Gaussian Process Models for Preliminary Low-Thrust Trajectory Optimization

Master Thesis

Lieve Bouwman

Delft University of Technology

TU Delft

Gaussian Process Models for Preliminary Low-Thrust Trajectory Optimization

by

Lieve Bouwman

Dissertation for the degree of

Master of Science

at The Astrodynamics and Space Missions Section,
Delft University of Technology,
to be defended publicly on Friday, July 5th, 2019.

Supervisors:	Ir. K. J. Cowan	
	Ir. Y. Liu	
Thesis committee:	Prof. dr. ir. P. N. A. M. Visser,	Committee chair
	Ir. K. J. Cowan,	Supervisor
	Ir. Y. Liu,	Supervisor
	Dr. ir. A. Menucucci,	External examinator

This thesis is confidential and cannot be made public until 05-07-2021

TU Delft

Contents

Preface

This document is submitted to obtain the degree of Master of Science at the Delft University of Technology, faculty of Aerospace Engineering. It marks the end of my student time and my time at the Delft University of Technology, which I have enjoyed a lot. The work presented proposes a robust method to develop Gaussian process models for the preliminary optimization of low-thrust trajectories based on exponential sinusoid shaping. To the best of my knowledge, the presented work is original and might provide a useful scientific contribution to the design of (low-thrust) trajectories. As I would not have produced the same results without the support of others, I would like to thank my supervisors Ir. Kevin Cowan and PhD candidate Yuxin Liu. They have supported, adviced, and encouraged me during the past nine months. I feel very lucky with the amount of time they reserved for our weekly meetings. Due to their enthusiasm and positivity, I enjoyed working on my thesis project. As Yuxin is working himself on Gaussian processes as well, he has been able to provide me useful insights on this machine learning technique. Due to the wide experience Kevin has gained during his career, his strong reasoning and analytic insights have assisted me to improve the quality of my work. I feel exited for the presentation of our work at the Astrodynamics Specialist Conference and potentially the submission of the work to a journal.

In addition, I want to thank my family and friends who have always been supportive during my studies. After obtaining my BSc degree in Mechanical Engineering, I decided to switch to Aerospace Engineering, which has been one of the best choices I have ever made. I have always been intrigued by space missions, and I very much enjoyed the significant amount of knowledge and skills I gained during the past two years at the faculty of Aerospace Engineering. Furthermore, I feel grateful for all opportunities that I have received during my studies to be part of several student boards, committees and initiatives, and the opportunity to do three different internships, which have learned me to broaden my view and strengthen my skills on different dimensions.

Finally, I wish to express my gratitude towards all authors that are working on or have previously worked on low-thrust trajectory optimization and Gaussian process models. In the past months I have become very enthusiastic about the application of machine learning techniques, in particular Gaussian processes, on the complex optimization problems related to low-thrust mission design. I am both curious and exited to find out what the future will hold for the role of Gaussian processes in (low-thrust) trajectory optimization. Hopefully, in a couple of years, Gaussian process models will become the norm for (low-thrust) trajectory optimization and will play an important role in bringing space tourism or colonization to reality.

Lieve Bouwman
Delft, June 2019

Abstract

Low-thrust trajectories can benefit the search for propellant-optimal trajectories, but increases in modeling complexity and computational load remain a challenge for efficient mission design and optimization. In this work, a procedure for developing models utilizing Gaussian process regression (GPR) and classification (GPC) is proposed to perform computationally efficient optimization while obtaining acceptable accuracies for trajectories based on exponential sinusoid shaping. In the field of machine learning, Bayesian inference models based on a Gaussian process (GP) are a flexible and computationally efficient tool to infer target values given some observations. The goal of this work is to predict a combination of values of input variables (i.e. the observations) which correspond to a shape-based trajectory with the smallest total velocity increment (ΔV) or smallest propellant mass fraction (J_m). In order to build the GP models, training samples are generated using an exponential sinusoid shape to approximate low-thrust trajectories for the computation of direct interplanetary transfers. A GPC model is constructed to assess whether a given combination of values for a three-dimensional input vector corresponds to a feasible trajectory. A GPR model is then developed to predict the total ΔV or J_m corresponding to the input vectors resulting in feasible trajectories, thereby replacing the entire computation of the exposin shape and the integration along the shape. To investigate the applicability of the GP based method, GPR models, which map a six-dimensional input vector containing shape information to a ΔV or J_m value, are developed, thereby replacing only the integration along the shape.

In order to develop a GP model that fits the problem at hand, the underlying functions and parameters should be selected rationally. In this work, a novel model development procedure is proposed to ensure that the mean function, covariance function, likelihood function, inference method, and hyperparameters, which dominate the performance of the models, are chosen rationally in terms of mean absolute percentage error (MAPE) and prediction time. The development procedure consists of three parts. First, candidates which are promising in terms of MAPE are separately sought for the mean function, covariance function, likelihood function, and inference method. Next, these candidates are combined into several competing GP models, from which the GP model resulting in the smallest MAPE is chosen. Finally, the hyperparameters are optimized using the conjugent gradient method. In each part of the development procedure, k-fold cross validation is used to increase the robustness of the model validation. To assess the validity of the choices made during the proposed model development procedure, a robustness analysis is performed. Using the procedure outlined above, five GP models are constructed and presented in this work. For each model, the number of training samples is determined iteratively to strike a balance between accuracy and prediction time. The models are tested on transfer trajectories from Earth to Mars and Ceres and from Mars to Earth, and their performance, in terms of MAPE and processing time, is compared to that of more common optimization techniques such as differential evolution in combination with the exponential sinusoid and other shape-based methods.

The results demonstrate that the computation time can significantly be reduced while achieving promising MAPE's, especially when the goal is to locate regions of feasible or near-optimal trajectories. When assessing 1,000,000 candidate input vectors, efficiency improvements of 150 times could be obtained using prediction instead of trajectory computation. This speed advantage increases for an increasing number of candidates. The robustness analysis performed here provides confidence in the choices made during the model development procedure. Furthermore, it is found that the model which maps three variables directly to a ΔV or J_m value performs better than the one trained with full shape information, which demonstrates the strength of GP models as applied to low-thrust trajectory optimization. The developed GP models for the test case from Earth to Mars are tested for their dependency on this mission scenario, and comparable results in terms of MAPE and prediction time are achieved when the models are applied to a mission scenario with a different or smaller range for departure date (t_0) and/or time-of-flight (TOF) or a different target planet than defined for the Earth-Mars mission scenario as presented in this work.

The work provided in this paper provides other researchers in the field of astrodynamics with a procedure necessary to apply GP models on low-thrust trajectory design and a reference against which to validate their

performance. In addition, given the promising performance as described in this work, researchers are encouraged to apply GP models to related applications in the field of astrodynamics, such as high-thrust trajectories including gravity assists or the combinatorial problem in multiple gravity-assist mission planning. To the best of the authors' knowledge, the GP-based modeling procedure presented in this paper is the first to be applied to low-thrust interplanetary trajectory design.

Nomenclature

Note: this nomenclature applies to the appendices and is an addition to the nomenclature provided in the conference paper in Chapter 2.

Abbreviations and acronyms

AGS	= adaptive grid search
CPU	= computation
DE	= differential evolution
GP	= Gaussian process
GPC	= Gaussian process classification
GPR	= Gaussian process regression
GS	= grid search
NEP	= nuclear electric propulsion
SEP	= solar electric propulsion
TOF	= time-of-flight

Symbols

$\boldsymbol{\theta}$	= vector of hyperparameters
$\boldsymbol{X}$	= training input matrix
$\boldsymbol{x}$	= input vector
$\boldsymbol{Y}$	= training output vector
ΔV	= velocity increment [km/s]
$\mu(\boldsymbol{x})$	= mean function
θ	= polar angle [rad]
ε	= error measure
a	= normalized thrust acceleration [-]
C_r	= crossover probability
F	= thrust acceleration [m/s^2]
F_p	= mutation probability
g_0	= standard gravitational acceleration [m/s^2]
I_{sp}	= specific impulse [s]
J_m	= propellant mass fraction [-]
$K(\boldsymbol{X},\boldsymbol{X})$	= covariance matrix
$k(\boldsymbol{x},\boldsymbol{x}^*)$	= covariance function
k_0	= scaling factor [km]
k_1	= dynamic range parameter [-]
k_2	= winding parameter [-]
N	= number of revolutions
r	= radial distance [km]
t	= time [s]
V	= velocity vector [km/s]

Subscripts

0	= initial value
f	= final value
r	= radial component
t	= tangential component
ard	= automatic relevance determination
iso	= isotropic length-scale
LT	= low-thrust
PP	= piecewise polynomial
RQ	= rational quadratic

SE	= squared exponential

Superscripts

$\ddot{\square}$	= second derivative with respect to time t
$\dot{\square}$	= derivative with respect to time t
$\square^*$	= prediction data from Gaussian process
$\square^T$	= transpose of a vector

1 Introduction

In the design of space missions, it is a major challenge to bring significant amounts of payload mass to planets in our Solar System. In the last decades, electric propulsion systems have proven to be successful during space missions, which allow to increase the payload mass by reducing propellant mass [26] [27]. Electric propulsion systems are characterized by their very low levels of thrust, which result in the necessity of long thrusting periods during interplanetary flight to achieve sufficient ΔV. As those propulsion systems are highly efficient, these ΔV's could be obtained using limited propellant mass. The continuous-thrusting related to low-thrust missions increases the complexity of the dynamics of the spacecraft. As a consequence, the optimization of low-thrust trajectories becomes an actual challenge. To keep the time related to optimization of low-thrust trajectories limited, preliminary optimization, where moderate accuracy is generally accepted in return for fast computation, becomes important. The trajectories found during preliminary optimization could later on be used as initial guess for more refined optimization of the trajectory. In the past years, several analytic shape-based methods have been developed for the preliminary design of low-thrust trajectories [23] [5] [20] [8]. Using a shape-based method, the trajectory is assumed to follow a specific shape.

All shape-based methods have in common that they contain free parameters, which should be optimized to obtain optimal trajectories. Usually, trajectories are optimized for minimum ΔV, minimum propellant mass, minimum TOF, or a combination of them. Finding the globally optimal trajectory, i.e. the overall best trajectory, can be very challenging. Many optimization techniques have been developed to find global optima. Several different ways to categorize them could be distinguished, e.g. deterministic versus heuristic [10]. Each of these methods has its own characteristics regarding computation (CPU) speed and robustness (i.e. their ability to converge to global optima). A similarity between these optimization techniques is the necessity of trajectory propagation and integration. Starting from a set of initial conditions, first propagation is performed to determine the state of the spacecraft at any instance of time (i.e the trajectory). Integration along the shape is required to define the TOF and the efficiency (e.g. the required ΔV). As opposed to these optimization techniques, the entire trajectory computation could be replaced by prediction making use of machine learning algorithms. Using prediction, propagation and integration steps are no longer required, and the goal is to predict trajectory efficiency from a set of initial conditions. The main advantage of prediction is the significant efficiency improvements that could be obtained. Machine learning algorithms could be divided into several types: supervised learning, unsupervised learning, reinforcement learning, feature learning, anomaly detection, sparse dictionary learning, and association rules [14]. GPR and GPC are types of supervised learning, where a set of data contains both the desired inputs and outputs. A function able to predict the efficiency of trajectories from initial conditions should be learned from example trajectories.

To address the applicability of GP models on low-thrust trajectory optimization, the main part of this thesis report is written as a conference paper manuscript, which will be submitted to the Astrodynamics Specialist Conference, hosted by the American Astronautical Society (AAS) and cohosted by the American Institute of Aeronautics and Astronautics (AIAA). It is intended to be submitted to a journal afterwards. The conference paper is named as follows:

"Gaussian Process Models for Preliminary Low-Thrust Trajectory Optimization"

Chapter 2 covers the manuscript of the paper. This paper includes another abstract and introduction, after which the modeling of low-thrust trajectories according to the exponential sinusoid is explained. Next, the GP-based method, together with the proposed model development procedure, is presented. This is followed by the test cases against which the method is tested, the selected GP models, and the results obtained. Finally, the most important conclusions are discussed. Chapter 2 is followed by several appendices that support the work that has been done, but will not be included in the conference paper itself.

2

Conference paper

.

Gaussian Process Models for Preliminary Low-Thrust Trajectory Optimization

L. Bouwman*, Y. Liu†, and K. J. Cowan‡
Delft University of Technology, 2629 HS Delft, The Netherlands

Abstract*—Low-thrust trajectories can benefit the search for propellant-optimal trajectories, but increases in modeling complexity and computational load remain a challenge for efficient mission design and optimization. In this paper, an approach for developing models utilizing Gaussian Process (GP) regression and classification is proposed to perform computationally efficient optimization while obtaining acceptable accuracies for trajectories based on exponential sinusoid shaping. The goal of this work is to predict a combination of values of input variables which corresponds to a shape-based trajectory with the smallest total velocity increment (ΔV) or propellant mass fraction (J_m). A GP classification model is constructed to assess whether a given combination of values of input variables corresponds to a feasible trajectory. GP regression models are developed to predict the total ΔV and J_m corresponding to a combination of shape parameters, which can replace the required integration along the shape. In addition, advanced regression models are developed to predict the target values while requiring only three input parameters, thereby replacing the entire shape computation. In order to develop a GP model that fits the problem at hand, the underlying functions and parameters should be selected rationally. In this work, a novel model development approach is proposed to ensure that the mean function, covariance function, likelihood function, inference method, and hyperparameters, which dominate the performance of the models, are chosen rationally in terms of mean absolute percentage error (MAPE) and prediction time. Using this approach, GP models are developed and tested on transfer trajectories from Earth to Mars and Ceres, and from Mars to Earth, and their performance, in terms of MAPE and prediction time, is compared to that of more common optimization techniques in combination with the exponential sinusoid and other shape-based methods. The results demonstrate that the computation time can significantly be reduced while achieving promising MAPE's, especially when the goal is to locate regions of feasible or near-optimal trajectories. The proposed model development procedure is tested for robustness, which provides confidence in the proposed approach. Furthermore, it is found that the models which map three input variables directly to a ΔV or J_m value perform better than the ones trained with shape information, which demonstrates the strength of GP models as applied to low-thrust trajectory optimization.

Nomenclature

Symbols

α	= angle of attack [rad]
$\boldsymbol{\theta}$	= vector of hyperparameters
$\boldsymbol{X}$	= training input matrix
$\boldsymbol{x}$	= input vector
$\boldsymbol{Y}$	= training output vector
ΔV	= velocity increment [km/s]
γ	= flight-path angle [rad]
μ	= gravitational parameter [m^3/s^2]
$\mu(\boldsymbol{x})$	= mean function
ϕ	= phase angle [rad]
ψ	= transfer angle [rad]
θ	= polar angle [rad]
ε	= error measure
a	= normalized thrust acceleration [-]
C_r	= crossover probability
F	= thrust acceleration [m/s^2]
F_p	= mutation probability
g_0	= standard gravitational acceleration [m/s^2]
I_{sp}	= specific impulse [s]
J_m	= propellant mass fraction [-]
$K(\boldsymbol{X},\boldsymbol{X})$	= covariance matrix
$k(\boldsymbol{x},\boldsymbol{x}^*)$	= covariance function
k_0	= scaling factor [km]
k_1	= dynamic range parameter [-]
k_2	= winding parameter [-]
M	= training set size
N	= number of revolutions
r	= radial distance [km]
t	= time [s]
V	= velocity vector [km/s]

Subscripts

0	= initial value
f	= final value
r	= radial component
t	= tangential component
LT	= low-thrust
MAE	= mean absolute error
MAPE	= mean absolute percentage error
RMSE	= root mean square error

Superscripts

$\ddot{\square}$	= second derivative with respect to time t
$\dot{\square}$	= derivative with respect to time t
$\square^*$	= prediction data from Gaussian Process
$\square^T$	= transpose of a vector

I Introduction

Trajectories flown with low-thrust rocket engines, such as ion propulsion systems, are attractive due to their propellant efficiency and high reliability [1]. This becomes especially important for interplanetary missions where large velocity changes are generally required. To obtain ΔV values required for interplanetary flight, the engine has to operate for thousands of hours, which complicates

*Graduate Student, Faculty of Aerospace Engineering, Kluyverweg 1; lievebouwman@gmail.com.
†Ph.D. Candidate, Faculty of Aerospace Engineering, Kluyverweg 1; Yuxin.Liu@tudelft.nl.
‡Education Fellow and Lecturer, Faculty of Aerospace Engineering, Kluyverweg 1; K.J.cowan@tudelft.nl.

the dynamics of the vehicle and results in challenging trajectory design and optimization. Most of the methods that are used to solve this optimization challenge for low-thrust trajectories are based on a non-linear programming problem. Although these methods are successful in the detailed design of a trajectory, they fail to efficiently explore a large number of candidate trajectories. For the preliminary design of trajectories, analytical methods have proven to be powerful tools. As first introduced by Petropoulos [2], popular analytical methods make use of shapes to approximate the spacecraft's trajectory. The shape-based method designed by Petropoulos makes use of an exponential sinusoid and is, due to its wide applicability, a well-known shape-based method. A method to generalize the implementation of the exponential sinusoid by using the multi-revolution Lambert's problem was proposed by Izzo [3]. The resulting algorithm is able to efficiently locate near-optimal trajectories, but a time-consuming search process is required to locate these trajectories. Depending on the shape-based method and the computer used, most shapes take about 0.3-2 s computation (CPU) time per trajectory [1], which may result in a rather slow process when a large search space needs to be investigated. Therefore, with the increasing interest for low-thrust trajectories, there is the necessity of a fast calculation when exploring a search space for a specific mission.

Gaussian Process regression (GPR) and classification (GPC) are useful techniques to predict a large number of target values within limited CPU time. D.G. Krige [4] was the first to propose the GPR method with the goal to estimate the distribution of gold. Since then, GPR has been applied four times in the field of aerospace: 1) to evaluate the aerodynamic coefficient of a spaceplane [5], 2) to assess the accessibility of main-belt asteroids [6], 3) to model the gravity field of small bodies [7] and 4) to evaluate the Mars entry terminal state [8]. GPC has not been used before to classify problems related to aerospace. The main advantage of a method based on a GP is that it is flexible and computationally efficient to handle the relation between inputs and outputs in both regression and classification problems [6]. As a GP model is built up of functions, in theory an unlimited number of different models could be developed. Generally, the GP model for a specific problem is selected by trial and error [6] or based on models used for comparable problems, leaving the functionality of the model as a black box. In this paper, a procedure for developing models using a GP is proposed to further speed up the preliminary optimization of low-thrust trajectories based on exponential sinusoid shaping. The robustness of the choices made in this development procedure is assessed.

This paper starts with discussing the implementation of the exponential sinusoid according to the multi-revolution Lambert's problem, with the purpose to generate training data for the GP models. Next, the basics of GPR and GPC will be discussed, together with the proposed development procedure for models based on a GP. It will be explained how the performance of the models and the robustness of the model development procedure will be assessed. Models have been developed for three mission test cases. The ones for the mission test case from Earth to Mars are presented and their robustness is measured. The developed models for the other two test cases are presented in the wppendix. The performance reached for each of the test cases, in terms of accuracy, CPU time, and the ability to locate regions of near-optimal trajectories, is discussed, followed by the conclusions.

II Modeling Low-Thrust Trajectories Using the Exponential Sinusoid

A. Theory of the Exponential Sinusoid

We will start by looking at the equations of motion of a point mass in a Newtonian gravity field when some thrust is acting on the particle, given in polar coordinates [9]:

$$\begin{cases} \ddot{r} - r\dot{\theta}^2 + \mu/r^2 = F\sin\alpha \\ \ddot{\theta}r + 2\dot{\theta}\dot{r} = F\cos\alpha \end{cases} \tag{1}$$

where μ is the gravitational parameter of the central body spacecraft system, F is the magnitude of the thrust acceleration, and α is the angle of attack. F can be normalized by the local gravitational acceleration using Equation 2.

$$a \equiv F/(\mu/r^2) \tag{2}$$

The general equation of a pure exponential sinusoid (exposin) is defined by [2]:

$$r = k_0 \exp\left[k_1 \sin\left(k_2\theta + \phi\right)\right] \tag{3}$$

where k_0, k_1, k_2 and ϕ are constants. k_0 is a scaling factor. k_1 is the dynamic range parameter, which controls the ratio of apoapsis distance to periapsis distance. k_2 is called the winding parameter, since it determines the number of revolutions until reaching apocenter. The phase angle ϕ defines the orientation of the exposin in its plane. The values of the parameters should be chosen such that the resulting trajectory fulfills the boundary conditions for the initial and final position and the time of flight (TOF).

The idea of Petropoulos' work [2] is to find a solution for the equations of motion using Equation 3. A tangential thrust profile is assumed, such that $\alpha = \gamma$. The flight-path angle γ, which is equal to $dr/d\theta$, can be written as follows:

$$\tan\gamma = k_1 k_2 \cos\left(k_2\theta + \phi\right) \tag{4}$$

As thrust tangential to the velocity vector maximizes the energy gain and thus the velocity change [9], this assumption is deemed reasonable.

Using Equations 1, 3, and 4, and the tangential thrust assumption, the normalized thrust acceleration can, after some derivations (refer to Section IX), be derived as:

$$a = \frac{(-1)^n \tan\gamma}{2\cos\gamma}\left[\frac{1}{\tan\gamma^2 + k_1 k_2^2 s + 1} - \frac{k_2^2(1-2k_1 s)}{(\tan\gamma^2 + k_1 k_2^2 s + 1)^2}\right] \tag{5}$$

where

$$s \equiv \sin k_2\theta + \phi \tag{6}$$

when $n = 0$, the thrust is directed along the velocity vector and when $n = 1$ against the velocity vector.

The TOF can be computed by integrating the inverse of the angular velocity over the polar angle [3]:

$$\begin{aligned} \text{TOF} &= \int_{t_0}^{t_f} dt = \int_{\theta_0}^{\theta_f} \frac{dt}{d\theta} d\theta \\ &= \int_{\theta_0}^{\theta_f} \sqrt{r^3(\tan\gamma^2 + k_1 k_2^2 s + 1)/\mu} \quad d\theta \\ &= \sum_{i=1}^{i_{\max}} \sqrt{r^3(\tan\gamma^2 + k_1 k_2^2 s + 1)/\mu} \quad \Delta\theta_i \end{aligned} \tag{7}$$

where $i_{\max}$ is the specified number of intervals. Equation 7 can be solved using a numerical integration technique.

B. Implementation Using Lambert's Approach

Although Petropoulos' work [2] was quite successful from a numerical point of view, it lacked a solution to go from a generic point P_1 to another point P_2 in a given fixed amount of time. In the case of ballistic arcs, this problem, named "Lambert's problem", admits a number of solutions. In line with this work, Izzo proposed the "multi-revolution Lambert's problem" for exponential sinusoids, which is a convenient approach for the implementation of the exposin [3]. It starts with assuming k_2 fixed and known, which reduces the problem to all exposins that are defined by only three free parameters, namely k_0, k_1, and ϕ. After simplifications of the equations, it can be derived that with r_0, r_f, ψ, and the number of revolutions N required, there exists for an assumed value of k_2 a class $S_{k2}[r_0, r_f, \psi, N]$ of exposins passing through r_0 and r_f, with ψ the transfer angle. An exposin is classified as a valid trajectory whenever the condition $k_1 k_2^2 < 1$ is fulfilled. Within a defined class, the only free parameter is γ_0, which determines the differences within the family of exposins, namely the TOF. A numerical method can be used to find the intersection between a given required TOF and the TOF curve corresponding to a class of exposins. With the value for γ_0 defined, the geometric parameters (k_0, k_1, and ϕ) can be obtained (refer to Section IX for the equations).

Using this generic approach, the problem is to choose the shape parameter k_2 for each exposin. A direct interplanetary low-thrust transfer between two bodies will therefore have a three dimensional input vector $\boldsymbol{x} = [t_0, k_2, \text{TOF}]$, where t_0 is the departure date. When this input vector is known, the ephemerides of the departure planet at departure of the spacecraft and of the arrival planet at arrival should be evaluated. Finally, the unique exposin matching the requirement on TOF has to be found.

C. Total Velocity Increment Computation

The first method to measure the efficiency of trajectories is by looking at the total velocity increment (ΔV) required to fly a specific exposin. This value consists of a low-thrust part along the arc (ΔV_{LT}) and two impulsive shots at departure (ΔV_0) and arrival (ΔV_f).

The ΔV_{LT} can be determined by integration of the thrust acceleration F over the flight time t, which in turn can be obtained as a function of polar angle θ:

$$\Delta V_{\text{LT}} = \int_0^{t_f} F dt = \int_0^{\theta_f} \frac{F}{d\theta} d\theta \tag{8}$$

A numerical technique can be used to perform the integration.

Impulsive shots at departure and arrival are necessary to match the velocity vectors of the departure (V_{dep}) and arrival (V_{arr}) planets to the velocity vectors required at the initial ($V_{0,\text{exp}}$) and final ($V_{f,\text{exp}}$) position of the exposin, illustrated by Equations 9 and 10, respectively .

$$\Delta V_0 = V_{0,\text{exp}} - V_{\text{dep}} \tag{9}$$

$$\Delta V_f = V_{\text{arr}} - V_{f,\text{exp}} \tag{10}$$

The velocity at any point along the exposin can be computed using the radial and tangential components [**?**]:

$$V_r = r\dot{\theta}\tan\gamma \tag{11}$$

$$V_t = r\dot{\theta} \tag{12}$$

where, after some derivations (refer to Section IX), the derivative of the angular rate is found using Equation 13.

$$\dot{\theta}^2 = \left(\frac{\mu}{r^3}\right)\frac{1}{\tan\gamma^2 + k_1 k_2^2(\sin k_2\theta + \phi) + 1} \tag{13}$$

The velocity vectors along the exposin should be converted from polar coordinates to Cartesian coordinates to match the velocities of the planets.

D. Propellant Mass Fraction Computation

A second method to specify the efficiency of trajectories is by looking at the propellant mass fraction, using [10]:

$$J_m = 1 - \exp\left(-\frac{\Delta V_0 + \Delta V_f}{I_{\text{sp,chem}} \cdot g_0} - \frac{\Delta V_{\text{LT}}}{I_{\text{sp,LT}} \cdot g_0}\right) \tag{14}$$

where $I_{\text{sp,chem}}$, $I_{\text{sp,LT}}$, and g_0 are the specific impulses for the chemical engine, low-thrust engine, and the standard gravitational acceleration, respectively. As payload mass increases when propellant mass fraction decreases, the most optimal trajectories are those with the lowest value of J_m. Since the efficiency of the trajectories depend not only on the total ΔV, but also on the relative magnitude between ΔV_{LT} and $\Delta V_0 + \Delta V_f$, it could be argued that J_m is a better parameter to assess the optimality of trajectories.

III Generation of Trajectories

In order to train any GP model, training data has to be generated. In this work, the exposin is implemented according to the multi-revolution Lambert's problem to model low-thrust trajectories. The problem is narrowed down by assuming a transfer using an exposin with one revolution ($N = 1$). The goal is to locate regions of near-optimal trajectories within a defined three-dimensional input space $[t_0, k_2, \text{TOF}]$. It is assumed that the spacecraft has a mass of 1000 kg and is equipped with an ion propulsion engine and a chemical rocket engine, with specific impulses of 3000 s and 350 s, respectively. The maximum achievable thrust acceleration ($F_{\max}$) of the ion propulsion engine is taken as $3 \cdot 10^{-4}$ km/s^2 [1].

Input vectors with uniform randomly distributed values for t_0, k_2, and TOF, within the defined bounds, are fed into a program written by the authors, which computes for each input vector the corresponding exposin, using the approach as outlined in Section II. With the k_2 and t_0 values defined by the input vector, a class of exposins, if there exists any, can be computed (refer to Section IX for all equations). The Regula Falsi method [11] is then used to find within this class the unique exposins that matches the required TOF, which is in turn evaluated using the Midpoint method [12]. The data generation program is implemented using the ECLIPJ2000 reference frame with the Sun as center of the reference frame. The JPL DE405 ephemerides [13] are used to evaluate the ephemerides of the departure and arrival planets at times t_0 and t_0+TOF, respectively. In accordance with the work of Izzo [3], the centers of mass of the arrival and departure planets are taken, respectively, as the r_0 and r_f positions. One-way light time and stellar aberration corrections are applied to the states of the planets. With the unique exposin defined, it remains to compute the ΔV and J_m values corresponding to this trajectory. Again the Midpoint method [12] is used for the numerical integration of ΔV_{LT}. Using the criterium, as defined by Gondelach [1], of a 0.1% accuracy in ΔV required to rank different trajectories well, 1000 integration steps have been selected. As the Midpoint method is a one-step method, first the thrust acceleration required along the low-thrust arc should be evaluated at 1000 points along the trajectory, using Equations 2 and 5. The impulsive velocities can be computed by matching the velocities of the planets to those of the exposin (refer to Section IX), after which J_m can be found. As the main purpose of the exposin is to provide a good initial guess for a detailed numerical optimization, only the gravitational acceleration of the Sun is taken into account and all other perturbations are ignored, which is in accordance with the work of both Petropoulus [2] and Izzo [3].

Finally, three criteria have been specified to classify whether there belongs a feasible exposin to a specific input vector:

1) the condition $k_1 k_2^2 < 1$ should be fulfilled [2],
2) the difference between the required TOF and the actual TOF of a trajectory has to be smaller than 1 second, and
3) the thrust profile required to follow the low-thrust arc should not exceed the maximum available thrust.

The program described above, used to generate training data, will be referred to as "the data generation program".

IV Gaussian Process Models for Prediction of Optimal Trajectories

Due to the constraints on feasible trajectories as discussed in the previous section, not all input vectors contain an exposin that can actually be flown. Before predicting near-optimal trajectories in terms of ΔV and J_m out of a set of candidate input vectors, it should first be evaluated whether there belongs a feasible exposin to each of these input vectors. This is where classification using a GPC model becomes useful. The mapping function, which associates the "feasible" or "not feasible" output with the input vector, is defined as the predicting target:

$$\begin{aligned} f_{\text{GPC}} &: \mathbb{R}^3 \to \mathbb{R}, \\ \boldsymbol{x} &\mapsto Q. \end{aligned} \tag{15}$$

where $\boldsymbol{x} = [t_0, k_2, \text{TOF}]$ and $Q = -1$ (not feasible) or $Q = 1$ (feasible).

When each input vector is classified as feasible or not feasible, a GPR model is used to predict the ΔV or J_m values corresponding to the feasible trajectories:

$$\begin{aligned} f_{\text{GPR},1} &: \mathbb{R}^3 \to \mathbb{R}, \\ \boldsymbol{x} &\mapsto \Delta V \quad \text{or} \quad \boldsymbol{x} \mapsto J_m. \end{aligned} \tag{16}$$

where $\boldsymbol{x} = [t_0, k_2, \text{TOF}]$. This model will be referred to as "GPR model 1". Because it directly maps the input vector to a ΔV or J_m value, it is only trained with shape parameter k_2. To investigate the applicability of the GP based method, a second GPR model has been developed ("GPR model 2") that maps the entire shape information onto a ΔV or J_m value, thereby replacing only the integration step:

$$\begin{aligned} f_{\text{GPR},2} &: \mathbb{R}^6 \to \mathbb{R}, \\ \boldsymbol{x} &\mapsto \Delta V \quad \text{or} \quad \boldsymbol{x} \mapsto J_m. \end{aligned} \tag{17}$$

where $\boldsymbol{x} = [r_0, \psi, k_0, k_1, k_2, \phi]$.

For a GP model, the prediction of an output value given an input, can be achieved using a Bayesian inference operating on a GP in function space. In a GP, every random collection of variables has a multivariate normal distribution [14]. In this work, the input vector $\boldsymbol{x}$ is considered as the random variable following a Gaussian distribution. A GP over the function $f(\boldsymbol{x})$ is defined by [14]:

$$f(\boldsymbol{x}) \sim GP(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}^*)) \tag{18}$$

where $\mu(\boldsymbol{x})$ and $k(\boldsymbol{x}, \boldsymbol{x}^*)$ are the mean function and covariance function, respectively. $\boldsymbol{x}^*$ represents a point in the input domain. The GP does not contain any prior information until some data is observed. It is therefore

necessary to train any GP model with information provided by training samples. For M training samples, a training set is constructed as:

$$\mathscr{D} = \{\boldsymbol{X} = [\boldsymbol{x_1}, \boldsymbol{x_2}, ..., \boldsymbol{x}_\mathrm{M}]^T, \boldsymbol{Y} = [y_1, y_1, ..., y_\mathrm{M}]^T\} \tag{19}$$

where $\boldsymbol{X}$ and $\boldsymbol{Y}$ are training input matrix and output vector, respectively. The training inputs $\boldsymbol{x}_i, 1 \leq i \leq M$ are generated following a random uniform distribution in the input domain and are all normalized between 0 and 1. Furthermore, $\boldsymbol{Y}$ is considered to be free of noise as the training outputs are obtained by the data generation program.

It is assumed that the ΔV and J_m values of M^* input vectors need to be assessed. We define the predicting input and output vectors as $\boldsymbol{X}^* = [\boldsymbol{x}_1^*, \boldsymbol{x}_2^*, ..., \boldsymbol{x}_{\mathrm{M}^*}^*]$ and $\boldsymbol{Y}^* = [\Delta V_1^*, \Delta V_2^*, ..., \Delta V_{\mathrm{M}^*}^*]$ or $\boldsymbol{Y}^* = [J_{m_1}^*, J_{m_2}^*, ..., J_{m_{\mathrm{M}^*}}^*]$, respectively. This allows to define the joint distribution of the training outputs $\boldsymbol{Y}$ and the predicting outputs $\boldsymbol{Y}^*$ as:

$$\begin{bmatrix} \boldsymbol{Y} \\ \boldsymbol{Y}^* \end{bmatrix} \sim \mathscr{N}\left(\begin{bmatrix} \mu(\boldsymbol{X}) \\ \mu(\boldsymbol{X}^*) \end{bmatrix}, \begin{bmatrix} K(\boldsymbol{X}, \boldsymbol{X}) & K(\boldsymbol{X}, \boldsymbol{X}^*) \\ K(\boldsymbol{X}^*, \boldsymbol{X}) & K(\boldsymbol{X}^*, \boldsymbol{X}^*) \end{bmatrix} \right) \tag{20}$$

where the covariance matrix $K(\boldsymbol{X}, \boldsymbol{X})$ is given by:

$$K(\boldsymbol{X}, \boldsymbol{X}) = \begin{bmatrix} k(\boldsymbol{x_1}, \boldsymbol{x_1}) & k(\boldsymbol{x_1}, \boldsymbol{x_2}) & ... & k(\boldsymbol{x_1}, \boldsymbol{x}_\mathrm{M}) \\ k(\boldsymbol{x_2}, \boldsymbol{x_1}) & k(\boldsymbol{x_2}, \boldsymbol{x_2}) & ... & k(\boldsymbol{x_2}, \boldsymbol{x}_\mathrm{M}) \\ ... & ... & ... & ... \\ k(\boldsymbol{x}_\mathrm{M}, \boldsymbol{x_1}) & k(\boldsymbol{x}_\mathrm{M}, \boldsymbol{x_2}) & ... & k(\boldsymbol{x}_\mathrm{M}, \boldsymbol{x}_\mathrm{M}) \end{bmatrix} \tag{21}$$

The likelihood function defines the probability density of the observations given the parameters and inference concerns the prediction of new targets given a dataset and the associated GP model [14].

A. Model Development Procedure

It is desired to develop a GP model where [6]

1) the probability of obtaining correct training outputs given the model is maximal and
2) the errors on predicting outputs are limited to an acceptable level.

As a GP is defined by its mean function, covariance function, likelihood function, and inference method, a combination of them where 1) and 2) are satisfied should be found. The selected mean function, covariance function, and likelihood function will contain parameters, called the "hyperparameters", which have to be optimized in order to improve the performance. Therefore, "model development" in this paper refers to

a) the selection of the mean function, covariance function, likelihood function and inference method and
b) the optimization of the hyperparameters.

To measure the performance of the developed GPR models, it is decided to use three error measures: the mean absolute error (MAE), the mean absolute percentage error (MAPE), and the root mean square error (RMSE), respectively given by Equations 22, 23, and 24.

$$\boldsymbol{\varepsilon}_\mathrm{MAE}(M) = \frac{1}{M_T} \sum_{i=1}^{M_T} |T_{\mathrm{DG},i} - T_{\mathrm{GPR},i}| \tag{22}$$

$$\boldsymbol{\varepsilon}_\mathrm{MAPE}(M) = \frac{100\%}{M_T} \sum_{i=1}^{M_T} \left| \frac{T_{\mathrm{DG},i} - T_{\mathrm{GPR},i}}{T_{\mathrm{DG},i}} \right| \tag{23}$$

$$\boldsymbol{\varepsilon}_\mathrm{RMSE}(M) = \sqrt{\frac{1}{M_T} \sum_{i=1}^{M_T} (T_{\mathrm{DG},i} - T_{\mathrm{GPR},i})^2} \tag{24}$$

In the above equations, M is the training set size, M_T the test set size, and $T_{\mathrm{DG},i}$ and $T_{\mathrm{GPR},i}$ the target values (ΔV, or J_m) obtained from the data generation program and the GPR model, respectively. For the GPC model only the MAPE value is of interest, which is found using Equation 25.

$$\boldsymbol{\varepsilon}_\mathrm{MAPE,GPC}(M) = \frac{100\%}{M_T} \sum_{i=1}^{M_T} \frac{|Q_{\mathrm{DG},i} - Q_{\mathrm{GPR},i}|}{2} \tag{25}$$

In theory, one could define infinitely many mean functions and covariance functions, and thus infinitely many GP models. As we cannot theoretically define the optimal GP model for a specific application and using a trial and error procedure is generally highly time consuming, a model development procedure is proposed, which consists of the following three parts:

i) Preliminary model selection phase
ii) Combined model selection phase
iii) Hyperparameter optimization phase

During the first two parts, the same method is used, as visualized in Figure 1. In the following subsections, the details of each selection phase are discussed.

1. Preliminary Model Selection Phase

During the preliminary phase, candidates which are promising in terms of MAPE are separately sought for the mean function, the covariance function, the likelihood function, and the inference method. Each of them is defined as a "candidate to be selected", as referred to in Figure 1. While this procedure is performed for one candidate (e.g. the mean function), default settings are used for the other candidates. The selected default settings are no mean function, a squared exponential covariance function with automatic relevance determination, a Gaussian likelihood function, and Gaussian inference for both training and prediction. The dataset contains 250 samples on which k-fold cross validation is applied [14], such that $k-1$ subsets are selected for training and the remaining subset for validation, i.e. defining the accuracy of the models. This process is repeated in total k times. In this work, k is set to 5. For each of the candidates separately, all possible functions that were provided by the gpml toolbox in MATLAB [15], where for the composite functions up to two base functions are combined, are evaluated. The procedure is repeated four times with different data sets to increase the robustness of validation, after which the average MAPE for each function is computed. For the mean function, covariance function, and inference method, five functions resulting

in the lowest MAPE values are selected. The single best likelihood function is selected.

2. Combined Model Selection Phase

In the combined model selection phase, the top five functions for the mean function, covariance function, and inference method, and the top likelihood function, as selected in the preliminary phase, are combined into a GP model. In total, 45 different GP models have to be evaluated according to the procedure as depicted in Figure 1, where the GP model is the "candidate to be selected". A dataset containing 1250 samples is used and the procedure is repeated once, after which the optimal GP model based on MAPE value is selected. One could optionally take the CPU time for prediction into account as a second selection criterion.

3. Hyperparameter Optimization

During the third phase model training is performed, where the values of the hyperparameters have to be selected. The most common way to optimize the hyperparameters ($\boldsymbol{\theta}$) is by using the maximum likelihood method, where the optimal values of the hyperparameters are found by minimizing the following objective function [14]:

$$\begin{aligned} J(\boldsymbol{\theta}) &= -\log p(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{\theta}) \\ &= \frac{1}{2}\boldsymbol{Y}^T K^{-1}(\boldsymbol{X},\boldsymbol{X})\boldsymbol{Y} + \frac{1}{2}\log|K(\boldsymbol{X},\boldsymbol{X})| + \frac{M}{2}\log 2\pi \end{aligned} \tag{26}$$

with p the likelihood function. To keep the procedure consistent, all hyperparameters are initialized with zero's, after which the conjugent gradient method is used to find optimal values [14]. 1000 training samples are used and the validation is performed by 1000 test samples. It is expected that multiple local optima exist for the values of the hyperparemeters. Although a gradient-based marginal likelihood optimization method might converge to a local optimum instead of the global one, it is chosen due to its extremely fast execution power. Furthermore, as the optima found for the values of the hyperparameters result in a satisfactory behavior, it is decided that using a more powerful optimization technique is not highly required. In other work [16], random searches have been advised for the optimization of hyperparameters. However, due to the large number of hyperparameters of the models under consideration, this approach is undesirable for preliminary optimization as it would require long CPU times to find optimal combinations of hyperparameters.

B. Training Set Size

With the GP models developed, it remains to determine the number of training samples. Training any GP model with a larger number of samples will generally improve the performance. However, as a consequence, the CPU time needed for prediction will also increase. There exists a trade-off in the number of training samples. Although we are unable to define the optimal training set size [17], it is desired to take a training set size that strikes a balance between accuracy and computational burden. Therefore, the MAE is taken as the evaluation criterion for determining the training set size of the regression models [6]. The training set size is initialized to 50 samples, and is increased until a balance between prediction time and accuracy is reached, using the following increment update equation:

$$M_{s+1} = M_s + m \tag{27}$$

where m is the specified increment of training samples, taken as 50. For every M_s training samples, $n+1$ MAE's are evaluated using:

$$E(M_s, m, n) = [\boldsymbol{\varepsilon}_{\text{MAE}}(M_s), \boldsymbol{\varepsilon}_{\text{MAE}}(M_s + m), ..., \boldsymbol{\varepsilon}_{\text{MAE}}(M_s + nm)]^T$$

where n is set to 10. Next, the difference between the maximum and minimum values in vector E is computed:

$$\Delta\boldsymbol{\varepsilon}_{\text{MAE}} = \max(E(M_s, m, n) - \min(E(M_s, m, n) \tag{28}$$

For each update step, the procedure is repeated five times with different training samples and the average values are taken, with the purpose to increase the robustness of validation. When $\Delta\boldsymbol{\varepsilon}_{\text{MAE}}$ becomes smaller than a specified threshold, the increment update process is stopped. For both the regression models this threshold is defined as 0.01 km/s when predicting ΔV and 0.001 when predicting J_m. For the classification model, the same approach is used, but as only the MAPE is of interest, $\Delta\boldsymbol{\varepsilon}_{\text{MAPE,GPC}} \leq 0.1\%$ is taken as the threshold. Using this approach, it can be assured that the GP models are stable and a balance between accuracy and CPU time for prediction is obtained.

C. Robustness of Model Development Procedure

As in this work a novel model development procedure is proposed, a robustness analysis with respect to the choices made during this procedure (e.g. the size of the dataset) is required. All choices are made with the goal to optimize the trade-off between CPU time of the development process and robustness of the validation. It should be verified whether these choices are made rationally and whether choosing a more elaborate model development procedure would result in the development of models performing better in terms of MAPE. The procedure as previously proposed will be referred to as "the baseline model development procedure" and the models developed using this procedure as the "baseline models".

1. Preliminary Model Selection Phase

The size of the dataset used in the preliminary phase is set to 250 samples. It is expected that the larger the dataset, the more robust the validation of the models. To test whether 250 samples is enough during the preliminary phase, a comparison with the performance of the final developed models when using a five times larger dataset is performed. The baseline model development procedure is used, but with a dataset of 1250 samples during the preliminary phase. The models resulting from this modified procedure are compared to the baseline models in terms of MAPE.

2. Combined Model Selection Phase

In the baseline model development procedure only the top performing GP model is selected during the combined phase. However, the best performing model in the combined phase will not necessarily perform best after hyperparameter optimization is applied. Therefore, the development procedure is modified by selecting the top three performing models during the combined phase to apply the hyperparameter optimization on all three of them. All other settings are kept the same as in the baseline model development procedure. The MAPE of the models resulting from this modified development procedure are compared to that of the baseline models.

3. Hyperparameter Optimization

As it is likely that the conjugent gradient method converges to a local optimum, it should be verified how much the final performance deviates when the hyperparameters belonging to another (local or global) optimum are selected. Therefore, while keeping the preliminary and combined stage unchanged, differential evolution is used to initialize the values of the hyperparameters, after which the conjugent gradient method is applied to determine the actual values of the hyperparameters. Combining these two methods, it is more likely to converge to the global optimum instead of a local one. The settings for the differential evolution are taken as $F_p = 0.75$, $C_r = 0.8$, and $I = 20D$, with I the number of individuals and D the dimension of the problem (i.e. the number of hyperparameters to be optimized) [18]. Again, the performance of the models resulting from this modified procedure are compared to that of the baseline models.

To assess the robustness of the choices made in each phase of the development procedure, Equation 29 is used:

$$\%_{\text{robustness}} = (1 - \frac{\text{MAPE}_{\text{model},x} - \text{MAPE}_{\text{model},0}}{\text{MAPE}_{\text{model},0}}) \cdot 100\% \quad (29)$$

where x and 0 indicate the models resulting from the deviated and baseline model development procedure, respectively.

V Test Cases

To assess the performance of the GP-based method for the preliminary optimization of low-thrust trajectories, in terms of accuracy and CPU time, it has to be tested against mission test cases. According to the proposed model development procedure, GP models are developed for three test cases: rendezvous missions from Earth to Mars, from Mars to Earth and from Earth to Ceres. Characteristics of each of these target planets are provided in Table I, and the mission test cases are further detailed in this section.

A. Rendezvous Mission from Earth to Mars

Since the Earth-Mars transfer has been used before to test the performance of other shape-based methods [1] [19] [20], it is selected as a test case in this work, to compare the results obtained with the developed GP models to external results computed by others. Therefore, the results reported in Table III are used, which cover the shaping techniques and DITAN. DITAN is a trajectory optimization tool and can be considered as state of the art for low-thrust interplanetary trajectory design [20].

TABLE I: Characteristics of target candidates for test cases: semi-major axis, eccentricity, and inclination (with respect to the mean ecliptic and equinox of J2000), orbital period and synodic period (with respect to Earth) [21]

	a (AU)	e (-)	i (deg)	Orbital period (days)	Synodic period (days)
Mars	1.5237	0.0934	1.8506	687	780
Earth	1.0000	0.0167	0.0000	365	-
Ceres	2.7671	0.0758	10.593	1682	467

The bounds for the input space are selected in accordance with the work of Novak and Vasile [19] as t_0 :[58848, 61769] MJD and TOF: [500, 2000] days. k_2 is selected to lie within the bounds [0.01, 1], allowing for up to 50 revolutions around the Sun, a number unlikely to be exceeded in practice. The launch window between Jan 1, 2020 and December 31, 2027 is large enough to contain almost four synodic periods of the Sun-Earth-Mars system, which is equal to 780 days on average.

B. Rendezvous Mission from Mars to Earth

The developed GP models should ideally be tested against as many test cases as possible. Therefore, it is chosen to test the performance against a mission from an outbound to an inbound target, namely a rendezvous mission from Mars to Earth. No relevant results produced by other researchers are available for this test case, but the performance should be compared to that of the Earth-Mars test case, and to be able to do a fair comparison, the same input space is chosen.

C. Rendezvous Mission from Earth to Ceres

The third mission test case is selected as a rendezvous mission from Earth to the asteroid Ceres. Due to its much larger inclination than Mars, it is expected to be a challenging target in terms of predicting feasible and near-optimal trajectories. As this test case has not been used before in low-thrust trajectory optimization, the same input space as for the Earth-Mars test case is chosen, to allow for comparison between both test cases.

VI Selected GP Models

Using the baseline model development procedure as proposed in Section IV, five GP models are selected for each of the test cases: a GPC model, GPR model 1 with target value ΔV, GPR model 1 with target value J_m, GPR model 2 with target value ΔV and GPR model 2 with target value J_m. The developed models for the Earth-Mars test case, the equations of the underlying functions [15], and the values of the hyperparameters found, are presented in this

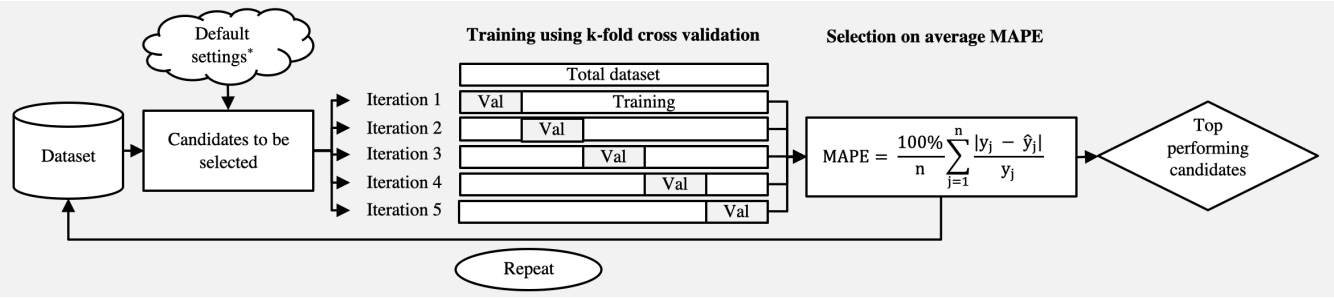


Fig. 1: Model selection and training method

section. Furthermore, a method to assess the robustness of the developed models is discussed. The developed models for the Mars-Earth and Earth-Ceres test cases are provided in Section X.

A. Selected Models for a Rendezvous Mission from Earth to Mars

1. Classification Model

For the classification model, the mean function is selected as the sum of a constant and a polynomial mean function, with two polynomials:

$$\mu(\boldsymbol{x}) = \mu_{\text{polynomial}}(\boldsymbol{x}) + \mu_{\text{constant}}(\boldsymbol{x}) \tag{30}$$

$$\mu_{\text{polynomial}}(\boldsymbol{x}) = \sum_{i=1}^{D}\sum_{j=1}^{d} a_{ij}\cdot \boldsymbol{x}_i^j \tag{31}$$

$$\mu_{\text{constant}}(\boldsymbol{x}) = c \tag{32}$$

where D is the dimension of the input space, d the number of polynomials and a_{ij} and c hyperparameters to be selected.

The covariance function is selected as the sum of a piecewise polynomial covariance function with automatic relevance determination (PPard) and a squared exponential covariance function with automatic relevance determination (SEard):

$$k(\boldsymbol{x},\boldsymbol{x}^*) = k(\boldsymbol{x},\boldsymbol{x}^*)_{\text{PPard}} + k(\boldsymbol{x},\boldsymbol{x}^*)_{\text{SEard}} \tag{33}$$

$$k(\boldsymbol{x},\boldsymbol{x}^*)_{\text{PPard}} = s^2_{f,\text{PPard}}\cdot \max(1-r,0)^{j+d}\cdot f(r,j) \tag{34}$$

$$k(\boldsymbol{x},\boldsymbol{x}^*)_{\text{SEard}} = s^2_{f,\text{SEard}}\cdot e^{-(\boldsymbol{x}-\boldsymbol{x}^*)'\cdot \text{inv}(P)\cdot \frac{\boldsymbol{x}-\boldsymbol{x}^*}{2}} \tag{35}$$

where the distance r is defined as:

$$r = \sqrt{(\boldsymbol{x}-\boldsymbol{x}^*)'\text{inv}(P)(\boldsymbol{x}-\boldsymbol{x}^*)} \tag{36}$$

with the P matrix diagonal with automatic relevance determination parameters $\ell_1^2,\ldots,\ell_D^2$. Furthermore, d is the number of polynomials (in this case 1), $j = \text{floor}(D/2)+d+1$, and s_f^2 is the signal variance. The function $f(r,j)$ is equal to $1+r(j+1)$ for $d=1$.

Finally, Gaussian likelihood is selected, given by Equation 37, using Leave-One-Out (LOO) inference for training and Gaussian noise inference for prediction:

$$p(y_i|f_i) = \frac{e^{\frac{-(y_i-f_i)^2}{2s_n^2}}}{\sqrt{2\pi s_n^2}} \tag{37}$$

where f is a scalar latent function value and s_n is the standard deviation of the noise.

As the classification model has a three dimensional input vector, the following hyperparameters have to be set:

$$\begin{cases} \boldsymbol{\theta}_C^M = & [a_{11},a_{12},a_{21},a_{22},a_{31},a_{32},c]^T \\ \boldsymbol{\theta}_C^C = & [\ell_{1,P},\ell_{2,P},\ell_{3,P},s_{f,P},\ell_{1,S},\ell_{2,S},\ell_{3,S},s_{f,S}]^T \\ \boldsymbol{\theta}_C^L = & \log(s_n) \end{cases}$$

where M, C, and L indicate the hyperparameters of the mean, covariance, and likelihood function, respectively, and subscripts P and S stand for "PPard" and "SEard", respectively.

2. Regression Model 1 for Prediction of ΔV

The mean function is selected as the product of a polynomial mean function ($d=3$) and a constant mean function:

$$\mu(\boldsymbol{x}) = \mu_{\text{polynomial}}(\boldsymbol{x})\cdot \mu_{\text{constant}}(\boldsymbol{x}) \tag{38}$$

The covariance function is selected as the product of PPard ($d=2$) and a rational quadratic covariance function with automatic relevance determination (RQard):

$$k(\boldsymbol{x},\boldsymbol{x}^*) = k(\boldsymbol{x},\boldsymbol{x}^*)_{\text{PPard}}\cdot k(\boldsymbol{x},\boldsymbol{x}^*)_{\text{RQard}} \tag{39}$$

$$k(\boldsymbol{x},\boldsymbol{x}^*)_{\text{RQard}} = s^2_{f,\text{RQard}}[1+\frac{(\boldsymbol{x}-\boldsymbol{x}^*)'\text{inv}(P)(\boldsymbol{x}-\boldsymbol{x}^*)}{2\alpha}]^{-\alpha} \tag{40}$$

with α a shape parameter and $f(r,j)$ in Equation 34 equal to $1+r(j+2)+(j^2+4j+3)/3r^2$ for $d=2$.

The same likelihood function and inference method for training and prediction are selected as for the GPC model.

The following hyperparameters need to be identified for this GPR model with a three dimensional input vector:

$$\begin{cases} \boldsymbol{\theta}_{R1_{\Delta V}}^M = & [a_{11},a_{12},a_{13},a_{21},a_{22},a_{23},a_{31},a_{32},a_{33},c]^T \\ \boldsymbol{\theta}_{R1_{\Delta V}}^C = & [\ell_{1,P},\ell_{2,P},\ell_{3,P},s_{f,P},\ell_{1,R},\ell_{2,R},\ell_{3,R},s_{f,R},\alpha]^T \\ \boldsymbol{\theta}_{R1_{\Delta V}}^L = & \log(s_n) \end{cases}$$

where subscript R indicates the "RQard" covariance function.

3. Regression Model 1 for Prediction of J_m

When the goal is to predict the propellant mass fraction J_m instead of ΔV, the continuous target values for the regression models differ significantly. Therefore, the model development procedure is repeated.

The mean function is selected as the second power of a constant mean function:

$$\mu(\boldsymbol{x}) = \mu_{\text{constant}}(\boldsymbol{x})^P = c^P \tag{41}$$

with P the power, in this case 2.

The same covariance function is selected as for the prediction of ΔV, but with $d = 1$ for PPard. Instead of LOO inference for training, as selected for the model when predicting ΔV, Laplace inference is selected.

The following hyperparameters have to be set:

$$\begin{cases} \boldsymbol{\theta}^M_{R1_{Jm}} = & c \\ \boldsymbol{\theta}^C_{R1_{Jm}} = & [\ell_{1,P}, \ell_{2,P}, \ell_{3,P}, s_{f,P}, \ell_{1,R}, \ell_{2,R}, \ell_{3,R}, s_{f,R}, \alpha]^T \\ \boldsymbol{\theta}^L_{R1_{Jm}} = & \log(s_n) \end{cases}$$

4. Regression Model 2 for Prediction of ΔV

Selecting the mean function as the sum of a polynomial mean function ($d = 4$) and a linear mean function results in the smallest MAPE:

$$\mu(\boldsymbol{x}) = \mu_{\text{polynomial}}(\boldsymbol{x}) + \mu_{\text{linear}}(\boldsymbol{x}) \tag{42}$$

$$\mu_{\text{linear}}(\boldsymbol{x}) = \sum_{i=1}^{D} c_i \cdot \boldsymbol{x}_i \tag{43}$$

with c_i hyperparameters to be selected.

The covariance function is selected as the product of the PPard covariance function ($d = 1$) and a squared exponential covariance function with isotropic lengthscale (SEiso):

$$k(\boldsymbol{x}, \boldsymbol{x}^*) = k(\boldsymbol{x}, \boldsymbol{x}^*)_{\text{PPard}} \cdot k(\boldsymbol{x}, \boldsymbol{x}^*)_{\text{SEiso}} \tag{44}$$

$$k(\boldsymbol{x}, \boldsymbol{x}^*)_{\text{SEiso}} = s^2_{f,\text{SEiso}} \cdot e^{-(\boldsymbol{x}-\boldsymbol{x}^*)' \cdot \text{inv}(P) \cdot \frac{\boldsymbol{x}-\boldsymbol{x}^*}{2}} \tag{45}$$

For the SEiso covariance function, P is ℓ^2 times the unit matrix, with ℓ an isotropic lengthscale.

In accordance with GPR model 1, Gaussian likelihood and Gaussian inference for prediction are selected. LOO likelihood is selected for training. As the input space is six-dimensional, the following hyperparameters have to be set:

$$\begin{cases} \boldsymbol{\theta}^M_{R2_{\Delta V}} = & [a_{11}, a_{12}, a_{13}, a_{14}, a_{21}, a_{22}, a_{23}, a_{24}, a_{31}, a_{32}, \\ & a_{33}, a_{34}, a_{41}, a_{42}, a_{43}, a_{44}, a_{51}, a_{52}, a_{53}, a_{54}, \\ & a_{61}, a_{62}, a_{63}, a_{64}, c_1, c_2, c_3, c_4, c_5, c_6]^T \\ \boldsymbol{\theta}^C_{R2_{\Delta V}} = & [\ell_{1,P}, \ell_{2,P}, \ell_{3,P}, \ell_{4,P}, \ell_{5,P}, \ell_{6,P}, s_{f,P}, \ell_S, s_{f,S}]^T \\ \boldsymbol{\theta}^L_{R2_{\Delta V}} = & \log(s_n) \end{cases}$$

5. Regression Model 2 for Prediction of J_m

Only one difference exist for GPR model 2 when predicting J_m instead of ΔV: in accordance with GPR model 1, the mean function is selected as the second power of a constant mean function.

The following hyperparameters have to be set:

$$\begin{cases} \boldsymbol{\theta}^M_{R2_{Jm}} = & c \\ \boldsymbol{\theta}^C_{R2_{Jm}} = & [\ell_{1,P}, \ell_{2,P}, \ell_{3,P}, \ell_{4,P}, \ell_{5,P}, \ell_{6,P}, s_{f,P}, \ell_S, s_{f,S}]^T \\ \boldsymbol{\theta}^L_{R2_{Jm}} = & \log(s_n) \end{cases}$$

Selected Values of the Hyperparameters

The optimal values that are selected for the hyperparameters of each of the five models are given below.

$$\begin{cases} \boldsymbol{\theta}^M_C = & [-0.1813, -0.6354, 0.3663, 0.3637, -1.3475, \\ & -2.0630, -1.9156]^T \\ \boldsymbol{\theta}^C_C = & [-1.6814, -0.9755, -1.0673, -0.1345, 0.4923, \\ & 0.4958, 1.7350, -1.7049]^T \\ \boldsymbol{\theta}^L_C = & -1.0167 \end{cases}$$

$$\begin{cases} \boldsymbol{\theta}^M_{R1_{\Delta V}} = & [0.0213, 0.0220, 0.0150, 0.0187, 0.0229, \\ & 0.0097, 0.0245, 0.0250, 0.0072, 0.0384]^T \\ \boldsymbol{\theta}^C_{R1_{\Delta V}} = & [-2.1976, 1.7751, -1.7550, 1.9828, -1.3113, \\ & 3.0219, -0.4880, 1.9828, -0.0960]^T \\ \boldsymbol{\theta}^L_{R1_{\Delta V}} = & 2.3302 \end{cases}$$

$$\begin{cases} \boldsymbol{\theta}^M_{R1_{Jm}} = & 0.0000 \\ \boldsymbol{\theta}^C_{R1_{Jm}} = & [-2.2217, 0.0925, -1.5596, -2.7126, 2.7644, \\ & -0.1544, 1.5578, -0.5312, -0.6622]^T \\ \boldsymbol{\theta}^L_{R1_{Jm}} = & -4.1403 \end{cases}$$

$$\begin{cases} \boldsymbol{\theta}^M_{R2_{\Delta V}} = & [0.0169, 0.0315, 0.000, 0.0314, 0.0191, \\ & 0.0213, 0.0206, 0.0314, 0.000, 0.0206, \\ & 0.0137, 0.0137, 0.0228, 0.0348, 0.000, \\ & 0.0136, 0.0110, 0.0104, 0.0244, 0.0398, \\ & 0.000, 0.0090, 0.0091, 0.0082, 0.0169, \\ & 0.0315, 0.000, 0.0314, 0.0191, 0.0213]^T \\ \boldsymbol{\theta}^C_{R2_{\Delta V}} = & [2.3813, 2.1613, 0.000, -2.2179, 0.6417, \\ & 1.3370, 2.6585, -0.5689, 2.6585]^T \\ \boldsymbol{\theta}^L_{R2_{\Delta V}} = & 3.5765 \end{cases}$$

$$\begin{cases} \boldsymbol{\theta}^M_{R2_{Jm}} = & 0.0000 \\ \boldsymbol{\theta}^C_{R2_{Jm}} = & [6.3993, 5.3731, 0.0000, 0.8819, 4.6385, \\ & 4.7228, 0.0119, -0.4210, 0.0119]^T \\ \boldsymbol{\theta}^L_{R2_{Jm}} = & -10.8042 \end{cases}$$

B. Assessing the Robustness of the Developed Models

When one wants to apply the provided GP models, referred to as "the initial models", on a slightly different mission scenario than the Earth-Mars test case as selected in this work ("the initial mission scenario"), one could either use the provided models or repeat the model development procedure. In this subsection, it is tried to assess the robustness of the developed models (i.e. their dependency on the specified mission scenario), by testing their performance in terms of MAPE and prediction time when applied to slightly different mission scenarios. The goal is to provide some confidence for the application of these models on mission scenarios with a variation from the one discussed in this paper. To assess this robustness, the initial models are tested on 15 different mission scenarios, by applying three deviations along five mission design dimensions:

i) A different range for t_0
   a) Initial: t_0 = Jan 1 2020 - Dec 31 2027
   b) Deviation 1: t_0 = Jan 1 2028 - Dec 31 2035
   c) Deviation 2: t_0 = Jan 1 2036 - Dec 31 2043
   d) Deviation 3: t_0 = Jan 1 2044 - Dec 31 2051

ii) A smaller range for t_0 and/or TOF
   a) Initial: t_0 = Jan 1 2020 - Dec 31 2027, TOF = 500-2000 days
   b) Deviation 1: t_0 = Jan 1 2020 - Dec 31 2023, TOF = 500-1250
   c) Deviation 2: t_0 = Jan 1 2020 - Dec 31 2027, TOF= 500-875 days
   d) Deviation 3: t_0 = Jan 1 2020 - Dec 31 2021, TOF= 500-2000 days

iii) A larger range for t_0 and/or TOF
   a) Initial: t_0 = Jan 1 2020 - Dec 31 2027, TOF= 500-2000 days
   b) Deviation 1: t_0 = Jan 1 2020 - Dec 31 2035, TOF= 500-3500
   c) Deviation 2: t_0 = Jan 1 2020 - Dec 31 2027, TOF= 500-6500 days
   d) Deviation 3: t_0 = Jan 1 2020 - Dec 31 2051, TOF= 500-2000 days

iv) A different target planet
   a) Initial: target = Mars
   b) Deviation 1: target = Ceres
   c) Deviation 2: target = Pallas
   d) Deviation 3: target = Vesta

v) A different number of revolutions for the exposin
   a) Initial: $N = 1$
   b) Deviation 1: $N = 2$
   c) Deviation 2: $N = 3$
   d) Deviation 3: $N = 4$

While one of these mission design parameters is varied, all other settings are kept equal to the initial mission scenario. For each of the 15 new mission scenarios, training and test samples are generated. The initial models are applied on all 15 mission scenarios, with the number of training samples equal to those presented in Table II. The difference in performance with respect to the initial mission scenario is computed as:

$$\%_{\text{difference}} = \frac{\text{MAPE}_{\text{scenario,x}} - \text{MAPE}_{\text{scenario,0}}}{\text{MAPE}_{\text{scenario,0}}} \cdot 100\% \qquad (46)$$

where x and 0 indicate the deviated and initial mission scenarios, respectively.

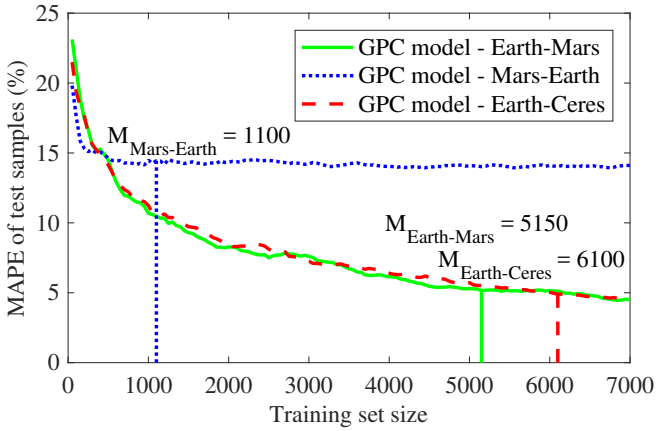


Fig. 2: Relationship between MAPE of test samples and number of training samples for GPC models for all three test cases

VII Numerical Results and Discussion

In the following, the performance of the developed GP models is discussed in terms of accuracy, CPU time, and their ability to locate regions of near-optimal trajectories, for each of the three test cases as defined in Section V. To evaluate this performance, test samples with a uniform random distribution in the input space are used for all GP models. Because in previous work [6] [7] the ratio between training and test samples varied between 1:1 and 1:5, and it is expected that the number of training samples required lies between 1000 and 5000, 1000 test samples are selected. The number of training samples at which a balance between accuracy and prediction time is obtained, is determined using the outlined approach in Section IV. This is demonstrated by Figure 2, which shows the resulting number of training samples for the GPC models for all test cases. It becomes clear that the performance of the GPC models barely improves when more training samples than the number selected by this iterative procedure are used. The prediction time for each of the models is computed as the CPU time for the prediction of 1000 samples and is averaged over 1000 runs. Furthermore, the results are discussed for the robustness analysis of the model development procedure, as described in Section IV, and the robustness of the developed models, as outlined in Section VI. All algorithms are implemented in MATLAB 2017a and are executed on a desktop computer with an Intel 2 GHz processor and 8.0 GB memory operating on a 64-bit macOS platform.

A. Rendezvous Mission from Earth to Mars

1. Numerical Results of GP Models

The determined number of training samples, the corresponding MAE, MAPE, and RMSE values, and the prediction times are provided in Table II. The prediction time is dependent on both the number of training samples and the complexity of the model.

TABLE II: Numerical results for a rendezvous mission from Earth to Mars

Model	#training samples	MAE (km/s) or (-)	MAPE (%)	RMSE (km/s) or (-)	CPU time prediction (s)
GPC	5150	-	5.16	-	9.63
GPR 1 - ΔV	4100	7.49	23.16	13.07	6.43
GPR 1 - J_m	2350	0.01	1.38	0.02	1.45
GPR 2 - ΔV	2700	33.26	119.08	40.01	2.16
GPR 2 - J_m	2350	0.03	3.74	0.06	1.60

a) Classification Model

Out of the 1000 test samples that are predicted, on average 948 are predicted correctly. Out of the 52 that are predicted incorrectly, 29 unfeasible trajectories are predicted as feasible and 23 feasible trajectories are predicted unfeasible. GP models have not been applied before to classification problems in the aerospace field. In several other classification applications [22] [23] [24] [25], unrelated to this work, accuracies between 1.5% and 33% have been found. Compared to these numbers, the MAPE of 7.49% as reached in this work, is located near the lower bound of this range. It should however be noted that these MAPE values are not the best indicator for the performance achieved in this work, but are provided here to give the reader an idea which range of values could be obtained for classification problems.

b) Regression Models

When evaluating the performance of each of the GPR models, it becomes clear that 1) the performance is better when predicting J_m instead of ΔV and 2) the performance of GPR model 1 is better than that of GPR model 2. It is expected that GPR models are, in general, better in predicting targets with a smaller ratio between the minimum and maximum target value. As the values for J_m differ between 0.48 and 1.00, and for ΔV between 6.06 and $\pm$130 km/s, this could explain the first observation, and is referred to as "hypothesis 1". The second observation could be explained by the fact that GPR model 2 is mainly trained with shape information, while GPR model 1 is trained with parameters containing information on the input space. Therefore, GPR model 2 should be able to accurately predict ΔV_{LT}, but is expected to contain less information on the impulsive velocities required at departure and arrival. This hypothesis is referred to as "hypothesis 2".

GPR has been applied once before for the prediction of ΔV values. In the work of Shang and Liu [6] it was used to predict the ΔV values required to access main-belt asteroids using (high-thrust) transfer trajectories, and MAE's of 0.06-0.12 km/s have been found. It should be noted that the to be predicted values ranged within 6-12 km/s, which is a much smaller ratio between the minimum and maximum value than the one used in this test case, from which it is expected that high accuracies could be obtained more easily. As a result, the MAE's obtained for the GPR models in this work when predicting ΔV are worse, starting from 7.49 km/s for GPR model 1. The found MAPE's in the work of Shang and Liu range between 0.58% and 1.33%, which comes close to the value obtained in this work for GPR model 1 when predicting J_m. Another way to assess the performance is in comparison with other machine learning algorithms. In the work of Li [16], the MAPE's of several machine learning algorithms applied on low-thrust fuel optimization problems have been listed. For eight machine learning algorithms, the corresponding MAPE's varied between 0.33% and 3.66%. Similar MAPE's are found for both GPR models when predicting J_m.

2. Optimal Trajectories

The best trajectories found previously with other shape-based methods ([1] [19] [26]), together with the optimization techniques used, are presented in Table III. To find the overall best trajectory belonging to the exposin shape, three optimization techniques are selected: a differential evolution [27], a grid search and a grid search with mesh refinements. The total ΔV values and maximum accelerations corresponding to the optimal trajectories in terms of ΔV modeled by the exposin shape are shown in Table III. The optimal value for J_m is found using the differential evolution and is given in Table VIII. In this work, the globally optimal trajectory is defined as the overall best trajectory found by either the differential evolution, the grid search, or the adaptive grid search.

a) Differential Evolution

The classical version of the DE is chosen, as developed by Storn and Price [27]. To determine the optimal values for the mutation probability F_p and crossover probability C_r used in the differential evolution (DE), F_p is varied between 0.5 and 0.8, and C_r between 0.85 and 0.95, as optimal values for comparable problems were found to lie within these ranges [1] [28]. From trial and error the number of individuals I turns out to be optimal between 50 and 100. Using a grid of 0.05 for F_p and C_r, and 10 for I, the optimal settings are found to be 0.5, 0.85 and 100 for F_p, C_r, and I, respectively. The input vectors corresponding to the optimal trajectories found in terms of ΔV and J_m are given by $\boldsymbol{x}$ =[61175.64, 0.4007, 618.04] and $\boldsymbol{x}$ = [61358.04, 0.4359, 949.05], respectively.

b) Grid Search

A grid search (GS) is implemented, where the grid is specified as: $t_0 = [58848 : 15 : 61769]$, $k_2 = [0.01 : 0.025 : 1]$, and TOF = $[500 : 20 : 2000]$, in line with the work of Novak

and Vasile [19]. At each grid point, the ΔV and J_m values corresponding to this input vector are computed, and the optimal trajectory in terms of ΔV is found to belong to input vector $\boldsymbol{x}$ = [61173.00, 0.3850, 620.00].

c) Adaptive Grid Search

An adaptive grid search (AGS) is implemented, where a coarse grid is used that is refined around the obtained interim solution. The initial grid (G_0) contains 75 grid points, where five points are spaced equally for all three dimensions. After evaluation of all 75 grid points, the grid is refined around the interim solution by halving the length of the search interval (G_1). This process of refining the grid is repeated i times, until the solution at G_i is the same as G_{i-10} and therefore convergence is reached. The input vector for the optimal trajectory obtained with the AGS in terms of ΔV is given by $\boldsymbol{x}$ = [62701.02, 0.1970, 562.74].

From Table III, it can be observed that with all three optimization techniques less optimal trajectories in terms of total ΔV are found for the exposin shape than for the other shape-based methods. As the GP models are trained with transfer trajectories based on the exposin shape, they will not be able to find trajectories with ΔV values smaller than 6.06 km/s, which is therefore used as the global optimum to compare the performance of the GP models against. As given in Table VIII, J_m = 0.48 is used as the global optimum for the propellant mass fraction.

When the GPC model and GPR model 1 are placed in series, it is possible to predict ΔV and J_m belonging to feasible trajectories given a set of input vectors, thereby replacing the functionality of the data generation program. The more samples that are fed to the GP models, the higher the probability that trajectories with ΔV and J_m values close to the global optima are found. On the other hand, the corresponding prediction time also increases. Figure 3 shows a trade-off between the optima found and the corresponding prediction time, at different numbers of samples to be predicted. These plots are produced as follows. The specified number of samples (e.g. 10,000) is fed to the GPC model. The GPC model is trained with the number of training samples as provided in Table II. The trajectories classified by the GPC model as feasible are fed into GPR model 1. The predicted values for each of the input vectors are ranked, and the top 50 input vectors are selected. The actual target values are computed for these 50 input vectors, making use of the data generation program. Averaged over 50 runs, the target value of the best trajectory is given as a dot in Figure 3. The error-bar shown spans the range between the worst and best values found for the best trajectory within these 50 runs. This procedure is performed for both the prediction of ΔV and J_m. For the prediction of J_m, a decreasing average target value and increasing prediction time, at an increasing number of prediction samples, is clearly visible. Furthermore, it can be observed that the error bars get smaller for a larger number of samples to be predicted, indicating that the reliability of the average J_m value increases. Such a clear trend is not observed for the prediction of ΔV, and the target values barely get better with increasing samples to be predicted, although the average prediction time does increase. This observation could be explained by the fact that the accuracies achieved for the prediction of J_m are much better than those for the prediction of ΔV, and the resulting optimal trajectories are therefore more reliable. Additionally, it becomes clear from Figure 3a that the optimal trajectories found deviate somewhat from the global optima, which could be explained as follows. When producing 100,000 randomly generated input vectors, the smallest ΔV values found generally lie between 6.25 and 6.30 km/s and the smallest J_m values between 0.49 and 0.52. Furthermore, as the training set contains only 4100 or 2350 training samples for the prediction of ΔV and J_m respectively, the models are likely not trained with (near-)globally optimal trajectories. When the goal is to obtain globally optimal trajectories, it should be investigated how the performance of the GPR models could be improved by adding the globally optimal trajectories to the training set.

Although the GP models are not most suitable for finding globally optimal trajectories, they are a powerful tool for the prediction of a large input space with the goal to locate the regions of feasible or near-optimal trajectories. Classifying the feasibility of 1,000,000 candidate input vectors and predicting the ΔV or J_m values corresponding to the feasible trajectories, is achieved within a total computation time for prediction of less than 1400 seconds. When assessing these values with the data generation program, this will take more than one week. Training the models takes about 30 s CPU time per model, but as the hyperparameters are already provided in this work, training the model is unnecessary and only prediction has to be repeated. The CPU time required to generate the training samples takes 0.7 s per trajectory times the amount of training samples required, in this case 5150. For these purposes, the CPU time can be reduced approximately 150 times. It should be noted that for an increasing number of candidates to be assessed, the reduction in CPU time is even larger as prediction time does not increase linearly with prediction samples, which is observed in Figure 3a.

The distributions computed by the data generation program, referred to as "the actual distributions", and the ones predicted by the GP models are shown in Figure 4. A repetitive pattern is observed for both the ΔV and J_m values of feasible trajectories. This pattern is periodic with the synodic period of the Sun-Earth-Mars system, which is on average 780 days. Since the predicted distributions closely resemble the actual distributions, it is concluded that the GP models are indeed a powerful tool in locating the regions of feasible and near-optimal trajectories. Furthermore, trajectories that are located in near-optimal regions could be fed as initial guess to more refined optimization techniques, which are more likely able to find globally optimal trajectories.

TABLE III: Required ΔV and maximum thrust accelerations $F_{\max}$ corresponding to minimum-ΔV trajectories found by different methods for a rendezvous mission to Mars (*= [1],**= [19]***= [20])

Method	**Optimization technique**	**ΔV [km/s]**	**$F_{\max}$ [10^{-4} m/s^2]**
Hohmann***	-	5.50	-
Hodographic - time*	Nelder-Mead	5.77	1.5
Hodographic - polar angle*	Nelder-Mead	5.81	1.6
Spherical**	Grid Search	5.74	2.2
Pseudo-equinoctial***	Evolutionary Branching	5.83	1.6
DITAN***	Direct Finite Element Transcription + Sparse optimizer	5.66	1.5
Exponential sinusoid	Differential Evolution	6.24	1.2
Exponential sinusoid	Grid Search	6.25	1.2
Exponential sinusoid	Adaptive Grid Search	6.06	1.2

3. Predicting ΔV Along the Low-Thrust Arc

The total ΔV required for an interplanetary transfer is built up of a part delivered by the chemical engine and a part along the the low-thrust arc, delivered by the ion propulsion engine. To proof hypothesis 2, used for the explanation that GPR models 1 are performing better than GPR models 2, the prediction of solely ΔV_{LT} for the Earth-Mars mission test case is discussed in this subsection. As only ΔV_{LT} has to be predicted, J_m is linearly related to this value and therefore not of importance. The model development procedure is repeated for GPR model 1 and GPR model 2, with as target value ΔV_{LT}. The same input space as for the Earth-Mars mission test case is used.

The determined number of training samples and the numerical results achieved for both models are presented in Table IV. It can be observed that the performance of GPR model 2 is significantly better than that of GPR model 1, which is contrary to the performance of the GPR models for target value ΔV. This observation could be explained by the fact that GPR model 2 is trained with full shape information (input vector defined as $\boldsymbol{x} = [r_0, \psi, k_0, k_1, k_2, \phi]$), and is therefore able to accurately predict the value for ΔV_{LT}. As for most transfer trajectories, the largest part of ΔV is built up of the impulsive ΔV's, the performance of GPR model 2 is much worse when predicting ΔV. The values found for ΔV_{LT} for the Earth-Mars mission test case range within $\pm$3.20 km/s and $\pm$16.60 km/s. As the ratio between minimum and maximum target values is much smaller than for the target value ΔV, and better performance is reached for both GPR models when predicting ΔV_{LT}, the results obtained for this test case support hypothesis 1 as well.

TABLE IV: Numerical results for the prediction of ΔV_{LT} for a rendezvous mission from Earth to Mars

Model	**#training samples**	**MAE (km/s) or (-)**	**MAPE (%)**	**RMSE (km/s) or (-)**	**CPU time prediction (s)**
GPR 1 - ΔV	5050	0.20	2.18	0.33	3.98
GPR 2 - ΔV	1100	0.04	0.54	0.12	0.15

TABLE V: Numerical results for a rendezvous mission from Mars to Earth

Model	**#training samples**	**MAE (km/s) or (-)**	**MAPE (%)**	**RMSE (km/s) or (-)**	**CPU time prediction (s)**
GPC	1100	-	13.55	-	0.31
GPR 1 - ΔV	5000	9.35	31.24	14.75	11.22
GPR 1 - J_m	2350	0.02	2.14	0.03	1.41
GPR 2 - ΔV	1100	47.82	187.06	49.10	0.28
GPR 2 - J_m	1000	0.06	7.12	0.09	0.17

B. Rendezvous Mission from Mars to Earth

1. Numerical Results of GP Models

The numerical results are given in Table V. It is observed that for all models slightly worse performance is achieved than for the Earth-Mars test case. Better performance is obtained for GPR models 1 than GPR models 2, and the models perform significantly better when predicting J_m instead of ΔV, which is in line with the Earth-Mars test case. Hypotheses 1 and 2 are supported by both observations.

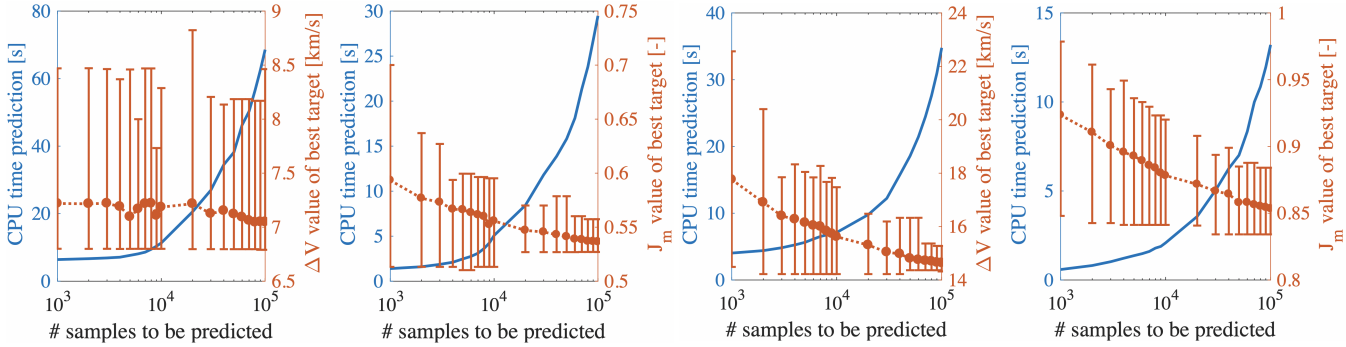
2. Optimal Trajectories

The optimal trajectories in terms of ΔV and J_m are provided in Tables VII and VIII, respectively, together with the corresponding input vectors at which these value are achieved. The DE, GS, and AGS are applied on this optimization problem, and the optima are found using the DE. When one would produce the trade-off and distribution plots for the Mars-Earth test case, results comparable to those shown in Figures 3 and 4 are observed. Refer to Section XI for all results. In line with the Earth-Mars test case, it can be concluded that the GP models are not perfectly suited for the computation of globally optimal trajectories, but they significantly outperform other optimization techniques in efficient exploration of a large search space for feasible and near-optimal trajectories.

C. Rendezvous Mission from Earth to Ceres

1. Numerical Results of GP models

The numerical results achieved for the Earth-Ceres mission test case are presented in Table VI. The achieved performance for the GPC model is slightly better than the one reached for the Earth-Mars test case. The GPR models show significantly better performance than for the other two test cases. Using hypothesis 1, this could be explained by the fact that for the Earth-Ceres test case the ratios between minimum and maximum values of ΔV and J_m



(a) Trade-off plots for Earth-Mars mission test case

(b) Trade-off plots for Earth-Ceres mission test case

Fig. 3: Trade-off plots for the Earth-Mars test case (left) and the Earth-Ceres test case (right)

TABLE VI: Numerical results for a rendezvous mission from Earth to Ceres

Model	#training samples	MAE (km/s) or (-)	MAPE (%)	RMSE (km/s) or (-)	CPU time prediction (s)
GPC	6100	-	4.90	-	12.22
GPR 1 - ΔV	3900	3.96	9.03	7.36	4.02
GPR 1 - J_m	2100	0.01	0.55	0.01	0.57
GPR 2 - ΔV	2100	37.07	96.07	38.78	1.21
GPR 2 - J_m	2250	0.02	1.63	0.02	0.43

are smaller. Consistent with the other two test cases and in accordance with hypothesis 2, GPR models 1 are showing better performance than GPR models 2.

2. Optimal Trajectories

The optimal trajectories in terms of ΔV and J_m values are provided in Tables VII and VIII, respectively. The DE, GS, and AGS are applied on this optimization problem, and the optima are found using the DE.

The trade-off between the number of prediction samples and target values found is shown in Figure 3b. It is observed that both the ΔV and J_m values steadily decrease at an increasing number of samples to be predicted, and as the error-bars are getting smaller, the reliability increases. The difference between Figures 3a and 3b for target value ΔV could be explained by the higher accuracies reached for the Earth-Ceres than for the Earth-Mars test case. Although the results for finding globally optimal trajectories are better for the Earth-Ceres mission test case than for the other two, the main strength of the GP models is still found in the efficient evaluation of a large input space. The distribution plots for the Earth-Ceres test case can be found in Section XI.

D. Robustness of the Model Development Procedure

The robustness analysis as detailed in Section IV is applied on both the GPC model and GPR model 1 with target value ΔV, for the rendezvous mission from Earth to Mars. The development procedure for the GPR model turned out to be highly robust, especially in the preliminary and combined phase, where (according to Equation 29) a robustness

TABLE VII: Globally optimal trajectories in terms of ΔV for all test-cases using the traditional optimization techniques (DE, AGS, and GS)

Mission test-case	ΔV (km/s)	t_0 (MJD)	k_2 (-)	TOF (days)
Earth-Mars	6.06	62701.02	0.1970	562.74
Earth-Mars low-thrust arc	3.15	61263.21	0.5837	507.33
Mars-Earth	6.34	61157.18	0.4341	580.45
Earth-Ceres	13.85	60290.56	0.5506	1007.22

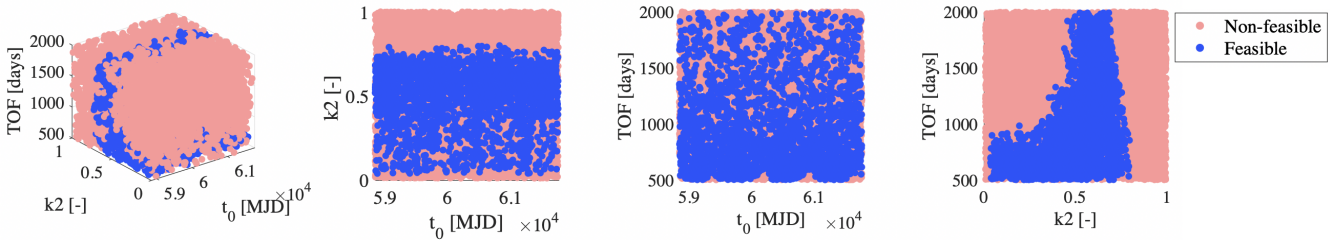
TABLE VIII: Globally optimal trajectories in terms of J_m for all test-cases using the traditional optimization techniques (DE, AGS, and GS)

Mission test-case	J_m (-)	t_0 (MJD)	k_2 (-)	TOF (days)
Earth-Mars	0.48	61358.04	0.4359	949.05
Mars-Earth	0.39	59599.01	0.3728	651.99
Earth-Ceres	0.76	59197.22	0.2639	1178.24

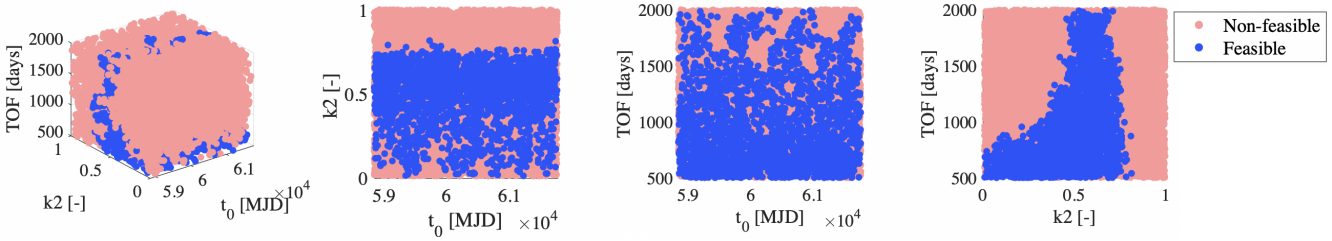
of more than 99% is reached. A robustness of 96.1% is obtained for the hyperparameter optimization phase, after performing 50 iterations, which takes more than 100 hours. For the application of the proposed procedure on the development of a GPC model, again a robustness of more than 99% is reached in the preliminary phase. The robustnesses achieved in phases 2 and 3, of respectively 92% and 86%, are lower than the ones achieved for the GPR model. It should be taken into account that the deviated procedure requires significantly more CPU time than the baseline model development procedure.

E. Robustness of the Developed Models

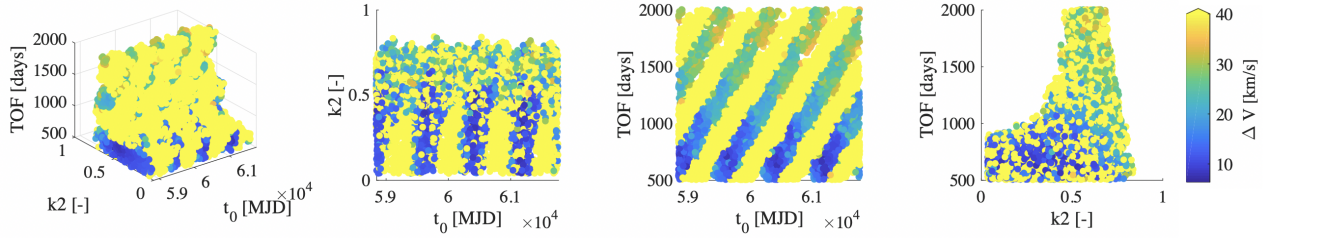
The numerical results for all 15 different mission scenarios of the Earth-Mars test case, achieved with the initial GP models, is assessed. Along the five mission design dimensions as specified in Section VI, the averaged results over the three deviations is taken. The MAPE values obtained for the averaged deviated mission scenarios, together with those for the initial mission scenario, are provided for all GP models in Figure 5 (left). The right plot in Figure 5 shows the relative difference in MAPE with respect to the initial mission scenario, which is computed using Equation 46. Positive percentages indicate a larger MAPE, and therefore



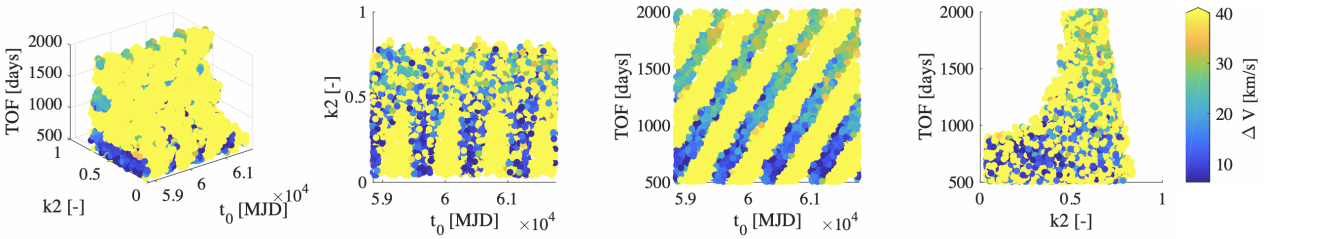
(a) Actual distribution of feasible trajectories



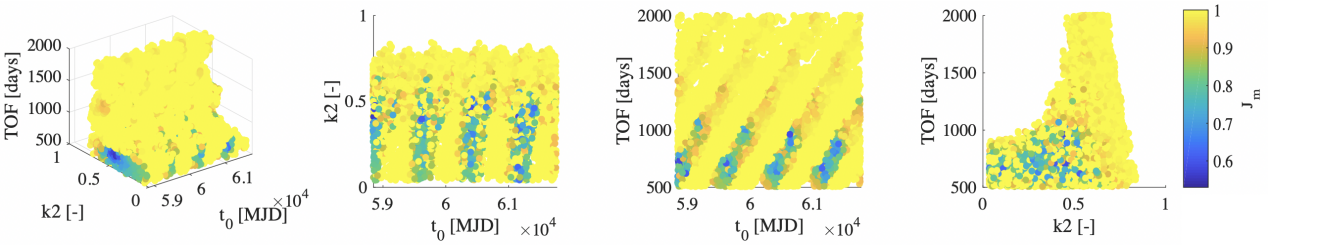
(b) Predicted distribution of feasible trajectories



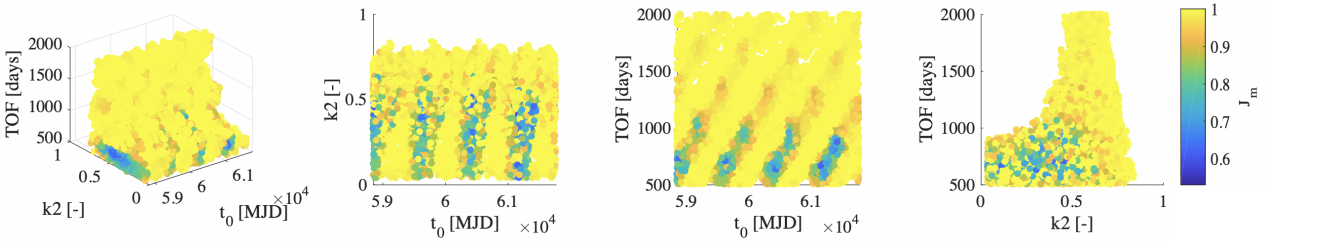
(c) Actual ΔV distribution



(d) Predicted ΔV distribution



(e) Actual J_m distribution



(f) Predicted J_m distribution

Fig. 4: Plots of actual and predicted distributions for the Earth-Mars mission test case

worse accuracies. It can be observed that for all models better performance is reached when they are applied on a mission scenario with a smaller range for t_0 and/or TOF. This result is expected, as generally less training samples are required for a smaller range between the input bounds [14]. As the same numbers of training samples as for the initial mission scenario are used, higher accuracies could be obtained. When the models are applied on a test case with a different range for t_0, but of the same size, the performance is comparable to that of the initial mission scenario. Worse performance is obtained when the models are applied on mission scenarios with a larger range for t_0 and/or TOF, as generally more training samples are required. When the models are applied on mission scenarios with a different target planet, better performance is achieved for all regression models. As the ratio between minimum and maximum ΔV and J_m values for missions to Ceres, Pallas and Vesta is smaller than for a mission to Mars, the higher accuracies obtained for the GPR models support hypothesis 1. For mission scenarios with a different number of revolutions, significantly worse performance is observed.

VIII Conclusions

In this paper, a novel procedure is proposed for the development of models utilizing GP regression and classification, to perform computationally efficient, preliminary optimization for direct, low-thrust trajectories. The low-thrust trajectories are modeled using the exponential sinusoid shape. The analysis performed in this work demonstrates that the proposed procedure is (highly) robust, especially where the development of GPR models is concerned. GP models have been developed for test cases from Earth to Mars and Ceres, and from Mars to Earth. In all test cases, the main advantage of the GP-based method is its efficiency in locating regions of feasible and near-optimal trajectories, especially when a large input space is explored. For the evaluation of 1,000,000 candidate trajectories, the GP-based prediction method is approximately 150 times faster versus numerical trajectory computation. This speed advantage increases for an increasing number of candidates.

GPR models trained with only three input space parameters showed higher accuracies than the ones trained with full shape information. This observation is caused by the fact that the shape-trained GPR models can accurately predict ΔV_{LT}, but are not informative about the impulsive ΔV's. Additionally, significant higher accuracies were achieved when predicting propellant mass fraction values (J_m) instead of ΔV. As one of the main objectives of space missions is to decrease J_m, models able to accurately predict J_m could be of significant importance in the preliminary design of low-thrust missions. From the results presented in this paper, the hypothesis is developed which states that GPR models can reach higher accuracies when the ratio between the minimum and maximum target values gets smaller. This hypothesis is supported by the results of all three test cases, but further work is necessary to confirm this. MAPE values obtained for the regression models with target value J_m ranged between 0.55% and 2.14%, which is in line with other work [6]. For the classification of feasible exposins, MAPE values between 4.90% and 13.55% were achieved. When the GPC and GPR models are placed in series, it is possible to predict (regions of) near-optimal trajectories, while requiring only three input parameters. Trajectories located in these regions can be used as initial guesses for more refined optimization techniques. The models presented have been tested for dependency on the Earth-Mars test case as presented in this paper. Comparable results in terms of MAPE and prediction time were achieved when the models were applied on a mission scenario with a different or smaller range for t_0 and/or TOF or a different target planet than the Earth-Mars mission scenario as specified in this work.

References

[1] D. J. Gondelach and R. Noomen. Hodographic-Shaping Method for Low-Thrust Interplanetary Trajectory Design. *Journal of Spacecraft and Rockets*, 52(3):728–738, 2015.

[2] A. E. Petropoulos and J. M. Longuski. A shape-based algorithm for the automated design of low-thrust, gravity-assist trajectories. *Advances in the Astronautical Sciences*, 109 III(5):2321–2336, 2002.

[3] D. Izzo. Lambert 's problem for exponential sinusoids. Technical Report April, ESA/ESTEC, 2005.

[4] D. G. Krige. A statistical approach to some basic mine valuation problems on the Witwatersrand. *Journal of the chemical metallurgical & mining society of South Africa*, 52(6), 1951.

[5] R. Dufour, J. de Muelenaere, and A. Elham. Trajectory driven multidisciplinary design optimization of a sub-orbital spaceplane using non-stationary Gaussian process. pages 755–771, 2015.

[6] H. Shang and Y. Liu. Assessing Accessibility of Main-Belt Asteroids Based on Gaussian Process Regression. *Journal of Guidance, Control, and Dynamics*, 40(5):1144–1154, 2017.

[7] A. Gao and W. Liao. Acta Astronautica Efficient gravity field modeling method for small bodies based on Gaussian process regression. *Acta Astronautica*, 157(May 2018):73–91, 2019.

[8] A. Gao, G. Y. Wang, S. S. Wu, and T. Song. Efficient Evaluation of Mars Entry Terminal State Based on Gaussian Process Regression Efficient. In *IOP Conference Series: Materials Science and Engineering*, 2018.

[9] D. A. Vallado. *Fundamentals of Astrodynamics and Applications*. Microcosm Press, 2013.

[10] M. Dellnitz Vasile, O. Schütze, O. Junge, G. Radice. Spiral Trajectories in Global Optimisation of Interplanetary and Orbital Transfers Table of Contents. Technical Report 0, 2006.

[11] M. Dowell and I. Jarratt. A modified regula falsi method for computing the root of an equation. *BIT Numerical Mathematics*, 11(2):168–174, 1971.

[12] P. C. Hammer. The Midpoint Method of Numerical Integration. *Mathematics Magazine*, 31(4):193–195, 1958.

[13] navigation and Facility NAIF: The ancillary Information. The SPICE Toolkit, 2018.

[14] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning.*, volume 14. 2004.

[15] C. E. Rasmussen. Documentation for GPML Matlab Code version 4.2.

[16] H. Li, S. Chen, D. Izzo, and H. Baoyin. Deep Networks as Approximators of Optimal Transfers Solutions in Multitarget Missions.

[17] J. Pericchi, O. Berger, and R. Luis. Training Samples in Objective Bayesian Model Selection. *The Annals of Statistics*, 32(3):841–869, 2004.

[18] M. Vasile and E. Minisci. Analysis of Some Global Optimization Algorithms for Space Trajectory Design. *Journal of Spacecraft and Rockets*, 47(2), 2010.

[19] D. M. Novak and M. Vasile. Improved Shaping Approach to the Preliminary Design of Low-Thrust Trajectories. *Journal of Guidance, Control, and Dynamics*, 34(1):128–147, 2011.

[20] P. de Pascale and M. Vasile. Preliminary Design of Low-Thrust Multiple Gravity-Assist Trajectories. *Journal of Spacecraft and Rockets*, 43(5):1065–1076, 2006.

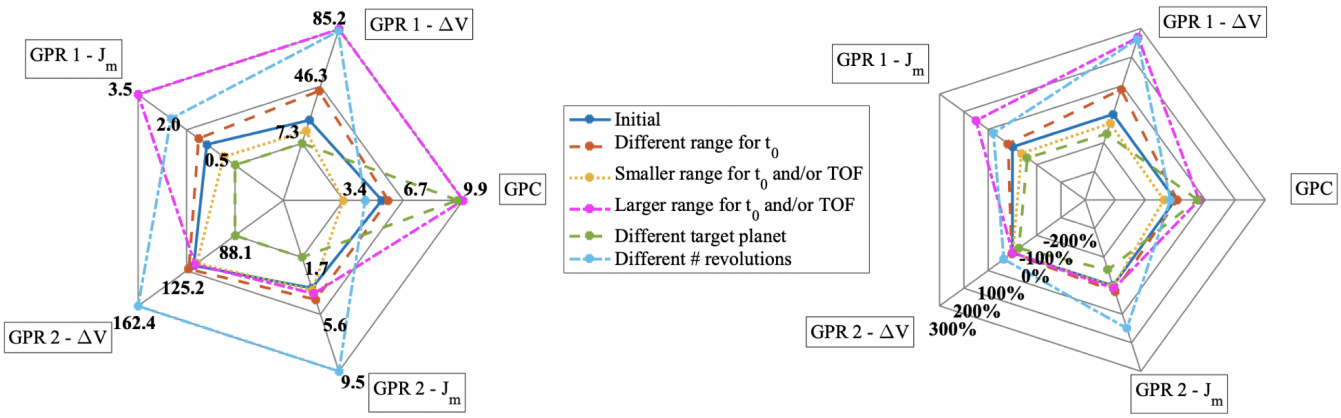


Fig. 5: Spider plots indicating the robustness of the developed models for the Earth-Mars test case. Left: MAPE values, Right: relative difference in MAPE with respect to initial mission scenario (taken as 0%)

[21] J. J. Lissauer and I. de Pater. *Fundamental planetary science*. Cambridge University Press, 2013.

[22] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Active Learning with Gaussian Processes for Object Categorization. (October), 2007.

[23] C. E. Rasmussen. Assessing Approximate Inference for Binary Gaussian Process Classification. 6:1679–1704, 2005.

[24] R. M. Neal. Regression and Classification Using Gaussian Process Priors. 1998.

[25] C. K. I. Williams and D. Barber. Bayesian Classification With Gaussian Processes. 20(12):1342–1351, 1998.

[26] M. Vasile, O. Schütze, and O. Junge. Spiral trajectories in global optimisation of interplanetary and orbital transfers. *ESA Technical Report*, 31(0):0–45, 2006.

[27] R. Storn and K. Price. Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359, 1997.

[28] J. Englander. Automated Trajectory Planning for Multiple-Flyby Interplanetary Missions. 2013.

IX Appendix: Derivations

A. Derivations of Exponential Sinusoid Equations

Making use of a trajectory that is shaped like an exposin, the dynamics must fit the kinematic equations (Equation 1). Given Equation 3, its first and second derivatives are computed as:

$$\dot{r} = r\dot{\theta}k_1k_2c \tag{A.1}$$

$$\ddot{r} = r[\ddot{\theta}k_1k_2c + (\dot{\theta}k_1k_2c)^2 - \dot{\theta}^2k_1k_2^2s] \tag{A.2}$$

where

$$c = \cos k_2\theta + \phi \tag{A.3}$$

$$s = \sin k_2\theta + \phi \tag{A.4}$$

Then using Equations 1, 2 and A.2, Equation A.5 is derived.

$$\dot{\theta}^2 = \left(\frac{\mu}{r^3}\right)\frac{ak_1k_2c\cos\alpha - a\sin\alpha + 1}{(k_1k_2c)^2 + k_1k_2^2s + 1} \tag{A.5}$$

As the assumption of tangential thrusting is made, $\alpha = \gamma$, and making use of Equation 4, Equation A.5 can be simplified to:

$$\begin{aligned}\dot{\theta}^2 &= \left(\frac{\mu}{r^3}\right)\frac{a\tan\gamma\cos\gamma - a\sin\gamma + 1}{\tan\gamma^2 + k_1k_2^2s + 1}\\ &= \left(\frac{\mu}{r^3}\right)\frac{a\sin\gamma - a\sin\gamma + 1}{\tan\gamma^2 + k_1k_2^2s + 1}\\ &= \left(\frac{\mu}{r^3}\right)\frac{1}{\tan\gamma^2 + k_1k_2^2s + 1}\end{aligned} \tag{A.6}$$

Furthermore, using Equations 1 and 2, the normalized thrust acceleration could be expressed as:

$$\begin{aligned}a &= \left(\frac{\mu}{r^3}\right)F\\ a &= \left(\frac{\mu}{r^3}\right)\frac{\ddot{\theta}r + 2\dot{\theta}\dot{r}}{\cos\gamma}\end{aligned} \tag{A.7}$$

Taking the derivative of both sides of Equation A.5 results in:

$$\begin{aligned}2\dot{\theta}\ddot{\theta} &= \left(\frac{\mu}{r^3}\right)\frac{-k_1k_2^3c\dot{\theta}}{(\tan\gamma^2 + k_1k_2^2s + 1)^2}\\ \ddot{\theta} &= \left(\frac{\mu}{r^3}\right)\frac{-k_1k_2^3c}{2(\tan\gamma^2 + k_1k_2^2s + 1)^2}\end{aligned} \tag{A.8}$$

Combining Equations A.1, A.6, A.7 and A.8 allows to express the normalized thrust acceleration a as:

$$\begin{aligned}a &= \left(\frac{\mu}{r^3}\right)\frac{-\left(\frac{\mu}{r^2}\right)\frac{k_1k_2^3c}{(\tan\gamma^2+k_1k_2^2s+1)^2} + 2r\dot{\theta}^2k_1k_2c}{\cos\gamma}\\ &= \left(\frac{\mu}{r^3}\right)\frac{-\left(\frac{\mu}{r^2}\right)\frac{k_1k_2^3c}{(\tan\gamma^2+k_1k_2^2s+1)^2} + 2\left(\frac{\mu}{r^2}\right)\frac{k_1k_2c}{\tan\gamma^2+k_1k_2^2s+1}}{\cos\gamma}\end{aligned} \tag{A.9}$$

which can after simplification be written as:

$$a = \frac{-1}{2\cos\gamma}\left[\frac{\tan\gamma}{\tan\gamma^2 + k_1k_2^2s + 1} - \frac{k_1k_2^3c - 2k_1k_2^2s\tan\gamma}{(\tan\gamma^2 + k_1k_2^2s + 1)^2}\right] \tag{A.10}$$

To make sure that the normalized thrust acceleration is always a positive value, independent of whether the thrust

is directed along or against the velocity vector, the final expression for a is given by Equation A.11.

$$a = \frac{(-1)^n \tan\gamma}{2\cos\gamma}\left[\frac{1}{\tan\gamma^2 + k_1 k_2^2 s + 1} - \frac{k_2^2(1-2k_1 s)}{(\tan\gamma^2 + k_1 k_2^2 s + 1)^2}\right] \tag{A.11}$$

B. Derivations of Lambert's Exposin Implementation

To compute a class of exposins belonging to a certain value of k_2, several derivations have been made in the work of Izzo [3], which are repeated here for the sake of completeness.

The sign of k_1 is given by Equation A.12:

$$\frac{k_1}{|k_1|}\sqrt{k_1^2 - \frac{\tan\gamma^2}{k_2^2}} = \frac{\ln\frac{r_0}{r_f} + \frac{\tan\gamma_0}{k_2}\sin k_2\bar{\theta}}{1-\cos k_2\bar{\theta}} \tag{A.12}$$

and the magnitude of k_1 follows from Equation A.13:

$$k_1^2 = \left(\frac{\log\frac{r_0}{r_f} + \frac{\tan\gamma_0}{k_2}\sin k_2\bar{\theta}}{1-\cos k_2\bar{\theta}}\right)^2 + \frac{\tan\gamma_0{}^2}{k_2^2} \tag{A.13}$$

where $\bar{\theta}$ is given by:

$$\bar{\theta} = \psi + 2\pi N \tag{A.14}$$

Next, ϕ and k_0 can be computed using Equations A.15 and A.16, respectively.

$$\phi = \arccos\left(\frac{\tan\gamma_0}{k_1 k_2}\right) \tag{A.15}$$

$$k_0 = \frac{r_0}{\exp k_1 \sin\phi} \tag{A.16}$$

It is important to find out which of the exposins belonging to a class $S_{k2}[r_0,\ r_f,\ \psi,\ \mathrm{N}]$ are actually feasible trajectories, i.e. that it is possible to follow that trajectory by tangential thrusting. Therefore, the condition $k_1 k_2^2 < 1$ must hold. Making use of Equation A.13 and rewriting it such that this condition holds, Equation A.17 is found:

$$\left(\frac{k_2^2\log\frac{r_0}{r_f} + k_2\frac{\tan\gamma_0}{k_2}\sin k_2\bar{\theta}}{1-\cos k_2\bar{\theta}}\right)^2 + k_2^2\tan\gamma_1{}^2 < 1 \tag{A.17}$$

which makes it possible to find analytically the interval for γ_0 for which feasible trajectories exist. In order to have a feasible trajectory within the class $S_{k2}[r_0,\ r_f,\ \psi,\ \mathrm{N}]$, it must hold for γ_0 that:

$$\tan\gamma_0 = \frac{k_2}{2}\left[-\log\frac{r_0}{r_f}\cot\frac{k_2\bar{\theta}}{2} \pm \sqrt{\Delta}\right] \tag{A.18}$$

in which:

$$\Delta = \frac{2(1-\cos k_2\bar{\theta})}{k_2^4} - \log\frac{r_1}{r_2}^2 \tag{A.19}$$

It can easily be seen that if $\Delta < 0$, no feasible exposins exist for the class $S_{k2}[r_0,\ r_f,\ \psi,\ \mathrm{N}]$.

C. Derivations of Impulsive Thrust Computation

To compute the impulsive thrust required at departure and arrival, the following equations are used:

$$\Delta V_0 = \sqrt{(V_{0,\mathrm{exp}} - V_{\mathrm{dep}})_x^2 + (V_{0,\mathrm{exp}} - V_{\mathrm{dep}})_y^2 + (V_{0,\mathrm{exp}} - V_{\mathrm{dep}})_z^2} \tag{A.20}$$

$$\Delta V_f = \sqrt{(V_{f,\mathrm{exp}} - V_{\mathrm{arr}})_x^2 + (V_{f,\mathrm{exp}} - V_{\mathrm{arr}})_y^2 + (V_{f,\mathrm{exp}} - V_{\mathrm{arr}})_z^2} \tag{A.21}$$

where V_{dep} and V_{arr} are the velocity vectors of the departure and arrival planet, respectively, and are retrieved from the JPL ephemerides [13]. The velocity at any point along the exposin can be expressed in polar coordinates using:

$$V_r = V_{\mathrm{mag}}\frac{\tan\gamma}{\sqrt{1+\tan\gamma^2}} \tag{A.22}$$

$$V_t = V_{\mathrm{mag}}\frac{1}{\sqrt{1+\tan\gamma^2}} \tag{A.23}$$

where γ is the flight-path angle at a specific point along the exposin and V_{mag} the magnitude of the velocity vector, which is found using Equations A.24, A.25 and A.26.

$$t = \dot{\theta} r \tag{A.24}$$

$$r = \dot{r} = r k_1 \cos(k_2\theta + \phi) k_2\dot{\theta} \tag{A.25}$$

$$V_{\mathrm{mag}} = \sqrt{t^2 + r^2} \tag{A.26}$$

The next step is to transform the velocities along the exposin from polar to Cartesian coordinates. Therefore, Equation A.27 is used:

$$V_{\mathrm{exp}} = \begin{bmatrix} V_r \cdot u_{r,x} \\ V_r \cdot u_{r,y} \\ V_r \cdot u_{r,z} \end{bmatrix} + \begin{bmatrix} V_t \cdot u_{t,x} \\ V_t \cdot u_{t,y} \\ V_t \cdot u_{t,z} \end{bmatrix} \tag{A.27}$$

where

$$u_r = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} \frac{1}{\sqrt{r_x^2 + r_y^2 + r_z^2}} \tag{A.28}$$

$$u_t = \begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} \frac{1}{\sqrt{c_x^2 + c_y^2 + c_z^2}} \tag{A.29}$$

in which vector c can be computed as:

$$\begin{bmatrix} c_x \\ c_y \\ c_z \end{bmatrix} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} \times \left(\begin{bmatrix} r_{0,x} \\ r_{0,y} \\ r_{0,z} \end{bmatrix} \times \begin{bmatrix} r_{f,x} \\ r_{f,y} \\ r_{f,z} \end{bmatrix}\right) \tag{A.30}$$

with × indicating the cross product. Position vectors r_0 and r_f are retrieved from the JPL ephemerides [13].

X Appendix: Selected Models for Test Cases

The GP models that are selected for the Mars-Earth and Earth-Ceres test cases are presented in this appendix. The selected values of the hyperparameters are given below. The selected mean functions, covariance functions, likelihood functions, and inference methods, for each of the five models for the Mars-Earth test case and Earth-Ceres test case, are provided in Tables A.I and A.II, respectively. For a more detailed explanation of the functions used, the reader should refer to the book by Rasmussen [14].

A. Hyperparameters for the Rendezvous Mission from Mars to Earth

The following optimal values are selected for the hyperparameters of the five models for the Mars-Earth transfer:

$$\begin{cases} \boldsymbol{\theta}_C^M = & [-0.1097, 0.2258, 0.1179, -0.1207, -0.1235, \\ & 0.0513, 0.2418, -0.2567, 0.2698, -0.1874]^T \\ \boldsymbol{\theta}_C^C = & [3.7278, -0.8838, -0.0063, -0.5790, \\ & 1.3901, 0.3949, 0.4116, -1.3126]^T \\ \boldsymbol{\theta}_C^L = & -0.5100 \end{cases}$$

$$\begin{cases} \boldsymbol{\theta}_{R1_{\Delta V}}^M = & [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, \\ & 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, \\ & 0.0000, 0.0000, 0.0000, 0.0000, 0.0000]^T \\ \boldsymbol{\theta}_{R1_{\Delta V}}^C = & [-0.1751, 8.0956, -0.2784, 3.2476, 1.0834, \\ & 6.5319, 0.9135, 3.2476, 0.5967]^T \\ \boldsymbol{\theta}_{R1_{\Delta V}}^L = & 2.6782 \end{cases}$$

$$\begin{cases} \boldsymbol{\theta}_{R1_{Jm}}^M = & [0.0000, 0.0000]^T \\ \boldsymbol{\theta}_{R1_{Jm}}^C = & [-2.1440, -0.0417, -2.0412, -2.4980, \\ & 3.6555, 0.0165, 1.4693, -0.0371, -0.5043]^T \\ \boldsymbol{\theta}_{R1_{Jm}}^L = & -3.5807 \end{cases}$$

$$\begin{cases} \boldsymbol{\theta}_{R2_{\Delta V}}^M = & [0.0312, 0.0340, 0.0000, 0.0324, 0.0180, \\ & 0.0202, 0.0278, 0.0405, 0.0000, 0.0224, \\ & 0.0182, 0.0472, 0.0291, 0.0773, 0.0000, \\ & 0.0162, 0.0207, 0.0697, 0.0305, 0.1176, \\ & 0.0000, 0.0124, 0.0242, 0.0712, 0.0312, \\ & 0.0340, 0.0000, 0.0324, 0.0180, 0.0202]^T \\ \boldsymbol{\theta}_{R2_{\Delta V}}^C = & [0.3560, 0.4305, -0.2984, 0.2195, -0.0200, \\ & 0.0495, 0.8205, -1.2693, 0.9229, 0.0000, \\ & 0.7113, -2.4052, -3.0558, 3.3139, -1.3425]^T \\ \boldsymbol{\theta}_{R2_{\Delta V}}^L = & 3.8459 \end{cases}$$

$$\begin{cases} \boldsymbol{\theta}_{R2_{Jm}}^M = & [0.0000, 0.0000]^T \\ \boldsymbol{\theta}_{R2_{Jm}}^C = & [0.1475, -0.6733, 0.0000, 0.1402, -0.3092, \\ & -0.7930, -2.1455, 2.6543, 0.2887]^T \\ \boldsymbol{\theta}_{R2_{Jm}}^L = & -5.3471 \end{cases}$$

B. Hyperparameters for the Rendezvous Mission from Earth to Ceres

The hyperparameters that are selected for each of the five GP models for the Earth-Ceres test case are presented below. It can be observed that the values differ significantly from those of the Mars-Earth test case, which illustrates the importance of the hyperparameter optimization phase in the model development procedure.

$$\begin{cases} \boldsymbol{\theta}_C^M = & [-0.6501, 0.0176, 1.1996, -0.1683, -0.2297, \\ & -1.3976]^T \\ \boldsymbol{\theta}_C^C = & [-0.8514, 0.0292, 0.5066, 0.3878, 1.6103, \\ & -1.5113, -0.2563]^T \\ \boldsymbol{\theta}_C^L = & -5.5533 \end{cases}$$

$$\begin{cases} \boldsymbol{\theta}_{R1_{\Delta V}}^M = & [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, \\ & 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, \\ & 0.0000, 0.0000, 0.0000]^T \\ \boldsymbol{\theta}_{R1_{\Delta V}}^C = & [-2.8155, 7.4397, -1.7395, -5.9042, \\ & 7.1267, -0.6571]^T \\ \boldsymbol{\theta}_{R1_{\Delta V}}^L = & -7.8084 \end{cases}$$

$$\begin{cases} \boldsymbol{\theta}_{R1_{Jm}}^M = & [0.0000, 0.0000, 0.0000, 0.0000, 0.4080, \\ & 0.0000, 0.0000, 0.0000, 0.0000, 0.4080, \\ & 0.0000, 0.0000, 0.0000, 0.0000, 0.4080]^T \\ \boldsymbol{\theta}_{R1_{Jm}}^C = & [-2.7584, 7.6780, -1.8951, -5.7783]^T \\ \boldsymbol{\theta}_{R1_{Jm}}^L = & -26.9124 \end{cases}$$

$$\begin{cases} \boldsymbol{\theta}_{R2_{\Delta V}}^M = & [0.0000, 0.0000, 0.0000, 0.0000, 0.0000, \\ & 0.0000, 0.0000, 0.1067, 0.0000, 0.0000, \\ & 0.0000, 0.0000, 0.0000, 0.0000, 0.0000, \\ & 0.1067, 0.0000, 0.0000, 0.0000, 0.0000, \\ & 0.0000, 0.0000, 0.0000, 0.1067]^T \\ \boldsymbol{\theta}_{R2_{\Delta V}}^C = & [-0.0756, -0.7563, 0.0000, 0.0126, -0.5231, \\ & -0.4258, 2.6386, 2.7193, 4.1212]^T \\ \boldsymbol{\theta}_{R2_{\Delta V}}^L = & 3.3537 \end{cases}$$

$$\begin{cases} \boldsymbol{\theta}_{R2_{Jm}}^M = & [0.0000, 0.0000]^T \\ \boldsymbol{\theta}_{R2_{Jm}}^C = & [-1.2500, -2.5265, 0.0000, 2.4133, -0.5175, \\ & -0.3657, -5.8826, 16.3252, 0.0159]^T \\ \boldsymbol{\theta}_{R2_{Jm}}^L = & -15.0936 \end{cases}$$

XI Appendix: Results for Test Cases

A. Rendezvous Mission from Mars to Earth

The plots visualizing the trade-off between prediction time and best trajectory found for the Mars-Earth test case are provided in Figure A.1. The plots are comparable to those for the Earth-Mars test case, shown in Figure 3a. As worse accuracies are achieved for the prediction of ΔV than J_m, the GPR model when predicting ΔV is not capable of finding trajectories approaching the global optimum at an increasing number of prediction samples.

The distributions in the input space of feasible trajectories and the corresponding ΔV and J_m values are shown in Figure A.2. It can be observed that the distribution for the Mars-Earth test case closely resembles that of the Earth-Mars test case, as was shown in Figure 4. This result is in

TABLE A.I: Selected GP models for the rendezvous mission from Mars to Earth

Model	**GPC model**	**GPR model 1 - ΔV**	**GPR model 1 - J_m**	**GPR model 2 - ΔV**	**GPR model 2 - J_m**
Mean function	Sum of polynomial ($d = 3$) and constant mean	Product of polynomial ($d = 4$) and linear mean	Scaled version of a constant mean	Sum of polynomial ($d = 4$) and linear mean	Scaled version of a constant mean
Covariance function	Sum of PPard ($d = 1$) and SEard covariance	Product of PPard ($d = 1$) and RQard covariance	Sum of PPard ($d = 1$) and RQard covariance	Sum of RQard and SEard covariance	Sum of PPard ($d = 0$) and SEiso covariance
Likelihood function	Gaussian likelihood	Gaussian likelihood	Gaussian likelihood	Gaussian likelihood	Gaussian likelihood
Inference method	Training: LOO inference	Training: LOO inference	Training: Laplace inference	Training: LOO inference	Training: Laplace inference
	Prediction: Gaussian noise inference	Prediction: Gaussian noise inference	Prediction: Gaussian noise inference	Prediction: Gaussian noise inference	Prediction: Gaussian noise inference

TABLE A.II: Selected GP models for the rendezvous mission from Earth to Ceres

Model	**GPC model**	**GPR model 1 - ΔV**	**GPR model 1 - J_m**	**GPR model 2 - ΔV**	**GPR model 2 - J_m**
Mean function	Polynomial ($d = 2$) mean	Product of polynomial ($d = 4$) and constant mean	Weighted sum of 3 projected cosines mean (WSPC, $p = 3$)	Weighted sum of 3 projected cosines mean (WSPC, $p = 3$)	Scaled version of a constant mean
Covariance function	Sum of PPard ($d = 0$) and RQiso covariance	Sum of PPard ($d = 1$) and SEiso covariance	Maternard ($\nu = 1/2$) covariance	Sum of PPard ($d = 0$) and SEiso covariance	Sum of PPard ($d = 0$) and SEiso covariance
Likelihood function	Gaussian likelihood	Gaussian likelihood	Gaussian likelihood	Gaussian likelihood	Gaussian likelihood
Inference method	Training: Gaussian noise inference	Training: Gaussian noise inference	Training: Gaussian noise inference	Training: Gaussian noise inference	Training: Gaussian noise inference
	Prediction: Gaussian noise inference	Prediction: Gaussian noise inference	Prediction: Gaussian noise inference	Prediction: Gaussian noise inference	Prediction: Gaussian noise inference

line with expectation, as both mission test cases share the same synodic period of 780 days on average. It can again be concluded that, although the GP models fail in finding globally optimal trajectories, they are capable of efficiently predicting distributions that quite closely resemble the actual distributions.

B. Rendezvous mission from Earth to Ceres

For the Earth-Ceres test case, the distribution plots are provided in Figure A.3. A repetitive pattern for optimal trajectories is visible, which corresponds to the average synodic period of the Sun-Earth-Ceres system of 467 days on average. As the predicted distributions strongly resemble the actual distributions, it is again concluded that the GP models are powerful tools for the prediction of regions of feasible and near-optimal trajectories within limited CPU time.

C. Robustness of the Developed Models for the Rendezvous Mission from Earth to Mars

This subsection contains a more detailed discussion on the robustness of the developed models, of which the procedure was outlined in Section VI. Along five mission design dimensions, three deviations are applied. The averaged results along each of them were presented in Section VII. In this section, the results along each of these dimensions is examined separately. For the 15 different mission scenarios with a deviated mission design parameter from the initial mission scenario, the obtained accuracies, when using the

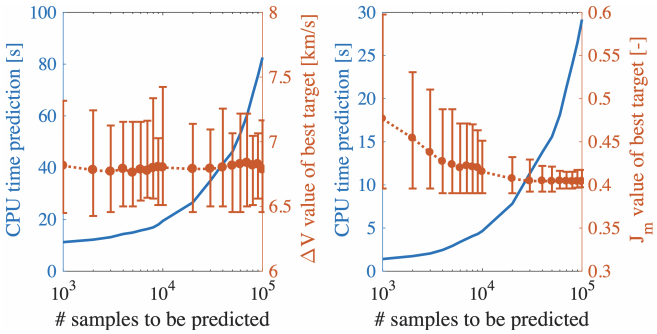
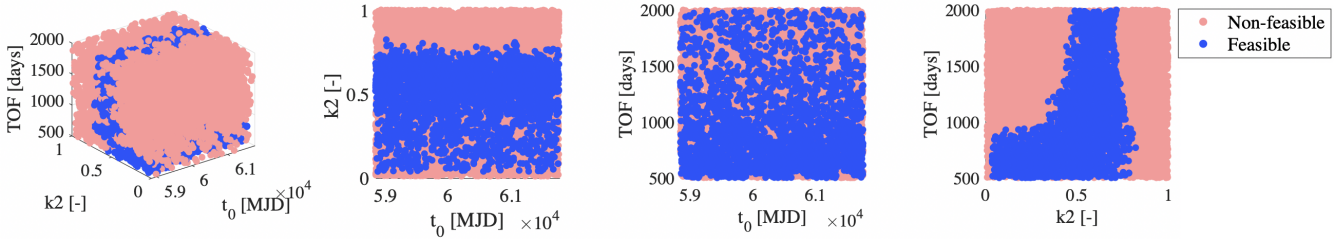
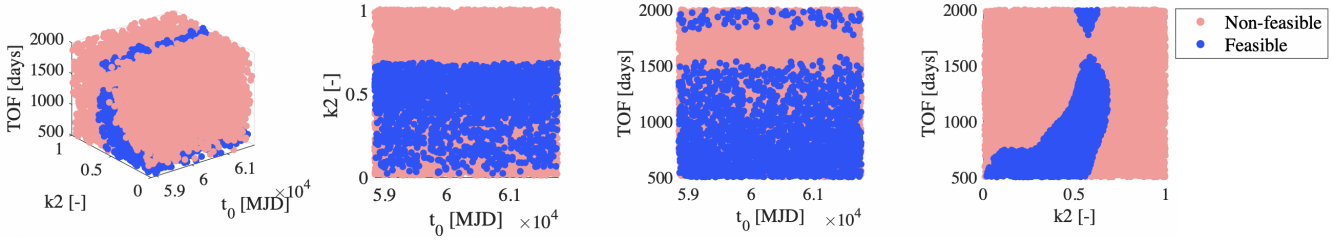


Fig. A.1: Trade-off plots for the Mars-Earth test case

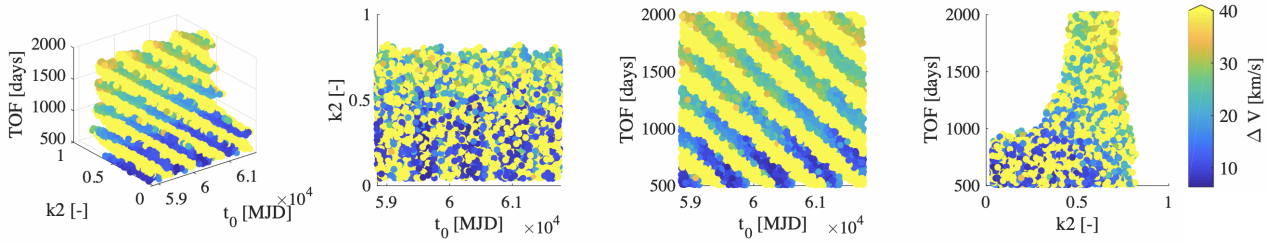
initial models given in Section VI, are shown in Figure A.4. From Figure A.4a it can be observed that, when applying the models on a mission scenario with a different range for t_0, in general the performance resembles that of the initial mission scenario. The largest differences are observed for GPR model 1 when predicting ΔV. Using hypothesis 1, this can be explained by the fact that the ratio between the minimum and maximum ΔV values is slightly larger for deviation 2 and 3 than for the initial mission scenario. When the models are applied on a mission scenario with a smaller range for t_0 and/or TOF, the performance gets better (refer to Figure A.4b). An exception is observed for GPR models 2, where the performance gets slightly worse for the second deviation. A possible explanation is found in hypothesis 2, since the percentage of ΔV used for the low-thrust acceleration (ΔV_{LT}) is smaller for deviation 2 than



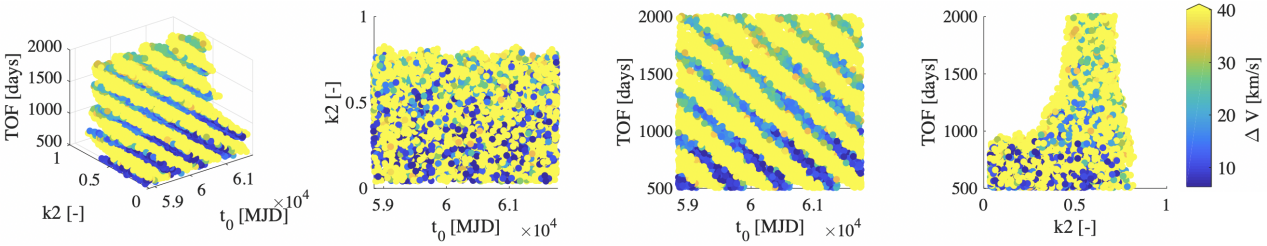
(a) Actual distribution of feasible trajectories



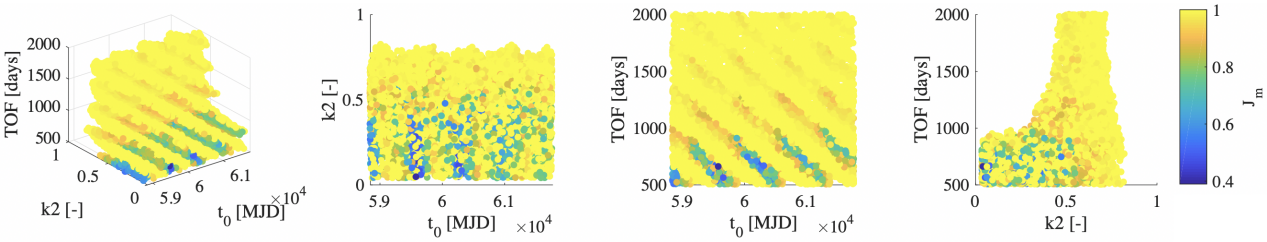
(b) Predicted distribution of feasible trajectories



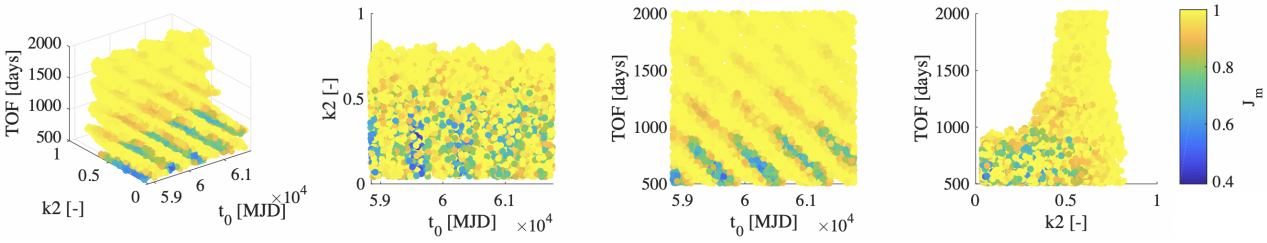
(c) Actual ΔV distribution



(d) Predicted ΔV distribution

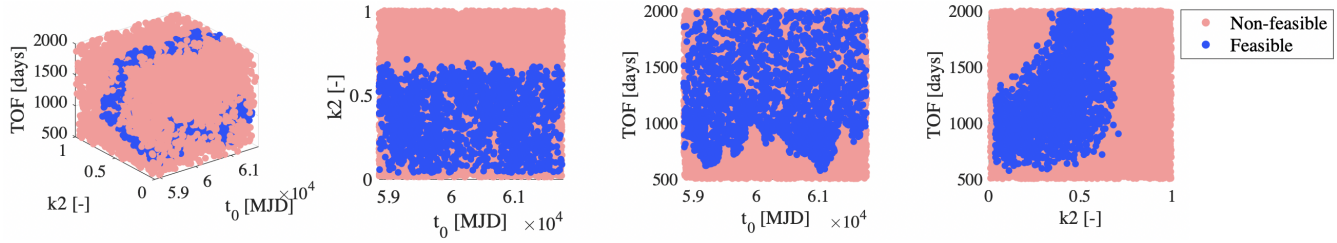


(e) Actual J_m distribution

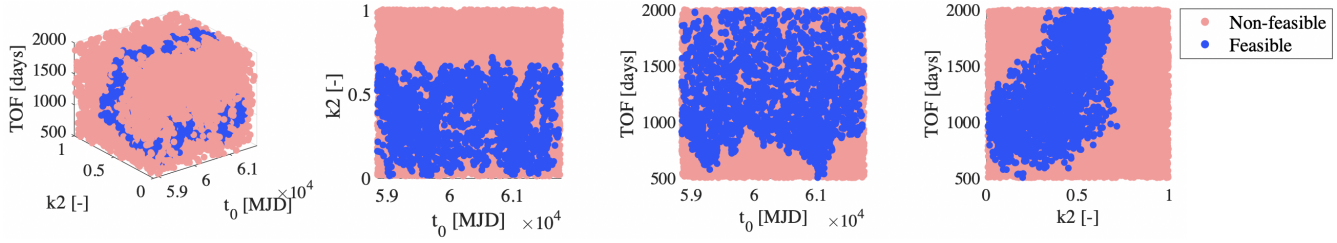


(f) Predicted J_m distribution

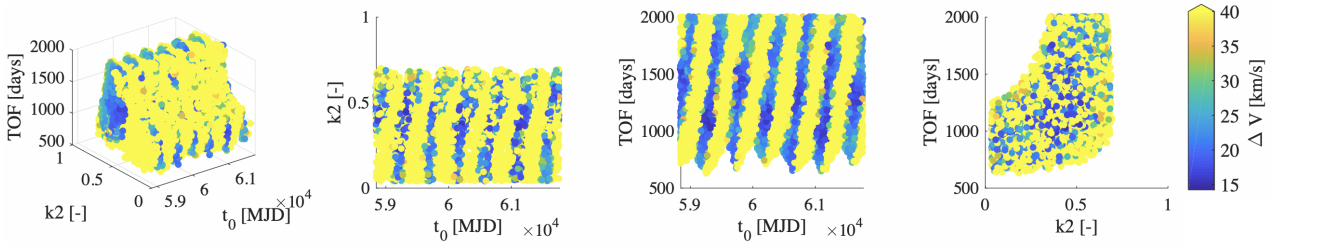
Fig. A.2: Plots of actual and predicted distributions for the Mars-Earth mission test case



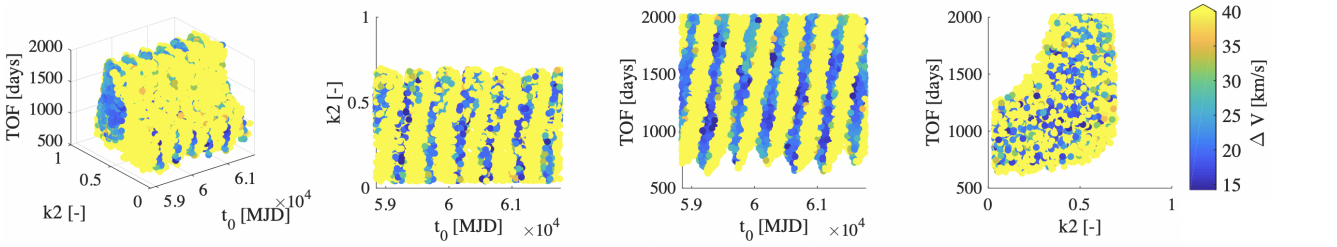
(a) Actual distribution of feasible trajectories



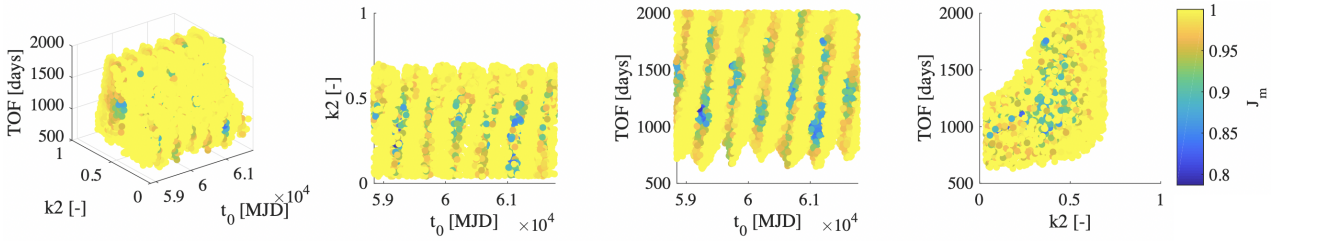
(b) Predicted distribution of feasible trajectories



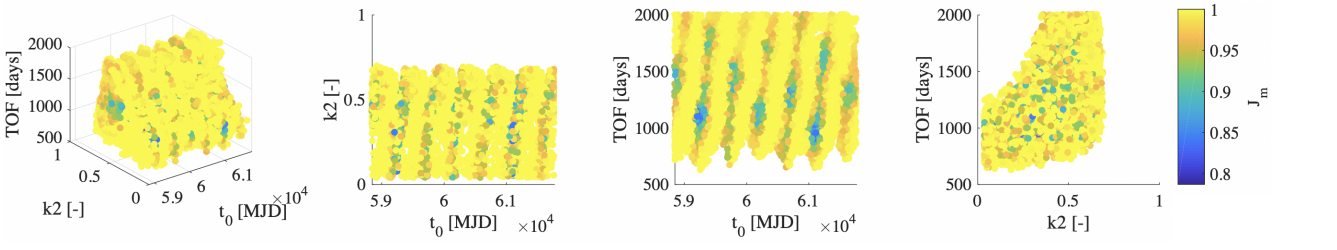
(c) Actual ΔV distribution



(d) Predicted ΔV distribution

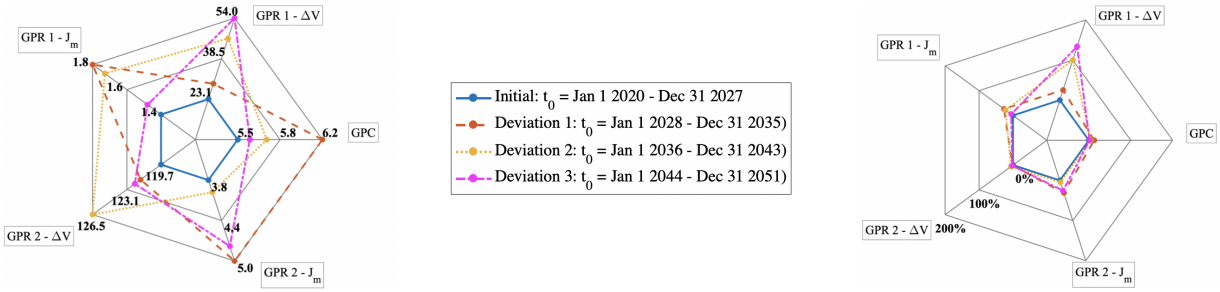


(e) Actual J_m distribution

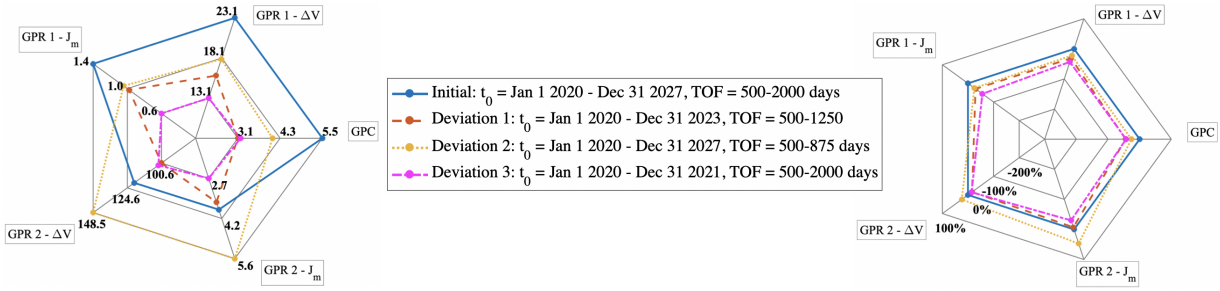


(f) Predicted J_m distribution

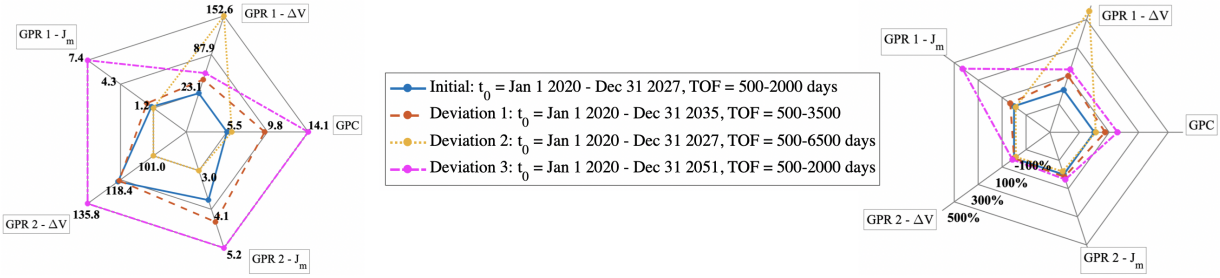
Fig. A.3: Plots of actual and predicted distributions for the Earth-Ceres mission test case



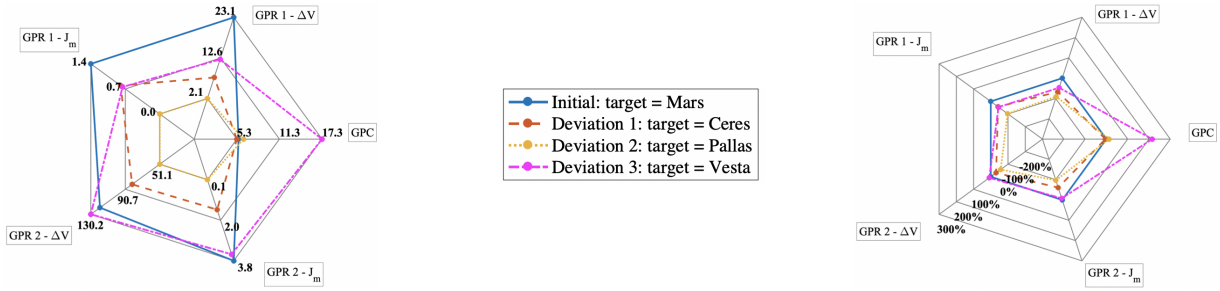
(a) Spider plots for different range of t_0



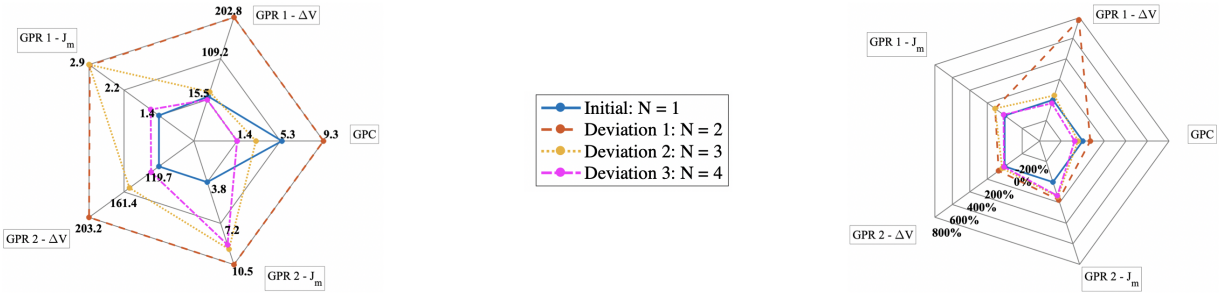
(b) Spider plots for smaller range of t_0 and/or TOF



(c) Spider plots for larger range of t_0 and/or TOF



(d) Spider plots for different target planet



(e) Spider plots for different number of revolutions

Fig. A.4: Spider plots indicating the robustness of the developed models for the Earth-Mars test case. Left: MAPE values, Right: relative difference in MAPE with respect to initial mission scenario (taken as 0%)

for the initial mission scenario. The better achieved results are in line with expectation, as with a decreasing range between input bounds, generally less training samples are required [14]. The opposite is true for a larger range of t_0 and/or TOF, as is observed in Figure A.4c, where the performance gets somewhat worse. For the performance of the mission scenarios to different target planets, it becomes clear from Figure A.4d that for the regression models the obtained MAPE's are, from high to low, ordered as: Pallas - Ceres - Vesta - Mars, which is inversely related to the order of the ratios of minimum to maximum target values. This ratio is the smallest for Pallas and the largest for Mars, which supports hypothesis 1. The performance for the GPC models is for all target planets comparable, except for Vesta, where worse performance is achieved. Finally, it remains to discuss the results when the initial models are applied on a mission scenario with a different number of revolutions. From Figure A.4e, it can be observed that in general the performance gets worse, especially for $N = 2$. The performance for the GPC models forms an exception, as higher accuracies are reached for $N = 3$ and $N = 4$. A possible explanation for these observations is the percentage of feasible trajectories. For $N = 2$, the percentage of feasible trajectories is larger than for $N = 1$. For $N = 3$, this percentage is lower than for $N = 1$, and for $N = 4$ even less feasible trajectories exist. This same rationale could be used to explain the worse performance of the GPC model for target planet Vesta. However, more work is required to confirm this rationale.

A

Orbital Mechanics

The work presented in Chapter 2 focuses on the preliminary optimization of low-thrust trajectories, which are modeled according to the exposin shape with the addition of impulsive ΔV's to approximate three-dimensional mission design. This section discusses the reference frame and coordinate systems used to implement the trajectory computation in the data generation program.

A.1. ECLIPJ2000 Reference Frame

The reference frame used in this work is the ECLIPJ2000, where the fundamental plane is the ecliptic plane. Most planetary orbits in the Solar System have relatively small inclinations to the ecliptic plane [31], which makes it convenient as a reference frame. The X-axis lies in the ecliptic plane with its direction towards the mean equinox at J2000. The Z-axis is orthogonal to the ecliptic plane and the Y-axis completes the right-handed system. The origin of the reference frame is taken as the center of the Sun.

A.2. Coordinate Systems

Two types of coordinate systems are used in this work: the Cartesian and polar coordinate systems, which are shown in Figures A.1 and A.2, respectively, and will be discussed here [7].

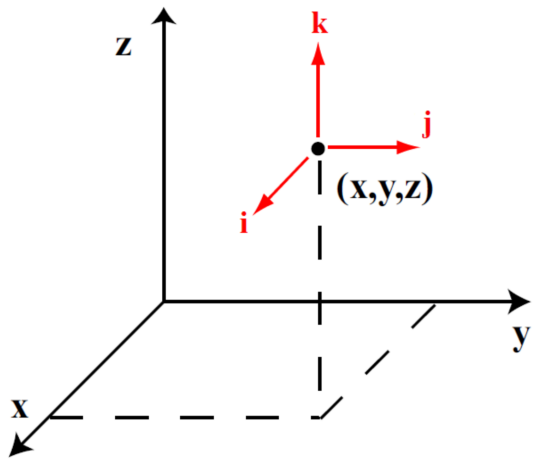


Figure A.1: Cartesian coordinate system

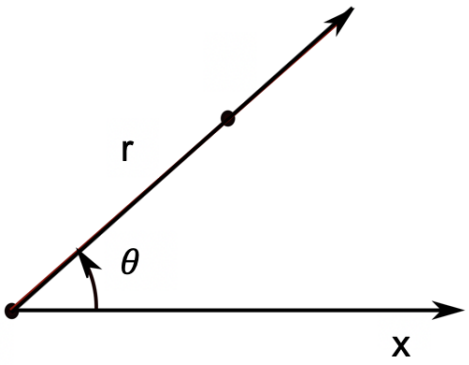


Figure A.2: Polar coordinate system

A.2.1. Cartesian Coordinate System

In the Cartesian coordinate system, the position and velocity of the point mass are specified by the linear distances (x, y, z) and their rate of change $(\dot{x}, \dot{y}, \dot{z})$ along three fixed orthogonal directions, which are, respectively, the X-,Y- and Z-axes. The coordinates of a point-mass in the Cartesian coordinate system can lie anywhere in

the interval $(-\infty,\infty)$. The position and velocity vectors of the planets are in this work expressed in Cartesian coordinates.

A.2.2. Polar Coordinate System

The polar coordinate system is a two-dimensional coordinate system, which is used to express the position along the exposin. This coordinate system is specified by the radial distance from the origin r and the polar angle θ, which indicates the positive rotation from the x-axis. Although the exposin is expressed in a two-dimensional coordinate system, three-dimensional mission design is approximated by the addition of impulsive ΔV's at both the departure and arrival positions.

B

Appendix: Low-Thrust Propulsion

Since this study focuses on the design of low-thrust trajectories, it is important that assumptions could be made on the low-thrust propulsion system, for example on the values of I_{sp} and F_{max}. Therefore, some basic knowledge about low-thrust propulsion techniques and understanding of the dynamics of low-thrusting is required. Contrary to high-thrust, thrust levels achieved with low-thrust propulsion systems are commonly in the order of about 0.1 N [33].

B.1. Fundamental Equations

The thrust of a rocket engine is given by [31]:

$$T = \dot{m} V_e \tag{B.1}$$

where $\dot{m}$ is the mass flow leaving the rocket engine nozzle per unit of time, and V_e is the exhaust velocity. The thrust can also be expressed as a function of available power W:

$$T = \sqrt{2\dot{m}W} = \frac{2W}{V_e} \tag{B.2}$$

From Equation B.2, it follows that for a fixed value of available power, the available thrust decreases with increasing exhaust velocity. Therefore, to achieve high thrust levels with limited propellant consumption, high exhaust velocities and high power levels are required. It also becomes clear from Equation B.2 that the thrust depends linearly on the input power. One of the key parameters of a rocket engine is the value of I_{sp}, which is expressed as the ratio of the thrust to the propellant mass flow in units of weight [30]:

$$I_{sp} = \frac{T}{\dot{m} g_0} = \frac{V_e}{g_0} \tag{B.3}$$

with $g_0 = 9.81$ m/s^2. The I_{sp} value gives an indication of the efficiency of a rocket engine. Generally, exhaust velocities of electric propulsion systems are high. Therefore, the I_{sp} values are high as well, which indicates that the electric propulsion system is highly efficient.

B.2. Characteristics of Electric Propulsion Missions

Several electric propulsion systems and their main characteristics are provided in Table B.1. It becomes clear that the thrust levels that can be achieved with current electric propulsion technology are very low and the values for specific impulse are very high. Especially for an ion engine, an I_{sp} of up to 6000 s can be obtained. As in this work an ion electric propulsion system is considered, the I_{sp} value is taken as 3500 s, which lies within the 2000-6000 s range.

Table B.1: Characteristics of various electric propulsion systems [33]

Propulsion system	I_{sp} [s]	**Electric power/thrust [kW/N]**	**Efficiency [%]**	**Thrust range [N]**
Resistojet	150-700	1.3-2	60	0.005-0.5
Arcjet	450-1500	6-15	91-95	0.05-5
Ion	2000-6000	22-36	87-91	5×10^{-6} - 0.5
Hall effect thruster	1500-2500	16-19	91-93	$5\cdot10^{-6}$ - 0.1
Magnetoplasmadynamic	2000	10-19		25-200
Pulsed plasma	1500	83-100	80	$5\cdot10^{-6}$ - 0.005
Pulsed inductive	2500-4000	40	70	2-200

Furthermore, the main characteristics of four low-thrust missions are shown in Table B.2. It can be seen that for BepiColombo a maximum thrust of 290 mN is feasible, which corresponds for a spacecraft of 1000 kg to a thrust acceleration of $2.9\cdot10^{-4}$ m/s^2. This compares to the thrust acceleration of Smart I of $2\cdot10^{-4}$ m/s^2 with a weight of 367 kg [7]. It is therefore assumed that for the preliminary design of low-thrust trajectories, thrust accelerations of up to $3.0\cdot10^{-4}$ m/s^2 are feasible with current technology. The weight of the spacecraft considered in this work is taken as 1000 kg.

B.3. Power Source

Mainly two types of power sources exist for electric propulsion systems: solar electric propulsion (SEP) and nuclear electric propulsion (NEP) [22]. In this work, only NEP is considered, as it is not dependent on the heliocentric radius distance, and the available power is therefore constant throughout the mission. The available power for NEP ranges between 5-300 kW [33].

Table B.2: Characteristics of electric propulsion systems used in past space missions [27][15][26][3]

Mission	I_{sp} [s]	**Input power [kW]**	**Thrust range [mN]**	ΔV **[km/s]**
Deep Space 1	1900 - 3200	0.5 - 2.3	19 - 92	4.5
Hayabusa	2900	0.3 - 11	5 - 25	22
Dawn	1900 - 3200	0.5 - 2.6	19 - 92	10.7
BepiColombo	4300	7 - 14	100 - 290	5.8

C

Appendix: Numerical Integration

In this work, the Midpoint method is used for numerical integration, which is a one step method that solves differential equations numerically. The integral is approximated by interpolating f in the midpoint $\bar{x}_k = (x_{k-1} + x_k)/2$ of m intervals I_k. The approximation of the integral is given by Equation C.1 [9]:

$$I_{mp}(f) = h \sum_{k=1}^{m} f(\bar{x}_k) \tag{C.1}$$

The error ε between the approximation and the exact value of the integral decreases, according to Equation C.2, with the second power of the spacing h:

$$\varepsilon_{mp} = I(f) - I_{mp}(f) = \frac{b-a}{24} h^2 \ddot{f}(x) \qquad \text{with} \quad x \in [a, b] \tag{C.2}$$

with $I(f)$ the exact value of the integral and $I_{mp}(f)$ the value of the integral approximated by the Midpoint method. The Midpoint method has been chosen due to the combination of small relative error and fast execution time, as will be explained here. From the equations for the computation of the normalized thrust acceleration a and the velocity change ΔV_{LT} (refer to Equations 5 and 8 in conference paper), it becomes clear that ΔV_{LT} along the low-thrust arc is computed by integration of the thrust acceleration F, which is in turn obtained by the computation of a (refer to Equation 2 in conference paper). When a one-step method is used, a has to be computed at the same number of steps as ΔV_{LT}. If a multi-step integration method (e.g. Runge-Kutta) would be implemented, a should be computed for a larger number of steps, thereby significantly increasing the CPU time and complexity of the program. To reduce CPU time and avoid this complexity in the data generation program, a one-step integration method is preferred over a multi-step method. Out of the possible one-step methods, four methods have been investigated: the Euler method, the Midpoint method, the Trapezoidal method, and Simpson's method [4], which have been tested for accuracy and computational speed. The latter three methods showed equal performance in terms of accuracy and CPU time. Worse accuracies were achieved for the Euler method at comparable CPU time. As the Midpoint, Trapezoidal, and Simpson's method resulted in the same performance, the Midpoint method is chosen due to its straightforward implementation.

D

Appendix: Optimization

The optimization techniques used in this work to find optimal trajectories are the differential evolution, the grid search, and the adaptive grid search. The conjugate gradient method is used for the optimization of the hyperparameter values. Each of them will be discussed here.

D.1. Differential Evolution (DE)

Storn and Price developed the method of DE [29], which is both efficient and able to find the overall best trajectory, i.e. the global optimum. In the DE a population of N solution candidates ($\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N$) is used, where each of the candidates is a vector containing the n optimization parameters. Firstly, an initial population is generated, which consists of candidates that could be located in the entire solution space. Next, for each candidate vector $\boldsymbol{x}_i$ a trial vector $\boldsymbol{z}_i$ is created. The candidate vector $\boldsymbol{x}_i$ is replaced by the trial vector $\boldsymbol{z}_i$ if the objective function value of vector $\boldsymbol{z}_i$ is lower than that of vector $\boldsymbol{x}_i$. This process is repeated until the global minimum or a specified number of iterations is obtained.

In the classical version of the DE, as developed by Storn and Price, three intermediate solution vectors ($\boldsymbol{x}_{p(1)}, \boldsymbol{x}_{p(2)}$ and $\boldsymbol{x}_{p(3)}$) are randomly generated for each solution vector $\boldsymbol{x}_i$. These are then, together with the mutation probability F_p, used to compute a new vector $\hat{\boldsymbol{x}}_i$ according to Equation D.1:

$$\hat{\boldsymbol{x}}_i = \boldsymbol{x}_{p(1)} + F_p \cdot (\boldsymbol{x}_{p(2)} - \boldsymbol{x}_{p(3)}) \tag{D.1}$$

The trial vector is found by applying crossover to $\boldsymbol{x}_i$ and $\hat{\boldsymbol{x}}_i$ in the following way:

$$\boldsymbol{z}_i^j = \begin{cases} \hat{\boldsymbol{x}}_i^j & \text{if} \quad R^j \leq C_R \quad \text{or} \quad j = W_i \\ \boldsymbol{x}_i^j & \text{if} \quad R^j > C_R \quad \text{and} \quad j = W_i \end{cases} \tag{D.2}$$

with j indicating the j^{th} component of the corresponding vector and $R_j \in (0,1)$ randomly determined for each j. C_R is the crossover probability, which is usually a value between 0.5 and 1.0, and W_i is a integer, which is randomly chosen between 1,2,..,n. This procedure ensures that $\boldsymbol{z}_i$ gets at least one parameter from $\hat{\boldsymbol{x}}_i$.

As the DE algorithm is both simple and straightforward and has demonstrated to be highly robust and efficient for complex optimization problems [1] [13], it is chosen in this work. Furthermore, it has also been used for the optimization of the decision vector in the multi-revolution Lambert problem for exposins proposed by Izzo [12].

D.2. Grid Search (GS)

In a GS, the input space is discretized using an enumerative search, i.e. a grid, after which the objective function is evaluated at each grid point. The grid point where the best value for the objective function is found, is assumed to be the global minimum. The GS method is disadvantageous for large search spaces, because either the computational burden becomes extensive or the grid spacing becomes very large, resulting in limited accuracy. Grid searches might be useful for optimization of the launch date t_0 and TOF by stepping through

the possible launch dates and transfer times in discrete steps, which was done in the work by Novak and Vasile [20].

D.3. Adaptive Grid Search (AGS)

The AGS provides a solution to overcome the disadvantage in CPU time of the GS technique [19], and has therefore been used as the third optimization technique. In the AGS, first a coarse grid (G_0) is used. After the evaluation of all grid points in the course grid, the grid is refined around the interim optimum by halving the length of the search interval (G_1). This procedure is repeated a specified number of times or until a specified optimal value is reached. In this work, it is repeated until the optimum at G_i is equal to that at G_{i-10} and it is therefore assumed that convergence is reached. Although it is advantageous with respect to CPU time, this technique is prone to converging to local optima.

D.4. Conjugent Gradient Method

The optimization of the marginal likelihood to find the hyperparameters, as discussed in Section IV in the conference paper, is performed using the conjugent gradient method, which is built-in in the gpml toolbox [24]. The one that is implemented is an algorithm for the numerical solution of linear equations, whose matrix is symmetric and positive-definite. A detailed description is provided in the book by Hestenes and Stiefel [11]. To find the hyperparameter values by maximizing the marginal likelihood, we look for the partial derivatives of the marginal likelihood w.r.t. the hyperparameters [25]:

$$\frac{\partial}{\partial \theta_j} \log p(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{\theta}) = \frac{1}{2}\boldsymbol{Y}^T K^{-1} \frac{\partial K}{\partial \theta_j} K^{-1} \boldsymbol{Y} - \frac{1}{2}\mathrm{tr}\left(K^{-1}\frac{\partial K}{\partial \theta_j}\right) \tag{D.3}$$

with tr the trace of a matrix. The complexity lies in inverting the covariance matrix K. Standard methods for the inversion of positive definite symmetric matrices require time $O(n^3)$ for inversion of an $n \times n$ matrix. After the determination of K^{-1}, the derivatives in Equation D.3 need to be computed, which requires only $O(n^2)$ per hyperparameter. As the computational overhead of computing derivatives is small, using a gradient-based optimizer is advantageous [25]. It should be noted that there is no guarantee that the gradient-based method does not converge to one of the local optima of the marginal likelihood. Empirical evidence indicates that, although local optima exist, it is not a devastating problem in terms of final performance to converge to one of the local optima instead of the global one [25].

E

Appendix: Gaussian Process Regression and Classification

GPR and GPC are both forms of supervised learning, which is the machine learning task of learning a function that maps an input to an output based on empirical data (i.e. the training dataset). It is therefore concerned with predicting the output at new inputs, when some observations of dependent output are known at certain independent inputs. Regression means that it is concerned with a continuous output. In classification problems, the output is a discrete value [25].

E.1. The Advantages of Using Gaussian Process Models

The advantage of a GP is its ability to provide a conscientious, practical, and probabilistic approach to learning in kernel machines. This is beneficial with respect to the interpretation of model predictions and provides a well-founded framework for learning and model selection. When considering processes which are Gaussian, the computations that are required for inference and learning become relatively easy. As a result, supervised learning problems in machine learning, which can be considered as learning a function from examples (i.e. training data), can be cast straight into the GP framework [25].

Neural networks became popular mainly because they allowed the use of adaptive basis functions, as opposed to linear models. The adaptive basis functions in a neural network could "learn" hidden features useful for the problem at hand. However, this adaptivity came at the cost of other problems, e.g. the lack of a principled framework. When using a GP, one is able to add prior knowledge and specifications about the shape of the model by selecting different kernel functions. In case this prior knowledge is not available, the model development procedure as proposed in this paper might offer a solution. GP's are mathematically equivalent to many models, including Bayesian linear models, spline models, large neural networks, and are closely related to e.g. support vector machines. However, under the GP viewpoint the models may be easier to handle and interpret. An example is that a GP directly captures the model uncertainty. GP models provide a distribution for the prediction value, rather than just one value as the prediction. For neural networks this is not the case. One of the main advantages of the GP framework is that it combines a sophisticated and consistent view with computational tractability [25].

E.2. The Basics of Gaussian Process Models

As discussed in Section IV of the conference paper, a GP over the function $f(\boldsymbol{x})$ is defined by [25]:

$$f(\boldsymbol{x}) \sim GP(\mu(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}^*)) \tag{E.1}$$

where $\mu(\boldsymbol{x})$ and $k(\boldsymbol{x}, \boldsymbol{x}^*)$ are the mean function and covariance function, respectively. Both functions are discussed in more detail hereafter, followed by the likelihood function and the inference method.

E.2.1. Mean Function

The mean function, specified as $\mu(\boldsymbol{x})$, dominates the forecast in regions far from the data. Therefore, the choice of the prior mean function can have a large impact on the data. In most cases, one is unsure about whether a trend is up or down, i.e. prior knowledge is not available. In that case, either a flat, often zero-offset, mean function could be selected, as is done in most papers [28] [6], or several different mean functions could be tried to select the best performing one. The latter is incorporated in the model development procedure as proposed in this work. When e.g. the function has a linear drift term, much better performance could be achieved by choosing a mean function that fits this trend. The model development procedure as proposed in this work tries to find a mean function that can fit the problem at hand. It should be noted that, when the input space becomes multi-dimensional, it is hard to visualize the distribution and it is therefore less likely that prior knowledge is available. When a mean function other than zero is selected, the mean function contains hyperparameters, which conceal domain knowledge regarding the deterministic component.

From the model development procedure, the following five mean functions turned out to be of importance for the problems considered in this work (refer to Section X in the conference paper):

$$\mu_{\text{polynomial}}(\boldsymbol{x}) = \sum_{i=1}^{D} \sum_{j=1}^{d} a_{ij} \cdot \boldsymbol{x}_i^j \tag{E.2}$$

$$\mu_{\text{linear}}(\boldsymbol{x}) = \sum_{i=1}^{D} c_i \cdot \boldsymbol{x}_i \tag{E.3}$$

$$\mu_{\text{constant}}(\boldsymbol{x}) = c \tag{E.4}$$

$$\mu_{\text{WSPC}}(\boldsymbol{x}) = \sqrt{\frac{2}{p}} \sum_{j=1}^{p} a_j \cdot \cos\left(\text{inv}(w_j) \cdot \boldsymbol{x} + b_j\right) \tag{E.5}$$

$$\mu_{\text{scaled}}(\boldsymbol{x}) = a \cdot \mu_0(\boldsymbol{x}) \tag{E.6}$$

with D the dimension of the input space, d the number of polynomials, p the number of projected cosines, and a, b, c, and w hyperparameters to be selected. Furthermore, $\mu_0(\boldsymbol{x})$ is the mean function to be scaled. It should be noted that significantly better performance is obtained for all models than what could be achieved with a zero mean function.

E.2.2. Covariance Function

The covariance function, specified as $k(\boldsymbol{x}, \boldsymbol{x}^*)$, represents some form of distance or similarity between input points. It is likely to assume that points with inputs $\boldsymbol{x}$, which are close together, are likely to have similar target values y, and therefore training points close to a test point should be informative about the prediction at that test point. It is the covariance function that defines this measure of nearness or similarity. The most important properties of covariance functions could be distinguished as [25]:

- *Stationary*: when a covariance function is a function of $\boldsymbol{x} - \boldsymbol{x}^*$ and is thus invariant to translations in the input space.
- *Isotropic (iso)*: when a covariance function is a function only of $|\boldsymbol{x} - \boldsymbol{x}^*|$ and is therefore invariant to all rigid motions.
- *Characteristic length-scale* ℓ: describes how smooth a function is. When the length-scale is small, it means that function values can change quickly. For large values of ℓ, functions can only change slowly. Furthermore, length-scales determine how far we can reliably extrapolate from the training data (i.e. how far do you need to move (along a particular axis) in the input space for the function values to become uncorrelated).
- *Signal variance* σ^2: is a scaling factor that determines the variation of function values from their mean. When σ^2 is small, this indicates that a function stays close to its mean value, while larger values allow for more variation. In case the signal variance is too large, the modeled function will start to chase outliers.
- *Automatic relevance determination*: when a covariance function implements automatic relevance determination (ard) [18], the inverse of the length-scale determines how relevant an input is. The covariance

will become almost independent of the input if it has a very large length-scale, and it will therefore effectively be removed from the inference. For the removal of irrelevant input, ard has been used successfully by several researchers [34].

The covariance functions that were selected for the models in this work, determined using the model development procedure, turned out to be all stationary, and are therefore only a function of Euclidean distance. Stationary covariance functions decay monotonically with $|\boldsymbol{x} - \boldsymbol{x}^*|$ and are always positive [25]. A downside of stationary GPs is that they fail to adapt to changeable smoothness in the function of interest, which is of particular importance when the function varies more quickly in certain parts of the input space than in others [25]. Using a non-stationary covariance function, the model is allowed to adapt to functions whose smoothness changes with the inputs and is thus able to fit properly a variable smoothness prior. A downside is that a non-stationary GP model requires significantly more parameters than a stationary GP model, especially for a growing dimension, losing the attractive simplicity of the stationary GP model. The result might be slow computation, which limits the feasibility of the model to approximately $M < 1000$ [21]. The families of covariance functions that are of importance in this work are discussed hereafter.

Squared Exponential Covariance Function

The standard form of the squared exponential (SE) covariance function is given by [25]:

$$k(\boldsymbol{x}, \boldsymbol{x}^*)_{\text{SE}} = \exp\left(-\frac{(\boldsymbol{x} - \boldsymbol{x}^*)^2}{2\ell^2}\right) \tag{E.7}$$

where ℓ is the characteristic length-scale. When the characteristic length-scale is short, the function would vary rapidly, and vice-versa for a long characteristic length-scale. The SE covariance function is infinitely differentiable (i.e. the GP with this covariance function has mean square derivatives of all orders) and is thus very smooth [25]. The SE is probably the most widely-used covariance function in the field of machine learning. In this work two variants of the SE covariance functions are used: one with isotropic distance measure (SEiso) and one with automatic relevance determination distance measure (SEard), respectively given by Equations E.9 and E.8:

$$k(\boldsymbol{x}, \boldsymbol{x}^*)_{\text{SEard}} = s^2_{f,\text{SEard}} \cdot e^{-(\boldsymbol{x}-\boldsymbol{x}^*)' \cdot \text{inv}(P) \cdot \frac{\boldsymbol{x}-\boldsymbol{x}^*}{2}} \tag{E.8}$$

$$k(\boldsymbol{x}, \boldsymbol{x}^*)_{\text{SEiso}} = s^2_{f,\text{SEiso}} \cdot e^{-(\boldsymbol{x}-\boldsymbol{x}^*)' \cdot \text{inv}(P) \cdot \frac{\boldsymbol{x}-\boldsymbol{x}^*}{2}} \tag{E.9}$$

with s^2_f the signal variance. For the covariance functions with SEard, the P matrix is diagonal with automatic relevance determination parameters $\ell^2_1, ..., \ell^2_D$. For SEiso, the P matrix is ℓ^2 times the unit matrix.

Rational Quadratic Covariance Function

The rational quadratic (RQ) covariance function is of the form:

$$k(\boldsymbol{x}, \boldsymbol{x}^*)_{\text{RQ}} = \left(1 + \frac{(\boldsymbol{x} - \boldsymbol{x}^*)^2}{2\alpha\ell^2}\right)^{-\alpha} \tag{E.10}$$

with $\alpha > 0$ and $\ell > 0$ [25].

In this work, the RQard and RQiso covariance functions are used, given by Equations E.11 and E.12, respectively:

$$k(\boldsymbol{x}, \boldsymbol{x}^*)_{\text{RQard}} = s^2_{f,\text{RQard}}[1 + \frac{(\boldsymbol{x} - \boldsymbol{x}^*)'\text{inv}(P)(\boldsymbol{x} - \boldsymbol{x}^*)}{2\alpha}]^{-\alpha} \tag{E.11}$$

$$k(\boldsymbol{x}, \boldsymbol{x}^*)_{\text{RQiso}} = s^2_{f,\text{RQiso}}[1 + \frac{(\boldsymbol{x} - \boldsymbol{x}^*)'\text{inv}(P)(\boldsymbol{x} - \boldsymbol{x}^*)}{2\alpha}]^{-\alpha} \tag{E.12}$$

with the P matrices equal to those for the SE covariance functions.

Piecewise Polynomial Covariance Function

The piecewise polynomial (PP) covariance function is specified by the number of polynomials d (equal to 0,1,2, or 3). These covariance functions are $2d$-times continuously differentiable, and thus the corresponding processes are d-times mean-square differentiable. The standard form of the covariance function is given by [25]:

$$k(\boldsymbol{x},\boldsymbol{x}^*)_{\text{PP}} = \max\left(1-\sqrt{\frac{|\boldsymbol{x}-\boldsymbol{x}^*|}{\ell^2}},0\right)^{j+d} \cdot f\left(\sqrt{\frac{|\boldsymbol{x}-\boldsymbol{x}^*|}{\ell^2}},j\right) \tag{E.13}$$

with j the signal variance, given by $j = \text{floor}(D/2)+d+1$.

Again, an ard and iso variant are used in this work:

$$k(\boldsymbol{x},\boldsymbol{x}^*)_{\text{PPard}} = s^2_{f,\text{PPard}} \cdot \max(1-r,0)^{j+d} \cdot f(r,j) \tag{E.14}$$

$$k(\boldsymbol{x},\boldsymbol{x}^*)_{\text{PPiso}} = s^2_{f,\text{PPiso}} \cdot \max(1-r,0)^{j+d} \cdot f(r,j) \tag{E.15}$$

where the distance r is defined as:

$$r = \sqrt{(\boldsymbol{x}-\boldsymbol{x}^*)'\text{inv}(P)(\boldsymbol{x}-\boldsymbol{x}^*)} \tag{E.16}$$

with the P matrices equal to those for the SE and RQ covariance functions. The function $f(r,j)$ is dependent on the number of polynomials d, which is given in Equation E.17 for the values of d used in this work.

$$\begin{cases} d=0: & f(r,j)=1 \\ d=1: & f(r,j)=1+r(j+1) \\ d=2: & f(r,j)=1+r(j+2)+(j^2+4j+3)/3r^2 \end{cases} \tag{E.17}$$

Matérn Covariance Function

The Matérn class of covariance functions is given by [25]:

$$k(\boldsymbol{x},\boldsymbol{x}^*)_{\text{MATÉRN}} = \frac{2^{1-\nu}}{\Gamma\nu}\left(\frac{\sqrt{2\nu}|\boldsymbol{x}-\boldsymbol{x}^*|}{\ell}\right)^{\nu} K_\nu\left(\frac{\sqrt{2\nu}|\boldsymbol{x}-\boldsymbol{x}^*|}{\ell}\right) \tag{E.18}$$

where Γ is the standard Gamma function, ν and ℓ are positive parameters of the covariance function, and K_ν is the modified Bessel function of second order. The values for ν considered in this work are $\nu = 1/2$, $\nu = 1/3$ and $\nu = 1/5$. The Matérn covariance function can be seen as a generalization of the SE covariance function. Due to its finite differentiability (for finite ν), it is generally better able to capture physical processes [25].

The ard version of the Matérn covariance function is used in this work and is given by:

$$k(\boldsymbol{x},\boldsymbol{x}^*)_{\text{MATÉRNard}} = f(\sqrt{(\nu)}\cdot r)\cdot e^{-\sqrt{0}\cdot r} \tag{E.19}$$

where $f(t)=1$ for $\nu=1/2$, $f(t)=1+t$ for $\nu=1/3$, and $f(t)=1+t+t^2/3$ for $\nu=1/5$.

E.2.3. Likelihood Functions

A likelihood function $p(\boldsymbol{Y}|\boldsymbol{f})$ is a conditional density defined for scalar latent function values $\boldsymbol{f}$ and output vector $\boldsymbol{Y}$. Likelihood functionality is of importance both during training and prediction. During training, the values of the hyperparameters are determined, which will be discussed in the following subection. How the likelihood becomes useful during prediction will be discussed here.

A prediction at input $\boldsymbol{x}^*$ using the dataset $\mathscr{D} = (\boldsymbol{X},\boldsymbol{Y})$ consists of the predictive mean μ_{y^*} and variance $\sigma^2_{y^*}$, which are computed from the latent marginal moments μ_{f^*} and $\sigma^2_{f^*}$ (i.e. the Gaussian marginal approximation $\mathcal{N}(f^*|\mu_{f^*},\sigma^2_{f^*})$), using [24]:

$$p(y^*|\mathscr{D},\boldsymbol{x}^*) = \int p(y^*|f^*)p(f^*|\mathscr{D},\boldsymbol{x}^*)df^* \tag{E.20}$$

$$= \int p(y^*|f^*)\mathcal{N}(f^*|\mu_{f^*},\sigma^2_{f^*})df^* \tag{E.21}$$

Using the moments of the likelihood $\mu_{f^*} = \int y^* p(y^*|f^*)dy^*$ and $\sigma^2_{f^*} = \int (y^* - \mu_{f^*}) p(y^*|f^*)dy^*$, the following (exact) expressions could be obtained for the predictive moments [24]:

$$\mu_{y^*} = \int \mu_{f^*} p(f^*|\mathscr{D}, \boldsymbol{x}^*)df^* \tag{E.22}$$

$$\sigma^2_{y^*} = \int [\sigma^{*2}_f + (\mu_{f^*} - \mu_{y^*})^2](f^*|\mathscr{D}, \boldsymbol{x}^*)df^* \tag{E.23}$$

Although several likelihood functions exist, only Gaussian likelihood is used in this work and is therefore the only one discussed here. As the posterior distribution is Gaussian, the computation can be performed analytically, which makes it the simplest likelihood function. Due to this simplicity, both training and prediction are executed faster than for other likelihood functions. Together with the fact that the achieved MAPE values are equal to or better than those obtained with other likelihood functions, it is selected for the GP models used in this work.

The Gaussian likelihood is given by [24]:

$$p(y_i|f_i) = \frac{e^{\frac{-(y_i - f_i)^2}{2s_n^2}}}{\sqrt{2\pi s_n^2}} \tag{E.24}$$

where s_n is the standard deviation of the noise, which is the only hyperparameter.

E.2.4. Inference Methods

Inference methods are used to compute the parameterization of the posterior, the (approximate) negative log marginal likelihood and its partial derivatives with respect to the hyperparameters [25]. Inference is therefore of importance during the training part, to compute the values of the hyperparameters, and during the prediction part, to compute an approximation to the posterior distribution of the latent variables f_i related to training cases $i = 1, ..., n$. This approximate posterior is assumed to be Gaussian. The three inference methods that are used in this work are discussed here.

Gaussian Noise Inference

When Gaussian noise inference is used in combination with Gaussian likelihood, as done in this work, computing the posterior reduces to computing the mean and covariance of a multivariate Gaussian, which can be done in an exact way by using matrix algebra. The Gaussian posterior resulting from this computation is exact.

Laplace Inference

For likelihoods which are differentiable, Laplace inference approximates the posterior by a Gaussian centered at its mean and matching its curvature. This inference method needs derivatives up to the third order for the fitting procedure. For more information, the reader is referred to the manual of the gpml toolbox [24].

LOO Inference

Using LOO inference, the posterior is approximated using a least-squares probabilistic method. A detailed explanation is provided in the book by Rasmussen [25].

E.3. Selection of the Hyperparameters

The mean, covariance, and likelihood functions previously discussed contain several hyperparameters, of which the optimal values have to be determined during the third phase of the model development procedure. The marginal likelihood is used to optimize the hyperparameter values, which is the integral of the likelihood times the prior:

$$p(\boldsymbol{Y}|\boldsymbol{X}) = \int p(\boldsymbol{Y}|\boldsymbol{f}, \boldsymbol{X}) p(\boldsymbol{f}, \boldsymbol{X}) d\boldsymbol{f} \tag{E.25}$$

The term "marginal" indicates that a marginalization is performed over the function values $\boldsymbol{f}$. In general, using Bayesian inference, the posterior is computed as:

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}} \tag{E.26}$$

which results for the computation of the hyperparameters in:

$$p(\boldsymbol{\theta}|\boldsymbol{Y},\boldsymbol{X}) = \frac{p(\boldsymbol{Y}|\boldsymbol{X},\boldsymbol{\theta})\,p(\boldsymbol{\theta})}{p(\boldsymbol{Y}|\boldsymbol{X})} \tag{E.27}$$

where the denominator, or the marginal likelihood, is given by Equation E.25.

The hyperparameters should be set such that the marginal likelihood is maximized. Therefore, the partial derivatives of the marginal likelihood w.r.t. the hyperparameters are sought. The optimal hyperparameters can be found by minimizing the objective function given in Equation 26 in the conference paper, making use of the conjugent gradient method, as was discussed in Section D.

E.4. Cross Validation for Model Selection

This subsection addresses how cross validation can be used for model selection. In cross validation the training set is split into two sets, of which one is actually used as training data and the other is used as validation set to monitor the performance [25]. A drawback of this might be that only part of the data set could be used for training. Furthermore, if the remaining validation set is small, the estimate of the performance might have a large variance. k-fold cross-validation could help resolve these problems. The data set is then split into k subsets of equal size. $k-1$ sets are used for training and a single subset is used for validation. This process is then repeated in total k times, such that each subset is selected once for validation. The benefit is that a large part of the data can be used as training data and the model is validated against all data. A slight drawback is that multiple models have to be trained. Values for k typically range between 3 to 10 [25], and is in this work set to 5.

F

Appendix: Verfication and Validation

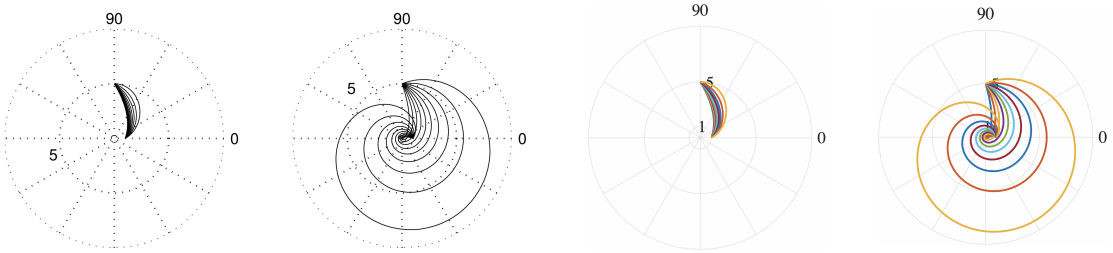
To assure that the results presented in this work are correct, verification is applied on several aspects of the work: the implementation of the exposin shape, the integration along the shape, and the optimization techniques used. For each of these aspects, both the method of verification and the results are provided.

F.1. Implementation of Exposin

F.1.1. Method

In order to ensure that the exposin shape is implemented correctly and the TOF and thrust acceleration along the arc are computed in the right way, two sources of verification have been used: the lecture notes proved by R. Noomen on exposins [19] and the paper written by Izzo [12].

The verification of the implementation of a class of exposins is done by reproducing the plots as provided in the work by Izzo. These plots, given in Figure F.1a, contain, for two different classes of exposins, all feasible exposins within these classes. The same classes of exposins are implemented in the algorithm used in this work (i.e. the data generation program), to find all feasible exposins belonging to these classes. The computation of the TOF is verified by another plot provided in Izzo's paper, which can be found in Figure F.2a. In this plot, the TOF versus the flight-path angle is given for the class $S_{1/12}[1, 1.5, \pi/2, N]$. This same class is implemented in the data generation program and the TOF at different flight-path angles is computed for $N = 0, 1, .., 5$. Finally, the computation of the thrust acceleration along the exposin should be verified. For this purpose, the plot provided in the lecture notes by R. Noomen [19] is used, which can be found in Figure F.3a. In this figure, the thrust acceleration versus the transfer angle is provided for an exposin defined by the parameters $\{k_0 = 1\,\text{AU}, k_1 = 0.5, k_2 = 2/9, \phi = -\pi/2\,\text{rad}\}$. In the data generation program the acceleration along the low-thrust arc is computed for an exposin with the same parameters.



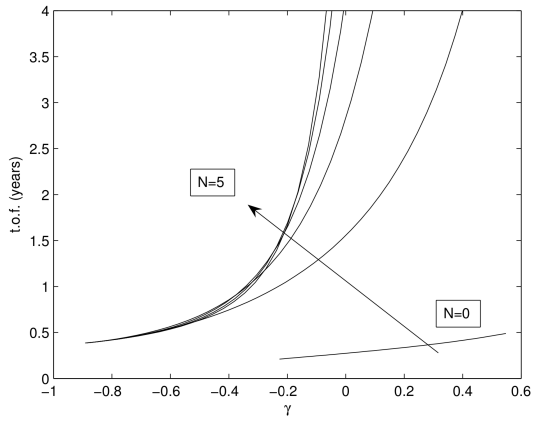
(a) Plots used for verification [12]

(b) Plots produced with data generation program

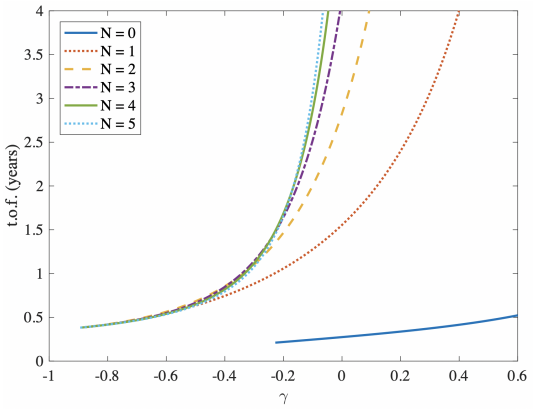
Figure F.1: Verification of exposin implementation: Feasible exposins in the classes $S_{1/2}[1, 5, \pi/2, 0]$ (left) and $S_{1/4}[1, 5, \pi/2, 1]$ (right)

F.1.2. Results

It can be observed that the produced plots for the class of exposins, the TOF curves, and the thrust acceleration, respectively given in Figures F.1b, F.2b, and F.3b, very closely resemble the plots used for verification. Although

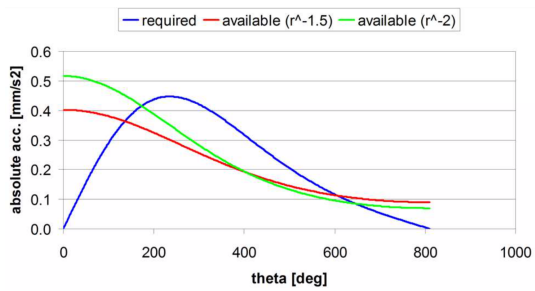


(a) Plot used for verification [12]

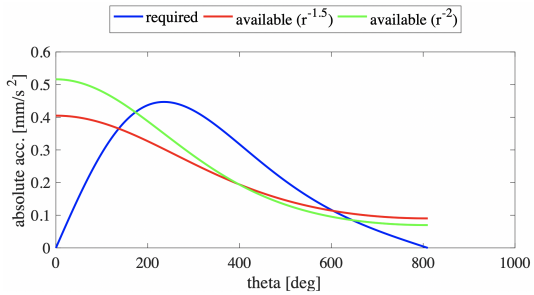


(b) Plot produced with data generation program

Figure F.2: Verification of TOF computation: Time-of-flight versus the flight-path angle for the class $S_{1/12}[1, 1.5, \pi/2, N]$



(a) Plot used for verification [12]



(b) Plot produced with data generation program

Figure F.3: Verification of thrust acceleration computation: Acceleration required for an exposin defined by parameters $\{k_0 = 1\text{AU}, k_1 = 0.5, k_2 = 2/9, \phi = -\pi/2\text{rad}\}$

no numerical data is available for quantitative comparison, from this qualitative verification it is concluded that the exposin shape is implemented in the right way and the corresponding TOF and thrust acceleration are computed correctly.

F.2. ΔV Computation

F.2.1. Method

With respect to the computation of ΔV, it is important to 1) verify the correct computation of the values for the impulsive shots (ΔV_0 and ΔV_f) and ΔV_{LT} along the low-thrust arc, and 2) validate the number of steps that is used in the Midpoint method for numerical integration of ΔV_{LT}. Additionally, it should be validated if this number of steps provides sufficient accuracy for the computation of the TOF.

The computation of the ΔV values is verified by the code provided by Chris Andre [2] in Python, which in turn used a paper of ESA for validation [32]. The reason for not directly verifying against this paper, is the fact that the necessary ephemeris data was not available in MATLAB. Five different input vectors have been chosen, which are provided in Table F.1. The verification data is generated by feeding these input vectors to the Python code. The same input vectors are fed into the data generation program written in MATLAB, and the absolute and relative differences of the ΔV values are compared.
For the numerical integration, it is important to determine the minimum number of steps required. In the work by Gondelach [7], it was assumed that ΔV needs to be known with an accuracy of 0.1% to be able to compare the ΔV's of different trajectories well. Therefore, an internal validation is performed to determine the number of steps at which this accuracy is achieved. Here a large number of steps, in this case 100,000,000, is assumed to lead to the exact value of ΔV and TOF. The input vector used corresponds to the best trajectory found using the exposin shape for the Earth-Mars mission test case, given by $\boldsymbol{x}$ = [62701.02, 0.1970, 562.74].

F.2.2. Results

The computed ΔV values for the five input vectors using the data generation program in MATLAB are given in Table F.1. The absolute and relative differences between the validation data and the computed values are presented in Table F.2. It can be observed that all absolute differences are smaller than $3 \cdot 10^{-6}$ m/s and all relative differences are smaller than $3 \cdot 10^{-8}$ %. As these values are negligible during preliminary optimization, it is concluded that the ΔV computation is done correctly.

The computed values for ΔV at different number of steps are given in Table F.3. As the value belonging to 100,000,000 is taken as the reference value, the absolute differences and relative differences are computed compared to ΔV achieved at 100,000,000 steps. Taking the criterium as defined by Gondelach into account [7], 400 steps would be sufficient. To ensure some margin of correct computation, the number of steps used in this work is set to 1000. When considering the TOF computation, it becomes clear from Figure F.4 that using 1000 steps results in a relative error smaller than $5 \cdot 10^{-6}$ %. As this error is significantly smaller than the one reached for the ΔV computation, 1000 steps is deemed sufficient for the TOF computation.

Table F.1: Verification of ΔV computation

Input vector				**Verification data**				**Results**			
t_0 (MJD)	k_2 (-)	TOF (days)	N (-)	ΔV_0 (m/s)	ΔV_f (m/s)	ΔV_{LT} (m/s)	ΔV (m/s)	ΔV_0 (m/s)	ΔV_f (m/s)	ΔV_{LT} (m/s)	ΔV (m/s)
55000	0.15	550	1	3256.5631	2624.0152	4591.3846	10471.9629	3256.5631	2624.0152	4591.3846	10471.9629
56000	0.25	600	1	60053.9099	45197.7661	6228.0339	111479.710	60053.9099	45197.7661	6228.0339	111479.710
59092	0.14	760	2	9803.2212	9183.0716	11123.3732	30109.6659	9803.2212	9183.0716	11123.3731	30109.6659
59990	0.83	1705	3	6491.9569	2950.8949	15044.7286	24487.5804	6491.9569	2950.8949	15044.7286	24487.5804
51020	0.11	1522	4	60276.1285	51440.7887	9498.2026	121215.120	60276.1285	51440.7887	9498.2026	121215.120

Table F.2: Differences between verification data and results

Input vector				**Absolute difference**				**Relative difference**			
t_0 (MJD)	k_2 (-)	TOF (days)	N (-)	ΔV_0 (m/s)	ΔV_f (m/s)	ΔV_{LT} (m/s)	ΔV (m/s)	ΔV_0 (%)	ΔV_f (%)	ΔV_{LT} (%)	ΔV (%)
55000	0.15	550	1	7.2897E-07	6.5298E-07	4.3319E-08	1.1931E-07	2.2385E-08	2.4885E-08	9.4349E-10	1.1393E-09
56000	0.25	600	1	2.4804E-08	6.8976E-09	2.1991E-08	5.3697E-08	4.1303E-11	1.5261E-11	3.5309E-10	4.8167E-11
59092	0.14	760	2	1.2481E-07	1.4728E-07	2.4665E-06	2.7386E-06	1.2732E-09	1.6038E-09	2.2174E-08	9.0954E-09
59990	0.83	1705	3	1.0004E-11	1.0004E-11	1.0004E-10	9.8225E-11	1.5411E-13	3.3903E-13	6.6498E-13	4.0112E-13
51020	0.11	1522	4	3.0996E-09	6.1991E-09	2.1149E-06	2.1056E-06	5.1423E-12	1.2051E-11	2.2267E-08	1.7371E-09

Table F.3: Validation of minimum number of steps required for ΔV computation using the Midpoint method (for $\boldsymbol{x}$ = [62701.02, 0.1970, 562.74])

# steps	**Computed ΔV**	**Absolute difference**	**Relative difference (%)**
100	6.038231276	0.023095754	0.381034618
200	6.049774442	0.011552588	0.190595029
300	6.053605264	0.007721767	0.127393993
400	6.055517506	0.005809524	0.095845747
500	6.05667934	0.004647691	0.076677777
1000	6.059004283	0.002322747	0.038320769
10,000	6.061094645	0.000232386	0.00383391
100,000	6.061303811	2.3219E-05	0.000383069
1,000,000	6.061324729	2.30101E-06	3.79622E-05
10,000,000	6.061326821	2.09182E-07	3.45109E-06
100,000,000	6.06132703	0	0

Table F.4: Validation of minimum number of steps required for TOF computation using the Midpoint method (for $\boldsymbol{x}$ = [62701.02, 0.1970, 562.74])

# steps	Computed TOF	Absolute difference	Relative difference (%)
100	562.7399885	0.002707824	0.000481183
200	562.7420193	0.000676954	0.000120295
300	562.7423954	0.000300868	5.34646E-05
400	562.7425271	0.000169238	3.00738E-05
500	562.7425880	0.000108313	1.92473E-05
1000	562.7426692	2.70782E-05	4.81182E-06
10,000	562.7426960	2.70846E-07	4.81296E-08
100,000	562.7426963	2.77703E-09	4.93481E-10
1,000,000	562.7426963	8.60609E-11	1.52931E-11
10,000,000	562.7426963	2.20552E-11	3.91924E-12
100,000,000	562.7426963	0	0

F.3. Optimization

F.3.1. Method

In this work, three different methods are used for the optimization of trajectories: the DE, the GS, and the AGS. It has to be verified that each of them is implemented correctly and converges to the global optimum. Therefore, two functions of which the global optimum is known are used as test cases [17]. The DE is verified by optimization of the Rastrigin function. The GS and the AGS are verified by optimization of the HimmelBlau function. The global minima of both functions are presented in Table F.5, together with the ones found using the optimization algorithms as implemented in this work. It should be noted that no verification is necessary for the conjugent gradient method as this optimization technique is built-in in the gpml toolbox [24].

F.3.2. Results

The optima found for the Rastrigin function and Himmelblau function, using respectively the DE, the GS, and AGS, are given in Table F.5. It can be observed that the optima found using the algorithms are exactly equal to the global optima [17], from which it is concluded that the optimization techniques are implemented correctly.

Table F.5: Verification of optimization techniques

Function to be optimized	Method	Global optima			Optima found with algorithm		
		x	y	$f(x,y)$	x	y	$f(x,y)$
Rastrigin	DE	0.00	0.00	0.00	0.00	0.00	0.00
Himmelblau	GS	3.00	2.00	0.00	3.00	2.00	0.00
Himmelblau	AGS	3.00	2.00	0.00	3.00	2.00	0.00

G

Appendix: Recommendations

Due to the promising results that can be achieved with GP models, some recommendations for further research into the application of GP models on low-thrust trajectories are discussed.

- In this work, the GP models are applied on direct low-thrust transfers that are approximated by the exposin shape. It could be investigated whether it is possible to obtain comparable or better results, in terms of accuracy and prediction time, when the GP models are applied on low-thrust transfer trajectories modeled or shaped in a different way. The following three deviations are recommended:

    1. Shaping the low-thrust trajectory with a different shape than the exposin, for example using hodographic shaping [8], pseudo-equinoctial shaping [20], or spherical shaping [5]. From Table III in Section VII in the conference paper, it becomes clear that these shape-based methods are able to find more optimal trajectories in terms of ΔV than the exposin, and might therefore be even more useful in the preliminary design of low-thrust trajectories.
    2. Considering low-thrust trajectories that include flyby's along planets or asteroids. In the work of Petropoulos [23], low-thrust trajectories containing flyby's were modeled making use of the exposin shape. Because trajectories that include flyby's are of significant importance in trajectory design, it will be highly valuable when GP models are developed for this application.
    3. Using detailed numerical computation instead of an analytic approach to generate low-thrust trajectories. It will take significantly more CPU time to generate training data, but if it is possible to achieve high accuracies for these models, the optimal trajectories will more closely resemble the actual optimal trajectories as less or no simplifications are made.

- Currently, the GP models are trained with data that is uniform randomly distributed within the input space. Since significantly the largest part of all feasible trajectories are non-optimal in terms of ΔV and J_m, a random uniform distribution will lead to a set of training data that contains very few trajectories with small ΔV and J_m values, and most likely not the global optimum. It is expected that potentially higher accuracies could be obtained and trajectories closer to the global optimum could be found when a larger percentage of the training data contains trajectories with small ΔV and J_m values. McKay et al. [16] has recommended two other methods for the sampling of training data: Stratified Sampling and Latin Hypercube Sampling, which are both further detailed in the literature study [4].

- Three different transfer trajectories with different target planets have been chosen as mission test cases in this work. Not enough time was available to try several other target planets within the Solar System. It is therefore recommended to develop GP models for transfers to planets closer to the Sun (e.g. Mercury or Venus), or planets further away (e.g. Saturn) than the targets currently investigated.

- Finally, no relevant papers have been published yet that define some logic in choosing a GP model for a specific application, when prior knowledge is not available. In this work, a method is developed that tries to provide some direction in choosing a GP model. However, it remains unanswered why specific GP models and functions within these models are performing better than others. A study could be directed towards this end, trying to find out whether some logic could be defined in the relationship between a specific application and the performance of a specific GP model.

Bibliography

Note: this bibliography applies to the appendices and is an addition to the bibliography provided in the conference paper

[1] M. M. Ali and A. Torn. Population set-based global optimization algorithms: some modifications and numerical studies. *Computers & Operations Research*, 31(10):1703–1725, 2004. doi: 10.1016/S0305-0548(03)00116-3.

[2] C. Andre. Exposin code Python: https://github.com/ChrisAndre/expsin. URL `https://github.com/ChrisAndre/expsin`.

[3] 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(1-2):2–20, 2010.

[4] L. Bouwman. Gaussian Process Regression for Low-thrust Trajectory Optimization - Literature study. Technical report, 2018.

[5] P. de Pascale and M. Vasile. Preliminary Design of Low-Thrust Multiple Gravity-Assist Trajectories. *Journal of Spacecraft and Rockets*, 43(5):1065–1076, 2006. ISSN 00653438. doi: 10.2514/1.19646.

[6] A. Gao, G. Y. Wang, S. S. Wu, and T. Song. Efficient Evaluation of Mars Entry Terminal State Based on Gaussian Process Regression Efficient. In *IOP Conference Series: Materials Science and Engineering*, 2018. doi: 10.1088/1757-899X/449/1/012010.

[7] D. J. Gondelach. A Hodographic-Shaping Method for Low-Thrust Trajectory Design. *MSc thesis: Delft University of Technology*, 2012.

[8] D. J. Gondelach and R. Noomen. Hodographic-Shaping Method for Low-Thrust Interplanetary Trajectory Design. *Journal of Spacecraft and Rockets*, 52(3):728–738, 2015. ISSN 0022-4650. doi: 10.2514/1.A32991.

[9] P. C. Hammer. The Midpoint Method of Numerical Integration. *Mathematics Magazine*, 31(4):193–195, 1958.

[10] L. R. Hellevik. Numerical methods for engineers. *Department of Structural Engineering; NTNU*, 2018. ISSN 03784754. doi: 10.1016/0378-4754(91)90127-O.

[11] M. R. Hestenes and E. Stiefel. *Methods of Conjugate Gradients for Solving Linear Systems.pdf*. 1952.

[12] D. Izzo. Lambert 's problem for exponential sinusoids. Technical Report April, ESA/ESTEC, 2005.

[13] D. Izzo and C. Bombardelli. Benchmarking different global optimisation techniques for preliminary space trajectory design. *International Astronautical Federation - 58th International Astronauti- cal Congress 2007,*, 6:4181–4190, 2007.

[14] S. Koziel and X. Yang. *Computational optimization, methods and algorithms*. Springer-Verlag Berlin Heidelberg, 2011.

[15] H. Kuninaka. Microwave Discharge Ion Engines onboard Hayabusa Asteroid Explorer. *AIP Conference Proceedings*, 997(1):572–581, 2008. doi: 10.1063/1.2931929.

[16] M. D. McKay, R. J. Beckman, and W. J. Conover. A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output From a Computer Code. *Technometrics*, 21(September 2013):37–41, 2012.

[17] T. Möller, I. Bernst, D. Panoglou, D. Muders, V. Ossenkopf, M. Röllig, and P. Schilke. Astrophysics Modeling and Analysis Generic Interface for eXternal numerical codes (MAGIX). *Astronomy & Astrophysics*, 21:1–11, 2013.

[18] R. M. Neal. Regression and Classification Using Gaussian Process Priors. 1998.

[19] R. Noomen. Lecture Notes AE4878 - Mission Geometry & Orbit Design: Space Mission Design: Exponential Sinusoids. *Delft University of Technology; The Netherlands*, 2018.

[20] D. M. Novak and M. Vasile. Improved Shaping Approach to the Preliminary Design of Low-Thrust Trajectories. *Journal of Guidance, Control, and Dynamics*, 34(1):128–147, 2011. ISSN 0731-5090. doi: 10.2514/1.50434.

[21] C. J. Paciorek and M. J. Schervish. Nonstationary Covariance Functions for Gaussian Process Regression. *Advances in Neural Information Processing Systems*, 2004.

[22] R. M. Patel. *Spacecraft Power Systems*. CRC Press, 2004. ISBN 9781420038217. doi: 1420038214.

[23] A. E. Petropoulos and J. M. Longuski. A shape-based algorithm for the automated design of low-thrust, gravity-assist trajectories. *Advances in the Astronautical Sciences*, 109 III(5):2321–2336, 2002. ISSN 00653438. doi: 10.2514/1.13095.

[24] C. E. Rasmussen. Documentation for GPML Matlab Code version 4.2. URL `http://www.gaussianprocess.org/gpml/code/matlab/doc/`.

[25] C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning.*, volume 14. 2004. ISBN 026218253X. doi: 10.1142/S0129065704001899.

[26] M. Rayman, T. Fraschetti, C. Raymond, and C. Russell. Dawn-A mission in development for exploration of main belt asteroids Vesta and Ceres. *Acta Astronautica*, 58(11):605–616, 2006.

[27] M. D. Rayman and S. N. Williams. Design of the First Interplanetary Solar Electric Propulsion Mission. *Journal of Spacecraft and Rockets*, 39(4):589–595, 2002. ISSN 0022-4650. doi: 10.2514/2.3848.

[28] H. Shang and Y. Liu. Assessing Accessibility of Main-Belt Asteroids Based on Gaussian Process Regression. *Journal of Guidance, Control, and Dynamics*, 40(5):1144–1154, 2017. ISSN 0731-5090. doi: 10.2514/1.G000576.

[29] R. Storn and K. Price. Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359, 1997. ISSN 09255001. doi: 10.1023/A:1008202821328.

[30] E. Stuhlinger. *Ion propulsion for space flight*. McGraw- Hill Series in Missile and Space Technology, 1964.

[31] D. A. Vallado. *Fundamentals of Astrodynamics and Applications*. Microcosm Press, 2013. ISBN 978-11881883180.

[32] M. Dellnitz Vasile, O. Schütze, O. Junge, G. Radice. Spiral Trajectories in Global Optimisation of Interplanetary and Orbital Transfers Table of Contents. Technical Report 0, 2006.

[33] J. R. Wertz. Mission Geometry: Orbit and Constellation Design and Management. *Book*, 2001. ISSN 19450699. doi: 10.1063/PT.4.0095.

[34] C. K. I. Williams and D. Barber. Bayesian Classification With Gaussian Processes. 20(12):1342–1351, 1998.