image credit: ESA

Delft University of Technology

Summary

In this thesis, a new method to approximate the cost function of Low-Thrust, Multiple-Gravity-Assist interplanetary trajectories using a Machine Learning surrogate is proposed. This method speeds up the optimization process without fine tuning of the surrogate parameters for every individual case. The computational cost of obtaining training data was identified as the main limitation when using Machine Learning methods for this purpose. Therefore, the surrogate was built with an Online Sequential Extreme Learning Machine Multi-Agent System (OS-ELM-MAS) due to its theoretical good performance when the training data is limited.

A high-fidelity global optimization problem was implemented, and a method to include the surrogate during the optimization process was designed. This method does not require specialized optimization algorithms. The parameters that control the interaction between the surrogate and the optimization process were identified and a procedure to obtain the best values was designed and applied. The final results show that the use of the surrogate improves the optimization results when evaluations of the cost function are computationally expensive. However, the values of the parameters that control the interaction between the surrogate and the optimization algorithm had to be carefully selected. The search for a general procedure to obtain these parameters without repeated tests is proposed for future research. Several applications to new optimization problems of the method developed in the thesis are also proposed for future research.

Contents

List of Figures

List of Tables

List of Acronyms

ANNs	Artificial Neural Networks
DE	Differential Evolution
ELM	Extreme Learning Machine
ELM-MAS	Extreme Learning Machine Multi-Agent System
GA	Gravity Assist
GB	Gradient Boosting
GPR	Gaussian Process Regression
LSODA	Livermore Solver for ODEs Automatic
LTP	Low-Thrust Propulsion
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MBH	Monotonic Basin Hopping
MGAT	Multiple-Gravity-Assist Trajectories
ML	Machine Learning
MM	MinMax scaler
NEO	Near Earth Object
NEP	Nuclear Electric Propulsion
NLP	Non-Linear Programming
ODE	Ordinary Differential Equation
OS-ELM	Online Sequential Extreme Learning Machine
OS-ELM-MAS	Online Sequential Extreme Learning Machine Multi-Agent System
pDE	Pygmo version of the self-adaptive Differential Evolution
PMP	Pontryagin’s Maximum Principle

PSO	Particle Swarm Optimization
relu	*rectifier linear unit*
RMSE	Root Mean Squared Error
S	Standard scaler
SA	Simulated Annealing
SEP	Solar Electric Propulsion
SF	Sims-Flanagan
sigmoid	*logistical sigmoid function*
SOI	Sphere Of Influence
tanh	*hyperbolic tangent function*
TOF	Time Of Flight
UAT	Universal Approximation Theorem

List of Symbols

AF	Activation Function
C_n	Number of individuals between updates of the error estimate
$\Delta \boldsymbol{v}_{GA}$	Change in the velocity vector of the spacecraft during the Gravity Assist
EPE	Expected Prediction Error
$f(\boldsymbol{x})$	Mapping of the parameters of the transfer to the parameters of the leg
g_0	Standard gravity
I_{sp}	Specific impulse
J_L	Propellant mass fraction required for an interplanetary leg
J_L^*	Optimal propellant mass fraction required for an interplanetary leg
$J_{(1)}^*$	Best cost value found for a transfer after several optimization runs
J	Fuel mass required to complete an interplanetary transfer
J^*	Optimal fuel mass required to complete an interplanetary transfer
$\boldsymbol{l}$	Vector of independent parameters that define an interplanetary leg
L'	Subspace of the leg parameters space in which feasible legs solutions exist
$\hat{L}'$	Estimated subspace of the leg parameters space in which feasible legs solutions exist
l	Number of legs in an interplanetary transfer
$\boldsymbol{l}_T$	Vector of parameters that define a complete interplanetary transfer
$\hat{L}_T'$	Estimated subspace of the transfer parameters space in which feasible transfer solutions exist
$\boldsymbol{l}_T^*$	Optimal vector of parameters that define a complete interplanetary transfer
m	Mass
M_k	Surrogate error estimation after k online training points
μ	Gravitational parameter of the Sun
μ_{J^*}	Mean of the final cost values for a transfer after several optimization runs

μ_p	Gravitational parameter of the planet for the Gravity Assist
N	Number of points used for initial training
NHU	Number of Hidden Units
$NHUP$	Number of Hidden Units Parent
N_I	Number of individuals per generation in the Differential Evolution algorithm
NM	Number of Models
n_{rev}	Number of revolutions in the Spherical shape-based method
n_{seg}	Number of segments in the Sims-Flanagan method
ν	Angle of the orbital plane during the Gravity Assist
p	Body visited during the interplanetary trajectory
ϕ	Azimuth angle of the relative velocity of the spacecraft with respect to the planet during the Gravity Assist
$\boldsymbol{R}$	Position vector of the planet
$\boldsymbol{r}$	Position vector of the spacecraft
ρ	Radius of the periapsis during the Gravity Assist
$\boldsymbol{s}$	State vector of the spacecraft
SI	Scaler for Inputs
σ_{J^*}	Standard deviation of the final cost values for a transfer after several optimization runs
SO	Scaler for Outputs
s	Pseudo-time used for the Sundman transformation in the Sims-Flanagan method
$\boldsymbol{T}$	Thrust vector of the spacecraft
t	Time
τ	Error threshold to accept surrogate predictions
θ	Polar angle of the relative velocity of the spacecraft with respect to the planet during the Gravity Assist
T_{max}	Maximum thrust magnitude allowed by the propulsion system
TOF	Time Of Flight of one leg of the trajectory
$\boldsymbol{T}'(t)$	Thrust policy required to follow a leg computed with the Spherical shape-based method
$\boldsymbol{T}^*(t)$	Optimal thrust policy during a leg
$\hat{\boldsymbol{T}}^*(t)$	Estimated optimal thrust policy during a leg
$\boldsymbol{V}$	Velocity vector of the planet
$\boldsymbol{v}$	Velocity vector of the spacecraft
$\boldsymbol{v}_a$	Velocity vector of the spacecraft after the Gravity Assist
$\tilde{\boldsymbol{v}}_a$	Relative velocity vector of the spacecraft with respect to the planet after the Gravity Assist
$\boldsymbol{v}_b$	Velocity vector of the spacecraft before the Gravity Assist

$\tilde{\boldsymbol{v}}_b$	Relative velocity vector of the spacecraft with respect to the planet before the Gravity Assist
v	Magnitude of the relative velocity of the spacecraft with respect to the planet during the Gravity Assist
$\boldsymbol{x}$	Vector of independent parameters that define a complete interplanetary transfer

1 Introduction

Missions to other planets or asteroids are very interesting from a scientific point of view. However, the achievements of such missions are hindered due to the large amounts of propellant required to reach a body beyond the orbit of Earth. Therefore, orbit optimization is a key part within the design phase of these missions. Moreover, two recent techniques resulted in an important increase of the objectives that can be achieved with the same amount of propellant: the Gravity Assist (GA) technique and the systems for Low-Thrust Propulsion (LTP). However, these advances also made the optimization of interplanetary trajectories a much more challenging problem. As a result, global optimization of interplanetary trajectories has become a very active field of research.

During a GA maneuver, a spacecraft makes a close approach to a planet, which causes a change in the heliocentric velocity of the spacecraft [4]. This maneuver allows the saving of the propellant for the change of velocity, but the trajectory optimization increases in complexity. In this thesis, only unpowered GAs are considered. Each GA is defined by the date of the encounter with the planet, the velocity vector of the spacecraft at the encounter, and two parameters that define the direction of the maneuver, and these parameters need to be optimized for each mission. This increases the size and complexity of the optimization problem proportionally with the number of GAs. For instance, the optimization of Multiple-Gravity-Assist Trajectories (MGAT) missions like BepiColombo [6], which has 7 GAs planned, is a very challenging task.

LTP systems are more fuel efficient than the traditional chemical engines, but at the cost of a greatly reduced thrust. The applications of LTP in interplanetary trajectories have been more limited due to the challenges that they present: The LTP system usually needs to be active for several months, which requires a very high reliability and presents a challenging optimization problem. The thrust direction can be freely chosen when the engine is active, so a wide range of trajectories are possible. Optimizing these trajectories is time consuming. However, LTP has been used successfully in missions to small bodies such as Deep Space 1 [7], Hayabusa [8], and Dawn [9].

The combination of GAs and LTP results in a very complex optimization problem. The computation time required to reach a global optimal solution is usually very high [10, 11]. In this thesis, the use of a Machine Learning (ML) method as a surrogate to replace the most expensive computations of the optimization process was studied. Previous works proved that it was possible to use ML methods to approximate the cost function of LTP trajectories [12]. References [13, 14] test different ML methods to approximate the cost of transfers between asteroids. Artificial Neural Networks (ANNs) have been studied as a method to predict the optimal control of a spacecraft from its state in References [15–17]. The cost function of interplanetary LTP trajectories was predicted using ANNs in References [18, 19] and using Gaussian Process Regression (GPR) in Reference [20]. In addition, the cost of trajectories in MGAT missions that use chemical propulsion was predicted using ANNs in Reference [21], and using GPR in Reference [22].

However, most of the methods presented do not consider the circumstances that are expected to be found in typical MGAT optimization problems. In particular, most of the ML methods used require large amounts of training data in order to make accurate predictions. The computation time required to generate this data may be longer than the time that would be saved by using the surrogate. Previous studies only focus on the surrogate accuracy without considering the complete optimization process [13, 14, 20, 22]. The ones that do consider the optimization process base their results on generating large amounts of data before the optimization [19, 21]. Instead, the objective of this thesis was to implement a ML surrogate that could be used as an off-the-shelf method for LTP MGAT optimization problems to produce medium-to-high fidelity solutions. This required the implementation of two complementary modules: a surrogate built with Online Sequential Extreme Learning Machine Multi-Agent System (OS-ELM-MAS), a specialized ML method, and an accurate trajectory computation procedure.

The Extreme Learning Machine (ELM) method [23] was selected as the best option to create the surrogate after analyzing the characteristics of the problem. This method has three main advantages: a good performance when the number of training points is low, it can be used online, and the Universal Approximation Theorem (UAT) indicates that it is capable of learning the underlying function [24, 25]. Online learning consists of learning from training points as they arrive sequentially instead of learning from a batch of training points available before training. In this thesis, online training was used to automatically adjust the number of training points generated to the minimum necessary for each case [26]. The online version of the ELM is the Online Sequential Extreme Learning Machine (OS-ELM) [27]. The low computational cost of training and evaluating the surrogate allowed the use of the combination of several OS-ELM models to reduce the overall error, known as an ensemble model. In this thesis, the combination was done based on the Extreme Learning Machine Multi-Agent System (ELM-MAS) [28]. The use of the ensemble model in an online fashion led to an innovative model named OS-ELM-MAS.

The computation of accurate trajectories did not require the use of new techniques. However, the joint implementation of the methods proposed for this study has never been done before. In the literature, the optimization of LTP MGAT was done by performing a global search using low-fidelity methods to define the trajectories [29–32], and performing a local optimization with a medium-to-high fidelity method from that solution in some cases [33–39]. However, the global minima of the higher fidelity methods might not be in the vicinity of the low fidelity method result. In this thesis, the global optimum of trajectories was computed directly with medium or high fidelity methods. This would require a large computation time if used as a standalone method, but the use of the surrogate allowed the speed-up of the process. The implementation of the global optimization for the more accurate method was challenging, and a result of this thesis was the implementation of this global optimization method, in addition to the one of the ML surrogate.

1.1. Objective and Research Question

The objective of this thesis was stated as:

To reduce the time required to compute the cost function associated with LTP interplanetary transfers in MGAT optimization by implementing a ML surrogate to approximate it.

To reach this objective, the following research question was answered:

"Is a ML method faster for computing the cost function in MGAT optimization than the present optimization methods for a given benchmark?"

In order to answer this question, three sub-questions and auxiliary sub-questions should be answered:

1. In which part of the MGAT optimization process is the ML surrogate used?
   (a) What advantages can be obtained by using it in that part?
   (b) How can the use of a ML surrogate be implemented in that part?
   (c) What is the accuracy that can be expected from the ML method?
2. Which are the inputs and outputs of the function to approximate?
   (a) Which ML method is used?
   (b) How is the training data generated?
   (c) What is the computational cost of each step?
   (d) Which optimization method is used for training?
3. What is the required accuracy for the ML method?
   (a) How is the accuracy evaluated?
   (b) Which methods are used as a reference?
   (c) What is the expected performance during the optimization when using the ML surrogate?

1.2. Report Structure

The structure of this report is the following one. A paper manuscript with the main findings of the thesis is presented in chapter 2. The MGAT optimization problem and its implementation, verification, and validation are presented in chapter 3. The implementation of the shape-based method used, and its verification and validation are presented in chapter 4. The implementation of the direct transcription method used, and its verification and validation are presented in chapter 5. The implementation of ML method used, the accuracy estimation, and the verification and validation of the surrogate are presented in chapter 6. Finally, the conclusions of the thesis and recommendations for future work are presented in chapter 7. The conclusions and recommendations presented in that chapter also include considerations about the implementation of the different methods that are not present in the paper.

2 Paper

In this chapter, a paper containing the main results of the thesis is presented. This paper was written adhering to the formatting and style requirements for papers to be submitted to the *AAS/AIAA Astrodynamics Specialist Conference*[1].

[1]http://www.univelt.com/FAQ.html#SUBMISSION (accessed May 13, 2020).

(Preprint) AAS XX-XXX

GLOBAL OPTIMIZATION OF LOW-THRUST INTERPLANETARY TRAJECTORIES USING A MACHINE LEARNING SURROGATE

Pablo Gómez Pérez[*], Kevin Cowan[†], and Yuxin Liu[‡]

In this work, we propose a new method to approximate the cost function of Low-Thrust, Multiple-Gravity-Assist interplanetary trajectories using a Machine Learning surrogate. This method speeds up the optimization process without fine tuning of the surrogate parameters for every individual case. We identified the computation time required to obtaining training data as the main limitation when using Machine Learning methods for this purpose. Therefore, we built the surrogate with an Online Sequential Extreme Learning Machine Multi-Agent System (OS-ELM-MAS) due to its theoretical good performance when the training data is limited. We defined a method to include the surrogate during the optimization process without needing specialized optimization algorithms, and identified the parameters that control the interaction between the surrogate and the optimization process. Finally, we showed that the use of the surrogate is beneficial when evaluations of the cost function have a high computational cost, as long as the parameters that control the interaction are carefully selected. We also showed that the optimal value of these parameters can be narrowed down based on the characteristics of the transfers, and a search over a broad range of values is not necessary.

INTRODUCTION

Interplanetary missions are very attractive from a scientific point of view. However, the payload capacity of these missions is severely limited by the large amounts of fuel required to reach bodies beyond the orbit of Earth. Therefore, orbit optimization is a critical part of their design process. This is a very complex task, and the difficulty is increased when incorporating propellant-saving advances such as the Gravity Assist (GA) maneuvers and the Low-Thrust Propulsion (LTP). These advances allowed to reduce the fuel required for interplanetary trajectories, but at the expense of an exponential increase of the difficulty of the optimization problem.[1,2] This is an issue especially during the preliminary mission design phase, as a large number of alternatives need to be considered to make an informed decision regarding the mission objective.[3] However, the long time required to obtain a globally optimal solution means that suboptimal solutions are often considered.[1,4] This is not an ideal situation as solutions that allow for higher payloads are missed.

The field of Machine Learning (ML) has gained attention in recent years due to increased capabilities of ML methods to approximate that are unknown or very costly to evaluate. In particular, Artificial Neural Networks (ANNs) are among the most versatile and successful ML methods. Both ANNs and other ML methods were studied as an option to speed up the optimization of both Multiple-Gravity-Assist Trajectories (MGATs) and trajectories with LTP. ANNs were used both to predict the optimal control at a given state[5,6] and to predict the fuel required to complete a whole transfer including an intermediate GA.[7] Moreover, several ML methods were tested to predict the fuel required to complete optimal transfers between asteroids[8,9] and it was found that Gradient Boosting[10] provided the best results. In addition, References 11 and 12 use ML

[*]M.Sc. Student, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands, pablogope@gmail.com
[†]Education fellow and Lecturer, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands, k.j.cowan@tudelft.nl.
[‡]Ph.D. Candidate, Faculty of Aerospace Engineering, Delft University of Technology, The Netherlands, yuxin.liu@tudelft.nl.

methods to estimate the fuel requirements of MGATs but for missions using traditional chemical propulsion. The methods used in previous approaches require large amounts of data to train the ML surrogates. Moreover, most previous work exclusively evaluates the overall accuracy of the approximation, but it does not evaluate the interaction between the surrogate and the optimization algorithm. In those cases where such interaction is studied, custom algorithms are proposed, specifically designed for the case studied.[11] Therefore, the main issue with previous approaches is that the computation time required for the generation of training data and for the fine tuning is not considered when evaluating the speed-up achieved with the surrogate. In this work, we selected a method to build a ML surrogate to predict the fuel cost of the trajectories that requires only a small amount of training data and has a fast training process. We tested the effects of this surrogate in the optimization using medium fidelity LTP models,[2] but we expect results to be extrapolatable to higher fidelity models.

The method we selected is a simplified version of the ANNs known as Extreme Learning Machine (ELM).[13] This method has a lower number of hyperparameters and a very fast training process. The simplicity of the method permits the combination of models to reduce the total error while still allowing for fast training and prediction. Therefore, we proposed the use of an Extreme Learning Machine Multi-Agent System (ELM-MAS)[14] method to combine several ELM surrogates. Moreover, we propose to train the surrogate online, so the model is updated during optimization. This allows the generation of training data only when the error of the surrogate predictions is expected to be high. This is the case at the beginning of the optimization, when few data is available, and when the optimization algorithm explores a new area of the input space. The online version of the ELM model is the Online Sequential Extreme Learning Machine (OS-ELM).[15] We presented a combined used of the OS-ELM and ELM-MAS into the innovative Online Sequential Extreme Learning Machine Multi-Agent System (OS-ELM-MAS) model.

To test the speed-up achieved with the surrogate, we focused on the effects of the surrogate when used in conjunction with an optimization algorithm. We identified a number of parameters that influence the interaction between the online surrogate and the optimization process, and we propose a method to select the most adequate values for these parameters. Finally, we used a Differential Evolution (DE)[16] global optimization algorithm in several MGATs cases and compared the results achieved with and without the surrogate replacing the cost function. The final objective was to achieve an implementation of the surrogate that does not require either fine tuning or previous training data and that can be used as an off-the-shelf tool for preliminary trajectory design.

This paper is structured as follows. The first section describes the optimization problem considered. The second section describes the numerical methods used to compute the fuel cost of the transfers. The third section describes the ML method used to build the surrogate. The fourth section describes the methodology used to evaluate the performance of the surrogate. The fifth section presents the results of the evaluation in three different MGATs. Finally, the last section contains a summary of the conclusions reached in this work.

MULTIPLE-GRAVITY-ASSIST OPTIMIZATION PROBLEM

The optimization of MGATs is especially challenging as every leg between planetary encounters needs to be optimized. This means that several optimization runs are required to compute a single trajectory. Moreover, the optimal combination of planetary encounter dates and GA parameters needs to be determined. A global search over these dates and parameters requires a large number of trajectory computations. Due to the high computational cost of each trajectory optimization, the use of the surrogate can be especially beneficial in these cases. This section defines in detail the optimization problem addressed in this work.

Spacecraft Dynamics

We follow the most common approach in the literature and use a simplified dynamic model for the computation of every individual transfer. In particular, we use the method of linked conics, which is used almost universally in the literature. We consider that the only forces acting on the spacecraft during the interplanetary flight are the spherical gravity of the Sun and the thrust provided by the LTP system. The planetary GAs are modeled as instantaneous changes in the velocity of the spacecraft at the position of the planet. We define the state of the spacecraft with the vector $\boldsymbol{s} = [\boldsymbol{r}^T, \boldsymbol{v}^T, m]^T$. The equations of motion are

$$\dot{\boldsymbol{r}} = \boldsymbol{v}, \qquad \dot{\boldsymbol{v}} = -\frac{\mu}{|\boldsymbol{r}|^3}\boldsymbol{r} + \frac{\boldsymbol{T}}{m}, \qquad \dot{m} = -\frac{|\boldsymbol{T}|}{I_{sp} g_0}, \tag{1}$$

where $\boldsymbol{r}$ is the position vector of the spacecraft, $\boldsymbol{v}$ its velocity vector, m its the mass, $\boldsymbol{T}$ the thrust vector, I_{sp} the specific impulse of the spacecraft, and $g_0 = 9.806\,65\,\mathrm{m\,s^{-2}}$ and $\mu = 1.327\,124\,400\,18 \times 10^{20}\,\mathrm{m\,s^{-2}}$ are the standard gravity and gravitational parameter of the Sun respectively.

Gravity Assist Computation

As mentioned before, the effects of the GAs are approximated as instantaneous changes in the velocity of the spacecraft. Given an incoming velocity vector $\boldsymbol{v}_b$, the change on velocity $\Delta\boldsymbol{v}_{GA}$ depends on the velocity of the planet $\boldsymbol{V}$, the radius of closest approach at the planet ρ, the orbit plane angle η with a reference plane, and the gravitational parameter of the planet μ_p. We use the method presented in Reference 4 to determine the state after a GA. The velocity after the GA, $\boldsymbol{v}_a$, is given by

$$\boldsymbol{v}_a = \boldsymbol{v}_b + \Delta\boldsymbol{v}_{GA}\left(\boldsymbol{v}_b, \boldsymbol{V}, \eta, \rho, \mu_p\right). \tag{2}$$

Optimization Problem Formulation

In this work, we considered the single-objective global optimization problem of finding the most fuel-efficient transfer between initial body p_0 and final body p_l for a given spacecraft. During the transfer, the spacecraft encounters $l-1$ intermediate bodies $p_1, \ldots, p_{l-1}$. A GA is performed if the encounter is with a planet, and the state remains unchanged if the encounter is with bodies of negligible gravitational attraction such as asteroids. We considered that the sequence of bodies to visit during the transfer remained constant for the whole optimization procedure. The characteristics of the spacecraft (i.e. initial mass m_0, maximum thrust T_{max} and specific impulse I_{sp}) were also considered fixed values for on optimization. A simple Nuclear Electric Propulsion (NEP) propulsion model was implemented, but our conclusions are expected to be extrapolatable to similar propulsion methods.

We consider a leg to be a segment of the trajectory between two consecutive body encounters, so the optimization problem includes l legs. Leg i is defined by the initial state $\boldsymbol{s}_0^{(i)} = [\boldsymbol{r}_0^{(i)T}, \boldsymbol{v}_0^{(i)T}, m_0^{(i)}]^T$, the final state $\boldsymbol{s}_f^{(i)} = [\boldsymbol{r}_f^{(i)T}, \boldsymbol{v}_f^{(i)T}, m_f^{(i)}]^T$, and the time of flight $TOF^{(i)}$. The trajectory of the leg is the optimal trajectory between the initial and final state. The value of $m_f^{(i)}$ is computed as the initial mass plus the change of mass during the leg:

$$m_f^{(i)} = m_0^{(i)} + \int_0^{TOF^{(i)}} \dot{m}\,dt. \tag{3}$$

Note that $\dot{m} \leq 0$. We define the cost function of the leg as the propellant mass fraction used during the leg:

$$J_L^{(i)} = J_L\left(\boldsymbol{s}_0^{(i)}, \boldsymbol{s}_f^{(i)}, TOF^{(i)}\right) = \frac{m_0^{(i)} - m_f^{(i)}}{m_0^{(i)}} = \int_0^{TOF^{(i)}} -\frac{\dot{m}}{m_0^{(i)}}\,dt = \int_0^{TOF^{(i)}} \frac{|\boldsymbol{T}(t)|}{m_0^{(i)} g_0 I_{sp}}\,dt. \tag{4}$$

To find the optimal control policy $\boldsymbol{T}^{*(i)}(t)$ that minimizes the cost function, we start by defining a general value function of the form

$$g(t, s) = \min_{\boldsymbol{u}}\left(\int_0^{t'} j\left(\boldsymbol{s}(t), \boldsymbol{u}(t)\right)dt + h(\boldsymbol{s}(t))\right), \qquad \text{and dynamics} \qquad \dot{\boldsymbol{s}}(t) = \boldsymbol{F}(\boldsymbol{s}(t), \boldsymbol{u}(t)),$$

where $\boldsymbol{u}(t)$ is the control law. The optimal control is the solution of the *Hamilton-Jacobi-Bellman equation*:

$$\dot{g}(\boldsymbol{s},t) = -\min_{\boldsymbol{u}} \left(\nabla g(\boldsymbol{s},t) \cdot \boldsymbol{F}(\boldsymbol{s},\boldsymbol{u}) + l(\boldsymbol{s},\boldsymbol{u}) \right), \qquad \text{subject to} \qquad g(\boldsymbol{s},t') = h(\boldsymbol{s}). \tag{5}$$

If $\boldsymbol{F}$, j, and h are continuous and bounded, and $\boldsymbol{s}(t')$ can be reached from $\boldsymbol{s}(0)$, there is an optimal value $\boldsymbol{u}^*$ which is a solution for Equation (5) and it is unique.[17] In our case, the relevant functions are

$$j = \frac{|\boldsymbol{T}|}{m_0^{(i)} g_0 I_{sp}}, \qquad h = 0, \qquad t' = TOF^{(i)}, \qquad \boldsymbol{u} = \boldsymbol{T},$$

and $\boldsymbol{F}$ is given by Equation (1). the relevant functions are continuous and bounded, so there is an unique optimal control policy $\boldsymbol{T}^{*(i)}$ which provides the optimal cost $J_L^{*(i)}$.

However, there is no known general solution for Equation (5). Instead, a numerical optimization is used to search for the function $\boldsymbol{T}^{*(i)}$ in every leg. A transcription method was used to define a problem in which a local optimization algorithm could be applied. This process is explained in detail in the Section Leg Computation. In a general case, the results obtained are not the true $\boldsymbol{T}^{*(i)}$ as the numerical optimization is subject to a series of additional constraints that make the problem tractable. Instead, an estimation of the optimal control policy $\hat{\boldsymbol{T}}^{*(i)}$ is provided by the local optimization procedure. This provides the value of the cost function associated with the leg used in practice:

$$\hat{J}_L^{*(i)} = J_L(\hat{\boldsymbol{T}}^{*(i)}) = J_L^{*(i)} + \epsilon^{(i)}, \tag{6}$$

where $\epsilon^{(i)}$ is the difference between the actual optimal value of the cost function and the estimated one.

The value $\hat{J}_L^{*(i)}$ depends on $\boldsymbol{s}_0^{(i)}$ and $\boldsymbol{s}_f^{(i)}$. However, the value of $m_f^{(i)}$ is not independent as it is given by Equation (3). Moreover, taking into account that $m_0^{(i)} = m_f^{(i-1)}$ and Equations (3) and (4), we get $m_0^{(i)} = m_0^{(i-1)}(1 - J_L^{(i-1)})$. Finally, the value of $m_0^{(1)}$ is constant for a problem. We define the vector of independent state variables that define an individual leg as

$$\boldsymbol{l}^{(i)} = [\boldsymbol{r}_0^{(i)T}, \boldsymbol{v}_0^{(i)T}, m_0^{(i)}, \boldsymbol{r}_f^{(i)T}, \boldsymbol{v}_f^{(i)T}, TOF^{(i)}].$$

This vector can take any values $\boldsymbol{l}_L^{(i)} \in L = \mathbb{R}^{14\times 1}$. However, the solution $\boldsymbol{T}^*$ only exists for values in $L' \subset L$. In practice, we consider $\hat{L}' \subseteq L'$ where a solution $\hat{\boldsymbol{T}}^*$ can be obtained. Finally, the overall cost function of the transfer, $J : \hat{L}'_T \to \mathbb{R}$ is the total propellant mass required

$$J(\boldsymbol{l}_T) = \sum_{i=1}^{l} \hat{J}_L^{*(i)} m_0^{(i)}, \qquad \text{where} \qquad \boldsymbol{l}_T = [\boldsymbol{l}^{(0)}, \boldsymbol{l}^{(1)}, \ldots, \boldsymbol{l}^{(l)}],$$

and $\boldsymbol{l}_T \in \hat{L}'_T = (\hat{L}')^l$. The outer optimization loop is the numerical procedure to obtain

$$\boldsymbol{l}_T^* = \arg\min_{\boldsymbol{l}_T \in \hat{L}'_T} J(\boldsymbol{l}_T). \tag{7}$$

The landscape of MGATs optimization problems usually contains a large number of local minima.[2] Therefore, a local optimizer is unlikely to converge to the global optimum. Instead, we use a global optimizer to search for the optimal solution.

Global Optimization Algorithm

The global optimization algorithm is used to look for the solution to Equation (7). We selected a DE[16] algorithm as global optimizer. In particular, we used the *Pygmo version of the self-adaptive Differential*

Evolution (pDE)[*]. This algorithm includes adaptation of all evolution parameters. In this work, we propose to replace the fitness function computation by the surrogate approximation during the optimization. The adaptation of the mutation parameters is expected to help the algorithm adapt to this change in the cost function. In addition, the only parameter that needs to be selected when using this algorithm is the number of individuals per generation N_I.

The pDE algorithm chosen can only handle box-bounding constraints on the optimization variables. However, the values of elements of $\boldsymbol{l}_T$ are not independent of each other, and they are subject to non-linear constraints: the position of the spacecraft at planet encounters has to match the position of the body it encounters at that time and the velocities before and after a GA are related through Equation (2). Notwithstanding, an analysis of the constraints allowed the definition of an alternative set of optimization variables that fulfill the constraints and are box bounded.

First, the position of the bodies can be obtained as a function of time from the planetary ephemerides. We define $\boldsymbol{R}_i(t)$ as the function that defines the position of body p_i at time t. Given a departure date t_0 for the transfer, we can compute the position at each encounter as

$$\boldsymbol{r}_f^{(i)} = \boldsymbol{r}_0^{(i+1)} = \boldsymbol{R}_i\left(t_0 + \sum_{j=1}^{i} TOF^{(j)}\right), \qquad \text{and} \qquad \boldsymbol{r}_0^{(1)} = \boldsymbol{R}_0(t_0). \tag{8}$$

Equation (8) shows that the positions at the encounters are fully defined by t_0 and the $TOF^{(i)}$ values, which can be chosen freely. Therefore, we selected these values as optimization variables. The lower and upper bounds for each of them are defined based on expert knowledge for each individual case. This is explained in detail in the Section Test Setup.

Second, the initial velocity of a leg depends on the final velocity of the previous one through Equation (2). The computation of the new velocity also requires the value of the velocity of the planet and the GA parameters η and ρ. The velocity of planet p_i can be computed again from an ephemeris as $\boldsymbol{V}_i(t)$, while the GA parameters at every encounter η_i and ρ_i can be freely chosen. The initial velocity at each leg is given by

$$\boldsymbol{v}_0^{(i+1)} = \boldsymbol{v}_f^{(i)} + \Delta\boldsymbol{v}_{GA}\left(\boldsymbol{v}_f^{(i)}, \boldsymbol{V}_i\left(t_0 + \sum_{j=1}^{i} TOF^{(j)}\right), \eta^{(i)}, \rho^{(i)}\right).$$

We used the values $\eta^{(i)}$ and $\rho^{(i)}$ as the design variables. For the GA orbit plane angle, the valid range is $\eta^{(i)} \in [0, 2\pi]$, while the range of GA radii allowed was set on every individual case depending on the characteristics of the planet encountered.

Finally, the values of $\boldsymbol{v}_0^{(1)}$ and $\boldsymbol{v}_f^{(i)}$ are not constrained by the dynamics of the problem. However, problems of scientific interest usually introduce constraints on the magnitude of the difference between the velocity of the body and the spacecraft at some of the encounters, such as a rendezvous with zero velocity at the end of the transfer, or a maximum departure velocity achievable with the available launchers. Therefore, we chose to represent the velocity magnitudes as the relative velocity with respect to the planet $\tilde{\boldsymbol{v}} = \boldsymbol{v} - \boldsymbol{V}$. We defined the relative velocity in spherical coordinates as

$$\tilde{\boldsymbol{v}} = v\sin\theta\cos\varphi\boldsymbol{e}_1 + v\sin\theta\sin\varphi\boldsymbol{e}_2 + v\cos\theta\boldsymbol{e}_3 \ ,$$

where v, θ and ϕ are respectively the magnitude, polar angle and azimuth angle of the velocity. The orthonormal base of vectors $\boldsymbol{e}_k$ is defined so the first vector is parallel to the velocity of the planet, the second one is parallel to the component of the position of the planet perpendicular to the velocity, and the third one as the cross product of the previous two. The equations that define these vectors are

$$\boldsymbol{e}_1 = \frac{\boldsymbol{V}}{|\boldsymbol{V}|}, \qquad \boldsymbol{e}_2 = \frac{\boldsymbol{R} - (\boldsymbol{R}\cdot\boldsymbol{e}_1)\,\boldsymbol{e}_1}{|\boldsymbol{R} - (\boldsymbol{R}\cdot\boldsymbol{e}_1)\,\boldsymbol{e}_1|}, \qquad \boldsymbol{e}_3 = \boldsymbol{e}_1 \times \boldsymbol{e}_2.$$

[*]`https://esa.github.io/pygmo2/algorithms.html#pygmo.de1220` (accessed on May 13, 2020).

We used this base to eliminate the interdependence between the GA dates and other GA parameters, as the change in orbital energy is very similar for GAs with the same relative velocity and GA parameters at different dates. The values of $\boldsymbol{v}_f^{(i)}$ are defined by the parameters $v^{(i)}$, $\theta^{(i)}$ and $\varphi^{(i)}$, and $\boldsymbol{v}_0^{(1)}$ is defined by $v^{(0)}$, $\theta^{(0)}$ and $\varphi^{(0)}$. The polar angle was limited to $\theta^{(i)} \in [\pi/2, 3\pi/2]$ as values outside that range would require orbits with large inclinations that we assumed to be of no interest for the cases considered. The azimuth angle $\varphi^{(i)}$ is allowed to take any value in $[0, 2\pi]$. Finally, the bounds of $v^{(i)}$ were defined based on the characteristic of each individual problem.

The final decision vector of the optimization problem is

$$\boldsymbol{x} = [t^{(0)}, v^{(0)}, \theta^{(0)}, \varphi^{(0)}, \mathit{TOF}^{(1)}, v^{(1)}, \theta^{(1)}, \varphi^{(1)}, \eta^{(1)}, \rho^{(1)}, \ldots, \mathit{TOF}^{(m)}, v^{(m)}, \theta^{(m)}, \varphi^{(m)}],$$

which is defined in a box-bounded subset $X \subset \mathbb{R}^{6l+2}$. The procedure detailed in the previous lines allows the definition the function $f : X \to L_T$ and the bounds of X. The final cost function used for the optimization is $J \circ f : X \to \mathbb{R}$. However, $L_T \not\subset \hat{L}'_T$, so not all $\boldsymbol{x}$ values resulted in valid input values. We can only know whether $f(\boldsymbol{x}) \in \hat{L}'_T$ after the function $J(f(\boldsymbol{x}))$ was evaluated. This is a non-linear constraint that cannot be eliminated. We handled this by assigning a weighted penalty to the cost of the individuals for which a solution could not be found.

Use of Machine Learning Surrogate

The objective of the surrogate is to approximate a cost function so the cost function does not need to be evaluated for every individual. In the optimization problem presented, there are three cost functions that can be approximated: $J(f(\boldsymbol{x}))$, $J(\boldsymbol{l}_T)$ or $\hat{J}^*_L(\boldsymbol{l})$. We decided to use the surrogate to approximate $\hat{J}^*_L(\boldsymbol{l})$ for three reasons: the function shape was expected to be more simple, the dimension of the input vector was lower, and it was evaluated m times for every evaluation of the other two candidate functions, increasing the number of training points available. In addition, we decided to represent the state of $\boldsymbol{l}$ in modified equinoctial elements.[18]

To minimize the number of training points required, the surrogate is trained online. This means that the surrogate is trained on the training data as it is generated and at the same time predictions are made with it. However, predictions are only made when the expected error is below a threshold τ. The error is estimated from the training points with a method explained in Section . An advantage of using an online method with respect to previous approaches[11] is the possibility to adapt the model if new points to be predicted are very different from the training points used. The surrogate may not be able to estimate the new points as accurately as expected when the optimization algorithm explores new areas of the input space. If this happens, the error increases and the surrogate predictions are not used until the error is below τ again. The decision to use the surrogate based on the estimated error is taken every time $\hat{J}^*_L(\boldsymbol{l})$ is evaluated. That meant that the cost of some legs may be estimated with the surrogate and the cost of other legs may be computed with the original optimization method for a single transfer.

The error estimation can only be updated when a point is computed with the numerical optimization method. With the method described, this will never happen if the error is below the threshold, as all points are predicted with the surrogate. To solve this problem, additional computations of the original cost function are performed even when the error is below the threshold. The frequency of this computations is controlled by the the parameter C_n. The error is estimated every C_n individuals and in consecutive legs. If the error is computed once on the leg i of the individual j, the next computation of the error will happen in the leg $i + 1$ of the individual $j + C_n$.

LEG COMPUTATION

The optimization process described requires a method to compute the cost of each leg, $\hat{J}^*_L(\boldsymbol{l})$. This section describes the numerical method used. The numerical method is a key part of this work as the training data for the surrogate model is generated using this method. In addition, we evaluated the performance of the optimization with the surrogate by comparing the results with the optimization using exclusively the numerical to compute the legs.

Several options are available to compute the cost of the legs when LTP is used. They differ in the fidelity of the trajectories obtained and the computation speed.[2] A surrogate is more advantageous in cases in which the computational cost is higher. Therefore, we used a method with high accuracy within the constraints of the dynamics described in Equation (1). In addition, the method used to compute $\hat{J}^*_L(\boldsymbol{l})$ determines the minimum expected approximation error achievable with the surrogate. The ability of the ML surrogate to approximate an arbitrary function is based on the Universal Approximation Theorem (UAT) for ELM.[19] However, this theorem only holds when the values of the function to approximate are unique for each input value. This is true for $J^*_L(\boldsymbol{l})$ as it is a solution of the *Hamilton-Jacobi-Bellman equation*. However, this is not true for a general $\hat{J}^*_L(\boldsymbol{l})$ as the solution can be affected by factors not considered in $\boldsymbol{l}$ such as the effect of integration tolerances and the random initial state. To analyze the effect of this in the error, we defined the expected prediction error as

$$EPE = \mathrm{E}\left[L\left(\hat{J}^{*\prime}_L(\boldsymbol{l}), \hat{J}^*_L(\boldsymbol{l})\right)\right] ,$$

where $\hat{J}^{*\prime}_L(\boldsymbol{l})$ is the surrogate prediction at a point $\boldsymbol{l}$ and $L(\hat{J}^{*\prime}_L(\boldsymbol{l}), \hat{J}^*_L(\boldsymbol{l}))$ is a loss function that defines the value of the error of a prediction at a point. We assume that the numerical procedure introduces a Gaussian noise δ_s with standard deviation σ_s corresponding to the effect of the factors not considered in $\boldsymbol{l}$. Considering the squared error loss $L(\hat{J}^{*\prime}_L(\boldsymbol{l}), \hat{J}^*_L(\boldsymbol{l})) = (\hat{J}^{*\prime}_L(\boldsymbol{l}) - \hat{J}^*_L(\boldsymbol{l}))^2$ the prediction error is bounded by[10]

$$EPE \geq \sigma_s^2 \; . \tag{9}$$

Estimating the actual value of σ_s is very difficult, as it requires to sample the value of $L(\hat{J}^{*\prime}_L(\boldsymbol{l}), \hat{J}^*_L(\boldsymbol{l}))$ in a large number of points. However, Equation (9) indicates that σ_s value has to be as small as possible in order to minimize the prediction error. This means that the numerical method to compute the cost of the legs should be selected so the influence of factors not included in $\boldsymbol{l}$ is as low as possible. We achieve this by choosing a method whose solution is independent of the initial guess. Moreover, we can also eliminate the effect of the random seed by choosing a deterministic method. Therefore, we decided to use a direct method for the leg computation as they make possible to use a deterministic local optimization. Nonetheless, these methods typically require a search trough several initial guesses when used as a standalone method.[3,20] This makes the result dependent on the initial guess, but we solved this problem by obtaining an initial guess with a lower fidelity method whose solutions are independent of the initial guess. Shape-based methods are commonly used for this purpose.[1,21] Therefore, the overall leg computation process is completely deterministic and not dependent on the initial guess as long as the shape-based method has an unique solution and a deterministic optimization algorithm is used for refining the initial guess.

Shape-Based Method

Shape-based methods assume an analytical shape for the trajectory of the leg. The thrust required to follow the shape is then computed and the cost function evaluated. These methods are usually faster than the alternatives when LTP is used.[2] However, the trajectories obtained are not optimal as they are restricted to the shape selected. In addition, thrust constraints are difficult to implement as they can only be checked once the trajectory has been computed.

We identified the Spherical[22] shape-based method as the best option to obtain the initial guess for the trajectory. This was based on the following set of requirements for the shape-based method: (1) it shall produce three-dimensional trajectories, (2) it shall be able to fully match the initial and final state, (3) it shall not require a numerical optimization. Shape-based methods provide different solutions depending one the number of revolutions, n_{rev}, selected for the trajectory. Therefore, the solution is not unique unless a n_{rev} value is selected. In this work, we selected the n_{rev} corresponding to the Spherical shape-based solution with the lowest fuel cost of all the possible solutions with n_{rev} in a predetermined range.

Direct Method

A direct method allows the transformation of the orbit optimization problem into a non-linear programming problem. These methods are usually considered to be medium fidelity.[2] We use a Sims-Flanagan (SF)[23] method as it is shown to be the most stable method,[2] although at the expense of higher computation times. We considered the higher convergence rate more important than computation speed when a surrogate is used. Moreover, we decided an increase in the accuracy of the solution was preferable to shorter computation times. Therefore, we used an alternative implementation of the SF method described in Reference 24. This implementation uses the Sundman transformation, continuous thrust along the segments instead of impulses at their middle points, and Taylor integration. The Sundman transformation is considered beneficial, specially for trajectories with a large difference of radii between initial and final state. Continuous thrust has the advantage of providing trajectories that are physically feasible, which is not granted with the impulse approximation of the original implementation. However, trajectories can no longer be considered Keplerian between impulses, although the increase in computation time is compensated in part by using Taylor integration.

The results from the shape-based method is the initial guess for the SF method. The values required for the transformation are the mean thrust values at each of the n_{seg} segments used, the final mass, and the total change in the variable s of the Sundman transformation, defined as $\Delta s = \int_0^{TOF} dt/|\boldsymbol{r}|$. A SF segment k has start time $t_{k,0}$ and end time $t_{k,f}$, such that $\int_{t_{k,0}}^{t_{k,f}} dt/|\boldsymbol{r}| = \Delta s/n_{seg}$, $t_{k,f} = t_{k+1,0}$, and $t_{0,0} = 0$. The initial guess for the thrust during segment k is given by the vector

$$\boldsymbol{T}_k = \bar{T}'_k \frac{\bar{\boldsymbol{T}}'_k}{|\bar{\boldsymbol{T}}'_k|}, \qquad \text{where} \qquad \bar{T}'_k = \frac{\int_{t_{k,0}}^{t_{k,f}} |\boldsymbol{T}'|dt}{t_{k,f} - t_{k,0}}, \qquad \bar{\boldsymbol{T}}'_k = \frac{\int_{t_{k,0}}^{t_{k,f}} \boldsymbol{T}'dt}{t_{k,f} - t_{k,0}},$$

and $\boldsymbol{T}'$ is the thrust obtained with the shape-based method.

Once the initial guess is obtained, the thrust values at each segment are optimized using a local optimization algorithm. We selected the Sequential Least SQuares Programming (SLSQP) algorithm from the nlopt package*. The optimization is subject to the constraints on the mazimum thrust magnitude

$$T_{k\alpha} < \max_{\forall t,\alpha} (T'_\alpha(t))\, | i \in \{1, \ldots, n_{seg}\}, \alpha \in \{x, y, z\}, \qquad |\boldsymbol{T}'_k| < T_{max} | k \in \{1, \ldots, n_{seg}\},$$

where $T_{k\alpha}$ are the Cartesian components of $\boldsymbol{T}_k$ and T'_α are the Cartesian components of $\boldsymbol{T}'$. The optimization is also subject to the mismatch constraint for the SF transcription[23]

$$|\boldsymbol{s}_h^+ - \boldsymbol{s}_h^-| < \epsilon_{SF},$$

where $\boldsymbol{s}_h^-$ is the state at the midpoint of the SF trajectory when integrated forward from the starting point, $\boldsymbol{s}_h^+$ is the state at the midpoint of the SF trajectory when integrated backwards from the ending point, and ϵ_{SF} is the tolerance allowed. In some cases, the algorithm is unable to find a solution that fulfills all the constraints. If that is the case, $\boldsymbol{l}_L \notin \hat{L}'$ and the leg transfer is considered unfeasible and the transfer is discarded. However, the model used for the surrogate has no way to identify unfeasible legs without performing the complete numerical optimization. Therefore, the surrogate provides numerical results when evaluated at legs where the numerical evaluation fails to converge. One of the main obstacles for the use of the surrogate is that these unfeasible legs are still used by the global optimization algorithm, and the final transfer selected may not be a feasible one. Therefore, the convergence rate of the SF computations influence the optimization results obtained when using the surrogate.

MACHINE LEARNING SURROGATE

As mentioned previously, we identified the ELM as a the most adequate ML method for the tool proposed in this work. This sections presents the details of this method.

*`https://nlopt.readthedocs.io/en/latest/NLopt_Algorithms/#slsqp` (accessed May 13, 2020).

Review of Extreme Learning Machines

ELMs[13] are based on ANNs with a single hidden layer. The peculiarity of this model is that the weights between the input later and the hidden layer are initialized randomly when the model is created, and remain frozen for the remaining of the training process. This allows the computation of the weights between the hidden and the output layer by least squares regression. Therefore, the result is always optimal for the training points in terms of Mean Squared Error (MSE). Moreover, the UAT is valid also for the ELMs.[19] Both additive nodes and radial basis function nodes are considered in the literature for the hidden units of the ELM, but the performance of both options is very similar.[13] Therefore, we considered only ELM with additive nodes as it is the most common option in ANNs.

The definition of an ELM requires an activation function $AF : \mathbb{R} \to \mathbb{R}$, which should be infinitely differentiable in any interval,[13] and a number of hidden units NHU. The training data contains N pairs of observations $(\boldsymbol{l}_i, \boldsymbol{y}_i) \in \mathbb{R}^{n_I \times 1} \times \mathbb{R}^{n_O \times 1}$, where $\boldsymbol{l}_i$ is the input vector at point i, $\boldsymbol{y}_i$ is the output vector at point i, and n_I and n_O are the lengths of the input and target vectors respectively. The training process is the following one.[13] First, the vectors $\boldsymbol{a}_i \in \mathbb{R}^{1\times n_I} \forall i \in \{1, \dots, NHU\}$ and $\boldsymbol{b} \in \mathbb{R}^{1\times NHU}$ are randomly initialized. In this work, each element is drawn from a random uniform distribution in the range $[-1, 1]$. Then, the *hidden layer output matrix* is computed as

$$\boldsymbol{H} = \begin{bmatrix} AF(\boldsymbol{a}_1 \cdot \boldsymbol{l}_1 + b_1) & \cdots & AF(\boldsymbol{a}_{NHU} \cdot \boldsymbol{l}_1 + b_{NHU}) \\ \vdots & \ddots & \vdots \\ AF(\boldsymbol{a}_1 \cdot \boldsymbol{l}_N + b_1) & \cdots & AF(\boldsymbol{a}_{NHU} \cdot \boldsymbol{l}_N + b_{NHU}) \end{bmatrix},$$

where $b_1, \dots, b_{NHU}$ are the elements of $\boldsymbol{b}$. Finally, the optimal set of output weights can be computed as

$$\hat{\boldsymbol{\beta}} = \boldsymbol{H}^\dagger \boldsymbol{Y}, \qquad \text{where} \qquad \boldsymbol{H}^\dagger = \left(\boldsymbol{H}^T\boldsymbol{H}\right)^{-1}\boldsymbol{H}^T, \qquad \text{and} \qquad \boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_1 \\ \vdots \\ \boldsymbol{y}_N \end{bmatrix}. \tag{10}$$

The prediction for a new input value $\boldsymbol{l}'$ is computed as

$$\hat{\boldsymbol{y}}' = \left[AF(\boldsymbol{a}_1 \cdot \boldsymbol{l}' + b_1) \quad \cdots \quad AF(\boldsymbol{a}_{NHU} \cdot \boldsymbol{l}' + b_{NHU})\right] \hat{\boldsymbol{\beta}}.$$

The Online Sequential ELM

The online version of the ELM is the OS-ELM.[15] The process of computing the weights can be adapted to the arrival of new training points so the weights always correspond to the least-squares optimal solution for the full set of points trained. This is done by updating $\hat{\boldsymbol{\beta}}$ with a recursive least-squares formula.[15] We consider only the case in which new training points arrive one by one. The total number of training samples is arbitrary, and the first N training samples are selected for initial training. First, the initial weights $\hat{\boldsymbol{\beta}}^{(0)}$ are computed with these N initial training samples as shown in Equation (10). The subsequent pairs of training points are $\left(\boldsymbol{l}^{(k)}, \boldsymbol{y}^{(k)}\right)$, where k indicates the position in which they arrive after the last point used for initial training. The weights are updated as

$$\hat{\boldsymbol{\beta}}^{(k+1)} = \hat{\boldsymbol{\beta}}^{(k)} + \boldsymbol{P}_{k+1}\boldsymbol{H}_{k+1}^T\left(\boldsymbol{y}^{(k+1)} - \boldsymbol{H}_{k+1}\hat{\boldsymbol{\beta}}^{(k)}\right),$$

where

$$\boldsymbol{H}_{k+1} = \left[AF\left(\boldsymbol{a}_1 \cdot \boldsymbol{l}^{(k+1)} + b_1\right) \quad \cdots \quad AF\left(\boldsymbol{a}_{NHU} \cdot \boldsymbol{l}^{(k+1)} + b_{NHU}\right)\right],$$
$$\boldsymbol{P}_{k+1} = \boldsymbol{P}_k - \boldsymbol{P}_k\boldsymbol{H}_{k+1}^T\left(\boldsymbol{I} + \boldsymbol{H}_{k+1}\boldsymbol{P}_k\boldsymbol{H}_{k+1}^T\right)^{-1}\boldsymbol{H}_{k+1}\boldsymbol{P}_k, \qquad \text{and} \qquad \boldsymbol{P}_0 = \left(\boldsymbol{H}^T\boldsymbol{H}\right)^{-1}.$$

The initial training shown in Equation 10 requires the rank of $\boldsymbol{H}$ to be NHU. In order to do achieve this, it is required that $N \geq NHU$. However, our tests showed that the pDE algorithm tended to generate the training points in groups of very similar input values. This caused the matrix inverse in Equation (10) to be ill-conditioned. This is known as collinearity of the samples and it is a known issue when using the least squares method.[25] The use of this matrix caused very high numerical errors in some random cases. However, we found that using $N = 2NHU$ eliminated this problem in the test cases considered as the chances of having at least NHU independent samples in $\boldsymbol{H}$ was very high. Lower N values resulted in very high errors in some tests.

This requirement of a minimum number of training points is one of the main limitations of the OS-ELM method. In general, higher number of hidden units mean better accuracy. However, this requires a larger number of points for initial training. This is relevant in this work, as generating additional training points has a high computation cost. As no prediction can be made before initial training is complete, the surrogate cannot be used online while these points are being generated.

OS-ELM Multi-Agent System

The training process of the OS-ELM is extremely fast compared to the time required to generate every training point. In our tests, the update of an average OS-ELM took $\sim 0.003\,\text{s}$ per data point, while the computation of each point took $\sim 120\,\text{s}$. This can be exploited by combining the predictions of several surrogates built with the same OS-ELM model but with different random initializations. The method used for the combination is know as the ensemble method. We designed an ensemble method based on the ELM-MAS.[14] This method uses a series of child ELMs and a parent ELM. The children work as standard ELMs. However, the predictions of all of them are used as inputs for the parent, whose output is a corrected prediction for the output value. Therefore, the input size for the parent ELM is $NM \cdot n_O$, where NM is the number of children. When training, the children are first trained as usual and then the predictions on the training points are used as inputs for the training of the parent.

The method in Reference 14 does not consider the possibility of online training. However, we included that option by replacing the ELMs with OS-ELMs. The initial training of the model is done as described for ELM-MAS. The online training is done by updating the children as usual for a OS-ELM with each training point and then using the updated prediction of the children at the point as input for the training of the parent OS-ELM. We named this implementation of the model OS-ELM-MAS. To the best of our knowledge, this is the first time the ELM-MAS has been used online. The use of OS-ELM-MAS proved to be beneficial in the cases tested as it consistently achieved lower errors than the standard OS-ELM.

Error Estimation

We selected the Mean Absolute Error (MAE) as the error measure to decide if the surrogate is accurate enough to be used during the optimization. In order to use the surrogate online as described, the error of the algorithm needs to be estimated in real time during the computation. In Reference 26, the prequential error of a model is argued to be an adequate option to estimate the performance of a model when it is trained online. This error estimation is computed from each point used for training before training on it. As the model has not trained on the sample, the sample works as a validation one. This way, the prequential error works as an estimation of the validation error. We selected the implementation of the prequential error with fading factor, M_k, which is computed as

$$M_k = \frac{S_k}{D_k}, \qquad \text{where} \qquad S_k = \epsilon_k + \alpha S_{k-1}, \qquad D_k = 1 + \alpha D_{k-1},$$

α is the fading factor, and ϵ_k the prediction error on sample k. We selected $\alpha = 0.999$ in this work. Our tests indicated that this value produces results close to the validation error as long as more than 20 samples are used to estimate the error.

TEST SETUP

The objective of this work was to design a method that can be used for the preliminary design of any MGATs with LTP. We described the method we designed in the previous sections. In this section, we present the procedure we followed to evaluate the performance of our method. In particular, we focused on the difference in performance with and without the ML surrogate.

Test Cases

We selected three different cases that represent a variety of past and proposed missions to evaluate the performance of our method. These missions were selected based on three criteria: (1) the expected improvement when using a ML surrogate, (2) the availability of results using similar methods in the literature, (3) the variety of number of legs, propulsion parameters and destinations. The cases matching these criteria were the two first legs of the Dawn mission (Dawn), an Earth-Earth-Jupiter (EEJ) transfer, and an Earth-Venus-Venus-Mercury-Mercury-Mercury (EVVMMM) transfer.

The parameters of each case sere selected based on the results in References 3 and 20. These references were considered to use the most similar procedure to the one used in this work to optimize the trajectories considered. In all cases, we used the same the range of departure dates, the spacecraft parameters, and the velocity constraints at the initial and final point of the trajectory that were used for the optimization in the reference. The number of revolutions used to obtain the initial shape-based solution was set to $n_{rev} \in \{0, 1, 2, 3\}$. The references do not specify the value used but none of the example trajectories shown completes more than 3 revolutions during a single leg. The TOF range allowed for the intermediate legs was not specified in the references. Instead, we selected the ranges in which the shape-based method could always find a solution during preliminary tests. Solutions outside these ranges had a very high cost when a solution could be found, so they were not considered of interest. The numerical values of the parameters described are shown in Table 1. Finally, all cases were computed with $n_{seg} = 10$ for the SF legs, as this value was used in Reference 20.

Table 1. Parameters of the cases tested.

Short name	Dawn[3]	EEJ[20]	EVVMMM[20]
Sequence	Earth-Mars-Vesta	Earth×2-Jupiter	Earth-Venus×2-Mercury×3
T_{max} (N)	0.368	2.26	0.34
I_{sp} (s)	2620	6000	3200
$m^{(0)}$ (kg)	2481	20 000	1300
$t^{(0)}$ (MJD2000)	7305 to 8401	7305 to 10 958	7305 to 10 762
$TOF^{(1)}$ (d)	100 to 2000	100 to 1000	100 to 1500
$TOF^{(2)}$ (d)	100 to 2000	1000 to 3000	100 to 1500
$TOF^{(3)}$ (d)	–	–	100 to 1500
$TOF^{(4)}$ (d)	–	–	100 to 1500
$TOF^{(5)}$ (d)	–	–	100 to 1500
$v^{(0)}$ (km s^{-1})	0 to 3.5	0 to 2	0 to 1.925
$v^{(1)}$ (km s^{-1})	0 to 9	0 to 15	0 to 9
$v^{(2)}$ (km s^{-1})	0	0	0 to 9
$v^{(3)}$ (km s^{-1})	–	–	0 to 9
$v^{(4)}$ (km s^{-1})	–	–	0 to 9
$v^{(5)}$ (km s^{-1})	–	–	0 to 0.5017

Dawn The parameters of this transfer are shown in the first column of Table 1. The ranges of the optimization variables were based on the values presented in Reference 3. The value of the maximum allowed relative velocity with respect to the Mars at the intermediate encounter was not reported in the reference, so we used twice the value of the optimal solution reported. In addition, the original spacecraft had a Solar Electric Propulsion (SEP) system, but we assumed a NEP system instead. This case was selected as an example of transfers to objects whose orbit has a semi-major axis similar to that of the Earth.

EEJ The parameters of this transfer are shown in the second column of Table 1. The ranges of the optimization variables were based on the values presented in Reference 20. The value of the maximum allowed relative velocity with respect to the Earth at the intermediate encounter was not reported in the reference, so we used twice the value of the optimal solution reported. We selected this case because the low-thrust transcription method used in the reference is also SF, although thrust is approximated as an impulse at the midpoint of the segments and Sundman transformation is not used. Therefore, an indication of the expected performance was available in the reference but results could not be directly compared. This case is an example of a transfer to an outer planet, and the final semi-major axis of the orbit is several times larger than the initial.

EVVMMM The parameters of this transfer are shown in the third column of Table 1. The reference values were obtained again from Reference 20. In this case, neither the maximum allowed velocity at the intermediate encounters or its optimal value were indicated in the reference. Instead, we used twice the largest velocity magnitude at any encounter in the similar transfer presented in Reference 3 as the maximum allowed relative velocity at any intermediate encounter. This case is an example of a transfer to an inner planet. In addition, it has a higher number of legs, which served as a way to evaluate the variations in the performance of the method proposed with the number of legs.

Benchmark

For the evaluation of the improvement achieved when using the surrogate, we defined a baseline optimization as one that used exclusively the numerical optimization to compute the fuel cost. The central hypothesis of this work was that the use of the surrogate allows for a faster optimization than with the baseline. We defined a benchmark to evaluate this hypothesis. Both the optimization with the surrogate and with the baseline were run for a fixed computation time, and the final results were compared. If the surrogate result was better, we hypothesize it was due to the additional function evaluations that could be performed in the same time. If the baseline result was better, the conclusion is that the errors introduced by the imperfect estimation by the surrogate were too large for a successful optimization. For these tests, comparing optimizations for a fixed computation time was considered a better measure than for a fixed number of function evaluations. The reasons are that the overhead corresponding to surrogate training and predictions was also considered this way and there was a great variation on the computation time of the cost function depending on the leg parameters.

The performance measure was the fuel cost value of the best individual computed during the optimization. When using the surrogate, the optimization algorithm did not report the true cost values, reporting instead the values estimated by the surrogate. To make a fair comparison, the true cost values of the individuals corresponding to the last population of the optimization with the surrogate were computed again with the numerical optimization procedure. To account for that, extra time was reserved at the end of the optimization with the surrogate. We set the run time to $3\,\mathrm{d} = 72\,\mathrm{h}$, but there were small individual variations as the pDE algorithm can only stop between generations. These variations were not considered to have an important effect on the final results as the cost of the best individual only improved in a small proportion of the generations computed. Therefore, the chances of one extra generation changing the result were low.

Parameters Test

Two parameters were expected to affect the results obtained with the optimization procedure when the OS-ELM-MAS is used: the error threshold τ, and the number of transfers between updates of the error estimation C_n. Moreover, the pDE algorithm required the definition of a number of individuals N_I. The high computation cost of each optimization run is expected to limit the number of test runs that can be performed when applying this method to the optimization of new missions. To solve this problem, we devised a procedure to search for the adequate values of the parameters in a small number of trials. The procedure is described in the following lines and was applied to each of the three test cases.

1. We run the optimization for several N_I values and a fixed random initial state for the pDE, and the most adequate N_I value was selected. Simultaneously, we run the optimization using the surrogate with the same random initial state and N_I values. The initial values $C_n = 4$ and $\tau = 0.06$ were selected by estimating the order of magnitude of the expected errors. The results of these optimizations helped define the C_n and τ ranges to be tested in the next step.
2. We tested several combinations of parameters to determine the best C_n and τ values. These combinations were selected based on the previous results. In general, a large difference between the predicted final value by the surrogate and the final value computed when reevaluating the final population with the numerical optimization means C_n was too large. However, small C_n values resulted in more SF computations, decreasing the speed of the optimization. When it comes to τ, the objective was to select a value that included most of the points so the surrogate was used during most of the time, but that excluded error peaks. As mentioned before, we hypothesize that these peaks correspond to points in which the optimization algorithm moves into a new area of the input space. The values of C_n and τ that resulted in a better surrogate performance were used in the next step.
3. We repeated the optimization with the final C_n and τ combination and 6 different random initializations for the pDE. All random initializations were used to obtain results both with the baseline and using the surrogate. Then, we tested the hypothesis that the means of the result with the surrogate and the baseline were different with a paired t-test. This test is described in the following section. The results from the 6 random initializations were the samples used for the paired t-test.

Paired T-Test

The final result of an optimization J^* was defined as the best J value of any individual computed during the optimization. As mentioned before, the final population obtained when using the surrogate was reevaluated using the SF method. The values of the reevaluated population were the ones considered as the final results. In addition, we also considered the cost values of the last individuals before the initial training of the surrogate was completed. Therefore, the final value for the optimization with the surrogate was the best individual among the recomputed final population and the last population before the initial training is completed.

As the optimization results are stochastic, we considered J^* to be a random variable with an unknown distribution. We evaluated the statistical significance of the results with the final parameter combination by performing a statistical hypothesis test. The samples were pairs of results obtained using the surrogate and with the baseline and using the same random initialization in both cases. These can be regarded as two dependent observations. The paired t-test[27] could determine whether the mean of the results with the surrogate was significantly different to the mean of the results with the baseline. The null hypothesis was that both means were equal. This test assumed the difference between the pairs of observations were normally distributed. The confidence value reported in the results corresponds to the confidence to reject the null hypothesis. If the null hypothesis was not rejected, the conclusion was that the surrogate had no effect on the optimization procedure. In case it was rejected, the conclusion was that the surrogate was beneficial if the mean of the results with the surrogate was lower, and that the surrogate was detrimental otherwise.

Probability of Obtaining the Best Result with the Surrogate

The mean of the result distributions was not the only relevant information that could be obtained by analyzing the distribution of the results. When designing a mission, it is common to repeat the optimization several times to mitigate the effects of the random initialization of the optimization algorithm.[3] When this is done, the only relevant result is usually the best one as that is the trajectory followed in the end. Assuming the Cumulative Distribution Function (CDF) of J^*, $F_{J^*}(J)$, and its Probability Density Function (PDF), $f_{J^*}(J)$, are known, the CDF and PDF of the best value after n tests are[28]

$$F_{J^*_{(1)}}(J) = 1 - \left(1 - F_{J^*}(J)\right)^n, \qquad \text{and} \qquad f_{J^*_{(1)}} = \frac{dF_{J^*_{(1)}}}{dJ} = n f_{J^*}\left(1 - F_{J^*}\right)^{n-1}$$

respectively. Therefore, the probability of obtaining a better overall value when using the surrogate is

$$P\left(J^{*S}_{(1)} > J^{*B}_{(1)}\right) = F_{J^{*S}_{(1)} - J^{*B}_{(1)}}(0) = \int_{-\infty}^{0} \left(\int_{-\infty}^{\infty} f_{J^{*S}_{(1)}}(J) f_{J^{*B}_{(1)}}\left(-\left(z - J\right)\right) dJ\right) dz, \tag{11}$$

where superscript S refers to results with the surrogate and B to results with the baseline. We computed this probability by assuming J^{*S} and J^{*B} followed normal distributions with the same mean μ_{J^*} and standard deviation σ_{J^*} as the samples used for the paired t-test. The integrals in Equation (11) were computed numerically. The results gave an estimation of the probability of obtaining a better result with the surrogate when the optimization is repeated n times.

OS-ELM-MAS Architecture Parameters

The OS-ELM-MAS model has some parameters that have to be defined before surrogate is created. These are know as architecture parameters, and are the following ones: the number of child OS-ELM-MAS, the number of hidden units of the OS-ELM-MAS, the number of hidden units of the parent OS-ELM-MAS, and the activation function in the hidden layer. Typically, the input and output data is normalized before passing it to the OS-ELM.[15] We considered the normalization procedure used as an architecture parameter. The architecture parameters selection aimed to achieve an adequate trade-off between prediction accuracy and lowest predicted value. The lowest predicted value was considered a relevant measure because very low predictions were more likely to be selected by the optimization algorithm.

We selected the parameters by testing both accuracy and lowest predicted value when training in the data from the legs computed during the optimization of the Dawn case with 20 individuals and no surrogate. This training data was not representative of all the test cases. However, we purposely limited the test to this data as we considered this accurately represented a realistic situation in which data for architecture selection was limited. The best combination was found to be 16 child OS-ELM-MASs with 128 hidden units each, 64 hidden units in the parent OS-ELM-MAS, and the *hyperbolic tangent* as activation in the hidden layer. In addition, both input and output data were scaled so their mean was 0 and standard deviation was 1 for each variable. The scaling factors were determined with the initial training data.

Computational Implementation

We implemented the method described in Python. We used the *Pykep* package[29] for the SF computations and astrodynamics related operations such as ephemerides lookup, the *Pygmo* package[30] for the optimization tools, and the *TensorFlow-OS-ELM* package[*] for the OS-ELM implementation. Finally, we implemented the spherical shape-based method using the Scipy tools[†] and following the description in the literature.[22] The full code of the implementation is available in GitHub[‡]

[*]`https://github.com/otenim/TensorFlow-OS-ELM` (accessed May 13, 2020).
[†]`https://www.scipy.org/` (accessed May 13, 2020).
[‡]`https://github.com/PabloGoPe/MScThesis` (accessed May 13, 2020).

RESULTS

This section presents the results the results obtained for each of the three cases and an analysis of the probability of obtaining a better result with the surrogate than with the baseline.

Dawn

The first step in the process to determine the best parameters of the optimization and the surrogate was to determine the adequate number of individuals for the pDE. In the Dawn, we run the optimization with $N_I \in \{20, 50, 100\}$. The evolution of the results during the optimization can be seen in the plot a) of Figure 1. From the results in the plot, it was clear that the case $N_I = 20$ outperformed the other two when the surrogate was not used. Therefore, this value was used for the remaining tests.

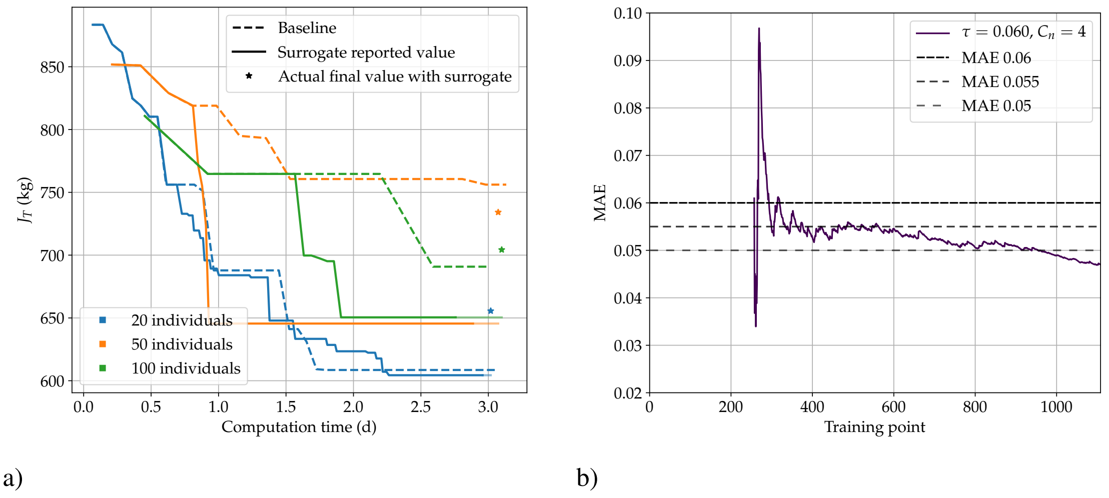


Figure 1. Results of initial test of Dawn case: a) best value obtained as a function of computation time and b) surrogate error estimation with $N_I = 20$.

The second step was to determine the best C_n and τ. The results presented in the plot a) of Figure 1 indicate that the difference between the expected and the actual value provided by the surrogate was within a 10% of the total cost. This value was considered adequate, so the value $C_n = 4$ was also considered adequate. In total, the values $C_n \in \{3, 4, 6\}$ were also tested to analyze the effect of C_n in the final result. When it comes to τ, the value selected was higher than the estimated error during most of the optimization process, as shown in plot b) of Figure 1. This indicated that the threshold was not useful to discriminate most error outliers. However, it left out the highest peaks, so the initial guess was not considered completely wrong. Therefore, the values selected to be tried were $\tau \in \{0.060, 0.055\}$, which were the same value and a slightly lower one. All the combinations of the C_n and τ values were tested. In addition, a test with $\tau = 0.050$ was also performed to evaluate even lower threshold values. However, this was not expected to provide good results so it was tested only with $C_n = 4$. The final values of the optimization using the surrogate with the parameter combinations previously selected are shown in Table 2. The test case with $\tau = 0.055$ and $C_n = 4$ reached a lower cost value than the baseline without surrogate. We concluded that the results agree with our expectations: the value $C_n = 4$ seemed to provided the optimal error update frequency, and $\tau = 0.055$ was adequate to leave out only the error outliers.

The third step was to determine the statistical significance of the results. The results of the tests for 6 random initializations of the pDE can be seen in Table 3. From the paired t-test results, we can conclude that the mean value of the optimization with the surrogate was higher than without. In addition, the best result we found had a very similar cost to the best result in Reference 3.

Table 2. Best J, final estimated MAE, number of cost function evaluations, number of leg computations with SF, and SF convergence rate for a range parameters in the Dawn case. The selected combination is shown in bold.

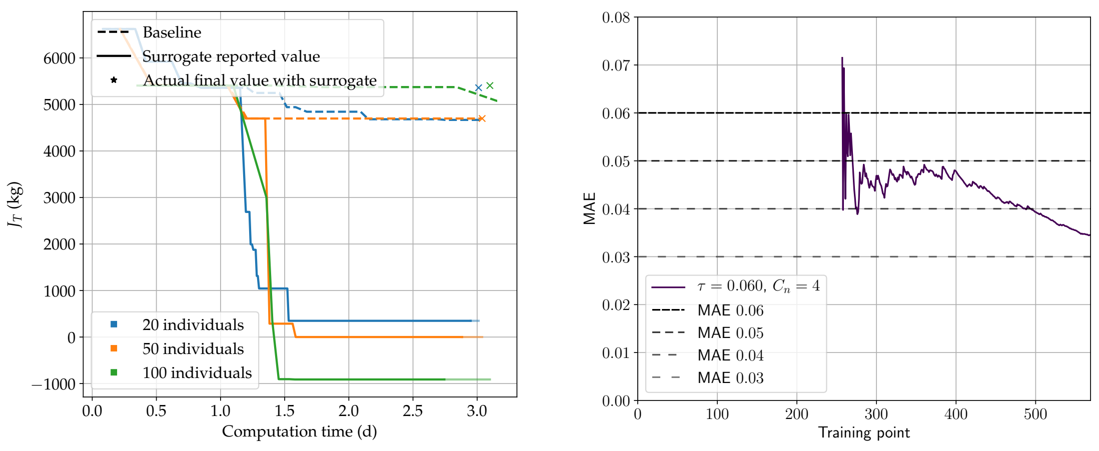
C_n	3		4			6		Baseline
τ	0.06	0.055	0.06	0.055	0.05	0.06	0.055	
J^* (kg)	676	609	656	**579**	713	674	687	609
Final MAE	0.0336	0.0318	0.0470	**0.0300**	0.0309	0.0435	0.0299	–
J evaluations	3560	3880	4360	**4340**	3660	5660	6200	920
SF evaluations	1337	1419	1302	**1361**	1365	1203	1337	1731
SF conv. rate	0.840	0.867	0.850	**0.896**	0.861	0.855	0.898	0.898

Table 3. Statistics of the results for the Dawn case. The confidence level is to reject the null hypothesis of the t-test (both means are equal). A negative confidence value indicates that the mean of the baseline cases is lower (i.e. better) than the mean using the surrogate.

Reference 3	Baseline			Surrogate			Confidence
$J^*_{(1)}$ (kg)	μ_{J^*} (kg)	σ_{J^*} (kg)	$J^*_{(1)}$ (kg)	μ_{J^*} (kg)	σ_{J^*} (kg)	$J^*_{(1)}$ (kg)	
534	656	27	609	686	63	579	$-86\,\%$

Earth-Earth-Jupiter

As in the Dawn case, we started the tests for the EEJ case by running the optimization with $N_I \in \{20, 50, 100\}$. The evolution of the results during the optimization can be seen in plot a) of Figure 2. The results showed that $N_I = 20$ outperformed the other two cases when the surrogate was not used. Therefore, the value used in the remaining tests was $N_I = 20$.



a) b)

Figure 2. Results of initial test of EEJ case: a) best value obtained as a function of computation time and b)surrogate error estimation with $N_I = 20$.

The second step was again to determine the best C_n and τ. This case had the particularity of having a low convergence rate when computing the legs with the SF method. This had two important consequences when using the surrogate: the number of training points available was much lower as only converged points could be used, and the chances of the surrogate selecting a valid transfer at the end of the optimization were low. First, the evolution of the predicted best value in plot a) of Figure 2 showed that the surrogate prediction were unrealistically optimistic. In some cases, the surrogate predicted negative fuel use. This was obviously wrong but those results could not be immediately disregarded as the true cost was higher but it might still be the best solution. The extreme values predicted could be mitigated by using lower C_n values. The values tested were $C_n \in \{1, 2, 3, 4\}$. When it comes to the τ value, the estimated MAE values were in general lower than in the Dawn case so the threshold values considered were $\tau \in \{0.06, 0.05, 0.04, 0.03\}$. The final values of the optimization using the surrogate with the parameter combinations previously selected are shown in Table 4. Notice that the same values were obtained for different parameters. The reason for this was that the MAE estimation reached values below all the thresholds very fast, so it did no make any difference for the optimization. The test case with $\tau = 0.03$ and $C_n = 3$ reached a lower cost value than the baseline without surrogate. We used these parameters for the remaining tests.

Table 4. Best J, final estimated MAE, number of cost function evaluations, number of leg computations with SF, and SF convergence rate for a range parameters in the EEJ case. The selected combination is shown in bold.

C_n	1		2		3			4		Baseline
τ	0.06[a,b]	0.03	0.06[a,b]	0.03	0.06[a]	0.04	0.03	0.06[a,b]	0.03	
J^* (kg)	4884	4681	5215	4681	5288	5359	**4674**	5359	4807	4667
Final MAE	0.0306	0.0299	0.0206	0.0287	0.0311	0.0363	**0.0297**	0.0345	0.0305	–
J evaluations	1880	1040	2560	1000	3540	2920	**1100**	3540	1100	960
SF evaluations	1664	1863	1529	1856	1537	1475	**1708**	1351	1771	1824
SF conv. rate	0.424	0.513	0.607	0.531	0.362	0.382	**0.506**	0.420	0.506	0.528

[a] Results identical for $\tau = 0.05$. [b] Results identical for $\tau = 0.04$.

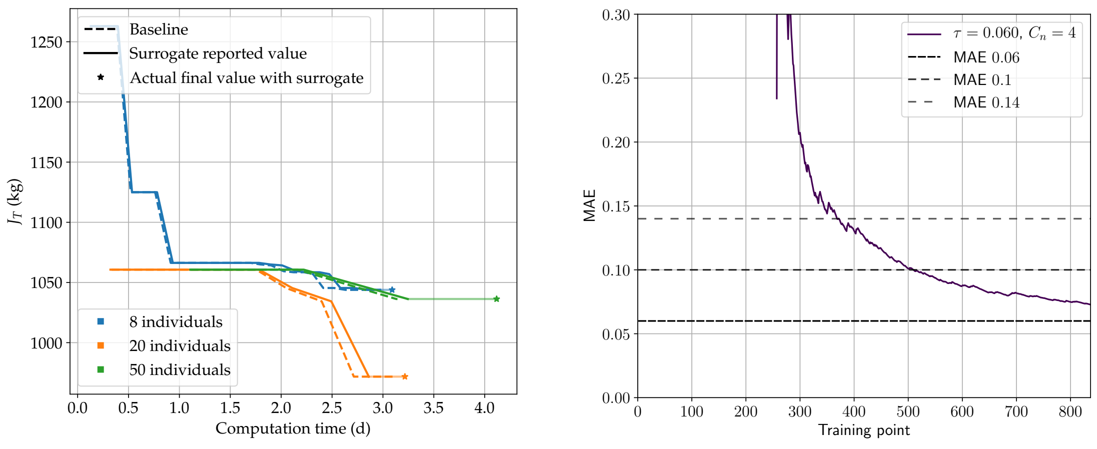
Finally, we evaluated the statistical significance of the results. The results of the test can be seen in Table 5. The results when the surrogate was used were almost identical to the ones of the baseline. The main cause for this was probably the low convergence of the SF results. The surrogate had a low number of training points and the τ value needed to be very restrictive to avoid excessively optimistic predictions on invalid points. As a consequence, the surrogate made predictions in very few cases, and had almost no effect on the results. The best result obtained had a higher cost value than the best result in Reference 20. We hypothesize that the reason was the different transcription method used in this work and in the reference. Our solution is guaranteed to be feasible, while the one in the reference might not be feasible.[24]

Table 5. Statistics of the results for the EEJ case. The confidence level is to reject the null hypothesis of the t-test (both means are equal). A negative confidence value indicates that the mean of the baseline cases is lower (i.e. better) than the mean using the surrogate.

Reference 20	Baseline			Surrogate			Confidence
$J^*_{(1)}$ (kg)	μ_{J^*} (kg)	σ_{J^*} (kg)	$J^*_{(1)}$ (kg)	μ_{J^*} (kg)	σ_{J^*} (kg)	$J^*_{(1)}$ (kg)	
2898	4669	129	4534	4675	135	4496	$-10\,\%$

Earth-Venus-Venus-Mercury-Mercury-Mercury

This case was different from the previous ones, as the number of legs of each transfer was 5 instead of 2. This meant that we expected longer computation times per individual than in the previous case. Therefore, the values $N_I \in \{8, 20, 50\}$ were used to roughly have the same number of leg computations per generation as in the Dawn and EEJ cases. The evolution of the results during the optimization can be seen in plot a) of Figure 3. In this case, the best result was obtained with $N_I = 20$. However, both the $N_I = 20$ and $N_I = 50$ cases required much longer computation times until the initial training of the surrogate was completed. This was a combination of two factors: a lower convergence rate and longer time for the computation of each individual point. This meant that only one population was computed with the surrogate before the computation time limit was reached when $N_I = 20$ and $N_I = 50$. Therefore, the value selected for the remaining tests was $N_I = 8$ as we considered that this value was more interesting as we could evaluate the effects of the surrogate on the optimization.



a) b)

Figure 3. Results of initial test of EVVMMM case: a) best value obtained as a function of computation time, and b) surrogate error estimation with $N_I = 8$.

The next step was to determine the best C_n and τ values. When the surrogate was used, the error estimation never reached values below $\tau = 0.06$. Therefore, the information relative to the selection of surrogate parameters obtained from this test was limited. The obvious conclusion was that a larger value of τ was required. We decided on $\tau \in \{0.100, 0.140\}$ for the following tests. The value of C_n was more difficult to determine as the effect of the surrogate predictions could not be evaluated. However, the convergence rate was high as in the Dawn case, and every leg required a longer computation time when computed with the SF method. In the Dawn case, $C_n = 4$ resulted in the best performance, and higher C_n values mean fewer SF computations per generation. Therefore, the values selected for the remaining tests were $C_n \in \{4, 6, 8\}$. The final values of the optimization using the surrogate with the parameter combinations previously selected are shown in Table 6. The test case with $\tau = 0.100$ and $C_n = 4$ reached a lower cost value than the baseline. These results agreed with our expectations: as in the Dawn case, the value $C_n = 4$ seemed to be adequate to estimate the error in real time, and $\tau = 0.100$ was low enough to leave out the higher error cases but still use the surrogate frequently enough to speed up the optimization.

Table 6. Best J, final estimated MAE, number of cost function evaluations, number of leg computations with SF, and SF convergence rate for a range parameters in the EVVMMM case. The selected combination is shown in bold.

C_n	4		6		8		Baseline
τ	0.10	0.14	0.10	0.14	0.10	0.14	
J^* (kg)	**963**	1066	1063	1066	1066	1066	1044
Final MAE	**0.0881**	0.0838	0.0937	0.0876	0.0898	0.0995	–
J evaluations	**1820**	3100	840	4120	3920	3980	580
SF evaluations	**698**	662	744	670	685	644	1053
SF conv. rate	**0.838**	0.796	0.817	0.797	0.788	0.828	0.901

We evaluated again the statistical significance of the results. The results of the test can be seen in Table

7. We concluded that the mean value of the optimization with the surrogate was lower than the optimization without the surrogate. In this case, the conditions were the most adequate for the use of the surrogate: each trajectory had a higher number of legs, which meant more computation time per individual, each leg had a higher computation time than in the previous cases, and the convergence rate of the SF method was very high, so chances of selecting an invalid trajectory when the surrogate was used were low. The best result obtained had a higher cost value than the best result in Reference 20. We hypothesize that the reason was the different transcription method used in this work and in the reference. Our solution is guaranteed to be feasible, while the one in the reference might not be feasible.[24]

Table 7. Statistics of the results for the EVVMMM case. The confidence level is to reject the null hypothesis of the t-test (both means are equal). A negative confidence value indicates that the mean of the baseline cases is lower (i.e. better) than the mean using the surrogate.

Reference 20	Baseline			Surrogate			Confidence
$J^*_{(1)}$ (kg)	μ_{J^*} (kg)	σ_{J^*} (kg)	$J^*_{(1)}$ (kg)	μ_{J^*} (kg)	σ_{J^*} (kg)	$J^*_{(1)}$ (kg)	
236	1088	68	998	1037	142	880	62 %

Best Result Statistics

We used Equation (11) to estimate $P(J^{*S}_{(1)} > J^{*B}_{(1)})(n)$ assuming that both J^{*S} and J^{*B} follow a normal distribution with the mean μ_{J^*} and the standard deviation σ_{J^*} presented in Tables 3, 5, and 7. The result is shown in Figure 4. In the Dawn and EVVMMM cases, the use of the surrogate was the best option if several random initializations were going to be used. However, the probability remains at around 50% independently of the number of tests performed for the EEJ case. This is a logical result as the distributions of J^{*S} and J^{*B} are almost identical in this case. These conclusions agree with our experimental results, in which $n = 6$ and the best results were obtained with the surrogate in all three cases.

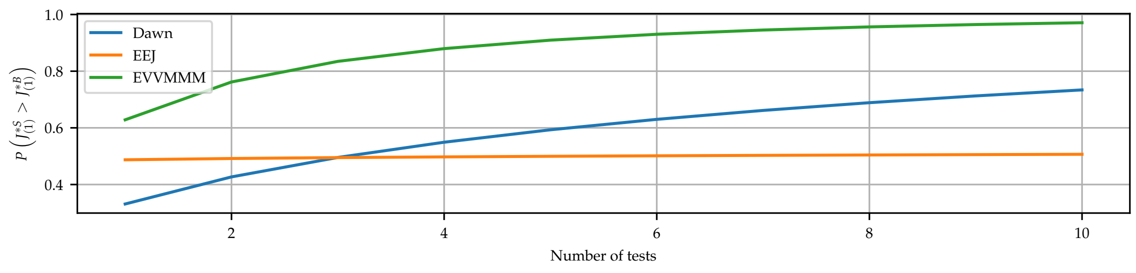


Figure 4. Probability of obtaining a better result when using the surrogate than with the baseline as a function of the number of cases tested.

CONCLUSIONS

The use of the surrogate during the optimization is always expected to be beneficial in the EVVMMM case. This case was the most promising for the use of the surrogate due to the longer computation time required for each trajectory computation. The Dawn case was not expected to be as adequate for the use of the surrogate because the cost function evaluations were faster, but the surrogate is also expected to be beneficial if more than 4 optimization with different initializations are performed simultaneously. Finally, the EEJ case had a very low convergence rate for the SF method so the optimal strategy for the surrogate was to be very conservative with the predictions, resulting in almost identical results to the baseline.

These results were obtained without adapting the OS-ELM-MAS architecture parameters to each case. In fact, the architecture was selected based on data from the Dawn case and it showed a better performance in the other two cases. This proved that the OS-ELM-MAS method can be applied to different problems without having to adapt the parameters. This may potentially save an important amount of computation time, as the data to test the parameters does not need to be generated and repeated training with different parameters combinations can be avoided. However, the tuning of the surrogate C_n and τ parameters was required to achieve better results with the surrogate than with the baseline. Therefore, the objective of having a method that could be applied without any tuning was not achieved. Nonetheless, there seems to be a relationship between the optimal parameters and the structure of the data for a problem. The proof of this is that we managed to find a combination of parameters for which the surrogate had a better performance than the baseline by testing a small number of carefully selected parameter combinations. We are confident that a general approach to estimate the optimal surrogate parameters from a very small data sample can be obtained in the future.

REFERENCES

[1] T. J. Debban, T. Troy McConaghy, and J. M. Longuski, "Design and optimization of low-thrust gravity-assist trajectories to selected planets," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, No. August, 2002, pp. 1–10, 10.2514/6.2002-4729.

[2] K. Alemany and R. D. Braun, "Survey of Global Optimization Methods for Low- Thrust , Multiple Asteroid Tour Missions," *AAS/AIAA Space Flight Mechanics Meeting January 2007, Sedona, Arizona*, 2007, pp. 1–20.

[3] J. A. Englander and B. A. Conway, "Automated solution of the low-thrust interplanetary trajectory problem," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 1, 2017, pp. 15–27, 10.2514/1.G002124.

[4] M. Vasile and P. De Pascale, "Preliminary Design of Multiple Gravity-Assist Trajectories," *Journal of Spacecraft and Rockets*, Vol. 43, jul 2006, pp. 1065–1076, 10.2514/1.19646.

[5] B. Dachwald, "Optimization of very-low-thrust trajectories using evolutionary neurocontrol," *Acta Astronautica*, Vol. 57, No. 2-8, 2005, pp. 175–185, 10.1016/j.actaastro.2005.03.004.

[6] C. Sánchez-Sánchez and D. Izzo, "Real-Time Optimal Control via Deep Neural Networks: Study on Landing Problems," *Journal of Guidance, Control, and Dynamics*, Vol. 41, mar 2018, pp. 1122–1135, 10.2514/1.G002357.

[7] E. D. Mendez Ramos, P. Mishra, S. Edwards, and D. Mavris, "Response Surface Regressions for Low-Thrust Interplanetary Mission Design," *AIAA SPACE*, AIAA SPACE Forum, American Institute of Aeronautics and Astronautics, sep 2016, doi:10.2514/6.2016-5651.

[8] D. Hennes, D. Izzo, and D. Landau, "Fast approximators for optimal low-thrust hops between main belt asteroids," *EEE Symposium Series on Computational Intelligence*, 2016, 10.1109/SSCI.2016.7850107.

[9] A. Mereta, D. Izzo, and A. Wittig, "Machine learning of optimal low-thrust transfers between near-earth objects," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Vol. 10334 LNCS, 2017, pp. 543–553, 10.1007/978-3-319-59650-1.

[10] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer series in statistics, New York: Springer,, 2nd ed., 2009, 10.1007/978-0-387-84858-7.

[11] C. Ampatzis and D. Izzo, "Machine learning techniques for approximation of objective functions in trajectory optimisation," *Proceedings of the International Joint Conference on Artificial Intelligence 2009 Workshop on Artificial Intelligence in Space*, Vol. 2009, No. September, 2009.

[12] H. Shang and Y. Liu, "Assessing Accessibility of Main-Belt Asteroids Based on Gaussian Process Regression," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 5, 2017, pp. 1144–1154, 10.2514/1.G000576.

[13] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, Vol. 70, No. 1-3, 2006, pp. 489–501, 10.1016/j.neucom.2005.12.126.

[14] C. T. Yaw, S. Y. Wong, K. S. Yap, and C. H. Tan, "Extreme Learning Machine neural networks for multi-agent system in power generation," *International Journal of Engineering and Technology (UAE)*, Vol. 7, No. 4, 2018, pp. 347–353, 10.14419/ijet.v7i4.35.22760.

[15] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, "A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks," *IEEE Transactions on Neural Networks*, Vol. 17, No. 6, 2006, pp. 1411–1423, 10.1109/TNN.2006.880583.

[16] R. Storn and K. Price, “Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces,” *Journal of Global Optimization*, Vol. 11, No. 4, 1997, pp. 341–359, 10.1023/A:1008202821328.

[17] M. Bardi and I. Capuzzo-Dolcetta, “Optimal control problems with continuous value functions: restricted state space,” *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations* (M. Bardi and I. Capuzzo-Dolcetta, eds.), ch. 4, pp. 227–284, Boston, MA: Birkhäuser Boston, 1997, 10.1007/978-0-8176-4755-1.

[18] M. J. H. Walker, B. Ireland, and J. Owens, “A Set of Modified Equinoctial Orbit Elements,” *Celestial Mechanics*, Vol. 36, aug 1985, pp. 409–419, 10.1007/BF01227493.

[19] G.-B. Huang, L. Chen, and C.-K. Siew, “Universal approximation using incremental constructive feedforward networks with random hidden nodes.,” *IEEE transactions on neural networks*, Vol. 17, jul 2006, pp. 879–892, 10.1109/TNN.2006.875977.

[20] C. H. Yam, D. Izzo, and D. D. Lorenzo, “Low-thrust trajectory design as a constrained global optimization problem,” *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 225, No. 11, 2011, pp. 1243–1251, 10.1177/0954410011401686.

[21] C. H. Yam, T. T. Mcconaghy, K. J. Chen, and J. M. Longuski, “Design of Low-Thrust Gravity-Assist Trajectories to the Outer Planets,” *55th International Astronautical Congress of the International Astronautical Federation, the International Academy of Astronautics, and the International Institute of Space Law*, International Astronautical Congress (IAF), American Institute of Aeronautics and Astronautics, oct 2004, pp. 1–16, doi:10.2514/6.IAC-04-A.6.02.

[22] D. M. Novak and M. Vasile, “Improved Shaping Approach to the Preliminary Design of Low-Thrust Trajectories,” *Journal of Guidance, Control, and Dynamics*, Vol. 34, jan 2011, pp. 128–147, 10.2514/1.50434.

[23] J. A. Sims and S. N. Flanagan, “Preliminary design of low-thrust interplanetary missions,” *Advances in the Astronautical Sciences*, Vol. 103, No. PART 1, 2000, pp. 583–592.

[24] C. H. Yam, D. Izzo, and F. Biscani, “Towards a High Fidelity Direct Transcription Method for Optimisation of Low-Thrust Trajectories,” *4th International Conference on Astrodynamics Tools and Techniques, 2010*, apr 2010, pp. 1–7.

[25] G. W. Stewart, “Collinearity and Least Squares Regression,” *Statistical Science*, Vol. 2, jan 1987, pp. 68–84.

[26] J. Gama, R. Sebastião, and P. P. Rodrigues, “On evaluating stream learning algorithms,” *Machine Learning*, Vol. 90, No. 3, 2013, pp. 317–346, 10.1007/s10994-012-5320-9.

[27] C. T. Le and L. E. Eberly, *Introductory Biostatistics*. Hoboken, UNITED STATES: John Wiley & Sons, Incorporated, 2016.

[28] H. A. David, *Order statistics*. New York: Wiley, 2nd ed., 1981.

[29] D. Izzo, “esa/pykep: The Gym and more.,” dec 2019, 10.5281/ZENODO.3560630.

[30] F. Biscani and D. Izzo, “esa/pagmo2: pagmo 2.15.0,” apr 2020, 10.5281/ZENODO.3738182.

3
Multi Gravity Assist Trajectories

The optimization of MGATs is a challenging problem. The parameters need to be optimized, in addition to the already challenging optimization of LTP trajectories. However, they usually lead to significant fuel savings that made possible most of the missions to very distant bodies. In this chapter, the details about the MGAT optimization problem considered are presented.

3.1. Problem definition

The definition of the problem considered is detailed in section Multiple-Gravity-Assist Optimization Problem of the thesis paper manuscript, page 7. The linked conics model used is the most common in the literature. In addition, Reference [3] showed that the inclusion of the planet gravity makes a small difference in the final results when studying a similar problem to the one considered here. The assumption that the Sphere Of Influence (SOI) of has negligible size was considered the main source of error during the computations.

3.1.1. Propulsion Model Selection

With respect to the orbit optimization procedure, the relevant parameters of the propulsion system are the maximum thrust available T_{max} and the specific impulse I_{sp}. In a general case, T_{max} and I_{sp} depend on the power available on board the spacecraft, and I_{sp} depends also on the thrust level [3, 38, 40]. Moreover, more complex dependencies are possible, such as a spacecraft fitted with several identical engines that can be turned on or off independently [41, 42].

In this thesis, a simple Nuclear Electric Propulsion (NEP) model was considered. The spacecraft was considered to have an internal constant power source for the propulsion system. The specific impulse I_{sp} was assumed to be constant during the whole transfer. This assumption is common in the literature [1, 2, 29, 30, 33–36, 39, 43–45]. T_{max} was also considered constant during the transfer, and any thrust magnitude $T \in [0, T_{max}]$ was considered possible. This is again a common assumption in the literature [2, 33–35, 39, 43, 44].

A common alternative to the NEP is the Solar Electric Propulsion (SEP), in which T_{max} depends on the solar power available. However, the described NEP model was selected over SEP or more complex alternatives due to its simplicity of implementation. Moreover, the results regarding the effects of the surrogate obtained with this model are expected to be extrapolatable to other propulsion models. First, the conclusions obtained in section Optimization Problem Formulation of the thesis paper manuscript, page 8 when using the *Hamilton-Jacobi-Bellman equation* are valid for any propulsion model as long as the thrust magnitude is bounded. Second, there is no reason to think that the shape of the cost function with a different model is going to be more difficult to approximate by the surrogate. However, this hypothesis was not tested in this work. The testing would have required the implementation and validation of the additional propulsion models and it was considered unfeasible in the restricted time available to complete the thesis.

3.2. Gravity assist

As mentioned in section Spacecraft Dynamics of the thesis paper manuscript, page 7, GA maneuvers are approximated as instantaneous changes of $\boldsymbol{v}$ at the position $\boldsymbol{R}$ of the planet. In this thesis, only unpowered GA were considered, i.e. no propulsion was used when the spacecraft was inside the SOI of the planet.

3.2.1. Implementation

As mentioned in section Gravity Assist Computation of the thesis paper manuscript, page 8, it is necessary to define a function $\Delta\boldsymbol{v}_{GA}$ such that

$$\boldsymbol{v}_a = \boldsymbol{v}_b + \Delta\boldsymbol{v}_{GA}\left(\boldsymbol{v}_b, \boldsymbol{V}, \eta, \rho, \mu\right)$$

to include the GA computation into the general optimization problem. This function is defined by following the procedure described in Reference [10]. The procedure is outlined in the following lines for easy reference.

First, the relative velocities with respect to the planet are defined as

$$\tilde{\boldsymbol{v}}_b = \boldsymbol{v}_b - \boldsymbol{V} \qquad \text{and} \qquad \tilde{\boldsymbol{v}}_a = \boldsymbol{v}_a - \boldsymbol{V}, \tag{3.1}$$

the angle γ is defined as

$$\gamma = \frac{\pi}{2} - \arccos\left(\frac{1}{\frac{\tilde{v}_b^2 \rho}{\mu} + 1}\right),$$

and the reference vector $\boldsymbol{n}_b$ is a vector perpendicular to $\tilde{\boldsymbol{v}}_b$ and contained in a predetermined plane. In this case, the x-z plane was selected, so the reference vector is

$$\boldsymbol{n}_b = \frac{\boldsymbol{u}_b \times [0,1,0]^T}{|\boldsymbol{u}_b \times [0,1,0]^T|},$$

where $\boldsymbol{u}_b = \tilde{\boldsymbol{v}}_b/|\tilde{\boldsymbol{v}}_b|$.

The rotation angle η defines a rotation with respect to the reference plane represented with the quaternion

$$\boldsymbol{q}_\eta = [\boldsymbol{u}_b^T \sin\eta, \cos\eta]^T.$$

The vector normal to the orbital plane during the GA is computed as

$$\boldsymbol{n}_\eta = \boldsymbol{Q}_\eta n_b,$$

where $\boldsymbol{Q}_\eta$ is the rotation matrix defined by $\boldsymbol{q}_\eta$. The rotation of the velocity is defined by quaternion

$$\boldsymbol{q}_\gamma = [\boldsymbol{n}_\eta^T \sin\gamma, \cos\gamma]^T.$$

Finally, the outgoing relative velocity is computed as

$$\tilde{\boldsymbol{v}}_a = \boldsymbol{Q}_\gamma \tilde{\boldsymbol{v}}_b,$$

where $\boldsymbol{Q}_\gamma$ is the rotation matrix defined by $\boldsymbol{q}_\gamma$. The final heliocentric velocity can be computed with Equation (3.1).

3.2.2. Verification

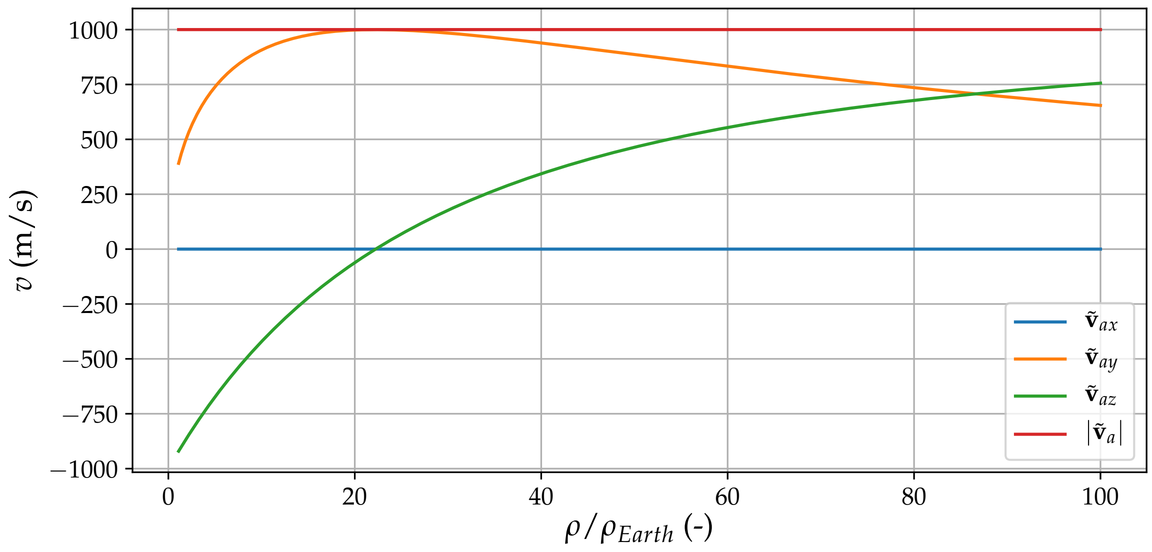


Figure 3.1: Components of the outgoing relative velocity after a GA as a function of the GA radius when $\tilde{v}_b = [0, 0, 1\,\mathrm{km\,s^{-1}}]^T$ and $\eta = 0$.

The verification of the gravity assist implementation is done by testing whether the behavior is the one expected in a simple case. The components of the outgoing

Table 3.1: Results of the GA around Venus with the parameters from Reference [4] used for validation.

Approach	ρ (km)	η (rad)	$\tilde{\boldsymbol{v}}_b$ ($\mathrm{m\,s^{-1}}$)	$\tilde{\boldsymbol{v}}_a$ ($\mathrm{m\,s^{-1}}$)	
				Reference [46]	Computed
Dark side	6352	0	$[2490, 2782, 0]^T$	$[-3289, 1766, 0]^T$	$[-3289, 1767, 0]^T$
Sunlit side	6352	$\pi/2$	$[2490, 2782, 0]^T$	$[2118, -3074, 0]^T$	$[2119, -3074, 0]^T$

velocity are computed as a function of the periapsis radius for an incoming relative velocity along the z-axis $\tilde{\boldsymbol{v}}_b = [0, 0, 1\,\mathrm{km\,s^{-1}}]^T$ and a GA on the y-z plane, i.e. $\eta = 0$. The components of $\tilde{\boldsymbol{v}}_a$ are shown in Figure 3.1. The behavior is the one expected. For large radius values, only a small change in velocity is produced. However, the deflection increases as the radius decreases, reaching the point in which $\tilde{\boldsymbol{v}}_a$ is perpendicular to $\tilde{\boldsymbol{v}}_b$. Past that point, the component in the opposite direction to the incoming velocity starts to grow again. In all cases, $|\tilde{\boldsymbol{v}}_b| = |\tilde{\boldsymbol{v}}_a|$ as expected.

3.2.3. Validation

The validation of the GA implementation is done with two tests. First, Reference [4] provides an example of a GA at Venus and provides the numerical values of the velocities after the GA. A fixed GA radius value (300 km over the surface of Venus) is considered and the GA occurs in the horizontal plane. The components of the velocity vector are given in a reference frame which has one axis oriented along the planetary velocity and the other one along the position. However, only the $\tilde{\boldsymbol{v}}_b$ components are given, so the orientation of the reference frame is irrelevant as long as it is kept consistent during the computations. Therefore, the velocity components were considered to be oriented along the x and y axes. Result are shown in Table 3.1. The values obtained closely match the ones in the reference so the validation was considered successful.

The second part of the validation was done by comparing with the GA computation function included in the Pykep package[1]. This function was not used in this work because its existence was only discovered after a newly implemented function had already been used for the computation of the results. This is an example of how the incomplete documentation of the Pykep and Pygmo packages was sometimes a problem during this thesis. Moreover, no mention was found in the documentation to the reference plane used to determine the rotation of the GA. Results could not be directly compared as the angle θ represents a different rotation in this thesis and in the function in the Pykep package. However, the locus of all velocities obtained should be the same once the complete range of possible values for η is tested. Therefore, the testing procedure was the following one: an arbitrary velocity ($\tilde{\boldsymbol{v}}_b = [1\,\mathrm{km\,s^{-1}}, 2\,\mathrm{km\,s^{-1}}, 3\,\mathrm{km\,s^{-1}}]^T$), planet (Mars), and radius ($\rho = 6000\,\mathrm{km}$) were selected and $\tilde{\boldsymbol{v}}_a$ was computed for all the possible η values with both the function

[1]https://esa.github.io/pykep/documentation/core.html#pykep.fb_prop (accessed May 13, 2020).

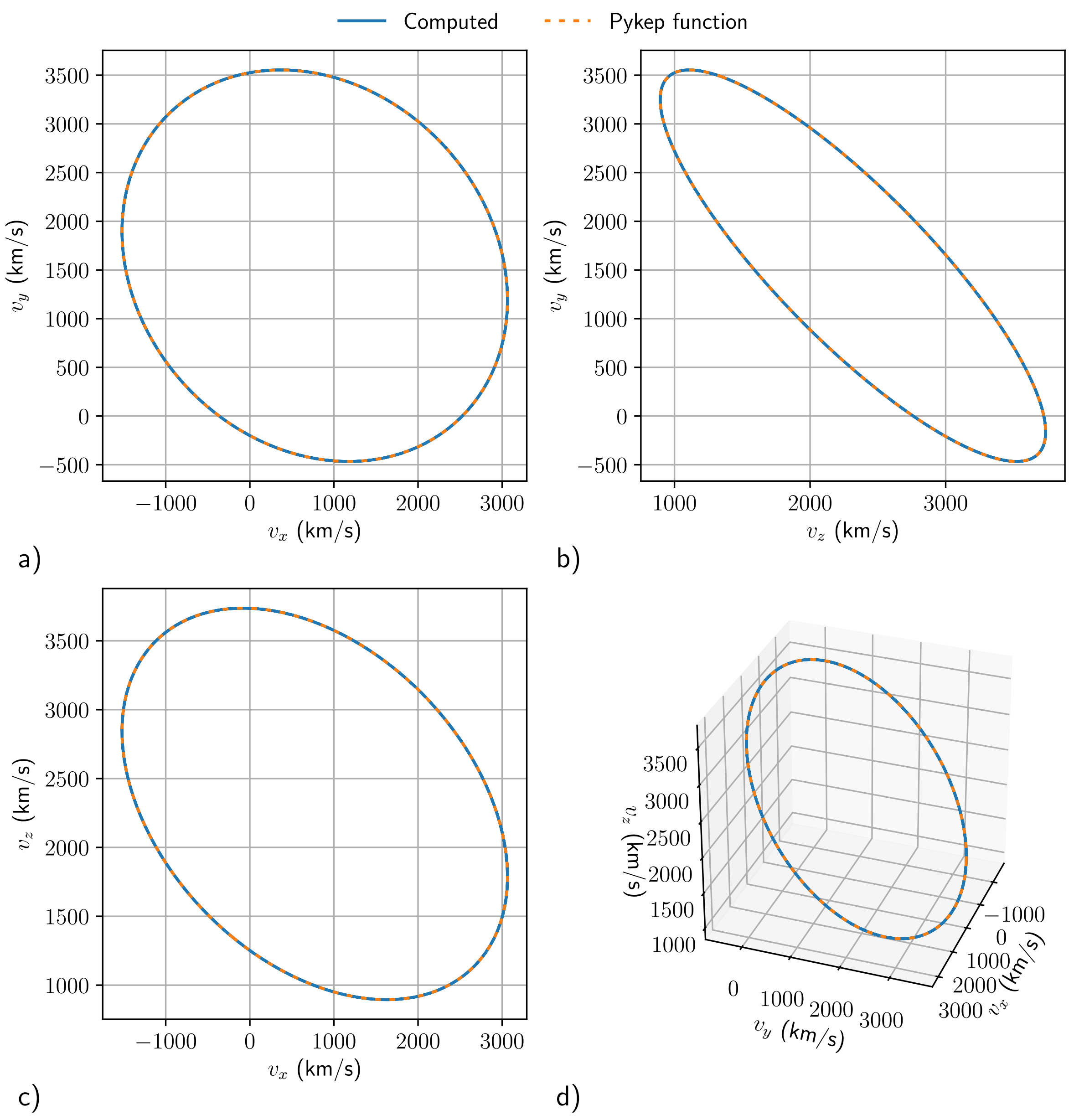


Figure 3.2: Locus of all possible $\tilde{v}_a$ for a test case computed with the function used in this thesis and with the function from Pykep: a) v_y as a function of v_x, b) v_y as a function of v_z, c) v_z as a function of v_x, d) 3D plot of the three components.

created in this work and the one from Pykep. When the locus of all $\tilde{v}_a$ values obtained was represented in the velocity component space, the curve for both functions should overlap perfectly. The results are shown in Figure 3.2. The results with both functions matched closely so the validation was considered successful.

3.3. Global Optimizer

This section outlines the global optimization algorithms that were considered for the global optimization described in section Optimization Problem Formulation of the thesis paper manuscript, page 8. It also describes the verification and validation process of the selected algorithm.

3.3.1. Algorithm Selection

The focus of this work was to speed-up the optimization algorithm so only state-of-the-art algorithms previously used for interplanetary trajectory optimization were considered: Monotonic Basin Hopping (MBH) [47], Simulated Annealing (SA) [48], Particle Swarm Optimization (PSO) [49] and Differential Evolution (DE) [50].

Both MBH and SA include a local optimization step that requires knowledge about the gradient of the cost function. This was a problem for the optimization problem considered here. No analytical expression for the gradients of the cost function was available so they would need to be computed with a numerical method. However, this would require a high number of cost function evaluations. If these computations were replaced by the surrogate, small errors in the predictions provided by the surrogate would introduce large errors in the gradient estimation. Therefore, these algorithms were discarded.

PSO algorithm was used in some orbit optimization problems before [45, 51]. However, the problems considered were different that the one in this thesis. Meanwhile, DE was applied successfully to the optimization of orbital maneuvers [52], MGAT [53–55], and low-thrust trajectories [56]. Therefore, DE was the algorithm selected for this work as there is a body of literature available. This was a key point both for selecting the most adequate implementation, and for the validation of the results.

Several versions of the DE evolution are available. As mentioned in section Global Optimization Algorithm of the thesis paper manuscript, page 9, the one used in this work is the Pygmo version of the self-adaptive Differential Evolution (pDE)[2]. This implementation of the algorithm includes methods considered in other algorithms to adapt the evolution parameters CR [57] and F [58]. In addition, it includes a method to adapt the mutation strategy among several options [59]. Therefore, there are no parameters to be set for the training of the algorithm apart from the number of individuals. Moreover, the adaptation of all the evolution parameters was considered beneficial when using the algorithm in conjunction with an online surrogate of the function. As explained in section 3.5, the cost function was replaced by a surrogate estimation during the optimization procedure. The surrogate estimation of the cost would have a slightly different shape than the original function due to the limits on the accuracy achievable, and the pDE algorithm was expected to be able to adapt to this new shape.

[2]https://esa.github.io/pagmo2/docs/python/algorithms/py_algorithms.html#pygmo.de1220 (accessed December 3, 2019).

3.3.2. Verification

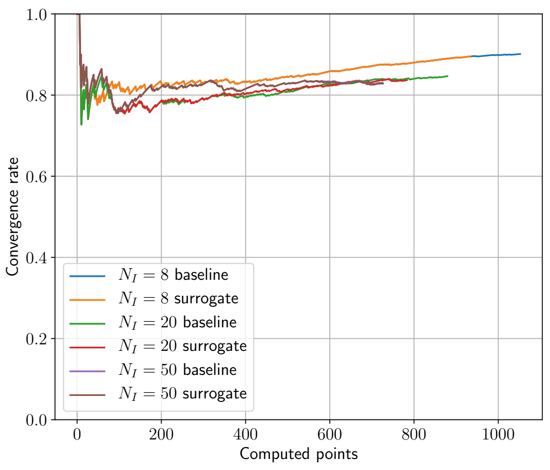
The pDE algorithm was already available in the Pygmo package and its implementation was expected to be correct. Therefore, the verification was focused on the correct interfacing between the cost function and the optimization algorithm. However, the algorithm is stochastic so the results depended on the random initialization used. This made the verification difficult as there was no direct way to tell if the results provided were the ones expected. Notwithstanding, the first part of the verification process were a set of sanity checks. First, some random individuals were selected, and it was tested whether the cost function values matched the ones reported by the optimization algorithm. This check was successful. Second, the best value of the cost function obtained was expected to decrease its value as optimization progresses. That was observed in the results in section Results of the thesis paper manuscript, page 20.

Nonetheless, the fact that the best value improved was not a guarantee that the algorithm was better than a random search. However, the rate of converged SF computations would remain constant for a random search, as the algorithm would keep trying points in the regions where $\boldsymbol{l} \notin \hat{S}'$ with the same frequency. If the optimization algorithm is working as expected, the convergence rate should increase as the optimization progresses, because the points that do not converge are heavily penalized. The convergence rate as a function of the total SF computations is shown in Figure 3.3 for the optimization tests used to determine the best number of individuals for the pDE in the Dawn, EEJ, and EVVMMM cases. It is clear that the convergence rate increased as the optimization progressed in all cases. This was a clear indication that the optimization algorithm generates new points in regions of the input space with a lower overall cost. Moreover, the configurations that achieved a higher convergence rate also achieved lower cost values during optimization.

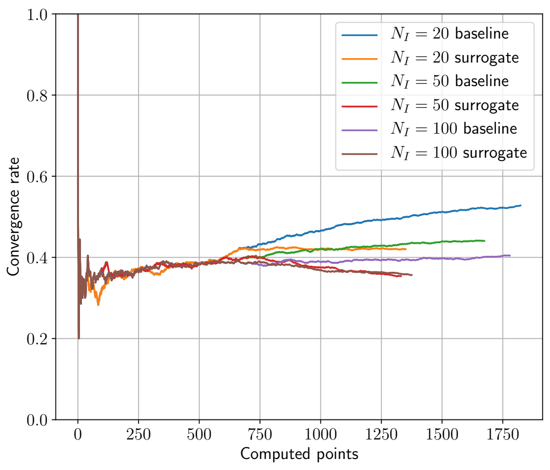
3.3.3. Validation

The external validation of the optimization tool was difficult as there are no instances of the optimization using the same cost function in the literature. The only similar problem was found in Reference [60], but a different SF implementation and different global optimization algorithms were used. The results might not be comparable because the convergence of the SF used in this work is more restrictive, as described in section 5.2. The best result found in the reference might not be a viable one with this method. Finally, the tool presented here is merely a prototype and its implementation was not optimized. The most obvious improvement that could be made is to implement the tool in a compiled language such as C++. The computations could be up to 10 times faster based on a comparison of the values for the shape-base method computation presented in subsection 4.3.2 and the ones in Reference [5]. However, the integration of all the independent tools required to obtain the results in a language such as C++ would have required much more time than it was available for this thesis.

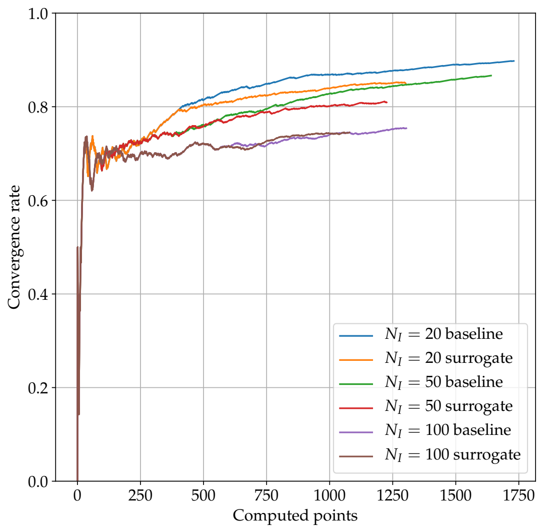
The validation procedure considered more interesting for this work was internal validation. The main question to answer regarding the method presented in this



a)



b)



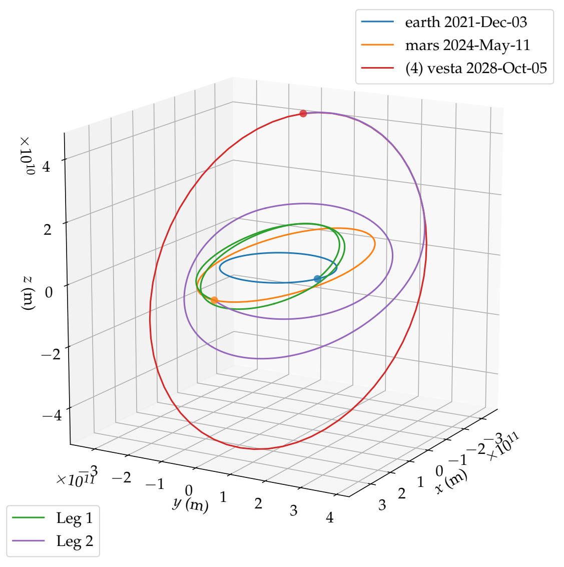
c)

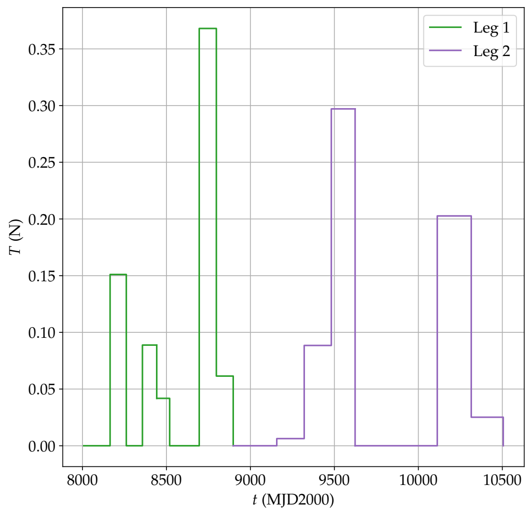
Figure 3.3: Convergence rate of all the points computed with Sims-Flanagan (SF) as a function of the total number of points computed during the optimization for: a) EVVMMM case, b) EEJ case, and c) Dawn case.

work is not whether it can produce better results than other tools available. Instead, the objective was to determine whether the inclusion of the surrogate improves the results with respect to the baseline optimization without surrogate. This approach was already discussed in detail in section Results of the thesis paper manuscript, page 20.

3.3.4. Example Trajectories

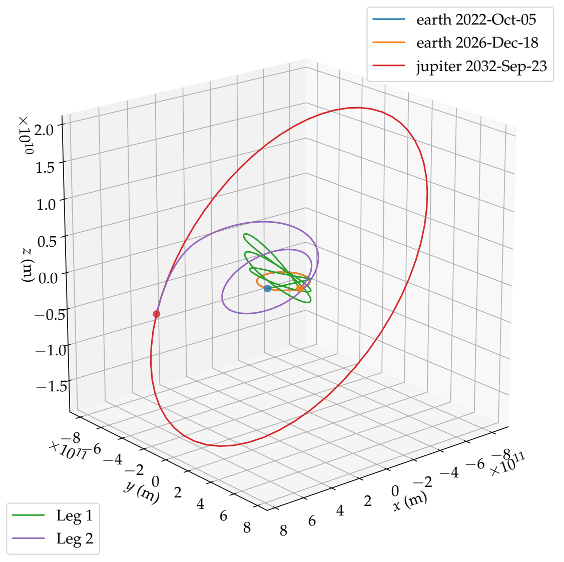
The best transfers found in the tests described in section Results of the thesis paper manuscript, page 20 are included here as an example of a complete optimization. The result for the Dawn case is shown in Figure 3.4, the result for the EEJ case is

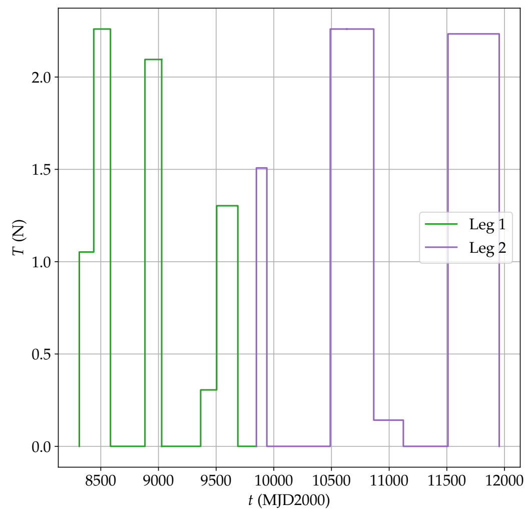




a) b)

Figure 3.4: Overall best transfer found for the Dawn case: a) trajectory, b) thrust as a function of time.

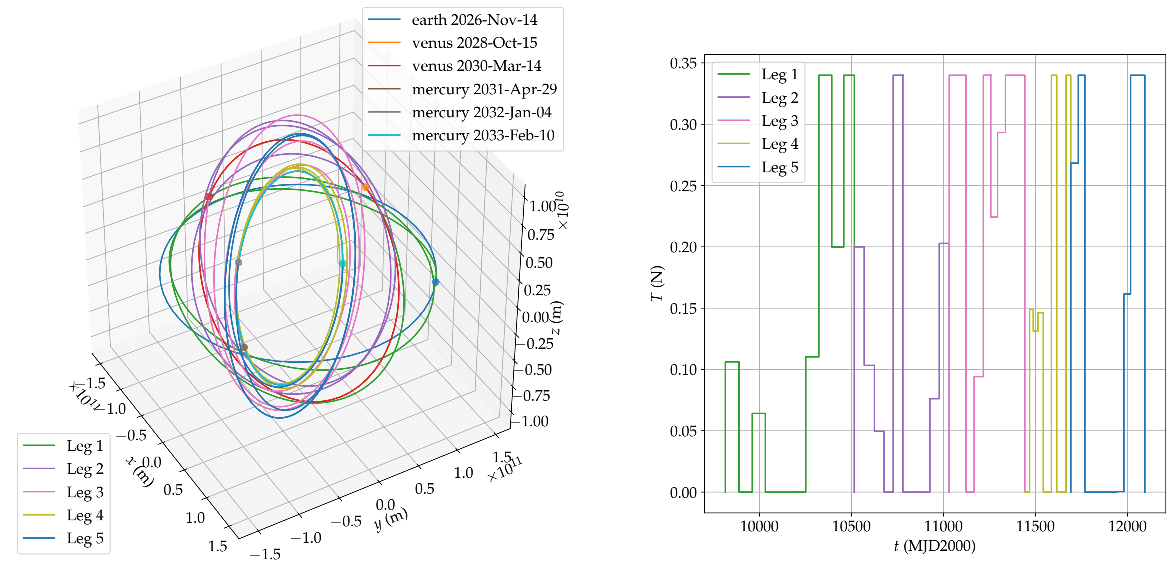




a) b)

Figure 3.5: Overall best transfer found for the EEJ case: a) trajectory, b) thrust as a function of time.

shown in Figure 3.5, and the result for the EVVMMM case is shown in Figure 3.6. Two characteristics are common to the transfers obtained in the three cases: the change in velocity due to the gravity assists can be clearly appreciated in the trajectory plots and the control is very close to bang-bang control. These were the results expected for the optimal trajectories, so the pDE algorithm seemed to work as expected.



a) b)

Figure 3.6: Overall best transfer found for the EVVMMM case: a) trajectory, b) thrust as a function of time.

3.4. Local Optimization

A certain transcription method was necessary to define the leg optimization problem in a way that could be implemented computationally. There are two main options in the literature [61]: indirect transcription, and direct transcription. In this work, a direct transcription method with a shape-based initial guess was identified as the best option, as explained in section Leg Computation of the thesis paper manuscript, page 11. Chapter 4 presents a detailed description of the shape-based method used to obtain the initial guess and chapter 5 of the direct method selected. In this section, the reasons for not considering the indirect methods are explained.

First, indirect methods make use of the Pontryagin's Maximum Principle (PMP) [62] to determine the necessary conditions for local minima. However, these conditions are not sufficient for a global minimum, so the solution is not necessary unique for a set of contour conditions [63, 64]. In general, the final solution achieved depends on the initial guess [44, 62]. Usually, several initial guesses need to be tried as the one that works for a particular leg might not be valid for a different one [65]. This introduces discontinuities in the final solution function that presumably increase the value of σ_s, as described in section Leg Computation of the thesis paper manuscript, page 11. The accuracy of the surrogate was expected to be too low to be used effectively if this was the case.

Second, indirect methods are typically difficult to automate [61] as the running time of a single optimization is usually very long. The method was implemented for this work, but a test in an example Earth-Mars leg required ~45 min to obtain a single valid result. This was an issue as enough data to train the surrogate could not be generated in a reasonable amount of time. Moreover, the time required to perform the experiments presented in this thesis would be in the order of several months for every test case.

At the beginning of the thesis, it was proposed to obtain a SF solution and an indirect method solution for some test transfers. Solutions obtained with an indirect method were expected to have lower cost than the ones obtained with the SF method. The difference between the pairs of solutions could be used as an estimate for the acceptable τ values. However, the time required to obtain values for a representative number of transfers would be too long due to the high computation time and the human intervention required by the indirect method.

3.5. Use of Machine Learning Surrogate

The problem described in section Optimization Problem Formulation of the thesis paper manuscript, page 8 has three different cost functions that can be substituted by the surrogate: $J^*(f(\boldsymbol{x}))$, $J^*(\boldsymbol{l}_T)$, or $\hat{J}^*(\boldsymbol{l})$. The function selected to substitute was $\hat{J}^*(\boldsymbol{l})$ for three reasons: the dimensionality of the input space is lower, the function shape is expected to be simpler, and the number of training points generated is higher during the same computation time.

First, the input dimensionality is determined by the number of elements in either $\boldsymbol{x}$, $\boldsymbol{l}_T$ or $\boldsymbol{l}$, depending on the cost function used. When $\boldsymbol{l}$ is the input, the total input dimensionality is always $p = 14$. When $\boldsymbol{l}_T$ is the input, the dimensionality is $p = 14m$, which means $p \geq 28$ assuming at least one GA is always performed. Most ML methods decrease their performance and require more training data when the input dimensionality is increased [66], due to a phenomenon known as the *curse of dimensionality* [67]. A possible alternative is to chose $\boldsymbol{x}$ as the input vector. In this case, the dimensionality is $p = 6l + 2 \geq 14$. The input dimensionality is the same if only one GA is performed, and larger if more GAs are performed.

Second, $J^*(\boldsymbol{l}_T)$ is a combination of all $\hat{J}^*_L(\boldsymbol{l}^{(i)})$ of the transfer, so the surrogate needs to predict simultaneously all of them. It is assumed this prediction is more complex than prediction individual $\hat{J}^*(\boldsymbol{l})$ values. Moreover, if $J_T(f(\boldsymbol{x}))$ is approximated instead, the surrogate needs to learn the additional transformation $\boldsymbol{l}_T = f(\boldsymbol{x})$ in addition to $J^*(\boldsymbol{l}_T)$. This is also expected to decrease the accuracy that can be achieved with the surrogate.

Third, the use of $\hat{J}^*(\boldsymbol{l})$ as target means that l training points are obtained after each transfer evaluation. However, only one is obtained per transfer evaluation if any of the two options is used. The training data available is one of the main limits to the accuracy achievable by the surrogate. Therefore, the increase in training data is a key advantage of using $\hat{J}^*(\boldsymbol{l})$ as target function to approximate.

4 Shape-Based Method

Shape-based methods have some advantages that make them useful in certain circumstances. Their main advantage is the fast computation when compared with other methods available for LTP trajectories. In this thesis, the shape-based method had two purposes: the computation of initial guesses for SF computation and the generation of a sample dataset for verification of the OS-ELM-MAS model as described in subsection 6.2.4. In this chapter, the details of the implementation of the shape-based method are provided.

4.1. Selection

Different shape-based methods, which use different shapes to represent the trajectories, are available in the literature [1, 29, 36, 68–71]. The shape of the trajectory is defined by a function of a set of parameters, which varies depending on the shape selected. The contour conditions of the leg introduce constraints on the parameter values. If the number of parameters is equal to the number of constraints, the constraints determine all the parameter values. If the number of parameters is smaller than the number of constraints, the not all constraints can be fulfilled. The standard alternative is to match the position constraints and ignore the velocity constraints [29]. Finally, several combinations of parameters can fit the constraints of the number is larger than that of constraints. In that case, the parameters can be optimized to minimize a cost function.

A common approach in the literature is to choose a shape that allows the analytical computation of the parameters that fulfill the contour conditions. However, the Time Of Flight (TOF) cannot be computed analytically in most cases. In those cases, the procedure is to compute the minimum number of parameters to match all the contour conditions but the TOF, and try different values for the remaining parameters until the TOF condition is met. When the number of parameters is larger than the number of constraints, a numerical optimization is performed with the TOF as an equality constraint.

Extensions of the base shape-based methods have been proposed to obtain more optimal solutions: a combination of thrust and coast arcs [30, 70], considering two main bodies in the trajectories [72], using transfers to intermediate orbits [73], and implementing approximate thrust level constraints [74]. These implementations introduce additional parameters, such as the switch points between thrust and coast arcs or the intermediate orbital parameters, which require numerical optimization.

As explained in section Shape-Based Method of the thesis paper manuscript, page 12, three requirements were defined for the shape-based method to be used in this work: (1) it shall produce three-dimensional trajectories, (2) it shall be able to fully match the contour conditions, (3) it shall not require a numerical optimization. Three methods in the literature match these requirements: Inverse Polynomial [68], Spherical [1], and Pseudoequinoctial [1]. In addition, three other methods allow for an arbitrary number of parameters in the shaping function and match the requirements when the number of parameters selected is equal to the number of constraints: Hodographic [71], Pseudoespectral [69], and Finite Fourier Series [75]. The Spherical shape-based method was selected as it reportedly produces the most optimal trajectories in terms of ΔV [75].

4.2. Spherical Shape-Based Theory

The spherical shape-based method was selected for this thesis. The method was developed for a PhD thesis [76], which resulted in a peer-reviewed publication [1]. However, some typos are present on the equations on those references. These typos are corrected in a posterior MSc thesis [5], which also provides a more detailed description of the implementation. In the following lines, the description of the method done in Reference [5] is reproduced. However, most of the intermediate results are not included as they were not considered of interest for this thesis.

The trajectories are described in spherical coordinates and defined as a function of the azimuthal angle ϑ. The shape for the radius and the elevation angle are respectively

$$\begin{aligned} R(\vartheta) &= \frac{1}{a_0 + a_1\vartheta + a_2\vartheta^2 + (a_3 + a_4\vartheta)\cos\vartheta + (a_5 + a_6\vartheta)\sin\vartheta}, \\ \varphi(\vartheta) &= (b_0 + b_1\vartheta)\cos\vartheta + (b_2 + b_3\vartheta)\sin\vartheta, \end{aligned} \tag{4.1}$$

where $a_0, \ldots, a_6$ and $b_0, \ldots, b_3$ are the parameters that describe the trajectory. The time is shaped trough its derivative as

$$T'(\vartheta) = \sqrt{\frac{D(\vartheta)R(\vartheta)}{\mu}}, \tag{4.2}$$

where μ is the gravitational parameter of the central body and $D(\vartheta)$ is the curvature of the trajectory, which can be computed as

$$D = -R'' + 2\frac{R'^2}{R} + R'\varphi'\frac{\varphi'' - \sin\varphi\cos\varphi}{\varphi'^2 + \cos^2\varphi} + R\left(\varphi'^2 + \cos^2\varphi\right).$$

The $\square'$ notation refers to the derivative with respect to ϑ.

The state of the spacecraft is represented in Cartesian coordinates for the computations in this thesis. The transformation from Cartesian coordinates to the spherical ones used in the shaping is

$$r(t) = \sqrt{x^2 + y^2 + z^2}, \qquad \vartheta(t) = \arctan 2\,(y, x), \qquad \varphi(t) = \arcsin\left(\frac{z}{r}\right), \tag{4.3}$$

where x, y and z are the components of the position in Cartesian coordinates, t is the time and

$$\arctan 2(y, x) = \begin{cases} \arctan(y/x) & \text{if } x > 0, \\ \arctan(y/x) + \pi & \text{if } x < 0 \text{ and } y \geq 0, \\ \arctan(y/x) - \pi & \text{if } x < 0 \text{ and } y < 0, \\ \pi/2 & \text{if } x = 0 \text{ and } y > 0, \\ -\pi/2 & \text{if } x = 0 \text{ and } y < 0, \\ \text{undefined} & \text{if } x = 0 \text{ and } y = 0. \end{cases}$$

The velocity vector in spherical coordinates is defined as

$$\boldsymbol{v} = \dot{\boldsymbol{r}} = \begin{bmatrix} v_r \\ v_\vartheta \\ v_\varphi \end{bmatrix} = \begin{bmatrix} \dot{r} \\ r\dot{\vartheta}\cos\varphi \\ r\dot{\varphi} \end{bmatrix}$$

where the time derivatives of the spherical components can be computed from Equation (4.3) given $\dot{x}$, $\dot{y}$ and $\dot{z}$. The derivative of the position vector with respect to ϑ is relevant for the computations and its expression is

$$\boldsymbol{r}' = \frac{dt}{d\vartheta}\dot{\boldsymbol{r}} = \frac{\boldsymbol{v}}{\dot{\vartheta}}.$$

The *function vector* is defined for the radius as

$$\boldsymbol{R}(\vartheta) = \begin{bmatrix} 1 \\ \vartheta \\ \vartheta^2 \\ \cos\vartheta \\ \vartheta\cos\vartheta \\ \sin\vartheta \\ \vartheta\sin\vartheta \end{bmatrix},$$

and for the elevation angle as

$$\boldsymbol{\varphi}(\vartheta) = \begin{bmatrix} \cos\vartheta \\ \vartheta\cos\vartheta \\ \sin\vartheta \\ \vartheta\sin\vartheta \end{bmatrix}.$$

The notation $R_i(\vartheta)$ and $\varphi_i(\vartheta)$ is used to refer to the i-th element of $\boldsymbol{R}$ and $\boldsymbol{\varphi}$ respectively. Two auxiliary functions are defined to simplify the notation:

$$\alpha(t) = -\frac{r'\varphi'}{\varphi'^2 + \cos^2\varphi}, \quad C(t) = -\frac{\mu t'}{r^2} + 2\frac{r'^2}{r} + r\left(\varphi'^2 + \cos^2\varphi\right) - r'\varphi'\frac{\sin\varphi\cos\varphi}{\varphi'^2 + \cos^2\varphi},$$

where

$$r' = \frac{\dot{r}}{\dot{\vartheta}}, \qquad \varphi' = \frac{\dot{\varphi}}{\dot{\vartheta}}, \qquad t' = \frac{1}{\dot{\vartheta}}.$$

This formulation has 11 free parameters, which allows the matching of initial and final positions and velocities, as well as the TOF. However, the computation of the TOF can only be done by integrating Equation (4.2), but it does not have an analytical expression. The procedure to compute the parameters is to select a fixed a_2 and compute analytically the parameters that match all the contour conditions but the TOF. Then, the TOF is computed by integrating T'. If it does not match the desired value, a new a_2 selected and the process repeated. The computation of all the other parameters for a a_2 fixed is done by solving the following linear system:

$$\boldsymbol{A}\begin{bmatrix} a_0 & a_1 & a_3 & a_4 & a_5 & a_6 & b_0 & b_1 & b_2 & b_4 \end{bmatrix}^T = \boldsymbol{b} - \boldsymbol{a}_{a_2}, \tag{4.4}$$

where

$$\boldsymbol{A} = \begin{bmatrix}
R_0(\vartheta_i) & R_1(\vartheta_i) & R_3(\vartheta_i) & \cdots & R_6(\vartheta_i) & 0 & \cdots & 0 \\
R_0(\vartheta_f) & R_1(\vartheta_f) & R_3(\vartheta_f) & \cdots & R_6(\vartheta_f) & 0 & \cdots & 0 \\
R'_0(\vartheta_i) & R'_1(\vartheta_i) & R'_3(\vartheta_i) & \cdots & R'_6(\vartheta_i) & 0 & \cdots & 0 \\
R'_0(\vartheta_f) & R'_1(\vartheta_f) & R'_3(\vartheta_f) & \cdots & R'_6(\vartheta_f) & 0 & \cdots & 0 \\
-r_i^2 R''_0(\vartheta_i) & -r_i^2 R''_1(\vartheta_i) & -r_i^2 R''_3(\vartheta_i) & \cdots & -r_i^2 R''_6(\vartheta_i) & \alpha_i \varphi''_0(\vartheta_i) & \cdots & \alpha_i \varphi''_3(\vartheta_i) \\
-r_f^2 R''_0(\vartheta_f) & -r_f^2 R''_1(\vartheta_f) & -r_f^2 R''_3(\vartheta_f) & \cdots & -r_f^2 R''_6(\vartheta_f) & \alpha_f \varphi''_0(\vartheta_f) & \cdots & \alpha_f \varphi''_3(\vartheta_f) \\
0 & 0 & 0 & \cdots & 0 & \varphi_0(\vartheta_i) & \cdots & \varphi_3(\vartheta_i) \\
0 & 0 & 0 & \cdots & 0 & \varphi_0(\vartheta_f) & \cdots & \varphi_3(\vartheta_f) \\
0 & 0 & 0 & \cdots & 0 & \varphi'_0(\vartheta_i) & \cdots & \varphi'_3(\vartheta_i) \\
0 & 0 & 0 & \cdots & 0 & \varphi'_0(\vartheta_f) & \cdots & \varphi'_3(\vartheta_f)
\end{bmatrix},$$

$$\boldsymbol{b} = \begin{bmatrix} 1/r_i \\ 1/r_f \\ -r_{r_i}/r_i'^2 \\ -r_{r_f}/r_f'^2 \\ C_i - 2r_{r_i}'^2/r_i \\ C_f - 2r_{r_f}'^2/r_f \\ \varphi_i \\ \varphi_f \\ r'_{\varphi_i}/r_i \\ r'_{\varphi_f}/r_f \end{bmatrix}, \qquad \boldsymbol{a}_{a_2} = \begin{bmatrix} a_2 R_2(\vartheta_i) \\ a_2 R_2(\vartheta_f) \\ a_2 R'_2(\vartheta_i) \\ a_2 R'_2(\vartheta_f) \\ -a_2 r_i R''_2(\vartheta_i) \\ -a_2 r_f R''_2(\vartheta_f) \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

The subindices i and f represent the values at the initial and final positions respectively. The final ϑ value is defined as

$$\vartheta_f = \begin{cases} \operatorname{arctan2}(y_f, x_f) + 2\pi n_{rev} & \text{if } \operatorname{arctan2}(y_f, x_f) > \operatorname{arctan2}(y_i, x_i), \\ \operatorname{arctan2}(y_f, x_f) + 2\pi(n_{rev} + 1) & \text{if } \operatorname{arctan2}(y_f, x_f) \leq \operatorname{arctan2}(y_i, x_i), \end{cases}$$

where n_{rev} is the number of revolutions of the leg.

Once the parameters are computed by solving Equation (4.4), the total time to complete the leg is computed as

$$t_T = \int_{\vartheta_i}^{\vartheta_f} T' d\vartheta. \tag{4.5}$$

For fixed number of revolutions and initial and final conditions, this value depends only on a_2. The correct value of a_2 is obtained by solving

$$t_T\,(a_2) - TOF = 0, \tag{4.6}$$

where TOF is the time of flight of the leg. The method used to solved this equation is detailed in subsection 4.3.2.

Finally, the shaped trajectory is represented again in Cartesian coordinates. The position is transformed as

$$\boldsymbol{r} = \begin{bmatrix} \cos\vartheta\cos\varphi \\ \sin\vartheta\cos\varphi \\ \sin\varphi \end{bmatrix},$$

the velocity as

$$\boldsymbol{v} = \frac{d\boldsymbol{r}}{dt} = \frac{d\boldsymbol{r}}{d\vartheta}\frac{d\vartheta}{dt} = \frac{1}{T'}\boldsymbol{r}',$$

and the acceleration as

$$\boldsymbol{a} = \frac{d\boldsymbol{v}}{dt} = \frac{d\boldsymbol{v}}{d\vartheta}\frac{d\vartheta}{dt} = \left(\frac{d^2\boldsymbol{r}}{d\vartheta^2}\frac{d\vartheta}{dt} + \right) = \frac{1}{T'^2}\boldsymbol{r}'' - \frac{T''}{T'^3}\boldsymbol{r}'.$$

However, the most important result are the thrust and mass as a function of time as they is used to transform the shape-based trajectory into a SF one. The acceleration that needs to by provided by the propulsion system can be computed using the dynamics presented in section Spacecraft Dynamics of the thesis paper manuscript, page 7 as

$$\boldsymbol{u} = \frac{\boldsymbol{T}'}{m} = \boldsymbol{a} + \frac{\mu}{r^3}\boldsymbol{r},$$

where $\boldsymbol{T}'$ is the thrust required to follow the trajectory (in this case, $\square'$ does not to refer to the derivative with respect to ϑ). The ΔV of the trajectory as a function of time is computed as

$$\Delta V(t) = \int_{t_i}^{t} |\boldsymbol{u}| dt = \int_{\vartheta_i}^{\vartheta(t)} |\boldsymbol{u}| T' d\vartheta. \tag{4.7}$$

The change of mass as a function of time can be computed as

$$m(t) = m_i e^{-\frac{\Delta V(t)}{I_{sp} g_0}}.$$

Finally, the thrust required to follow the trajectory is

$$\boldsymbol{T}'(t) = m\boldsymbol{u}.$$

4.3. Implementation

In the literature, the spherical shape-based method was implemented both in Matlab [1, 76] and in C++ [5]. In this thesis, Python was selected as the most convenient language as it allowed to easily merge the several tools required into a complete working algorithm. However, this meant that the implementation of the spherical method had to be done from scratch.

From the theory, it can be concluded that two key decisions needed to be made when implementing this method. The first one was the integration method used in Equations (4.5) and (4.7), which is discussed in subsection 4.3.1. The second one was the solver used to solve Equation (4.6), which is discussed in subsection 4.3.2.

4.3.1. Integrator Selection

Only integration methods implemented in the SciPy library[1] were considered. This library is widely used for scientific computation in Python and is regularly maintained, so it is well documented and reliable. There are two groups of integration methods that can be used in this problem. The first one is comprised by the methods that sample the function and use some expression to compute the area in the integration interval. The second one are the Ordinary Differential Equation (ODE) solvers. The ODE solvers are chosen as the implementation allows the creation of an interpolating function at the same time that the integration is performed. This interpolating function is very useful for computing the values of the solution at points different from the final one. That was required for the transformation of the shape-based solution into the initial guess for Sims-Flanagan as described in subsection 5.2.1.

Within this last group, several integration methods were available. However, there was a practical issue that limits the options that could be used. The issue was caused by the fact that all integration methods available are variable step size methods, which adapt the step size to match some predefined tolerances. However, Equation (4.1) shows that it is possible to obtain a value such that the denominator of the r function becomes zero, so $R \to \infty$ for certain combinations of parameters and ϑ. When this value is introduced in Equation (4.2), the result is $T' \to \infty$. If this situation was reached when integrating Equation (4.5), changes in the value of t_T would be very large for small changes in ϑ. Therefore, variable step size algorithms would need to select very small step sizes in order to keep the value of t_T within tolerances.

[1] https://docs.scipy.org/doc/ (accessed May 13, 2020).

The legs in which $T' \to \infty$ are obviously not valid as the TOF required is infinite. However, there is no strategy described in the literature neither found during the thesis that allows the prediction of this situation other than computing the leg. When attempted to compute with most of the ODE solvers available, extremely small step sizes are selected which makes the integration process very long. Moreover, these solvers have no way to automatically stop if the integration time is over a certain threshold or the step size is below a minimum, and they might take days to complete a single integration. This is an issue with the implementation of the solvers in Scipy and no workarounds could be found during the thesis.

The exception is the Livermore Solver for ODEs Automatic (LSODA) [77], which allows the definition of a minimum step size. With this solver, step size never decreased below the threshold even when tolerances were not met. This introduces certain error in the computation when T' takes very large values, but these trajectories are of no interest as the time of flight and radius were much larger than in any physically possible trajectory.

Table 4.1: Transfers used for the selection of parameters, verification, and validation of the spherical shape-based method. Initial body is Earth on all cases.

Case	Source	Final body	Starting date (MJD2000)	Final date (MJD2000)	n_{rev}
E-M	Figure 12 in [1]	Mars	7304	7804*	1*
E-ML	Figure 14 in [1]	1989ML	7529	8529*	2*
E-9P	Figure 16 in [1]	9P/Tempel	465	1565*	0*
E-N	Figure 18 in [1]	Neptune	7409	20409*	0*
B-M	Table 11.3 in [5]	Mars	8174	8754	1
B-ML	Table 11.11 in [5]	1989ML	7799	8399	1
B-9P	Table 11.21 in [5]	9P/Tempel	4124	5584	0
B-N1†	Table 11.17 in [5]	Neptune	7754	21254	0
B-N2	Table 11.18 in [5]	Neptune	10049	50549	10

* Approximate value obtained from source figure.
† Initial velocity with respect to Earth of 3 $\mathrm{km\,s^{-1}}$ tangential to its orbit.

The method selected required the definition of a relative tolerance δ_{rel}, an absolute tolerance δ_{abs}, and a minimum step size $\Delta\vartheta_{min}$. To reduce the number of parameters to consider, $\Delta\vartheta_{min} = \delta_{abs}$ was selected for all computations. The effect of these tolerances was tested in the test cases E-M, E-ML, E-9P and E-N defined in Table 4.1.

The solution for each combination of tolerances was obtained following the procedure described in subsection 4.3.2 to obtain the a_2 for the correct TOF. The value of different parameters along the trajectory was compared with a reference trajectory, which is computed with $\delta_{rel} = \Delta\vartheta_{min} = \delta_{abs} = 10^{-12}$. The differences were computed at all the ϑ values corresponding to the integration points of the reference trajectory. The following magnitudes were selected as representatives of the error for

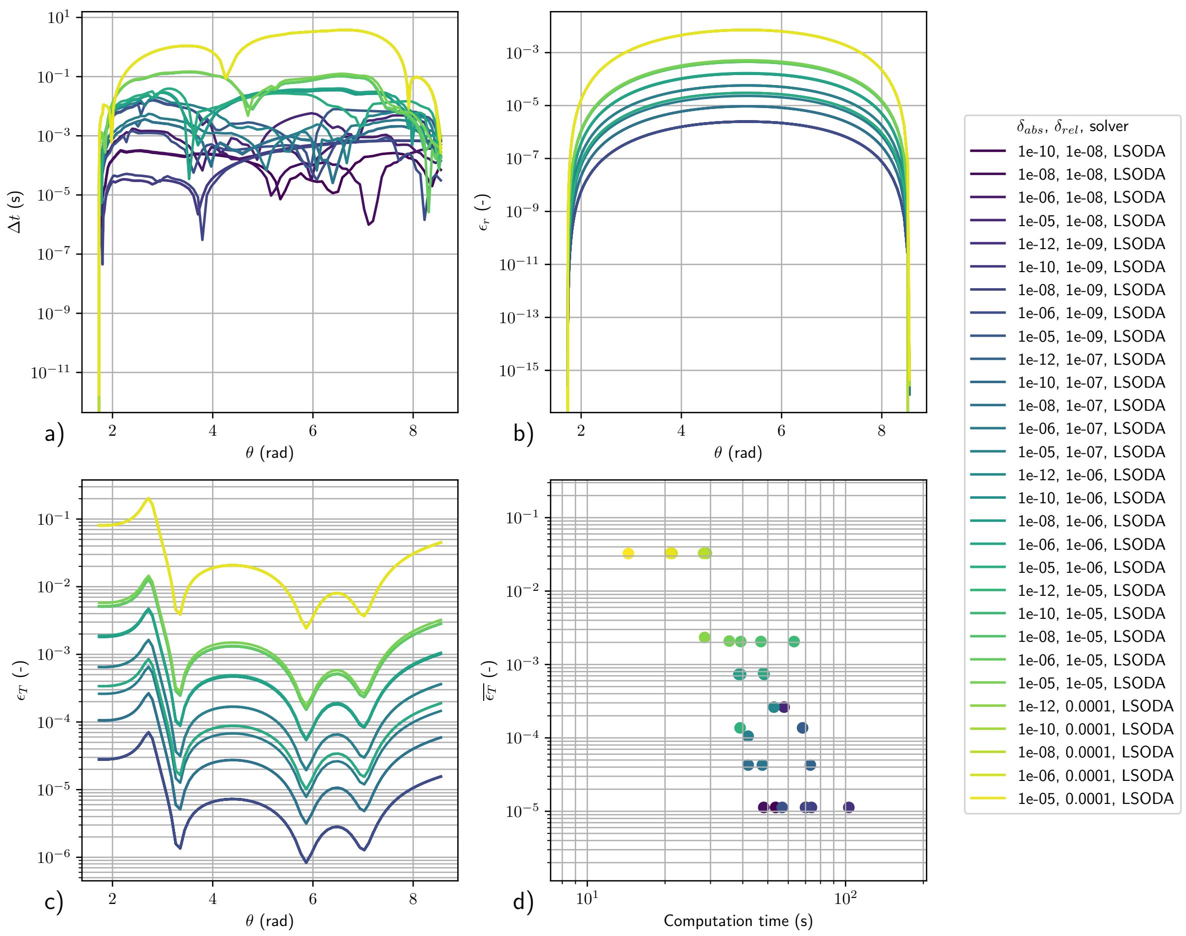


Figure 4.1: Error with respect to the reference trajectory when using different tolerances in the E-M case: a) in time as a function of ϑ, b) in relative position as a function of ϑ, c) in relative thrust as a function of ϑ, and d) in average relative thrust as a function of computation time.

each combination of tolerances:

$$\Delta t = |t - t_{ref}|, \qquad \epsilon_r = \frac{|\boldsymbol{r} - \boldsymbol{r}_{ref}|}{|\boldsymbol{r}_{ref}|}, \qquad \epsilon_T = \frac{|\boldsymbol{T}' - \boldsymbol{T}'_{ref}|}{|\boldsymbol{T}'_{ref}|}, \qquad \overline{\epsilon_T} = \frac{\int_{\vartheta_i}^{\vartheta_f} \epsilon_T d\vartheta}{\vartheta_f - \vartheta_i},$$

where the subindex ref indicates the reference case and the integral was computed using the trapezoid rule from the sampled points. The results are presented in Figures 4.1, 4.2, 4.3, and 4.4.

The value of $\overline{\epsilon_T}$ was considered the most important result as $\boldsymbol{T}'$ is the main result necessary to transform the shape-based trajectory into a Sims-Flanagan trajectory using the method described in subsection 5.2.1. The results show that there was a trade-off between accuracy and computation speed. Two general conclusions could be extracted from the results. The first one is that using values $\delta_{abs} < 10^{-6}$ increased the computation time without decreasing the error, so $\delta_{abs} = 10^{-6}$ was the most adequate value. The second one was that using values of $\delta_{rel} > 10^{-7}$ increased significantly the error but the gains on computation speed were very small, and the only significant increase in computation speed was achieved when $\overline{\epsilon_T} \approx 10^{-1}$, which

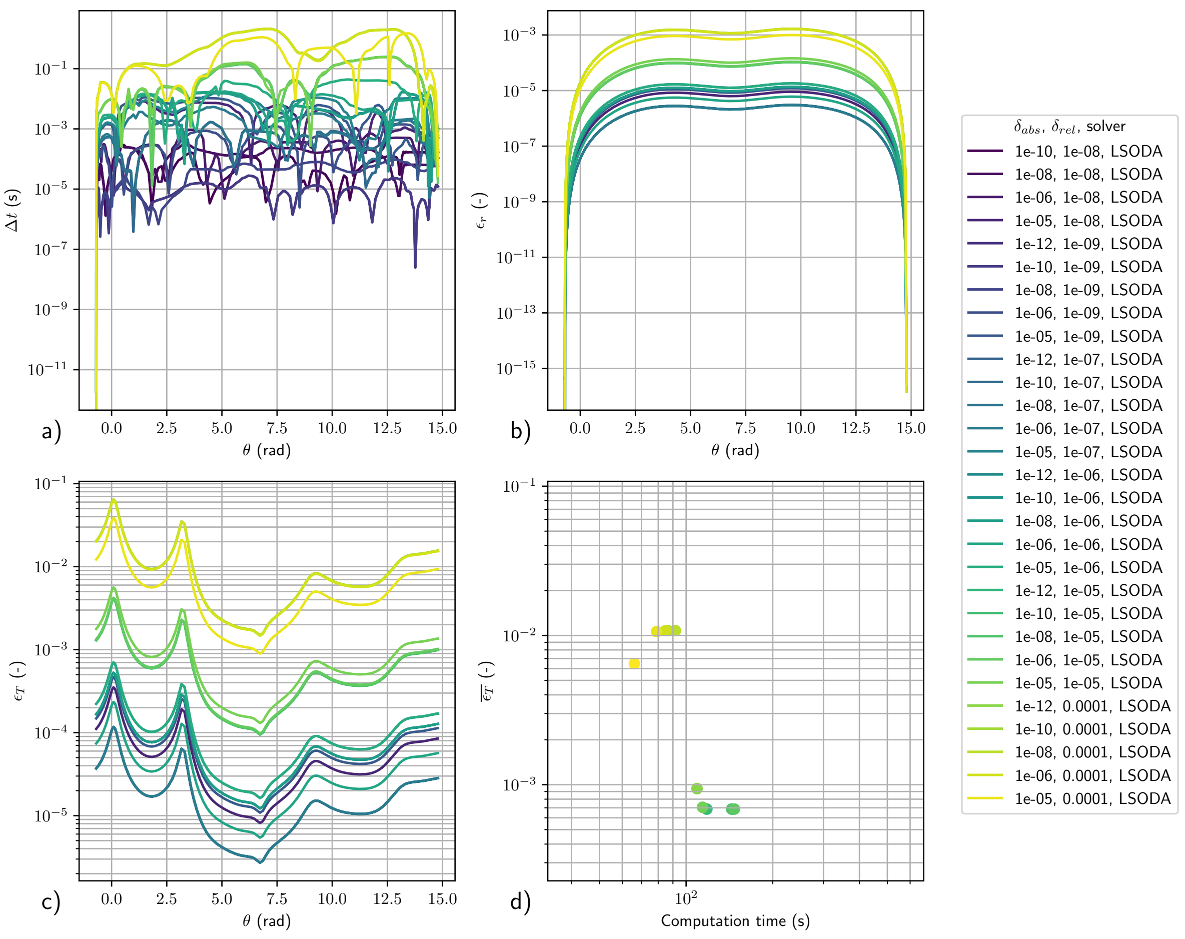


Figure 4.2: Error with respect to the reference trajectory when using different tolerances in the E-ML case: a) in time as a function of ϑ, b) in relative position as a function of ϑ, c) in relative thrust as a function of ϑ, and d) in average relative thrust as a function of computation time.

was considered too large. Therefore, the final tolerances selected were $\delta_{abs} = 10^{-6}$ and $\delta_{rel} = 10^{-7}$. The error was within bounds considered safe and the computation time was small compared with the time required to compute a SF transfer, as described in subsection 5.2.3.

4.3.2. Solver

The decision regarding the method used to solve Equation (4.6) was not trivial. Figure 4.5 illustrates the reason. The results in the figure were computed for an Earth-Mars transfer with departure at epoch 7305 MJD2000, different arrival dates and $n = 0$ i.e direct transfers. The TOF corresponding to a range of a_2 values was computed. It was found that the TOF seemed to converge asymptotically to a minimum TOF value as $a_2 \to -\infty$. As a_2 increased, there was a sudden increase of the TOF value until the point in which the trajectory reached $R \to \infty$, and no more solutions were available after that point. The oscillating solutions shown past that point correspond to numerical instabilities due to the integration of trajectories in which $R \to \infty$ at least in one point.

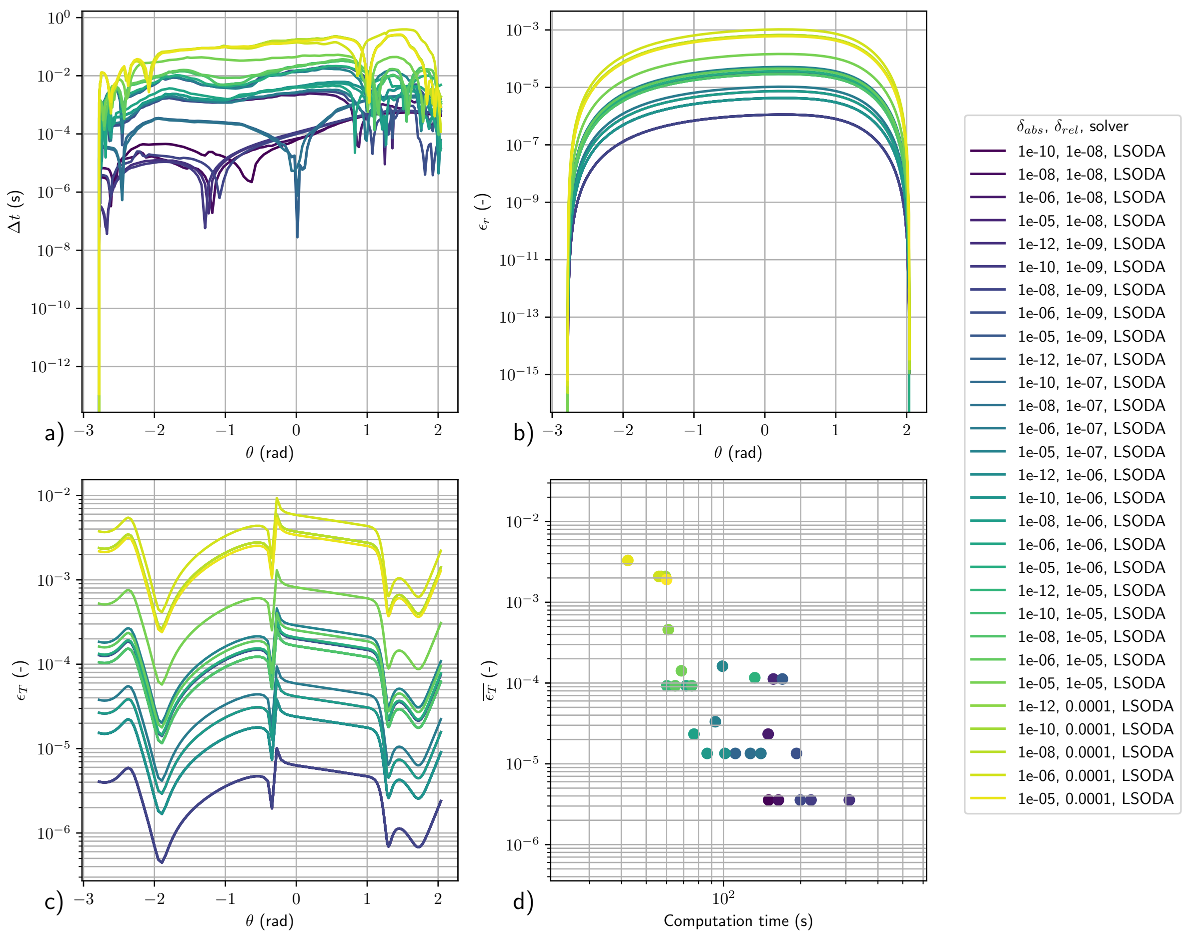


Figure 4.3: Error with respect to the reference trajectory when using different tolerances in the E-9P case: a) in time as a function of ϑ, b) in relative position as a function of ϑ, c) in relative thrust as a function of ϑ, and d) in average relative thrust as a function of computation time.

Most of the solutions lay in the relatively narrow range in which the TOF started to diverge from the asymptotic value and the discontinuity $R \to \infty$ was reached. Moreover, no way was found to predict the divergence point without performing a search like the one in the plot, neither during this thesis or in the literature. Moreover, the point at which the discontinuity happened varies in several orders of magnitude. For example, the line corresponding to an arrival on 7805 MJD2000 seems to be completely flat. However, it finally starts to increase and finally reaches the discontinuity for $a_2 \approx 7 \cdot 10^5$. Finally, starting at large negative values of a_2 and slowly increasing its value was also tried, but in some cases the function reproduced a mirrored version of shape presented, with the point of divergence in negative a_2 values.

To obtain a solution, several solvers were tested. The first one tried was the Newton method used in the literature [1, 5]. However, it was found that initial guess needed to be very close to the solution or a step would lie on the interval with discontinuities. Solutions were only achieved by manually setting the initial guess very close to the initial value. This situation was confirmed independently by a fellow student working on the implementation of the same spherical shape-based method

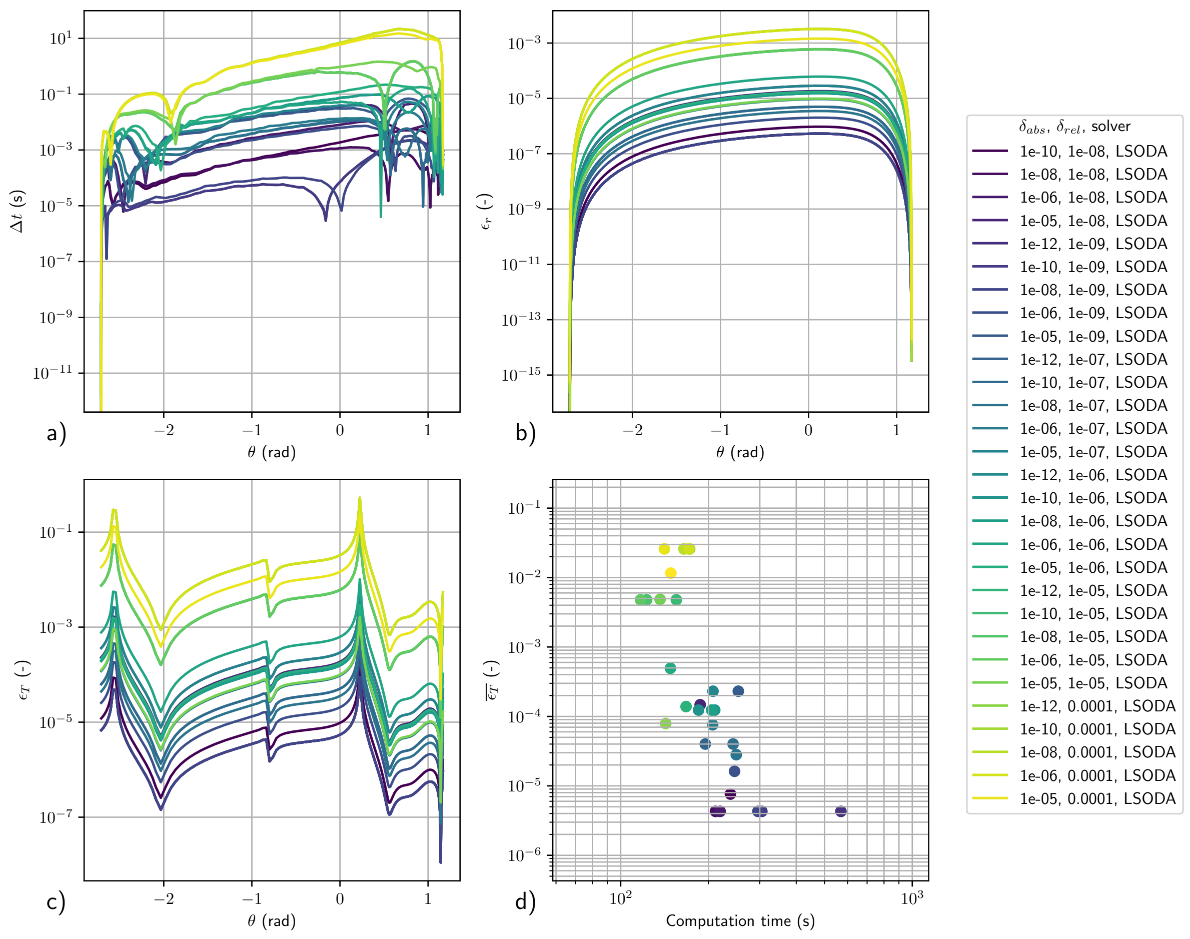


Figure 4.4: Error with respect to the reference trajectory when using different tolerances in the E-N case: a) in time as a function of ϑ, b) in relative position as a function of ϑ, c) in relative thrust as a function of ϑ, and d) in average relative thrust as a function of computation time.

in C++[2]. Instead, two set of methods were tested: gradient-based local optimizers from Pygmo[3] and bisection-based root finders from SciPy[4]. The summary of all solvers considered is presented in Table 4.2.

The algorithms were first tested on the test cases presented on Table 4.1. Algorithms that failed to reach a solution in at least one of those cases were discarded. The rest of algorithms were tested on the fast version of the grid searches described on Table 4.3. These grid searches are done using the parameters of the tests that resulted in the contour plots in Reference [1], but larger step sizes were used so computations could be completed in reasonable time. However, it was observed that a wrong solution was reached for individual combinations of departure dates and TOFs. The reason for this was that the solver selected a point in the range where R reaches zero, i.e. the fast oscillations on Figure 4.5. All solvers suffered these problem, but the

[2]M. Fayolle, personal communication.

[3]https://esa.github.io/pygmo2/overview.html#list-of-algorithms (accessed May 13, 2020).

[4]https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.root_scalar.html#scipy.optimize.root_scalar (accessed May 13, 2020).

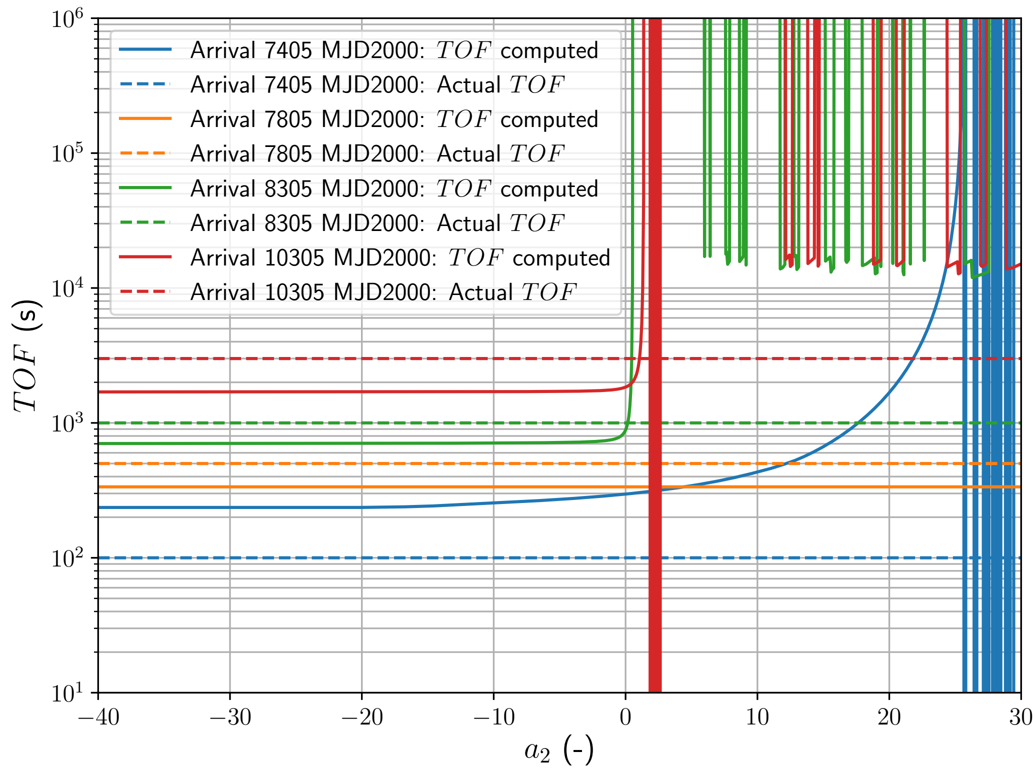


Figure 4.5: TOF as a function of a_2 values for spherical trajectories between Earth and Mars with departure at 7305 MJD2000, $n = 0$ and final state corresponding to Mars at different dates.

combinations at which it happened were different for each solver. In particular, SciPy solvers and Pygmo solvers failed at completely different points. Moreover, changing the initial guess for the optimizers also changed the occurrence of these failures to find the correct solution.

These problem of the solver selecting the wrong solution were solved with two complementary strategies. First, the initial guess was changed if the solution was not valid. Second, two solvers were combined to maximize the chances of at least one of them finding the correct solution. This total procedure might seem cumbersome and time consuming, but would only represent a small part of the total computation time of each leg. As explained in section Leg Computation of the thesis paper manuscript, page 11, a consistent initial guess for the directed method is a key part of the method presented in this work, so this robust method for looking for the correct solution was considered necessary.

The implementation of the changes in initial guess depends on the solver used. Pygmo solvers tended to get stuck when trying to explore the area where solutions were not valid as the fast oscillations in TOF reported by the solver prevented the correct computation of numerical gradients, which were required for all the solvers considered. However, having a wider search range increased the chances of finding

Table 4.2: Different solvers considered for solving Equation (4.6).

SciPy Bisection	Pygmo Unconstrained	Pygmo Constrained
bisect	bobyqa	mma*
brentq	newuoa*	ccsaq*
brenth	newuoa_bound*	slsqp*
ridder	neldermead	
toms748	sbplx	
	lbfgs	
	tnewton_precond_restart	
	tnewton_precond	
	tnewton_restart	
	tnewton	
	var2	
	var1	

* Discarded due to no convergence in preliminary tests

a solution. The implementation of these solvers also required to set maximum and minimum bounds of the search space. The procedure followed with these solvers was to start with large bounds $b_{a_2} = \pm 10^6$, and halve the bounds if no solution was found until $|b_{a_2}| < 10^{-12}$. Two initial guesses were used for each bound configuration, which were $a_2 = \pm\frac{3}{4}|b_{a_2}|$. This procedure proved to be the fastest in the individual test cases so it was the first one tried when looking for a solution.

In the case of the SciPy solvers, the procedure was the inverse. In this case, the solver was able to find values in the region where solutions are not reliable as the solvers did not rely on the gradient, so they were not affected by the fast oscillations. As a result, the search was started with the lowest bounds $b_{a_2} = \pm 10^{-12}$ and then increased. In this case, the solver requires that the value of the function at each bound has an opposite sign. Therefore, bounds were increased alternative in each trial to prevent the situation in which the sign of the solution at both extremes changed sign simultaneously. Moreover, it was found that increasing the bounds too fast caused the solver to skip two consecutive roots of the cost function, which cause the sign of the function to not change. However, increasing too slow made the solving process extremely slow. Therefore, the optimal solution was found to be to do a broad search increasing the bounds fast and then reducing the increase rate between trials if no solution was found. These solvers reported a solution even in most of the cases in which not a valid one could be found. Therefore, the condition $\Delta V < 10^5$ was set for the solution to be consider valid. If none of the solutions fulfilled this condition, the one with the lowest ΔV was considered the correct one.

The full description of the solution procedure can be seen in Algorithm 1. The solver used was observed to not have influence on the success rate but only on the computation speed. The bounds tested and increase rates were selected so all the test cases had a 100% success rate while keeping an optimal computation speed. In

Algorithm 1 Procedure to search for a valid solution of the spherical shape-based method.

```
1:  for all $n_{rev}$ in $R_{n_{rev}}$ do
2:      Set $n_{rev}$ and contour conditions in leg
3:      $b_{a_2} \leftarrow 10^6$
4:      $\mathcal{S} \leftarrow \{\}$
5:      while $b_{a_2} > 10^{-12}$ do
6:          $a_2 \leftarrow$Pygmo solver($-b_{a_2}$, $b_{a_2}$)
7:          if $t_T(a_2) - TOF < 10^{-6}$ then
8:              Add solution to $\mathcal{S}$
9:              Exit loop
10:         else
11:             $b_{a_2} \leftarrow b_{a_2}/2$
12:         end if
13:     end while
14:     if solution not found then
15:         $f_b \leftarrow 8$
16:         $\Delta V_{sol} \leftarrow \infty$
17:         while $\Delta V_{sol} > 10^5\,\mathrm{m\,s^{-1}}$ and $f_b > 0.2$ do
18:             $b^-_{a_2} \leftarrow -10^{-12}$
19:             $b^+_{a_2} \leftarrow 10^{-12}$
20:             $i \leftarrow 0$
21:             while $b^-_{a_2} > -10^{10}$ and $b^+_{a_2} < 10^{10}$ do
22:                 $a_2 \leftarrow$SciPy solver($b^-_{a_2}$, $b^+_{a_2}$)
23:                 if $t_T(a_2) - TOF < 10^{-6}$ then
24:                     Add solution to $\mathcal{S}$
25:                     Exit loop
26:                 else
27:                     $b^-_{a_2} \leftarrow b^-_{a_2} \cdot (1 + f_b \cdot (i \bmod 2))$
28:                     $b^+_{a_2} \leftarrow b^+_{a_2} \cdot (1 + f_b \cdot ((i+1) \bmod 2))$
29:                     $i \leftarrow i + 1$
30:                 end if
31:             end while
32:             Update $\Delta V_{sol}$ with solution $\Delta V$
33:             $f_b \leftarrow f_b/2$
34:         end while
35:     end if
36: end for
37: return solution in $\mathcal{S}$ with lowest $\Delta V$
```

Table 4.3: Grid search parameters used for the selection of parameters, and validation of the spherical shape-based method. Initial body is Earth on all cases.

Code	Final body	Starting date (MJD2000)	Dates step (d)	TOF (d)	TOF step (d)	n_{rev}
C-M	Mars	7305 to 10 227	15* or 900†	500 to 2000	20* or 1200†	1 to 5
C-ML	1989ML	7305 to 10 227	15* or 900†	100 to 1000	20* or 1200†	0 to 2
C-9P	9P/Tempel	0 to 5846	900†	400 to 1500	1200†	1
C-N‡	Neptune	7305 to 9496	900†	11 000 to 30 000	10000†	0

* For full tests.
† For fast tests.
‡ Initial velocity with respect to Earth of $3\,\mathrm{km\,s^{-1}}$ tangential to its orbit.

most of the cases, a solution could be found with only the Pygmo solver and the SciPy solver was not used. However, the use of the SciPy solver was key for the cases in which the Pygmo solver failed.

Table 4.4: Normalized time required to complete the grid search cases for each solvers combination. The selected combination is in bold.

	bisect	brentq	brenth	ridder	toms748
bobyqa	1.9105	1.8989	1.8784	1.8779	1.9195
neldermead	1.8511	1.8593	1.8587	**1.7372**	1.6378
sbplx	2.4472	2.3297	2.4090	2.4152	2.4572
lbfgs	3.8847	4.1515	4.5684	4.0776	3.9182
tnewton_precond_restart	4.4475	4.4257	4.3920	4.5105	4.5728
tnewton_restart	3.7311	3.7381	3.7280	3.7792	3.8043
tnewton_precond	4.5405	4.4571	4.4374	4.4275	4.4803
tnewton	3.6682	3.6589	3.6287	3.6398	3.7162
var2	3.8039	3.8649	3.8789	3.8822	3.9313
var1	3.8880	3.8983	3.8984	3.9260	3.9164
cobyla	8.8846	9.8699	9.8910	9.6734	9.9207

The only variables left to choose were the solvers. All the possible combinations of SciPy and Pygmo solvers were tested on the fast version of the cases in Table 4.3 and the computation time measured. To avoid excessive effect of cases that were longer in general, the time each combination takes to solve a particular case was normalized with the average of all the combinations. Then, all these normalized times were added and that was the final time score considered. The results are shown in Table 4.4. The combination selection was neldermead Pygmo solver and ridder SciPy

Table 4.5: Time required to complete each test case for the selected combination of solvers.

Solver 1	Solver 2	C-M*	C-ML*	C-9P*	C-N*
neldermead	ridder	179.6 s	389.0 s	334.3 s	294.0 s

* Fast test version.

solver. The actual time that the selected combination took to complete each test is shown in Table 4.5. This combination of solvers is the one to be used in the remaining of this work.

4.4. Verification and Validation

The implementation of the spherical shape-based method in the thesis was completely new, so a detailed verification and validation procedure was required. This procedure is described in this section.

4.4.1. Verification

The first part of the verification procedure was a series of unit tests. The procedure was to compute the values of the parameters in the E-Mars case from Table 4.1 and a value $a_2 = -0.03$ with the infinite precision software Mathematica[5]. The values obtained were compared with the ones provided during the computation of the same case with the implementation on this work.

The second part of the verification procedure were sanity checks. This were done during the parameter selection procedure described in Sections 4.3.1 and 4.3.2. The results of the computations are the ones expected and no unexplained phenomena arose so the implementation was considered correct.

4.4.2. Validation

The first validation procedure was to reproduce the results of several trajectories used as examples in the literature where the spherical shape-based method was presented. In Reference [1], there is one example transfer to each body, but parameters such as the final date, number of revolutions, or the thrust required to follow the trajectory are not provided and therefore they have to be estimated from the plots. These are the cases with codes E-{*target initial*} in Table 4.1. The thrust profile obtained and the one shown in the reference is shown in Figures 4.6, 4.7, 4.8, and 4.9. The thrust profile matches closely the reference in the E-M and E-ML cases. However, the thrust seems to be slightly different than in the reference in the E-9P and E-N cases. The reason for this was probably that the final date is not exactly the same as they have been obtained from the plot, which is not a very accurate procedure. Moreover, it is indicated in the reference that the results sometimes did not converge and a similar

[5]`https://www.wolfram.com/mathematica/` (accessed May 13, 2020).

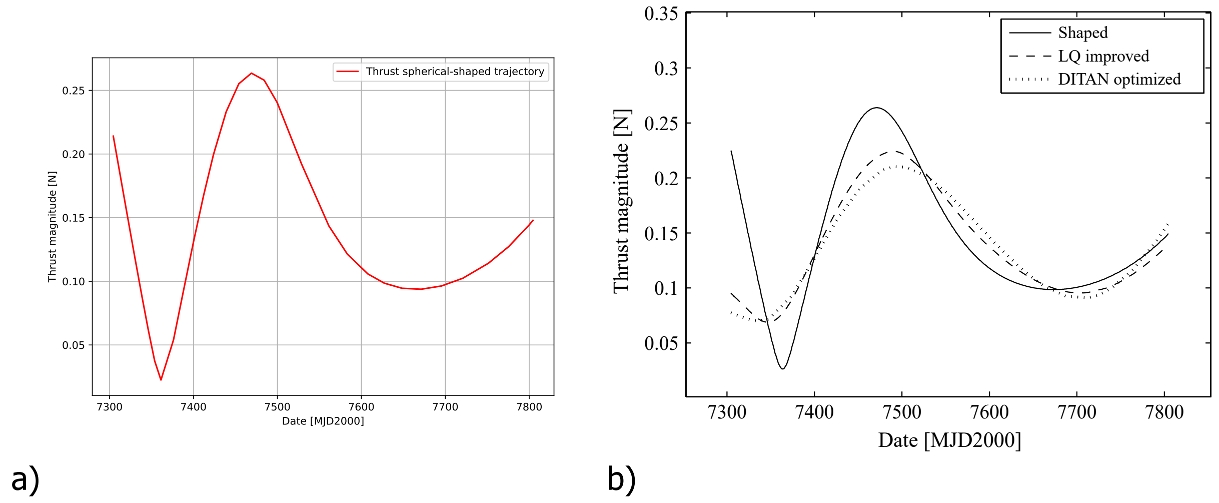


Figure 4.6: Thrust profiles in the E-M case: a) in this work, b) in Reference [1]

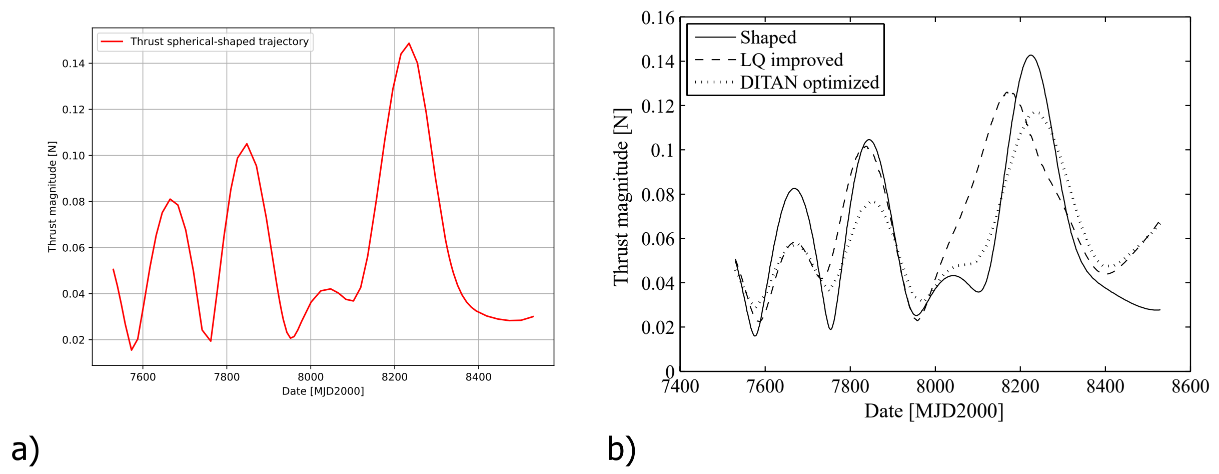


Figure 4.7: Thrust profiles in the E-ML case: a) in this work, b) in Reference [1]

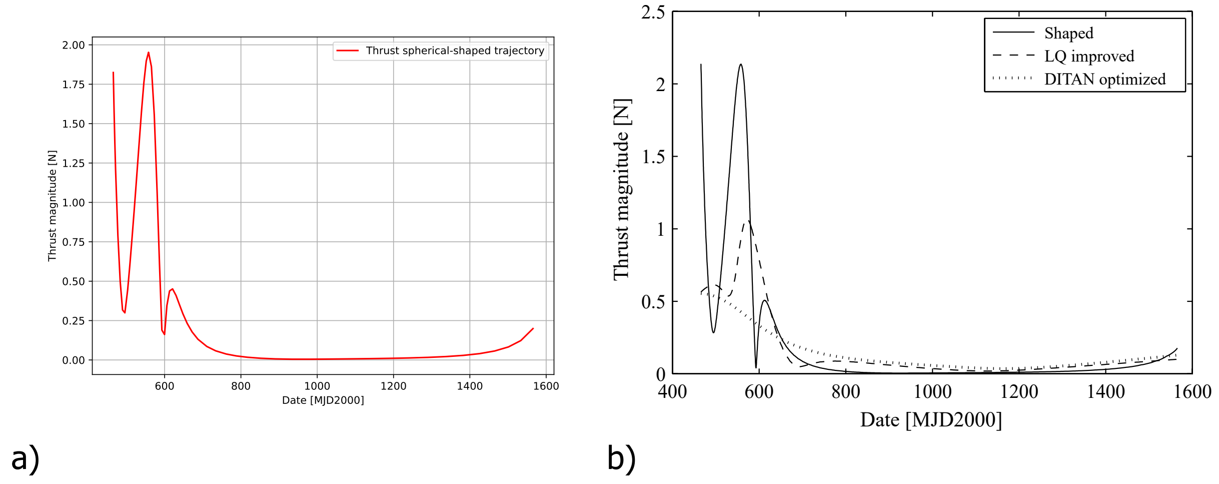


Figure 4.8: Thrust profiles in the E-9P case: a) in this work, b) in Reference [1]

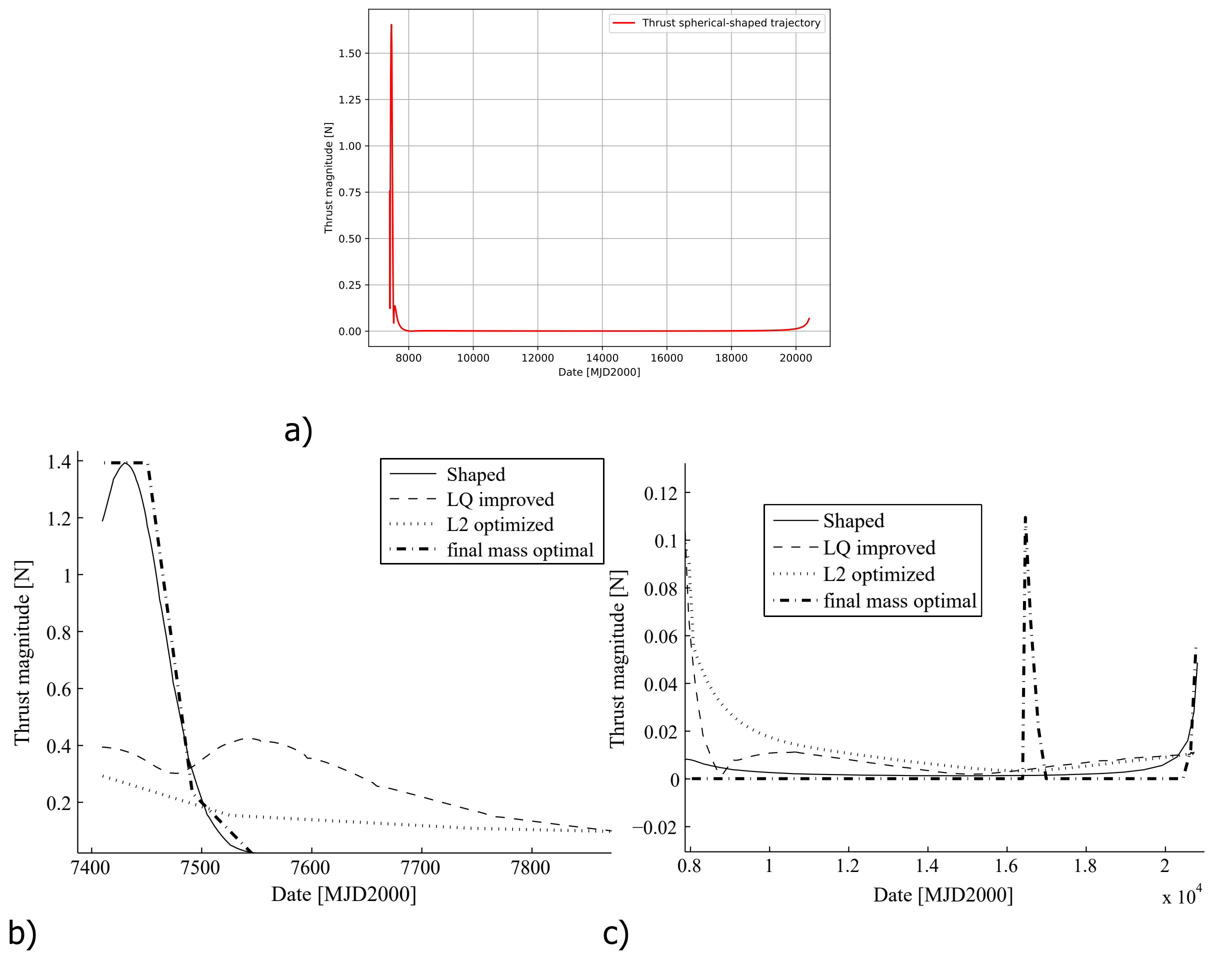


Figure 4.9: Thrust profiles in the E-N case: a) in this work, b) in Reference [1] before 8000 MJD2000, c) in Reference [1] after 8000 MJD2000

solution is used by modifying the time variable so the TOF matches the new value. That could be the case in these examples, but there is no way to know as no indication is given in the reference of how often this option was used.

A second set of individual transfers was available for comparison in Reference [5]. In the reference, the departure and arrival date and the final ΔV required to complete the best trajectory for each grid search are available. These trajectories are the ones with codes B-{*target initial*} in Table 4.1. These trajectories were reproduced in this work and the results obtained compared. The summary of the results can be seen in Table 4.6. It can be observed that the results obtained match quite closely the values in the reference, so the implementation seemed to be correct.

The final validation procedure was the reproduction of the ΔV contour plots presented in Reference [1]. These contour plots are done by selecting the best ΔV for any n_{rev} for each grid point. This contour plots are the result of the grid search with the parameters described in Table 4.3. In this case, the full version of the search was done. This required a long computation time so only the C-M and C-ML cases were

Table 4.6: Comparison of ΔV and a_{max} obtained with spherical shape-based method in Reference [5] and in this work for a selection of cases.

Code	ΔV (km s^{-1})		a_{max} (km s^{-2})	
	Reference [5]	This work	Reference [5]	This work
B-M	5.70	5.698	2.4×10^{-4}	2.3815×10^{-4}
B-ML*	4.53	4.530	1.8×10^{-4}	1.7715×10^{-4}
B-9P	11.51	11.588	$-^{\dagger}$	1.3922×10^{-3}
B-N1	15.48	15.329	$-^{\dagger}$	1.1501×10^{-3}
B-N2	24.16	24.161	$-^{\dagger}$	3.615×10^{-5}

* Apparent error in reference. Final state seems to be target body but at departure time. Results presented were obtained with that configuration.
† Value not provided in reference.

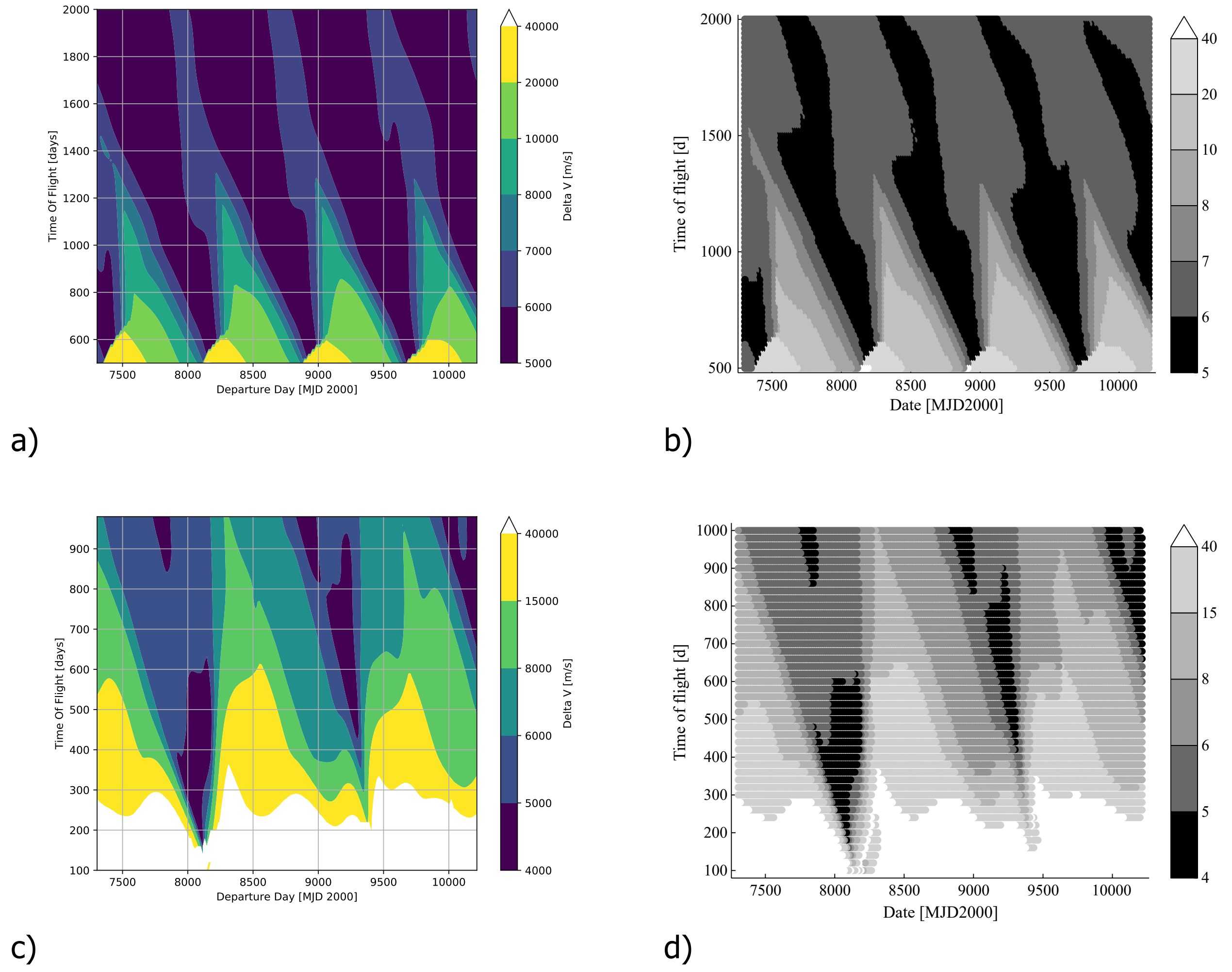


Figure 4.10: Contour plots of best ΔV: a) case C-M in this work, b) case C-M in Reference [1], c) case C-ML in this work, d) case C-ML in Reference [1]

tested. The comparison of the contour plots can be seen in Figure 4.10. However, there seems to be an error in the description of C-ML in Reference [1]. It is indicated that the number of revolutions tested is 1 or 2, but it was not possible to reproduce the contour plot without including 0 revolutions. This was thought to be a typo in the reference, which was supported by the fact that some other typos are present in equations as pointed out in Reference [5]. Apart from that, the contour plots seemed to match quite closely the expected results. The small differences might be caused by the use of the alternative solution method in the reference. However, there is again not information of how often that alternative is used in the reference so it was not possible to know if this is a reasonable assumption.

All the tests performed as part of the validation procedure produced the results expected from the references or have differences that can be traced to different decisions taken during the implementation. Therefore, the validation was considered successful.

5
Direct Transcription Method

As explained in section 3.4, direct methods were considered the best option for the computation of the interplanetary legs. A direct method allows the transformation of the orbit optimization problem into a Non-Linear Programming (NLP) problem. These methods are usually considered to be medium fidelity [61].

5.1. Method Selection

Several direct transcription methods are available in the literature. In References [78, 79], a method based on assigning values of thrust at fixed integration points is proposed. This method uses a mesh refinement strategy to increase the number of integration points and therefore the number of design variables in reasonable computation time. However, the computation is usually slower than with other methods. This method was used for situations in which the acceleration due to perturbations is larger than the thrust acceleration. In Reference [41], the SF method is presented. This method assumes the thrust to be continuous along fixed-time intervals, and the design variables are the thrust values at each interval. The representation of thrust values with series of polynomial [37, 38] and Chebyshev [80] functions was also studied. In these cases, the design variables are the coefficients of the series. Finally, a summary of possible collocation methods was done in Reference [81]. In this case, the design variables are the state variable values at some points in time, and an interpolating polynomial is built using those points.

The method selected for this thesis is the SF method. There are three reasons for this decision. First, it appears to have a more consistent performance in terms of optimality and computation time for general interplanetary optimization problems, although some other algorithms perform better in specific cases [61]. Second, it is used profusely in the literature [2, 3, 33–35, 42, 43, 60, 82, 83], which provides information about proven implementations and test cases for validation. Third, the package Pykep[84] provides a publicly available implementation of the method. This was particularly important in the context of this thesis, as several tools need to be used together and a new implementation of each individual tool would be too time consuming.

As explained in section Direct Method of the thesis paper manuscript, page 13, the implementation of SF used is the ”high fidelity” version presented in Reference [2]. This implementation differs from the original one [41] in two details. First, the thrust is considered a continuous force along a segment instead of an instantaneous impulse at the middle point. Second, the Sundman transformation is used to define the segments as a function of a variable s given by the equation $ds = dt/|\boldsymbol{r}|$.

These changes seek two objectives. First, the continuous thrust approximation has the advantage of providing feasible transfers as there are no discontinuities in the state (position, velocity, and mass) of the spacecraft. This is not the case with the traditional SF implementation. An example of this can be seen in Reference [2], where an optimization with the traditional SF method and with the high fidelity one are done compared. The result of the high fidelity method appears to be less optimal, but the traditional solution would most likely not be feasible. The increase in fidelity with respect to the previous case is achieved at he expense of higher computation times. However, the use of a ML surrogate is more useful when the cost of computing every leg is higher, so it is more interesting to test it in the high-fidelity case. Second, the use of the Sundman transformation allows for a better performance in cases in which the initial and final radius are very different. The transformation has the effect of causing segments to be shorter in time near the central body, which is where the spacecraft is expected to be moving faster.

5.2. Implementation

The Pykep package contains an implementation of the SF method selected. This was the implementation selected for this thesis. Using a finished implementation from a well-maintained package reduced the time required for implementation, verification and validation of the method.

5.2.1. Initial Guess

As explained in section Leg Computation of the thesis paper manuscript, page 11, the process to obtain a SF solution has to be deterministic. The chances of the SF method reaching a valid solution from an arbitrary initial guess are usually low. Instead, two methods are proposed in the literature to find a valid solution. The first one is to do a random search of the input space combined with local optimization in methods such as MBH [42], SA [85] or both [60]. However, these methods were discarded for this work due to the stochastic component of the search. The second method is to use the solution from a simpler method as initial guess. Lambert method is used in [35], and a shape-based method is used in [33, 34].

On this thesis, it was decided to use a shape-based method. It was found that the convergence of the SF method had a strong influence on the performance when the ML surrogate was used. The convergence increased with an initial guess close to a feasible solution, which the spherical shape-based method presented in chapter 4 provides.

Algorithm 2 Procedure to transform the solution obtained with the spherical shape-based method into valid SF inputs.

```
 1: $s(\theta') \leftarrow \int_{\theta_0}^{\theta'} \frac{t'}{r} d\theta$
 2: $\Delta s \leftarrow s(\theta_f)$
 3: $\delta s \leftarrow \Delta s / n_{seg}$
 4: $\boldsymbol{F}_T(\theta) \leftarrow \int_{\theta_0}^{\theta} \boldsymbol{T}' t' d\theta$
 5: $F_T(\theta) \leftarrow \int_{\theta_0}^{\theta} |\boldsymbol{T}'| t' d\theta$
 6: $s_0 \leftarrow 0$
 7: $\theta_i \leftarrow \theta_0$
 8: $\mathcal{T} \leftarrow \{\}$
 9: for all $i \in \{1, \dots, n_{seg}\}$ do
10:     $s_{i+1} \leftarrow s_i + \delta s$
11:     $\theta_{i+1} \leftarrow \arg\min_{\theta} (s(\theta) - s_{i+1})^2$
12:     $\delta t \leftarrow t(\theta_{i+1}) - t(\theta_i)$
13:     $\boldsymbol{T}_i \leftarrow (\boldsymbol{F}_T(\theta_{i+1}) - \boldsymbol{F}_T(\theta_i)) / \delta t$     ▯ The alternative implementation uses this $\boldsymbol{T}_i$ value.
14:     $\boldsymbol{u}_i \leftarrow \boldsymbol{T}_i / |\boldsymbol{T}_i|$
15:     $T_i \leftarrow (F_T(\theta_{i+1}) - F_T(\theta_i)) / \delta t$
16:     $\boldsymbol{T}_i \leftarrow T_i \boldsymbol{u}_i$
17:     $\boldsymbol{T}_i$ is added to $\mathcal{T}$
18: end for
19: return $\mathcal{T}$, $\Delta s$
```

The initial guess needed to be transformed into an input that can be used for the SF method. This was done as described in section Direct Method of the thesis paper manuscript, page 13. A more detailed description of the process can be seen in Algorithm 2. All the integrals were computed with the same integrator and tolerances used for the computation of the spherical shape-based method, which are described in subsection 4.3.1. The optimizer used to find the solution for θ_{i+1} is L-BFGS-B[1] from SciPy package. This solution was fast to obtain, so this optimizer was chosen due to its easy implementation. The inputs for the SF method are the set of thrust vectors $\mathcal{T}$, the total change in the s variable Δs, and the final mass of the trajectory obtained with the spherical shape-based method.

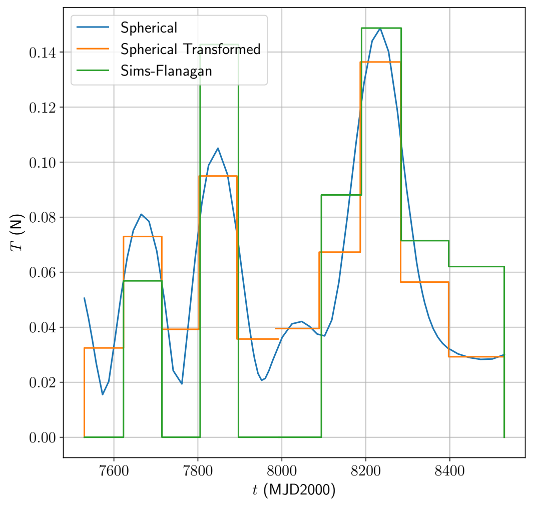
An alternative option was to transform the thrust as the average of its components. However, this caused the the final thrust to cancel out when the segments covered a significant part of a revolution around the Sun. The comparison between the two results for some of the cases described in Table 5.1 was is shown in Figures 5.1, 5.2, and 5.3. The transformation using the alternative method was not capable of properly

[1] https://docs.scipy.org/doc/scipy/reference/optimize.minimize-lbfgsb.html (accessed May 13, 2020).

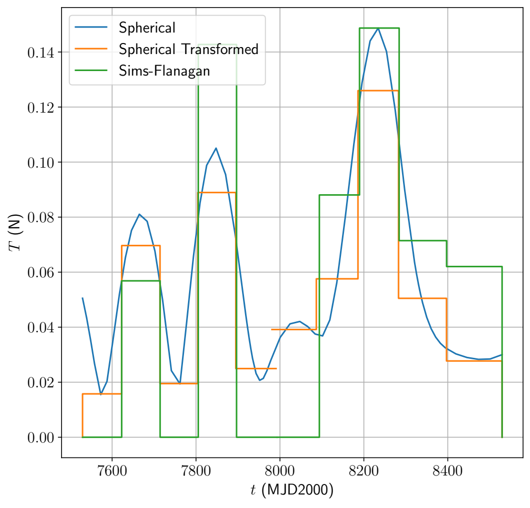
Table 5.1: Transfers used for the selection of parameters, verification, and validation of the SF implementation. Initial body is Earth on all cases.

Case	Source	Final body	t_0 (MJD2000)	t_f (MJD2000)	n_{rev}	n_{seg}	T_{max}/m_0 (mm s^{-2})	I_{sp} (s)
E-M	Figure 12 in [1]	Mars	7304	7804*	1*	10†	0.264‡	3000
E-ML	Figure 14 in [1]	1989ML	7529	8529*	2*	10†	0.149‡	3000
E-9P	Figure 16 in [1]	9P/Tempel	465	1565*	0*	10†	1.953‡	3000
E-N	Figure 18 in [1]	Neptune	7409	20409*	0*	10†	1.654‡	3000
B-M	Table 11.3 in [5]	Mars	8174	8754	1	10†	0.225‡	3000
B-ML	Table 11.11 in [5]	1989ML	7799	8399	1	10†	0.172‡	3000
B-9P	Table 11.21 in [5]	9P/Tempel	4124	5584	0	10†	1.333‡	3000
B-N1§	Table 11.17 in [5]	Neptune	7754	21254	0	10†	1.022‡	3000
B-N2	Table 11.18 in [5]	Neptune	10049	50549	10	50†	0.035‡	3000
S-Me¶	Figure 6 in [2]	Mercury	2655	4982	10*	30	0.140	3337
S-VX	Figure 2 in [3]	2002VX91	7509	8587	3*	100	0.085	3000
S-J	Table 9 in [34]	Jupiter	8869	10888	0 to 2*	30*	0.113	6000
S-S	Table 9 in [34]	Saturn	8317	10948	0 to 2*	30*	0.113	6000

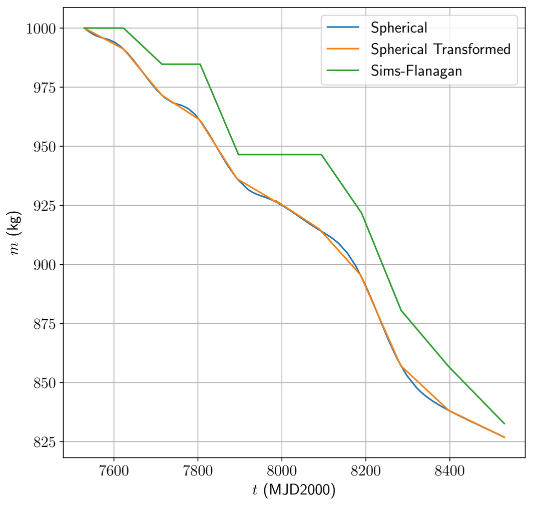
* Approximate value obtained from figure in source.
† Arbitrary value selected for tests.
‡ T_{max} value corresponds to maximum $|\boldsymbol{T}|$ value of shape-based trajectory.
§ Initial velocity with respect to Earth of 3 km s^{-1} tangential to its orbit.
¶ Initial velocity with respect to Earth of −2 km s^{-1} tangential to its orbit.



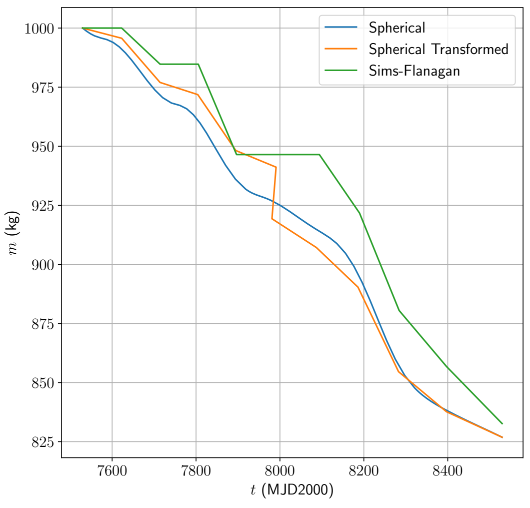
a)



b)



c)



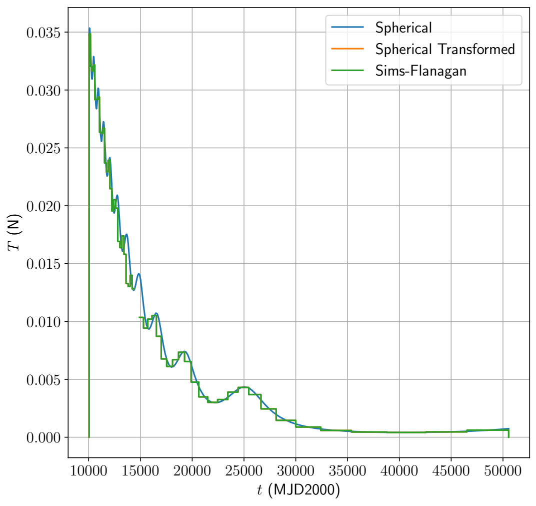
d)

Figure 5.1: SF solution for the E-ML case: a) actual thrust as a function of time, b) alternative thrust as a function of time, c) actual mass as a function of time, d) alternative mass as a function of time.

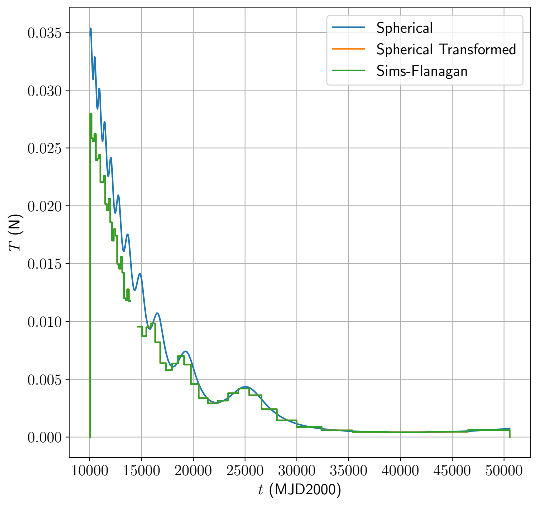
represent the thrust in the first part of the transfer, in which the change is fast. This led to a failed SF solution, while the method used in the thesis reached a converged solution.

5.2.2. Integrator

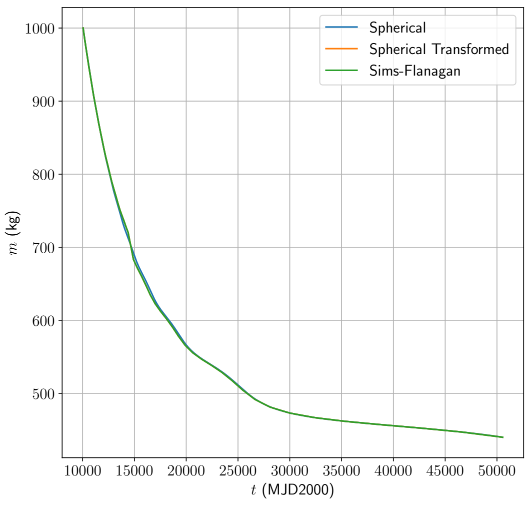
The implementation selected is designed to use the Taylor integration method. With this method, the state is only computed at the end points of the SF segments. This integration method is expected to have a lower computational cost than other meth-



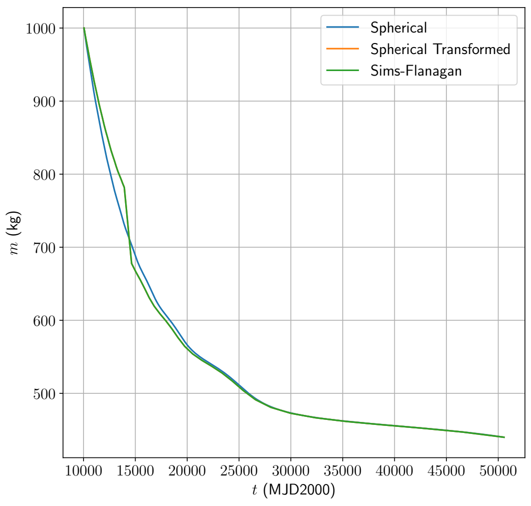
a)



b)



c)



d)

Figure 5.2: SF solution for the B-N2 case: a) actual thrust as a function of time, b) alternative thrust as a function of time, c) actual mass as a function of time, d) alternative mass as a function of time.

ods [2], so part of the increase in computation cost from the more complex SF is expected to be compensated. The Pykep implementation only allows requires one tolerance value for the integration, δ_{SF}. To evaluate the effects of this tolerance, the change in final predicted mass and computation time were considered the most relevant results. However, tests showed that there was no significant change of this values when modifying the tolerance several orders of magnitude with respect to the default value $\delta_{SF} = 10^{-8}$. Therefore, the default value was used in the computations.

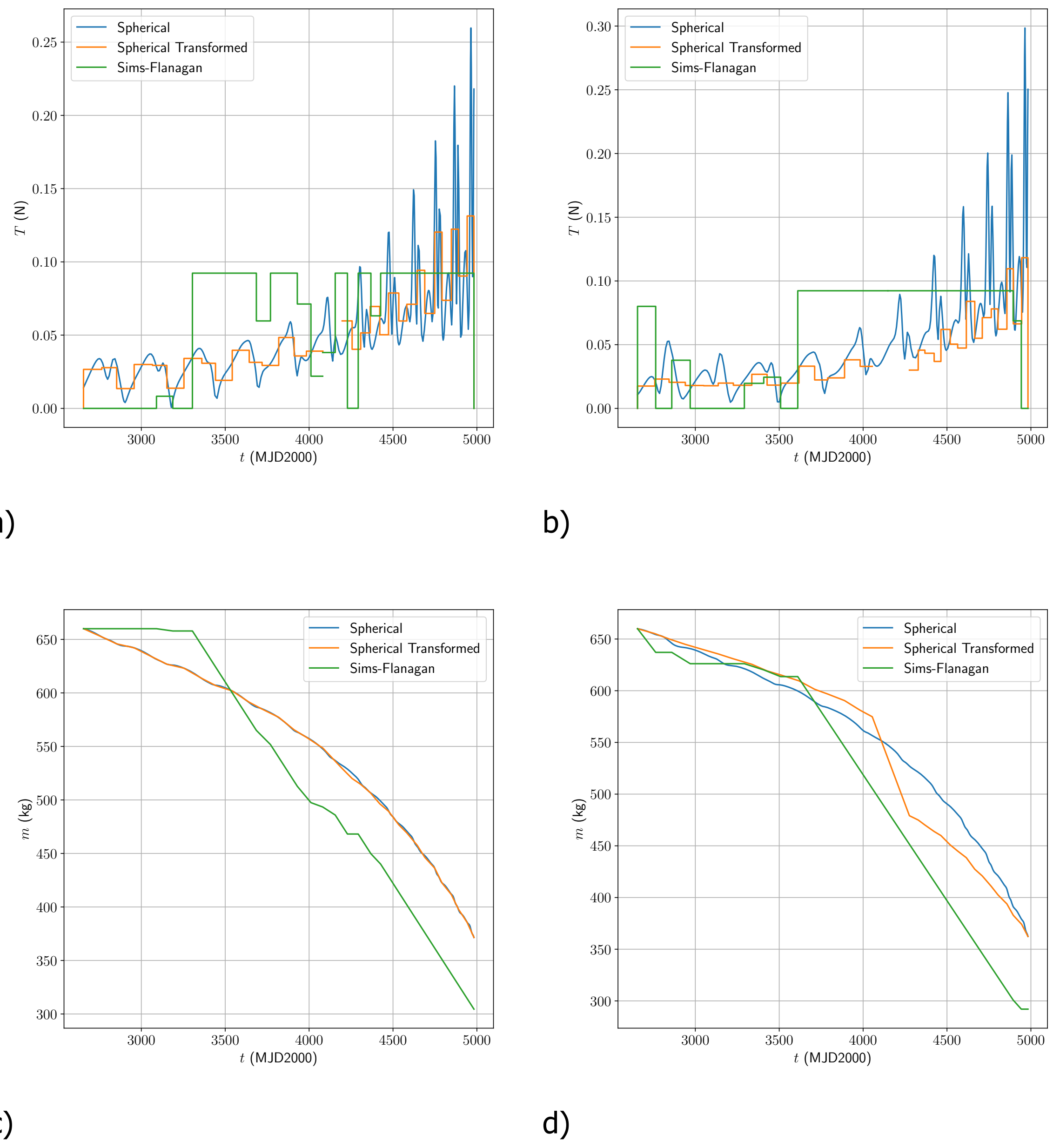


Figure 5.3: SF solution for the S-Me case: a) actual thrust as a function of time, b) alternative thrust as a function of time, c) actual mass as a function of time, d) alternative mass as a function of time.

5.2.3. Optimizer

The initial guess obtained as described in Section 5.2.1 could be used to obtain an optimal trajectory via NLP. The optimization variables were the Cartesian components of the thrust vector at each segment $\boldsymbol{T}_i$, the final mass m_f and the total change in the Sundman transformation variable $\Delta s'$. These variables were box bounded by the values indicated in Table 5.2. The value Δs corresponded to the change in the s variable in the shape-based trajectory, and m_0 to the initial mass of the leg.

The optimization problem included both inequality and equality constraints [2, 41].

Table 5.2: Bounds for vector variables of the SF method selected.

Bound	$T_{\alpha i}$	m_f	$\Delta s'$
Lower	$-T_{max}$	10^{-8}	$\Delta s/2$
Upper	T_{max}	m_0	$2\Delta s$

Table 5.3: Values used for nondimesionalization of the mismatch constrain during the SF optimization.

Units	Position	Velocity	Mass	Time
Reference value	1 au	1 au yr^{-1}	1 kg	1 yr
Value in SI units	149 597 870 691 m	4740.4705 m s^{-1}	1 kg	31 557 600 s

The inequality constraints was $|\boldsymbol{T}_i| < T_{max}$ at each segment. The equality constraints correspond to the mismatch between the state at the end of the forward integration of the first half oh the leg and at the and of the backward integration of the second half of the leg. The mismatch was nondimensionalized so the magnitudes were comparable. The values used for nondimensionalization are shown in Table 5.3. The tolerance allowed for each component of the normalized mismatch values was 10^{-6}. This might seem high in the case of the position, as it corresponds to ~150 km. However, it is considered reasonable taking into account that much larger errors were introduced by ignoring the SOI of planets during the GA.

The algorithms considered for this optimization task were the local optimization algorithms available in Pygmo[2]. Three of them could handle both equality and inequality constraints: compass_search, cobyla, and SLSQP. These algorithms were compared by testing on the cases presented in Table 5.1. The main criteria for algorithm selection was the number of successfully computed transfers. This was a more important objective that low computation times for this thesis, as explained in section 5.1. The computation time was considered for breaking ties, but it was not necessary to use it. Out of 13 cases, compass_search did not succeed in any of them, cobyla succeeded in 4 of them, and SLSQP succeeded in 8 of them. Therefore, SLSQP algorithm was used for all the subsequent computations. The Pygmo version of this algorithm is actually a wrapper for the implementation of the NLopt package[3].

This algorithm requires the cost function gradient. The expression of this gradient is not available, so a function available in Pygmo was used to estimate them with via finite differences. Two points are used for the estimation in each dimension, and the points are at a distance 10^{-8} of the units corresponding to each variable. Two stopping conditions were used in the algorithm: the minimum relative change in the input vector δx_{rel}, and the minimum relative change in the objective δf_{rel}. These values were set to $\delta x_{rel} = \delta f_{rel} = 10^{-4}$ as these were considered a reasonable values, as input values except the thrust and output value were not nondimensionalized. The

[2]https://esa.github.io/pygmo2/overview.html#local-optimization (accessed May 13, 2020).

[3]https://nlopt.readthedocs.io/en/latest/NLopt_Algorithms/#slsqp (accessed on May 13, 2020).

input thrust was nondimensionalized with T_{max} on each case and the tolerance on this variable was considered also reasonable. In addition, a limit of 20 min was enforced for the computation of the SF legs. This was included to avoid unusual cases of very long computations.

5.3. Verification

The implementation of the integration method and optimizer was considered reliable as it was obtained from well maintained packages. Instead, the verification of the SF method was focused on the correct transformation between the spherical shape-based method and the SF initial guess, and on the optimality of the solutions obtained with SF. The SF method was tested on the cases E-{*target initials*} and B-{*target initials*} described in Table 5.1, which are extensions to SF of the homonyms cases in Table 4.1.

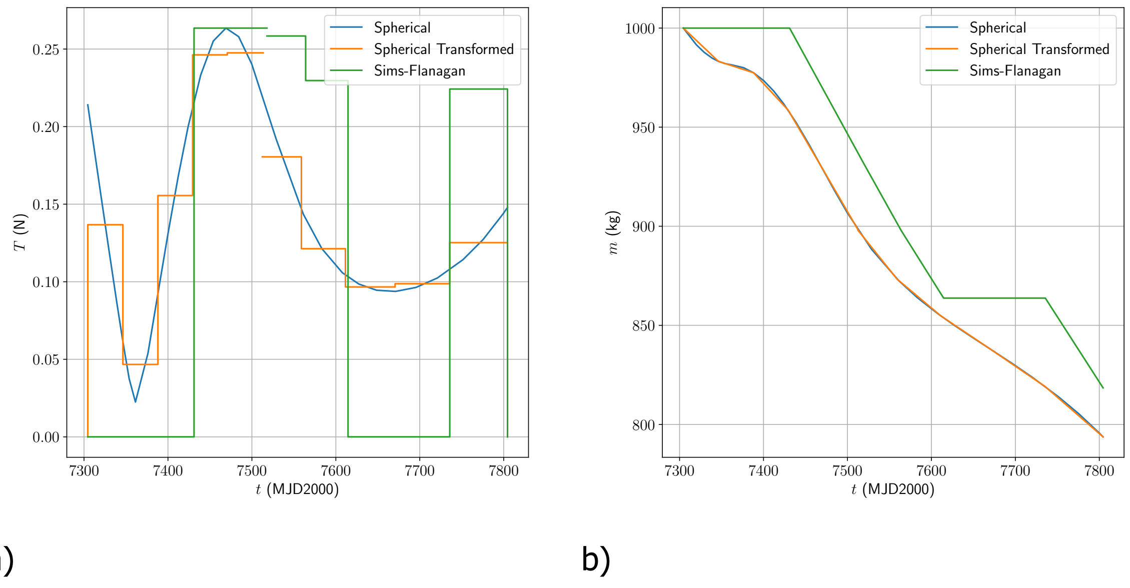
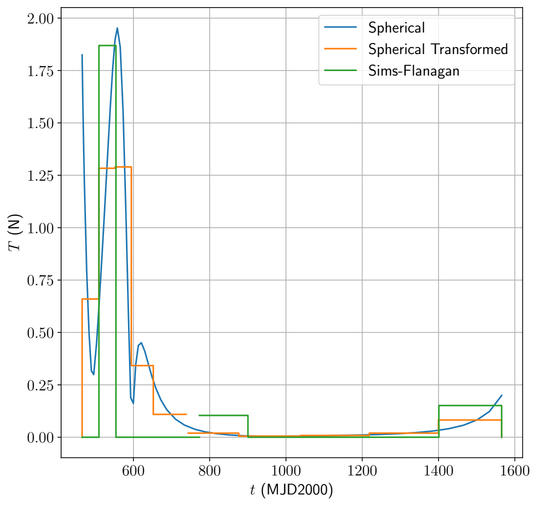
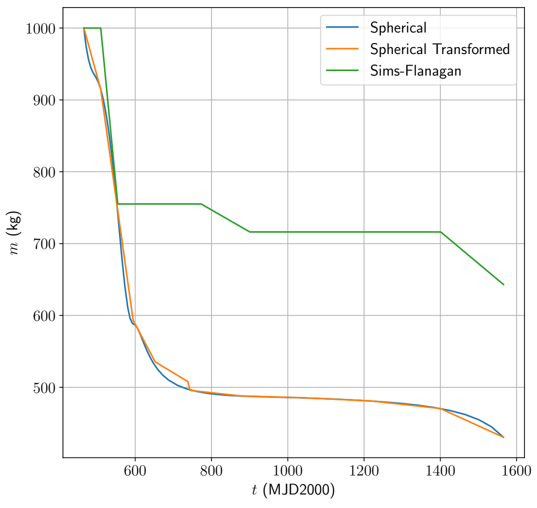


Figure 5.4: SF solution for the E-M case: a) thrust as a function of time, b) mass as a function of time.

Two results are expected from a proper implementation: the initial guess follows closely the spherical shape-based trajectory and the final result requires less propellant mass. Results are shown in Figures 5.1, 5.2, 5.4, 5.5, and 5.6. The results are the ones expected. First, the thrust profiles indicate that the initial guesses had thrust values corresponding to the averages of the shape-based thrust at each segment. In addition, the optimized SF thrust showed a bang-bang behavior in most cases as it was expected for the optimal control. Second, the mass evolution indicate that the initial guesses followed very closely the shape-based solution, and the optimized solution required less total mass in all cases.

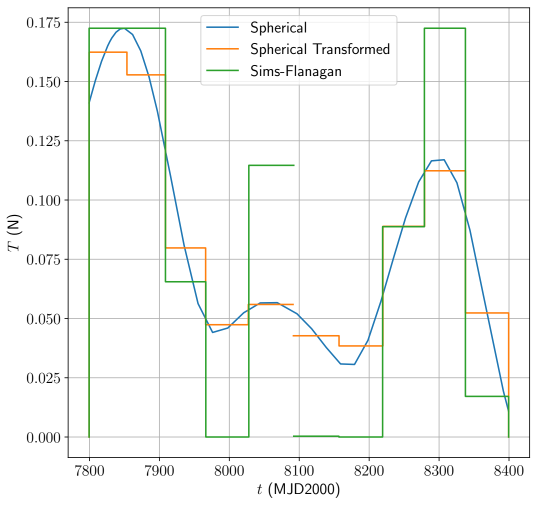
Finally, Figure 5.2 shows the limitations of this method. The SF optimization was not capable of finding a viable solution as the segments were to long to reproduce the fast changes in thrust in the shape-based method. This was the case even when

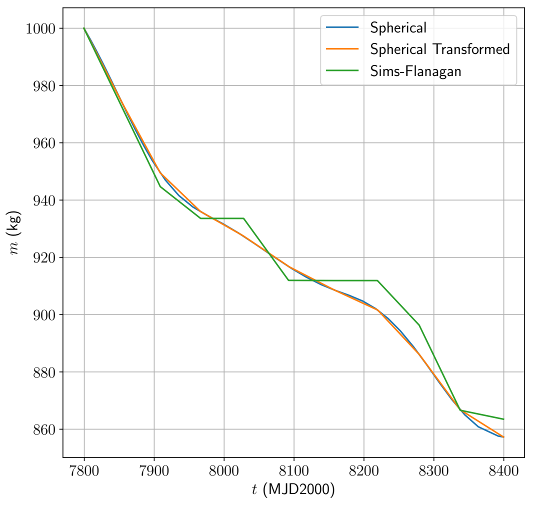




a) b)

Figure 5.5: SF solution for the E-9P case: a) thrust as a function of time, b) mass as a function of time.





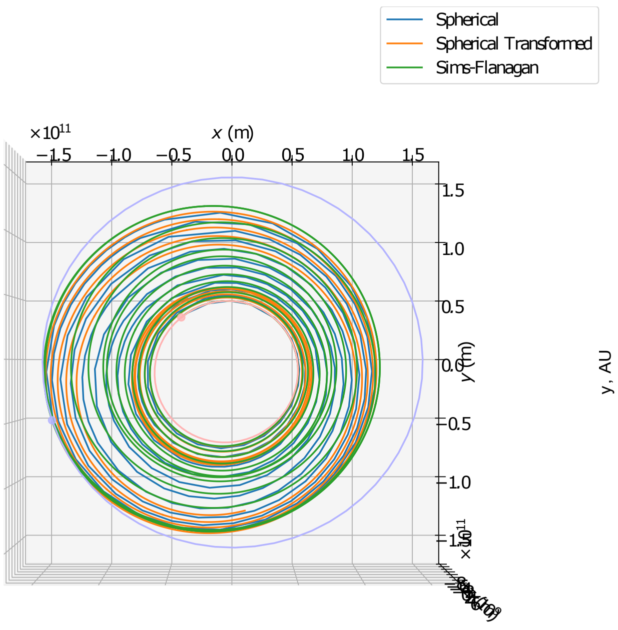
a) b)

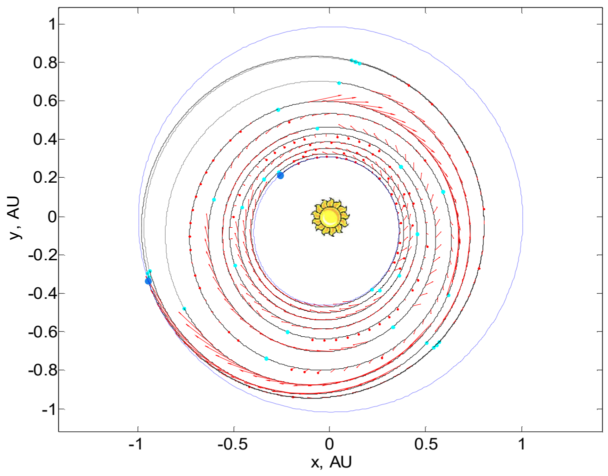
Figure 5.6: SF solution for the B-ML case: a) thrust as a function of time, b) mass as a function of time.

using a very large number of segments.

5.4. Validation

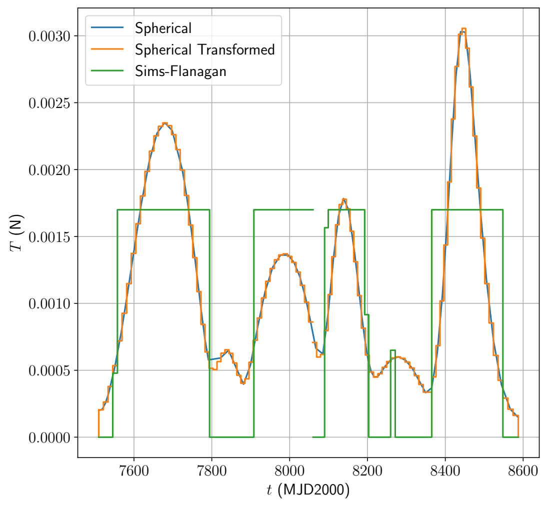
The validation procedure was to compare the results with examples in the literature. However, there was no result in the literature that could be directly compared. Some

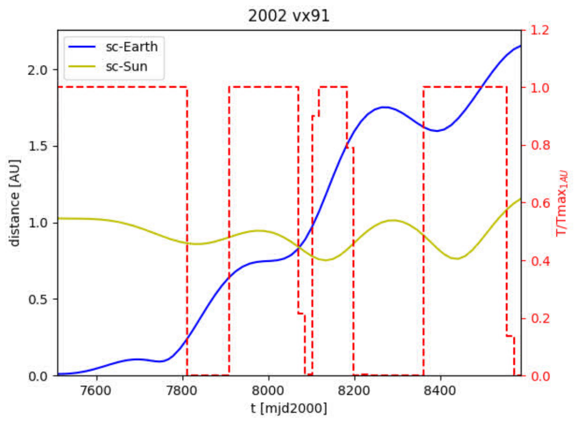




a) b)

Figure 5.7: Comparison of SF trajectories in the S-Me case: a) obtained in the thesis, b) in Reference [2].





a) b)

Figure 5.8: Comparison of SF thrust along the trajectory in the S-VX case: a) obtained in the thesis, b) in Reference [3].

references use the same implementation of the SF method but do not provide the detailed of the trajectories produced, so results can only be compared graphically. This was the case with cases S-Me and S-VX. The comparisons of the results are shown in Figures 5.7 and 5.8 for the S-Me and S-VX cases respectively. The results obtained are very similar to the one in the references. The small differences are likely caused by the different initial guesses. The references for cases S-J and S-S do provide numerical details about the best result. However, the optimization failed to converge with the SF method used in this thesis. In the reference, the simpler version of the SF method [41] was used. That implementation does not necessarily result in feasible trajectories, so the results in the reference might correspond to an unfeasible solution. Moreover, these cases are direct trajectories to outer planets that are not expected to be computed when optimizing a MGAT.

6
Machine Learning Surrogate

The objective of this thesis was to study the use of a ML surrogate to substitute the cost function computations while performing a global optimization of a MGAT. This chapter describes the implementation of the ML method selected and the rationale behind its selection.

6.1. Method selection

Previous research on the use of surrogates for estimation of the fuel cost of LTP trajectories found that the most accurate ML method to build the surrogate was Gradient Boosting (GB) [13, 14]. However, the situation considered in the reference is very different to the one in this thesis. First, the set of trajectories considered in the reference only includes transfers between Near Earth Object (NEO). All these transfers have very similar characteristics, as opposed to transfers between any pair of bodies in the solar system as considered in this thesis. Second, the number of points used for training is between 50 000 and 100 000. This number is much higher than the number of points expected to be available for training in this thesis, which is <2000.

Therefore, an alternative which could work with a much lower of training points was sought. As mentioned in section Use of Machine Learning Surrogate of the thesis paper manuscript, page 11, it was decided to train the ML online so the number of training points used was the minimum necessary. Several alternatives were considered: online version of GB, ANNs, GPR, non-parametric models, and OS-ELM.

6.1.1. Gradient Boosting

The GB and other tree-based algorithms [66] cannot be used online and require to train again for every new point. This requires a relative large computation time of ~20 s [13]. Moreover, the new model created every time would be completely different than the previous, so error estimation would not be reliable. Finally, the accuracy is expected to be much lower when only a small amount of training points are available. Therefore, these models were not used in this thesis.

6.1.2. Artificial Neural Networks

They were used as a surrogate in Reference [14], and to learn optimal state-control pairs in similar LTP problems [15–17]. However, they show a worse performance than the alternative method OS-ELM in small datasets when using a single layer [23, 27]. Increasing the number of layers does not help as the performance is only expected to decrease if not enough training data is available [86]. Moreover, a large number of hyperparameters need to be adjusted for an adequate training [87]. These can be adjusted by performing cross-validation to determine which combination of hyperparameters performs better [88, 89]. However, this process is time consuming, and was an important obstacle in previous master thesis on similar topics [19, 90]. In addition, there is high chance of overfitting to the case from which the data was generated. This means that the model will have a bad performance when used in a different case. An alternative approach is to test on data from several cases, but that requires a long computation time to cover all relevant options. Therefore, ANNs were not used in this thesis.

6.1.3. Nonparametric Methods

This label encompasses a great variety of methods. They are all based on estimating the objective function based exclusively on the values of previously computed points without making any assumption about the underlying distribution [91]. An example are kernel methods, that assign a new point the value of nearby points weighted with a function of the distance known as kernel [66]. These methods have some advantages for the application proposed here. The simpler versions do not require training and are inherently online as no parameters need to be determined and the values at new points are computed using the list of training points. This might be an issue for a large amount of training points, as the distance to every point needs to be evaluate every time a new point is predicted. However, that was not the case in this work. Nonetheless, these methods also have some issues if used in this thesis. The main one is that they have a limited generalization capability. Solutions are very unreliable when computed in areas where a low density of training points is available [91]. The optimization method is expected to produce points clustered around minima of the function, so the surrogate would have a very low error in small areas and very high error when evaluated at point outside those areas. Therefore, nonparametric methods were not used in this thesis.

6.1.4. Gaussian Process Regression

This method can be considered an advanced non-parametric method. It has been used successfully for the creation of surrogates for several applications, including the estimation of the cost of transfers to asteroids, using both conventional propulsion [22] and low-thrust [20]. This method shows a better generalization performance than the nonparametric methods, but still suffers when data is not uniformly distributed. Moreover, the selection of hyperparameters values is a necessary step to achieve reasonable accuracy. The process to obtain them is usually a gradient descent, so the time and data required to define this parameters is approximately the same that for the training of other models. This negates the advantages of using the model online as it needs to be trained beforehand. Therefore, GPR were not used in this thesis.

6.1.5. Extreme Learning Machine

The method described in section Machine Learning Surrogate of the thesis paper manuscript, page 13 was considered the most adequate for the application intended in this thesis. There are two main reasons for this. The first one is that the lack of optimization of the first set of weights acts as a normalization controlling the accuracy of the model. This is detrimental when a large amount of points are used as only a part of the model is optimized. In that case, an ANNs with the same number of hidden units but with both sets of weights optimized will probably have a better accuracy than the ELM. However, the situation is different when the number of training points is small. In this case, the ANNs would typically overfit to the samples [92], while we expect the ELM to produce a more general solution [23]. The second reason is the possibility of performing online training while providing the optimal solution to all the training points used.

6.2. OS-ELM-MAS Implementation

The description of the OS-ELM-MAS method used to create the surrogate is given in section Machine Learning Surrogate of the thesis paper manuscript, page 13. The implementation was done using the TensorFlow-OS-ELM package[1]. This package is publicly available but it seems to be a personal project from the creator. Therefore, the implementation is considered to require extensive validation and verification. The selection of the error measure, error implementation and architecture parameters were considered part of the implementation process. These procedures, and the related validation and verification are described in this section.

The package selected for the implementation uses the Tensorflow package[2] for matrix operations. This package usually allows for faster matrix operations than other Python implementations. However, it was found that Tensorflow and Pygmo had compatibility issues after all the modules of the thesis were implemented and integrated. A workaround was found to make the combination of two packages work, but it involved saving the Tensorflow models to the hard disk every time it was modified and loading them from disk every time a prediction was made. This introduced a large computational overhead when using the OS-ELM-MAS that reduced the efficacy of

[1] `https://github.com/otenim/TensorFlow-OS-ELM` (accessed May 13, 2020).

[2] `https://www.tensorflow.org/` (accessed May 13, 2020).

Table 6.1: Final estimated error measures for the optimization with the surrogate and the final configuration of parameters for each of the three cases tested.

Error measure	Dawn	EEJ	EVVMMM
MAE	0.029 12	0.029 70	0.088 10
MAPE	13.41%	36.08%	33.26%

the method. The final results were positive even including the excessive overhead, so even better results are expected if the implementation is refined.

6.2.1. Error measure

Some of the most common error measures in the literature [93] were considered for evaluating the accuracy of the surrogate. These were the Root Mean Squared Error (RMSE), the Mean Absolute Error (MAE), and the Mean Absolute Percentage Error (MAPE):

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} (\hat{y}_i - y_i)^2}{N}}, \qquad \text{MAE} = \frac{\sum_{i=1}^{N} |\hat{y}_i - y_i|}{N}, \qquad \text{MAPE} = \sum_{i=1}^{N} \left| \frac{\hat{y}_i - y_i}{y_i} \right| \frac{100}{N},$$

where N is the number of points for which the error is computed, and $\hat{y}_i$ and y_i are respectively the predicted value and the true value for point i. It is important to notice that RMSE = MAE when $N = 1$. This a very important conclusion for the error estimation method presented in section Error Estimation of the thesis paper manuscript, page 15. The error estimation would be identical for the RMSE and the MAE when the error is estimated from points that are received online one by one. Therefore, there is no difference between RMSE and MAE for the application considered in this work. Therefore, RMSE is not considered in the following discussion.

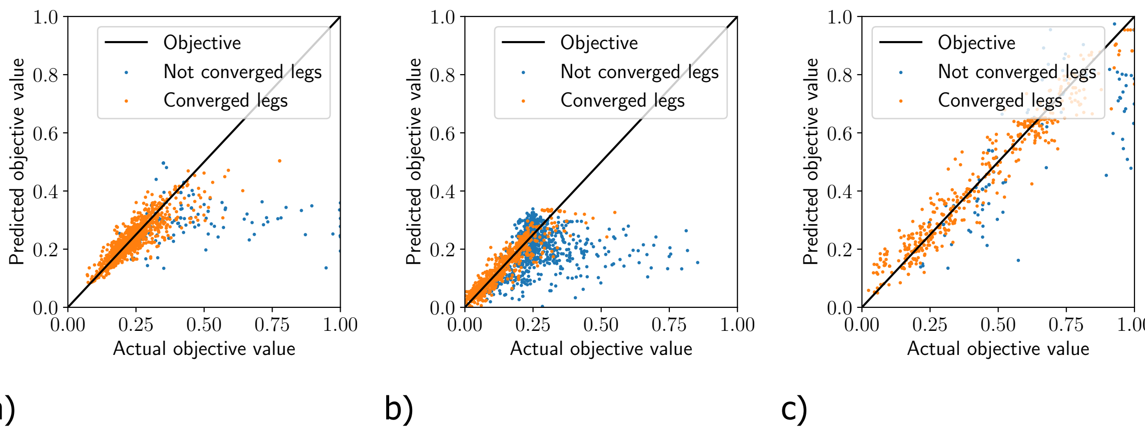


Figure 6.1: Final predicted values as a function of true values for the optimization with the surrogate and the final configuration of parameters for each of the three cases tested: a) Dawn, b) EEJ, and c) EVVMMM.

Among the main differences between the two remaining error measures is the importance of each point has in the final error value as a function of $|y_i|$. For a fixed $|\hat{y}_i - y_i|$, MAE weights identically points independently of the y_i value, while MAPE weights more points with smaller $|y_i|$. The optimization algorithm is more likely to select points with a lower $|y_i|$ as y_i corresponds to the cost of the leg in this case. Therefore, the MAPE measure could seem to be a better option as more weight is given to points that have more chances to be selected by the optimization problem. However, the tests shown that MAPE error had very high values for the test cases in which some transfers had very low $|y_i|$ values. To illustrate this, different final estimated error measures were computed for optimizations using the surrogate in the three test cases. The results corresponds to the final configurations of surrogate control parameters τ and C_n. The final estimated error values for different error measures are shown in Table 6.1, and the corresponding predictions on the points are shown in Figure 6.1. The accuracy of the predictions seems to be very similar for the Dawn and EEJ cases. However, the MAPE assigns a much larger error to the EEJ case. The reason is that the objective values are much lower in the EEJ case. This was not considered to be a adequate measure of the error, and the MAPE was not used for the computations in the thesis.

6.2.2. Error estimation

The only detailed summary of methods for online estimation of the error was found in Reference [26]. In the reference, four different types of estimated errors are proposed: the holdout error, the prequential error, the prequential error on sliding windows, and the prequential error with a fading factor.

For the holdout error, a test set is kept separate from the training data and the error is evaluated on this test set. The main drawback of this method is that a test set needs to be obtained and cannot be used for training. This is a problem in this thesis, as generating new data had a very high computational cost. Creating a representative test set would probably require more computation time that the time saved by using the surrogate. Therefore, this estimation of the error was not considered.

The prequential errors are calculated by computing the error for each new training point before the model is trained on that point. This way, the same training points can be used for validation and no extra data needs to be generated. The simpler version of the prequential error is to compute it as the average of the error on all the training points up to that moment. However, this estimation has a flaw. The error computed for every point cannot be updated after they are used for training. Therefore, the error on the older points corresponds to an older version of the model as it trained on new points since that error was computed. It is assumed that the accuracy of the model always increases when trained on new point. Therefore, these old errors are a pessimistic estimation of the true error as they were computed in a less accurate model. The older the error computations, the more pessimistic they are with respect to the current state of the model.

Two alternative types of prequential error are proposed in the reference to provide a more accurate estimate of the error. The first one is the prequential error over a moving average. This error is computed with a fixed number of the most recent error computations on the training points. The second one is the prequential error with a fading factor, which is described in section Error Estimation of the thesis paper manuscript, page 15. Reference [26] recommends the use of the prequential error with fading factor for applications similar to the one in this thesis, so that was the estimation used.

6.2.3. Architecture Selection

There were several architecture parameters that needed to be defined for the surrogate model used. The parameters to determine were the Number of Models (*NM*), Number of Hidden Units Parent (*NHUP*), and the Activation Function (*AF*) in the hidden units. The most common *AF* used in ANNs [94, 95] were considered. However, the *rectifier linear unit* (relu) activation was discarded as the performance was abysmally worse than the *hyperbolic tangent function* (tanh) and *logistical sigmoid function* (sigmoid) in preliminary tests. This did not agree with the literature, where it usually outperforms other *AF* [95]. However, it was hypothesized that the reason for this was the collinearity of the rows in matrix $\boldsymbol{H}$ for initial training. The input space had dimension 14, so that was the maximum range achievable of matrix $\boldsymbol{H}$ if no activation is used. The range $\tilde{N}$ required is achieved thanks to the nonlinearity introduced by the activation function. However, the relu activation is linear in most of its domain, so most of the samples would still be linear combinations after applying the activation function. Therefore, the initial training would provide erroneous values that would spoil the model. relus were not considered for the architecture parameters tests.

In addition to the previous parameters, the *SI* and *SO* also needed to be determined. Two options were considered: the Standard scaler (S)[3] transforms the data so the mean is 0 and the standard deviation is 1, and the MinMax scaler (MM)[4] transforms it so the lowest value has a transformed value of -1 and the highest value a transformed value of $+1$. The scaling factors were determined with the data used for initial training in all cases.

The Number of Hidden Units (*NHU*) was fixed to 128 and no variations were tested. The error was observed to decrease monotonically with *NHU*, but a larger *NHU* requires more points for initial training. In the tests, the computation of each point required ~120 s and the convergence rate of the SF computation was 0.7. Only converged solutions are used for training, and 20 extra points were required to make sure the error estimation is reliable. Therefore, the approximate minimum time required before the OS-ELM-MAS with *NHU* 128 could be used was $\sim 120\frac{128\cdot 2+20}{0.7}=$ ~47 300 s = ~790 min = ~13.1 h. This was considered a reasonable upper bound as the surrogate could still be used during most of the 72 h test period.

[3]`https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html` (accessed May 13, 2020).

[4]`https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html` (accessed May 13, 2020).

Table 6.2: Summary of parameters combinations tested.

Parameter	First iteration	Second iteration	Third iteration
NM	8	8 and 16	16
NHUP	32	16, 32, 64 and 128	16, 32, 64 and 128
NHU	128	128	128
AF	tanh and sigmoid	sigmoid	tanh
SI	S and MM	S	S
SO	S and MM	S	S

The Number of Models (*NM*) was limited to 16 during the tests. The reason for this limitation was related to the limits of the implementation of OS-ELM-MAS models in this thesis. As mentioned previously, issues with package compatibility forced to save every individual OS-ELM model to the disk, load it for predictions, and overwrite it for updates during training. This was a very inefficient implementation and caused the computation time required to train an OS-ELM-MAS to grow fast with *NM*. The limit on *NM* was intended to prevent the training of the OS-ELM-MAS model to require a significant amount of the total optimization time when used as a surrogate. However, the computation cost of the models is expected to decrease several orders of magnitude with a more refined implementation. This will allow to use larger *NM* without additional computation cost. The surrogates with larger *NM* are expected to have a better performance. Therefore, the results obtained in this thesis can be considered a conservative estimate of what can be achieved with an improved implementation of this method.

Performing a grid search over all the possible parameter values would have required a long computation time. Instead, the parameters were tested in groups. The values tested at each iteration are are shown in Table 6.2. The high computation cost for every optimization makes impossible to test which is the architecture that works better during a complete optimization run. Instead, the information about the legs computed during an optimization without surrogate was saved and then used as training data. It was simulated that the datapoints arrived one by one to the model as it happens when it is used as a surrogate for optimization. As mentioned in section Results of the thesis paper manuscript, page 20, the data used corresponded to the Dawn baseline with $N_I = 20$. Of the three cases tested, Dawn was the one with a lower variety of transfers, as all the bodies visited have similar orbits. Therefore, this case might not be considered representative of all the possible cases. This allowed to study the robustness of the architecture when applied to very different transfers. The conclusions were very positive as the surrogate improved its performance in the other two cases.

Each parameter combination was evaluated with respect to two criteria. The first one was the ability to accurately approximate the correct value of the objective function. This was measured with the final estimated MAE when the model was trained in all the data. This value was estimated as explained in section Parameters Test of the thesis paper manuscript, page 18. The value of the estimated MAE was found to have a constant offset between different combinations during the training process. Therefore, the final value was representative of the performance along the whole training procedure.

The second criterion was define by considering that the surrogate should not assign very low values to the points that did not have a valid SF solution. As mentioned in section Direct Method of the thesis paper manuscript, page 13, the surrogate cannot tell these points apart from the valid points, so predictions were returned when they were evaluated. If this prediction was to low, this points were selected over the valid points. To evaluate the capability of the surrogate to avoid this situation, a metric was defined as

$$F_{score} = \frac{F_{pred,min} - V_{0.5}}{V_{min} - V_{0.5}},$$

were $F_{pred,min}$ was the lowest prediction made on any failed point at any point during training, $V_{0.5}$ was the median of the true values of the valid points and V_{min} was the lowest true value of the valid points. The predictions on the failed points were made after each online training point, and $F_{pred,min}$ was the lowest prediction made at any point.

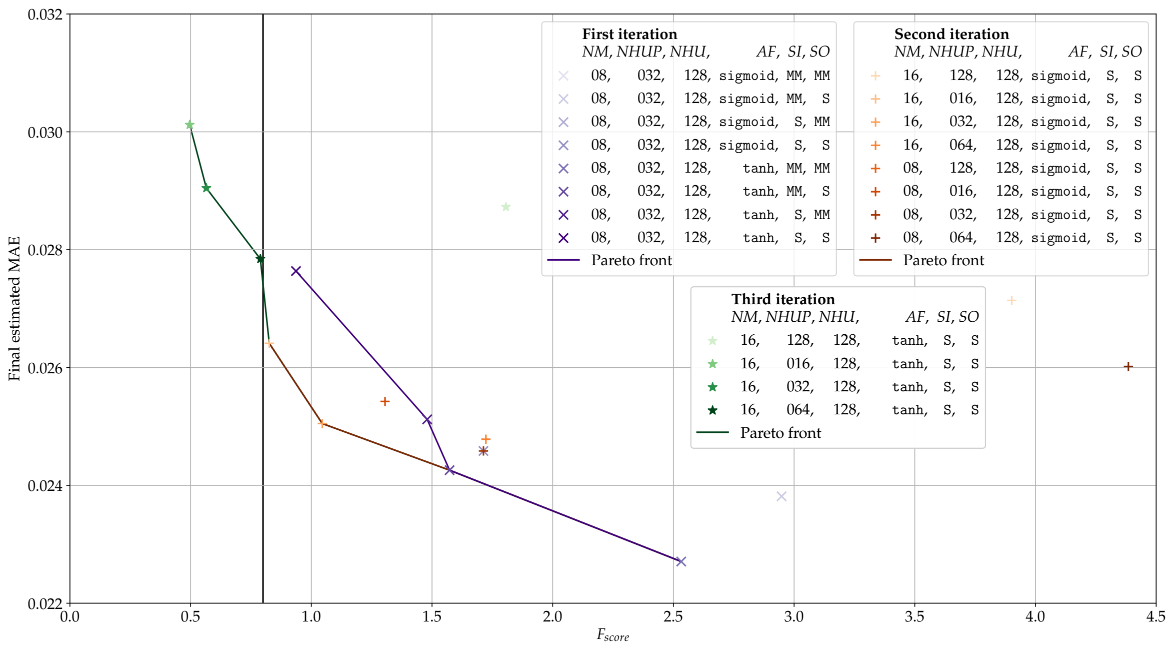


Figure 6.2: Values of estimated MAE and F_{score} for the architecture parameters tested.

The results for several parameter combinations with respect to these criteria are shown in Figure 6.2. The final value selected was the one with $F_{score} < 0.8$ and the lowest MAE. In addition, the results were also obtained with a simpler OS-ELM model using the same parameter combinations for *NHU*, *AF*, *SI*, and *SO*. The comparison between the two pareto fronts is shown in Figure 6.3. The results indicate that OS-ELM-MAS had clear advantages over OS-ELM.

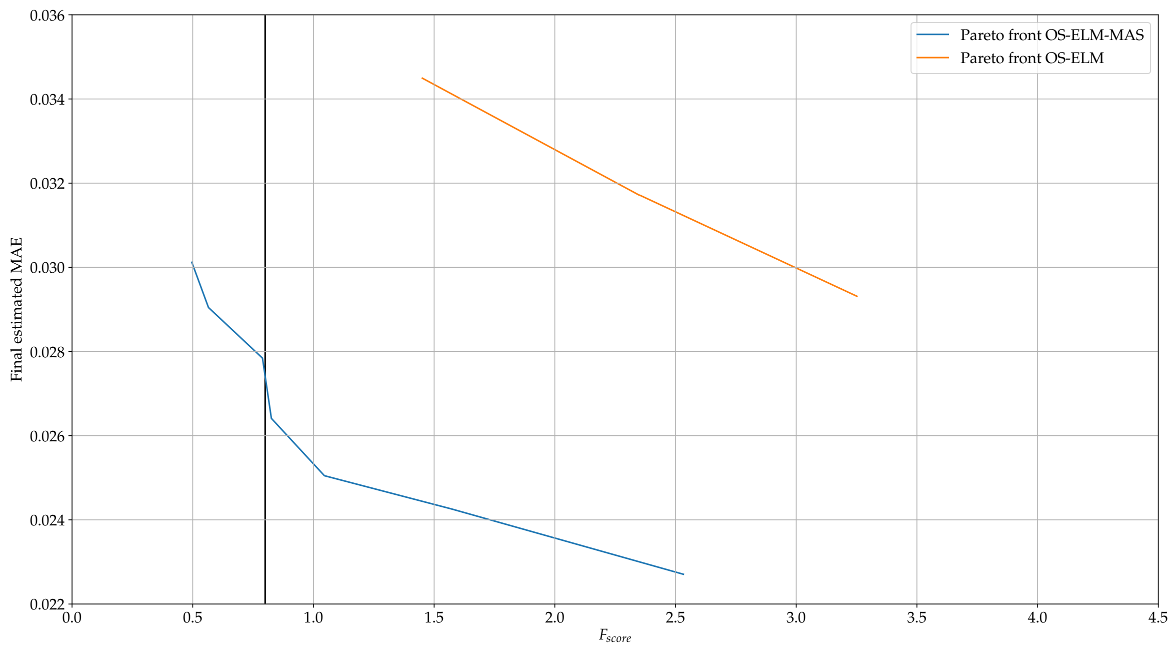
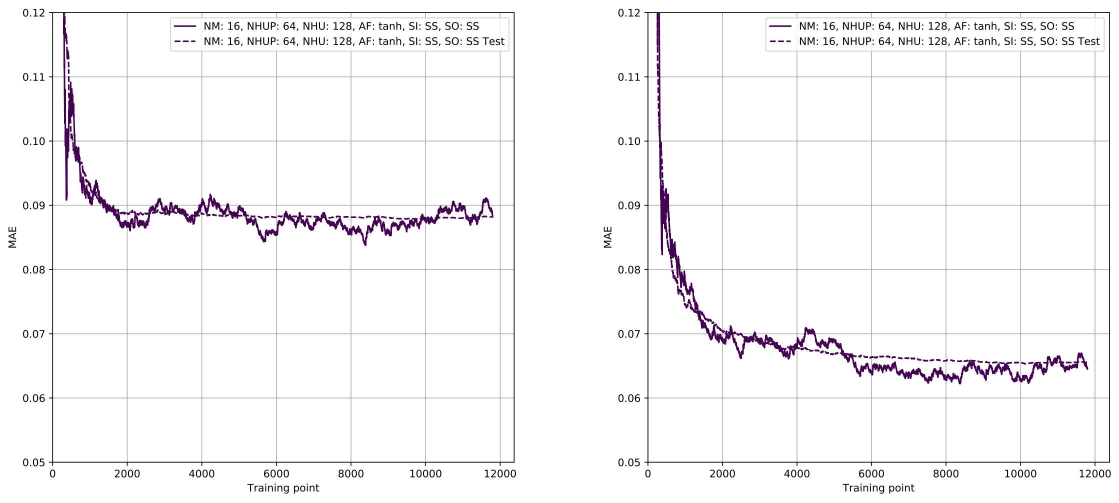


Figure 6.3: Comparison of optimal MAE and F_{score} values obtained when testing different architecture parameters for OS-ELM and OS-ELM-MAS.

6.2.4. Verification

There was no straightforward way to predict the results when training the OS-ELM-MAS. The verification of the model was based on testing on certain datasets and evaluating whether the results corresponded with what was expected. This process was done before data from transfers computed with SF was available. Instead, the verification was done with the data obtained during the grid search used to create the contour plots in subsection 4.4.2. This provided a large amount of data with similar characteristics to the SF transfers. The data from the C-M and C-ML cases was combined to create a single dataset. This provided a total of 23 595 training points. This number was much larger than the number expected to be available when using the surrogate. As the lack of data was not a problem, half the points were randomly split to be used for validation.

The estimated MAE and the MAE obtained in the dataset were computed when using both the OS-ELM and OS-ELM-MAS models. Results are shown in Figure 6.4. Three conclusions were obtained from the results. First, the error estimation method provided a good estimation of the test error, although it presented random variations around the validation error. Second, error decreased fast in the early points and stabilized after ~ 2500 training points. Third, OS-ELM-MAS reached lower error values than OS-ELM. This was the behavior expected from the model, so the OS-ELM-MAS model was considered verified. In addition, the initial error threshold $\tau = 0.06$ for the tests described in section Parameters Test of the thesis paper manuscript, page 18 was selected based on the results of this test.



a) b)

Figure 6.4: Estimated MAE and test set MAE when using: a) OS-ELM model, b) OS-ELM-MAS model.

Table 6.3: Average error of final OS-ELM model evaluated on the train and test points for some public datasets.

Source	Auto-MPG		Abalone		California housing	
	Train	Test	Train	Test	Train	Test
Thesis	0.0681	0.0867	0.0762	0.0774	0.1471	0.1697
Reference [27]	0.0680	0.0745	0.0754	0.0777	0.1303	0.1332

6.2.5. Validation

The validation process consisted on comparing the results on public datasets obtained with the model implemented in this work and the results in the literature. No results on public datasets for regression tasks are available in the reference where the ELM-MAS was presented [28]. Therefore, there is no reference results to compare the OS-ELM-MAS model. However, there are results available for OS-ELM [27], so these were the results compared. Three datasets are used: Auto-MPG[5], Abalone[6], and California housing[7]. The architecture parameters and split between train and test point is the same as in the reference. Tests are done with 50 different initialization and the error results presented correspond to the scaled datasets. The RMSE evaluated on the whole test or training dataset was used as a measure of the performance as it is the value present in the reference. Results are shown in Table 6.3. All values matched closely the values in the reference, so the validation was considered successful.

[5]https://archive.ics.uci.edu/ml/datasets/auto+mpg (accessed May 13, 2020).
[6]https://archive.ics.uci.edu/ml/datasets/Abalone (accessed May 13, 2020).
[7]https://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_california_housing.html (accessed May 13, 2020).

7 Conclusions and Future Work

The results presented in the previous chapters allow to answer the research questions introduced in section 1.1. Moreover, some possible extensions of this thesis were found when analyzing the results. These are presented in this chapter.

7.1. Objective and Research Question

The auxiliary questions formulated and the answers found are:

1. **In which part of the MGAT optimization process is the ML surrogate used?**

   It was considered to use the surrogate to substitute the cost function computation as it is the most time consuming part of the optimization. Two cost functions are computed during the optimization process: the global cost function for the whole transfer and the cost function for each leg. It was decided that approximating the cost function of the legs was the best option.

   (a) **What advantages can be obtained by using it in that part?**
   The cost function of the individual legs was expected to be easier to approximate with the surrogate than the global cost function. The shape was thought to be simpler and the dimensionality of the inputs was lower. In addition, more training points can be obtained in the same computation time. Finally, the computation of the leg cost is by far the most time consuming part of the global cost function computation, so the time saved by substituting it with the surrogate is virtually the same as by substituting the global cost function.

   (b) **How can the use of a ML surrogate be implemented in that part?**
   Online training was used for the ML surrogate. This allowed to train in the points obtained from the optimization as they were generated, but required

to introduce a method to decide when the predictions of the surrogate were accurate enough to be trusted.

(c) **What is the accuracy that can be expected from the ML method?**

Before the actual test cases were implemented, the accuracy of the surrogate was tested on data generated with the shape-based method. This indicated that the accuracy was reasonable to use the surrogate in optimization. However, the final accuracy results during the optimization with the surrogate varied significantly depending on the test case considered. This indicates that the accuracy is dependent on the particular transfers considered and the range of values in the objective function.

2. **Which are the inputs and outputs of the function to approximate?**

The inputs to the cost function to be substituted were the TOF of the leg and the initial and final states of the leg except the final mass. The output was the optimal propellant mass fraction of the leg.

(a) **Which ML method is used?**

The OS-ELM-MAS was selected as the most adequate method for this problem. The good performance for small number of training points, the small number of hyperparameters, and the fast training process are its key advantages. It was a novel method implemented by combining the OS-ELM and ELM-MAS method.

(b) **How is the training data generated?**

The training data was generated by using a SF direct transcription method. This method requires an initial guess for the solution that was obtained with a spherical shape-based method. This is the highest fidelity deterministic method available for the problem considered. A deterministic method was very important as any random effects on the values of the training data would have a detrimental effect on the accuracy achievable by the ML surrogate.

(c) **What is the computational cost of each step?**

The most costly part of the optimization process is the computation of the cost function for individual legs. The cost of training and predicting with the surrogate was expected to be very low. However, it was higher than initially thought due to a suboptimal implementation due to compatibility issues between the different packages used. Nonetheless, the computational cost of using the surrogate was still much lower than using the original cost function.

(d) **Which optimization method is used for training?**

The OS-ELM-MAS method selected allows the computation of the optimal weights with a least squares regression. In addition, these weights can be updated with new training points with an iterative least squares procedure. This way, the weights are always the optimal for all the training points considered.

3. **What is the required accuracy for the ML method?**

   Every case tested presented the best performance when using the surrogate for a different error threshold. This optimal threshold could only be determined by testing different threshold values. No general rule could be obtained to determine its value, but it seems to have a relation with the characteristics of the training data.

   (a) **How is the accuracy evaluated?**

   The accuracy was evaluated in real time during the optimization using the the prequential MAE with a fading factor. This error was considered to represent better the performance of the surrogate that the other options analyzed.

   (b) **Which methods are used as a reference?**

   The original idea was to use an indirect method to obtain high fidelity solutions so the difference between the SF result and the true optimal could be estimated. However, obtaining solutions with the indirect method required a very long computation time. Therefore, it was considered that a significant number of results could not be obtained in a reasonable computation time. Instead, the optimal error threshold was determined by evaluating the results of the optimization with surrogate and different threshold values.

   (c) **What is the expected performance during the optimization when using the ML surrogate?**

   The performance when using the surrogate was evaluated by comparing with the optimization without the surrogate. The initial plan was to compare the time that both the optimization with the surrogate and with the baseline take to reach an equivalent solution in terms of final cost. However, this was found to be very difficult to compare, as the solution reached depended on the initial random state of the optimization algorithm. Instead, it was decided to determine the expected value of the cost function after a fixed optimization time. The use of the surrogate was found to be more beneficial in the cases where the cost function had a higher computational cost and when the convergence rate of the SF method was high. These were the results expected.

After answering the auxiliary questions, the main research question can be answered:

"Is a ML method faster for computing the cost function in MGAT optimization than the present optimization methods for a given benchmark?"

It was not possible to compared the time required to reach cost threshold during the optimization with the surrogate and with the baseline. The comparison of the results for a fixed computation time was used as an alternative benchmark. For this benchmark, it was shown that the optimization with the surrogate was expected to obtain better results than the baseline without them in some cases. However, this required the selection of precise error threshold and frequency of update parameters. The best combination of these parameters could only be determined by testing the optimization with different parameters. Therefore, the total computation time is probably longer as this parameter determination step is time consuming. Nonetheless, the results seem to indicate that a general method to obtain these parameters may be obtained in the future.

Finally, the fulfillment of the objective of the thesis was evaluated:

To reduce the time required to compute the cost function associated with LTP interplanetary transfers in MGAT optimization by implementing a ML surrogate to approximate it.

The objective cannot be considered completely fulfilled as the use of the surrogate still requires a parameter selection process. However, it was proven that the use of a ML surrogate has the potential to reduce the computation time and it seems possible to eliminate the remaining obstacles in the future.

7.2. Future Work

Several recommendations for future work can be obtained from the results presented:

- To use the method presented without any prior parameter determination, it is necessary to find a general law to define the surrogate error threshold, τ, and number of individuals between error updates, C_n. The results seem to indicate that the optimal values are related to the characteristics of the training data: larger spread of objective values require a larger τ value, and lower SF convergence rates require a lower C_n. If more cases are tested, the exact dependency on these factors may be determined. If that is achieved, the parameters can be set automatically during the optimization from the characteristics of the initial training data.

- Among the main limitations of the surrogate is that it cannot tell apart the legs that have a feasible solution from the ones that do not have one. This causes the optimization algorithm to select combinations of parameters that are not feasible when using the surrogate. This has an important effect in cases in which the convergence rate is low such as the EEJ case. A possible alternative to this is to use an additional ML method to classify the legs as converged or not converged before computing them with the surrogate. In addition, this can be useful even if no surrogate is used as it will prevent the computation of SF legs that are predicted to not converge. Moreover, classification ML methods are widely available and they achieve very high performances. It was attempted to implement a classification version of the OS-ELM in this thesis but it could not be completed due to time constraints.

- The implementation of the methods presented was a prototype created as a proof of concept. Python was chosen as the programming language due to the easy implementation and the number of tools already available. However, the performance can probably be improved by implementing the method in a more efficient language such as C++. This is necessary to use the method for large scale optimization with a reasonable computation cost, and will allow to make additional tests to better understand the performance.

- The use of the surrogate presented is independent of the global optimization algorithm. A DE algorithm was chosen as it is a common choice in similar problems. However, there is no reason to think that the surrogate will have a worse performance with different gradient-free algorithms. However, this hypothesis needs to be tested.

- The tests cases considered assumed a NEP propulsion system for the spacecraft. The surrogate is expected to be able to approximate cost functions obtained with more complex propulsion systems and dynamics models with the same accuracy, as their overall shape is expected to be similar. This was not tested in this thesis due to the longer computation times that it would require. However, the use of the surrogate is more advantageous in this computationally expensive functions. Therefore, it is interesting to test whether the surrogate can actually approximate those functions with the same accuracy.

- The tests cases selected represent a wide variety of transfers between planets. However, transfers between several asteroids were not considered due to the limited number of cases that could be tested in the available time. However, the method presented can also be used for those missions. This is an interesting topic for future research.

List of References

[1] D. M. Novak and M. Vasile, *Improved Shaping Approach to the Preliminary Design of Low-Thrust Trajectories,* Journal of Guidance, Control, and Dynamics **34**, 128 (2011).

[2] C. H. Yam, D. Izzo, and F. Biscani, *Towards a High Fidelity Direct Transcription Method for Optimisation of Low-Thrust Trajectories,* in *4th International Conference on Astrodynamics Tools and Techniques, 2010* (2010) pp. 1–7.

[3] A. Mereta and D. Izzo, *Target selection for a small low-thrust mission to near-Earth asteroids,* ArXiv e-prints , 1 (2018), arXiv:1802.02418 [astro-ph.EP] .

[4] H. D. Curtis, *Orbital Mechanics for Engineering Students*, 3rd ed., edited by H. D. Curtis (Butterworth-Heinemann, Boston, 2014).

[5] T. Roegiers, *Application of the Spherical Shaping Method to a Low-Thrust Multiple Asteroid Rendezvous Mission*, Master thesis, Delft Univeristy of Technology (2014).

[6] J. Benkhoff, J. van Casteren, H. Hayakawa, M. Fujimoto, H. Laakso, M. Novara, P. Ferri, H. R. Middleton, and R. Ziethe, *BepiColombo-Comprehensive exploration of Mercury: Mission overview and science goals,* Planetary and Space Science **58**, 2 (2010).

[7] M. D. Rayman, P. Varghese, D. H. Lehman, and L. L. Livesay, *Results from the Deep Space 1 technology validation mission,* Acta Astronautica **47**, 475 (2000).

[8] J. Kawaguchi, A. Fujiwara, and T. Uesugi, *Hayabusa-Its technology and science accomplishment summary and Hayabusa-2,* Acta Astronautica **62**, 639 (2008).

[9] C. T. Russell and C. A. Raymond, *The Dawn Mission to Vesta and Ceres,* Space Science Reviews **15**, 3 (2012).

[10] M. Vasile and P. De Pascale, *Preliminary Design of Multiple Gravity-Assist Trajectories,* Journal of Spacecraft and Rockets **43**, 1065 (2006).

[11] G. Radice, M. Vasile, and P. Dilizia, *Advanced Global Optimisation Tools for Mission Analysis and Design,* in *2nd Workshop on Astrodynamic Tools and Techniques*, Vol. 31 (ESTEC, 2004).

[12] D. Izzo, M. Märtens, and B. Pan, *A survey on artificial intelligence trends in spacecraft guidance dynamics and control,* Astrodynamics **3**, 287 (2019), arXiv:1812.02948 .

[13] D. Hennes, D. Izzo, and D. Landau, *Fast approximators for optimal low-thrust hops between main belt asteroids,* EEE Symposium Series on Computational Intelligence (2016), 10.1109/SSCI.2016.7850107.

[14] A. Mereta, D. Izzo, and A. Wittig, *Machine learning of optimal low-thrust transfers between near-earth objects,* Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **10334 LNCS**, 543 (2017).

[15] B. Dachwald, *Optimization of very-low-thrust trajectories using evolutionary neurocontrol,* Acta Astronautica **57**, 175 (2005).

[16] C. Sánchez-Sánchez and D. Izzo, *Real-Time Optimal Control via Deep Neural Networks: Study on Landing Problems,* Journal of Guidance, Control, and Dynamics **41**, 1122 (2018).

[17] D. Izzo, C. Sprague, and D. Tailor, *Machine learning and evolutionary techniques in interplanetary trajectory design,* ArXiv e-prints , 1 (2018), arXiv:1802.00180 .

[18] E. D. Mendez Ramos, P. Mishra, S. Edwards, and D. Mavris, *Response Surface Regressions for Low-Thrust Interplanetary Mission Design,* in *AIAA SPACE*, AIAA SPACE Forum (American Institute of Aeronautics and Astronautics, 2016).

[19] L. Stubbig, *Investigating the use of neural network surrogate models in the evolutionary optimization of interplanetary low-thrust trajectories,* Master thesis, Delft University of Technology (2019).

[20] K. Cowan, L. Bouwman, and Y. Liu, *Gaussian Process models for preliminary low-thrust trajectory optimization,* in *Astrodynamics Specialist Conference* (Portland, ME, 2019).

[21] C. Ampatzis and D. Izzo, *Machine learning techniques for approximation of objective functions in trajectory optimisation,* Proceedings of the International Joint Conference on Artificial Intelligence 2009 Workshop on Artificial Intelligence in Space **2009** (2009).

[22] H. Shang and Y. Liu, *Assessing Accessibility of Main-Belt Asteroids Based on Gaussian Process Regression,* Journal of Guidance, Control, and Dynamics **40**, 1144 (2017).

[23] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, *Extreme learning machine: Theory and applications,* Neurocomputing **70**, 489 (2006), arXiv:1311.4555 .

[24] G.-B. Huang, L. Chen, and C.-K. Siew, *Universal approximation using incremental constructive feedforward networks with random hidden nodes.* IEEE transactions on neural networks **17**, 879 (2006).

[25] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, *Extreme Learning Machine for Regression and Multiclass Classification,* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) **42**, 513 (2012).

[26] J. Gama, R. Sebastião, and P. P. Rodrigues, *On evaluating stream learning algorithms,* Machine Learning **90**, 317 (2013).

[27] N.-Y. Liang, G.-B. Huang, P. Saratchandran, and N. Sundararajan, *A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks,* IEEE Transactions on Neural Networks **17**, 1411 (2006), arXiv:1407.1911 .

[28] C. T. Yaw, S. Y. Wong, K. S. Yap, and C. H. Tan, *Extreme Learning Machine neural networks for multi-agent system in power generation,* International Journal of Engineering and Technology (UAE) **7**, 347 (2018).

[29] A. E. Petropoulos, J. M. Longuski, and W. Lafayette, *Automated design of low-thrust gravity-assist trajectories,* in *Astrodynamics Specialist Conference*, Guidance, Navigation, and Control and Co-located Conferences No. c (American Institute of Aeronautics and Astronautics, 2000).

[30] A. E. Petropoulos and J. M. Longuski, *Shape-Based Algorithm for the Automated Design of Low-Thrust, Gravity Assist Trajectories,* Journal of Spacecraft and Rockets **41**, 787 (2004).

[31] D. Izzo, V. M. Becerra, D. R. Myatt, S. J. Nasuto, and J. M. Bishop, *Search space pruning and global optimisation of multiple gravity assist spacecraft trajectories,* Journal of Global Optimization **38**, 283 (2007).

[32] O. Schütze, M. Vasile, O. Junge, M. Dellnitz, and D. Izzo, *Designing optimal low-thrust gravity-assist trajectories using space pruning and a multi-objective approach,* Engineering Optimization **41**, 155 (2009).

[33] T. J. Debban, T. Troy McConaghy, and J. M. Longuski, *Design and optimization of low-thrust gravity-assist trajectories to selected planets,* AIAA/AAS Astrodynamics Specialist Conference and Exhibit , 1 (2002).

[34] C. H. Yam, T. T. Mcconaghy, K. J. Chen, and J. M. Longuski, *Design of Low-Thrust Gravity-Assist Trajectories to the Outer Planets,* in *55th International Astronautical Congress of the International Astronautical Federation, the International Academy of Astronautics, and the International Institute of Space Law*, International Astronautical Congress (IAF) (American Institute of Aeronautics and Astronautics, 2004) pp. 1–16.

[35] M. Okutsu, C. H. Yam, and J. M. Longuski, *Low-Thrust Trajectories to Jupiter via Gravity Assists from Venus, Earth, and Mars,* in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, Guidance, Navigation, and Control and Co-located Conferences No. August 2006 (American Institute of Aeronautics and Astronautics, 2006) pp. 1–14.

[36] P. De Pascale and M. Vasile, *Preliminary Design of Low-Thrust Multiple Gravity-Assist Trajectories,* Journal of Spacecraft and Rockets **43**, 1065 (2006).

[37] M. Vasile and S. Campagnola, *Design of low-thrust gravity assist trajectories to Europa,* Journal of the British Interplanetary Society **62**, 15 (2009).

[38] M. Vasile and F. Bernelli-Zazzera, *Optimizing low-thrust and gravity assist maneuvers to design interplanetary trajectories,* eprint arXiv:1105.1829 , arXiv:1105.1829 (2011).

[39] H. X. Shen, J. P. Zhou, Q. B. Peng, H. Y. Li, and J. T. Li, *Multi-objective interplanetary trajectory optimization combining low-thrust propulsion and gravity-assist maneuvers,* Science China Technological Sciences **55**, 841 (2012).

[40] L. Casalino, G. Colasurdo, and D. Pastrone, *Optimal Low-Thrust Escape Trajectories Using Gravity Assist,* Journal of Guidance, Control, and Dynamics **22**, 637 (1999).

[41] J. A. Sims and S. N. Flanagan, *Preliminary design of low-thrust interplanetary missions,* Advances in the Astronautical Sciences **103**, 583 (2000).

[42] J. A. Englander and B. A. Conway, *Automated solution of the low-thrust interplanetary trajectory problem,* Journal of Guidance, Control, and Dynamics **40**, 15 (2017).

[43] D. W. Parcher and J. A. Sims, *Earth gravity-assist trajectories to Jupiter using nuclear electric propulsion,* Advances in the Astronautical Sciences **123 III**, 2321 (2006).

[44] F. Jiang, H. Baoyin, and J. Li, *Practical Techniques for Low-Thrust Trajectory Optimization with Homotopic Approach,* Journal of Guidance, Control, and Dynamics **35**, 245 (2012).

[45] H. Yang, J. Li, and H. Baoyin, *Low-cost transfer between asteroids with distant orbits using multiple gravity assists,* Advances in Space Research **56**, 837 (2015).

[46] H. D. Curtis, *Interplanetary Trajectories,* in *Orbital Mechanics for Engineering Students*, edited by H. D. Curtis (Butterworth-Heinemann, Boston, 2014) 3rd ed., Chap. 8, pp. 429–484.

[47] H. R. Lourenço, O. C. Martin, and T. Stützle, *Iterated Local Search,* in *Handbook of Metaheuristics*, edited by F. Glover and G. A. Kochenberger (Springer US, Boston, MA, 2003) pp. 320–353.

[48] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Optimization by Simulated Annealing,* Science **220**, 671 (1983).

[49] R. Eberhart and J. Kennedy, *A new optimizer using particle swarm theory,* Proceedings of the Sixth International Symposium on Micro Machine and Human Science , 39 (1995).

[50] R. Storn and K. Price, *Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces,* Journal of Global Optimization **11**, 341 (1997).

[51] M. Pontani and B. A. Conway, *Particle swarm optimization applied to space trajectories,* Journal of Guidance, Control, and Dynamics **33**, 1429 (2010).

[52] W. Yao, J. Luo, M. Macdonald, M. Wang, and W. Ma, *Improved differential evolution algorithm and its applications to orbit design,* Journal of Guidance, Control, and Dynamics **41**, 935 (2018).

[53] A. D. Olds, C. A. Kluever, and M. L. Cupples, *Interplanetary mission design using differential evolution,* Journal of Spacecraft and Rockets **44**, 1060 (2007).

[54] M. R. Sentinella and L. Casalino, *Hybrid evolutionary algorithm for the optimization of interplanetary trajectories,* Journal of Spacecraft and Rockets **46**, 365 (2009).

[55] D. Izzo, L. F. Simões, M. Märtens, G. C. De Croon, A. Heritier, and C. H. Yam, *Search for a grand tour of the Jupiter Galilean moons,* GECCO 2013 - Proceedings of the 2013 Genetic and Evolutionary Computation Conference , 1301 (2013).

[56] A. Shirazi, J. Ceberio, and J. A. Lozano, *Evolutionary algorithms to optimize low-thrust trajectory design in spacecraft orbital precession mission,* in *2017 IEEE Congress on Evolutionary Computation, CEC 2017 - Proceedings* (2017) pp. 1779–1786.

[57] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Zumer, *Self-Adapting Control Parameters in Differential Evolution : A Comparative Study on Numerical,* IEEE Transactions on Evolutionary Computation **10**, 646 (2006).

[58] S. M. Elsayed, R. A. Sarker, and D. L. Essam, *Differential evolution with multiple strategies for solving CEC2011 real-world numerical optimization problems,* 2011 IEEE Congress of Evolutionary Computation, CEC 2011 , 1041 (2011).

[59] S. Das and P. N. Suganthan, *IEEE Trans. Evolutionary Computation*, Vol. 15 (2011) pp. 4–31.

[60] C. H. Yam, D. Izzo, and D. D. Lorenzo, *Low-thrust trajectory design as a constrained global optimization problem,* Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering **225**, 1243 (2011).

[61] K. Alemany and R. D. Braun, *Survey of Global Optimization Methods for Low-Thrust , Multiple Asteroid Tour Missions,* AAS/AIAA Space Flight Mechanics Meeting January 2007, Sedona, Arizona , 1 (2007).

[62] R. Bertrand and R. Epenoy, *New smoothing techniques for solving bang–bang optimal control problems—numerical results and statistical interpretation,* Optimal Control Applications and Methods **23**, 171 (2002).

[63] J. Gil-fernandez and M. A. Gomez-tierno, *Practical Method for Optimization of Low-Thrust Transfers,* **33**, 3 (2010).

[64] B. Pan, X. Pan, and P. Lu, *Finding Best Solution in Low-Thrust Trajectory Optimization by Two-Phase Homotopy,* Journal of Spacecraft and Rockets , 1 (2018).

[65] T. Guo, F. Jiang, and J. Li, *Homotopic approach and pseudospectral method applied jointly to low thrust trajectory optimization,* Acta Astronautica **71**, 38 (2012).

[66] T. Hastie, R. Tibshirani, and J. Friedman, *Elements of Statistical Learning*, 2nd ed., Springer series in statistics (Springer,, New York, 2009) pp. 1–745, arXiv:1010.3003 .

[67] E. Keogh and A. Mueen, *Curse of Dimensionality,* (2017).

[68] B. Wall, *Shape-Based Approximation Method for Low-Thrust Trajectory Optimization,* AIAA/AAS Astrodynamics Specialist Conference and Exhibit , 1 (2008).

[69] B. D. E. Vogeleer, *Automatic and Fast Generation of Sub-optimal and Feasible Low-Thrust Trajectories Using a Boundary-Value Pseudo-Spectral Method*, Master thesis, Delft Univeriesty of Technology (2008).

[70] O. Abdelkhalik and E. Taheri, *Approximate On-Off Low-Thrust Space Trajectories Using Fourier Series,* Journal of Spacecraft and Rockets **49**, 962 (2012).

[71] D. J. Gondelach and R. Noomen, *Hodographic-Shaping Method for Low-Thrust Interplanetary Trajectory Design,* Journal of Spacecraft and Rockets **52**, 728 (2015).

[72] E. Taheri and O. Abdelkhalik, *Fast Initial Trajectory Design for Low-Thrust Restricted-Three-Body Problems,* Journal of Guidance, Control, and Dynamics **38**, 2146 (2015).

[73] K. Zeng, B. Wu, and Y. Geng, *Two-phase shaping approach to low-thrust trajectories design between coplanar orbits,* Advances in Space Research **62**, 593 (2018).

[74] E. Taheri, I. Kolmanovsky, and E. Atkins, *Shaping low-thrust trajectories with thrust-handling feature,* Advances in Space Research **61**, 879 (2018).

[75] E. Taheri and O. Abdelkhalik, *Initial three-dimensional low-thrust trajectory design,* Advances in Space Research **57**, 889 (2016).

[76] D. M. Novak, *Methods and tools for preliminary low thrust mission analysis*, Phd, University of Glasgow (2012).

[77] A. C. Hindmarsh, *ODEPACK, A Systematized Collection of ODE Solvers,* IMACS Transactions on Scientific Computation **1**, 55 (1983).

[78] J. T. Betts, *Very low-thrust trajectory optimization using a direct SQP method,* Journal of Computational and Applied Mathematics **120**, 27 (2000).

[79] J. Betts and S. Erb, *Optimal Low Thrust Trajectories to the Moon,* SIAM Journal on Applied Dynamical Systems **2**, 144 (2003).

[80] C. H. Yam and J. M. Longuski, *Reduced parameterization for optimization of low-thrust gravity-assist trajectories: Case studies,* Collection of Technical Papers - AIAA/AAS Astrodynamics Specialist Conference, 2006 **3**, 1842 (2006).

[81] F. Topputo and C. Zhang, *Survey of direct transcription for low-thrust space trajectory optimization with applications,* Abstract and Applied Analysis (2014), 10.1155/2014/851720.

[82] G. J. Whiffen and J. A. Sims, *Application of the SDC optimal control algorithm to low-thrust escape and capture trajectory optimization,* Advances in the Astronautical Sciences **112 II**, 1361 (2002).

[83] J. Sims, P. Finlayson, E. Rinderle, M. Vavrina, and T. Kowalkowski, *Implementation of a Low-Thrust Trajectory Optimization Algorithm for Preliminary Design,* AIAA/AAS Astrodynamics Specialist Conference and Exhibit , 1 (2006).

[84] D. Izzo, *esa/pykep: The Gym and more.* (2019), 10.5281/ZENODO.3560630.

[85] N. Kuntikal, D. Mahadevaiah, and N. Kuntikal Doddi Mahadevaiah, *Analysis and Evaluation of Global Optimisation Algorithms for Gravity Assist Sequencing of Low Thrust Missions* (2017).

[86] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT Press, 2016) arXiv:arXiv:1011.1669v3 .

[87] Y. Bengio, *Practical recommendations for gradient-based training of deep architectures,* in *Neural Networks: Tricks of the Trade: Second Edition*, Vol. 7700 LECTU, edited by G. Montavon, G. B. Orr, and K.-R. Müller (Springer Berlin Heidelberg, Berlin, Heidelberg, 2012) 2nd ed., Chap. 19, pp. 437–478, arXiv:1206.5533 .

[88] J. Racine, *Consistent cross-validatory model-selection for dependent data: hv-block cross-validation,* Journal of Econometrics **99**, 39 (2000).

[89] V. Cerqueira, L. Torgo, and I. Mozetic, *Evaluating time series forecasting models: An empirical study on performance estimation methods,* , 1 (2019), arXiv:1905.11744 .

[90] J. M. H. Claes, *Thermosphere Modelling Using Machine Learning*, Master thesis, Delft University of Technology (2019).

[91] S. Theodoridis and K. Koutroumbas, *Classifiers Based on Bayes Decision Theory,* in *Pattern Recognition* (2009) Chap. 2, pp. 13–89.

[92] C. M. Bishop, *Oxidation Communications*, Vol. 27 (Springer-Verlag, Berlin, Heidelberg, 2006) pp. 140–147.

[93] M. V. Shcherbakov, A. Brebels, N. L. Shcherbakova, A. P. Tyukov, T. A. Janovsky, and V. A. evich Kamaev, *A survey of forecast error measures,* World Applied Sciences Journal **24**, 171 (2013).

[94] X. Glorot and Y. Bengio, *Understanding the difficulty of training deep feedforward neural networks,* Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics **9**, 249 (2010), arXiv:arXiv:1011.1669v3 .

[95] X. Glorot, A. Bordes, and Y. Bengio, *Deep sparse rectifier neural networks,* AISTATS '11: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics **15**, 315 (2011), arXiv:1502.03167 .