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

Gómez Pérez, P.; Liu, Y.; Cowan, K.J.

Publication date
2021

Document Version
Final published version

Published in
Advances in the Astronautical Sciences

Citation (APA)
Gómez Pérez, P., Liu, Y., & Cowan, K. J. (2021). Global Optimization of Low-Thrust Interplanetary Trajectories Using a Machine Learning Surrogate. In R. Wilson, J. Shan, K. Howell, & F. Hoots (Eds.), *Advances in the Astronautical Sciences* (Vol. 175, pp. 5147-5166). Article AAS 20-663 (Advances in the Astronautical Sciences; Vol. 175).

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

AAS 20-663

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

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

In this work, we propose a new method to approximate the cost function of Low-Thrust, Multiple-Gravity-Assist interplanetary trajectories using a Machine Learning surrogate. We identified the computation time required to obtain training data as the main limitation when using Machine Learning methods for this purpose so we present a strategy to build the surrogate with limited training data. We build an Online-Sequential Extreme Learning Machine Multi-Agent System (OS-ELM-MAS) surrogate due to its theoretical good performance when the training data is limited. In addition, we define a method to include the surrogate during the optimization process that can be used with any gradient-free algorithm, and study the effect of several surrogate parameters on the optimization results. Finally, several interplanetary trajectories are optimized with and without the surrogate. Employing the surrogate results in up to 12% lower fuel cost values after a fixed optimization time. The parameters that control the interaction have to be carefully selected to achieve this improvement, and we show that the optimal value of these parameters can be narrowed down based on the characteristics of the transfers.

INTRODUCTION

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

As an alternative, Machine Learning (ML) methods can be used to approximate functions that are unknown or very costly to evaluate. In previous work, ML methods have been proposed as surrogates for the cost function during the optimization of interplanetary transfers using both chemical rockets and LTP. Artificial Neural Networks (ANNs) were used to predict the optimal control at a given state when using LTP[5,6] and to predict the fuel required to complete a whole transfer including an intermediate GA.[7] Moreover, several ML methods were tested to predict the fuel required to complete optimal transfers between asteroids[8,9] and it was found that Gradient Boosting[10] provided the best results. In addition, ML methods were used to

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

estimate the fuel requirements of Multiple-Gravity-Assist Trajectories (MGATs) using traditional chemical propulsion[11,12] and LTP.[13]

The methods proposed require large amounts of training data, and the interaction between the surrogate and the optimization algorithm is not evaluated. The training data needs to be generated using the same cost function that the surrogate is replacing, so the computation time saved by using the surrogate is expected to be smaller than the computation time required to generate large amounts of training data. Moreover, most previous work exclusively evaluates the overall accuracy of the approximation, but it does not evaluate the interaction between the surrogate and the optimization algorithm. In the cases where such interaction is studied, custom algorithms are proposed.[11] Therefore, previous methods have limited applications in the optimization of MGATs. In this work, we aim to provide a way to incorporate a ML surrogate in any optimization algorithm with a very small computation time overhead. To achieve that objective, we selected a specialized ML method and a studied the interaction between the surrogate and the optimization algorithm. We tested the effects of this surrogate in the optimization using medium fidelity LTP models,[2] but we expect results to be extrapolatable to higher fidelity models.

The method we selected is a simplified version of the ANNs known as Extreme Learning Machine (ELM).[14] This method has a low number of hyperparameters and a very fast training process. The simplicity of the method permits the combination of models to reduce the total error while maintaining low training and prediction times. Therefore, we proposed the use of an Extreme Learning Machine Multi-Agent System (ELM-MAS)[15] method to combine several ELM surrogates. To minimize the number of training points required, the surrogate is trained online. This means that the surrogate is trained as the training data is generated and predictions are made at the same time. This way, the surrogate can be trained with the cost function values computed numerically during the optimization until the desired accuracy is reached. At that point, the surrogate replaces the numerical computation of the cost function, speeding up the optimization procedure. The online version of the ELM is the Online-Sequential Extreme Learning Machine (OS-ELM).[16] We presented a combined used of the OS-ELM and ELM-MAS into the innovative Online-Sequential Extreme Learning Machine Multi-Agent System (OS-ELM-MAS) method.

To test the speed-up achieved with the surrogate, we focused on the effects of the surrogate when used in conjunction with an optimization algorithm. We identified a number of parameters that influence the interaction between the online surrogate and the optimization process, and we propose a method to select the most adequate values for these parameters. Finally, we used a Differential Evolution (DE)[17] global optimization algorithm in several MGATs cases and compared the results achieved with and without the surrogate replacing the cost function. The final objective was to achieve an implementation of the surrogate that does not require either fine tuning or previous training data and that can be used as an off-the-shelf tool for preliminary trajectory design.

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

MULTIPLE-GRAVITY-ASSIST OPTIMIZATION PROBLEM

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

Spacecraft Dynamics

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

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

where $\boldsymbol{r}$ is the position vector of the spacecraft, $\boldsymbol{v}$ its velocity vector, m its mass, $\boldsymbol{T}$ the thrust vector, I_{sp} the specific impulse of the spacecraft, and $g_0 = 9.806\,65\,\mathrm{m\,s^{-2}}$ and $\mu = 1.327\,124\,400\,18 \times 10^{20}\,\mathrm{m^3\,s^{-2}}$ are respectively the standard gravity and gravitational parameter of the Sun.

Gravity Assist Computation

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

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

Optimization Problem Formulation

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

We consider a leg to be a segment of the trajectory between two consecutive body encounters, so the whole transfer consists of l legs. Leg i[†] is defined by the initial state $\boldsymbol{s}_0^i = [\boldsymbol{r}_0^{iT}, \boldsymbol{v}_0^{iT}, m_0^i]^T$, the final state $\boldsymbol{s}_f^i = [\boldsymbol{r}_f^{iT}, \boldsymbol{v}_f^{iT}, m_f^i]^T$, and the time of flight TOF^i. We define the trajectory of the leg as the optimal trajectory between the initial and final state. The value of m_f^i is computed as

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

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

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

[†]In this section, we use a superscript ($\square^i$) to indicate that a variable corresponds to leg i, an asterisk ($\square^*$) to indicate an optimal solution, a hat ($\hat{\square}$) to indicate a value estimated with a numerical optimization procedure, and a prime symbol ($\square'$) to indicate the subspace where a solution can be found.

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

$$g(t,s) = \min_{\boldsymbol{u}} \left(\int_0^{t_f} j\left(\boldsymbol{s}(t), \boldsymbol{u}(t)\right) dt + h(\boldsymbol{s}(t)) \right) \qquad \text{and dynamics} \qquad \dot{\boldsymbol{s}}(t) = \boldsymbol{F}(\boldsymbol{s}(t), \boldsymbol{u}(t)),$$

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

$$\dot{g}(\boldsymbol{s},t) = -\min_{\boldsymbol{u}} \left(\nabla g(\boldsymbol{s},t) \cdot \boldsymbol{F}(\boldsymbol{s},\boldsymbol{u}) + l(\boldsymbol{s},\boldsymbol{u}) \right), \qquad \text{subject to} \qquad g(\boldsymbol{s},t_f) = h(\boldsymbol{s}). \tag{5}$$

If $\boldsymbol{F}$, j, and h are continuous and bounded, and $\boldsymbol{s}(t')$ can be reached from $\boldsymbol{s}(0)$, then there is an optimal value $\boldsymbol{u}^*$, which is the solution of Equation (5) and unique.[18] In our case, the relevant functions are

$$j = \frac{|\boldsymbol{T}|}{m_0^i g_0 I_{sp}}, \qquad h = 0, \qquad t_f = TOF^i, \qquad \boldsymbol{u} = \boldsymbol{T},$$

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

However, there is no known general solution for Equation (5). Instead, a numerical optimization is used to search for $\boldsymbol{T}^{*i}$ when $\boldsymbol{s}_0^i$ and $\boldsymbol{s}_f^i$ are known. A transcription method is used to define a problem in which a local optimization algorithm can be applied. This process is explained in detail in the Leg Computation section. In a general case, the result obtained is not the true $\boldsymbol{T}^{*i}$, as the numerical optimization is subject to a series of additional constraints that make the problem tractable. Instead, an estimation of the optimal control policy $\hat{\boldsymbol{T}}^{*i}$ is provided by the local optimization procedure, which results in the leg cost used in practice:

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

where ϵ^i is the difference between the actual optimal value of the cost function and the estimated one.

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

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

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

$$J(\boldsymbol{l}_T) = \sum_{i=1}^{l} \hat{J}_L^{*i} m_0^i, \qquad \text{where} \qquad \boldsymbol{l}_T = [\boldsymbol{l}^{0T}, \boldsymbol{l}^{1T}, \ldots, \boldsymbol{l}^{lT}]^T \in \hat{L}'_T = (\hat{L}')^l.$$

The outer optimization loop is the numerical procedure to obtain

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

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

Global Optimization Algorithm

The global optimization algorithm is used to look for the solution to Equation (7). We selected a DE[17] algorithm as global optimizer. In particular, we used the *Pygmo version of the self-adaptive Differential Evolution* (pDE)[†]. This algorithm includes adaptation of all evolution parameters,and the only parameter that needs to be selected when using this algorithm is the number of individuals per generation N_I. This algorithm can only handle box-bounding constraints on the optimization variables. However, the values of the elements of $\boldsymbol{l}_T$ are not independent of each other, and they are subject to non-linear constraints: the position of the spacecraft at the time of the encounter with a body has to match that of the body and the velocities before and after a GA are related through Equation (2). Notwithstanding, an analysis of the constraints allows the definition of an alternative set of optimization variables that fulfill the constraints and are box bounded.

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

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

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

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

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

We use the values η^i and ρ^i as the optimization variables. For the GA orbit plane angle, the valid range is $\eta^i \in [0, 2\pi]$, while the range of GA radii allowed is set on every individual case depending on the characteristics of the planet encountered.

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

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

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

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

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

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

The final decision vector of the optimization problem is

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

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

Use of Machine Learning Surrogate

The objective of the surrogate is to approximate a cost function so said cost function does not need to be evaluated for every individual. In the optimization problem presented, there are three cost functions that can be approximated: $J(f(\boldsymbol{x}))$, $J(\boldsymbol{l}_T)$, or $\hat{J}_L^*(\boldsymbol{l})$. We decide to use the surrogate to approximate $\hat{J}_L^*(\boldsymbol{l})$ for three reasons: the function shape is expected to be simpler, the dimension of the input vector is lower, and it is evaluated m times for every evaluation of the other two candidate functions, increasing the number of training points available. In addition, we decided to represent the state variables in $\boldsymbol{l}$ using modified equinoctial elements.[19]

To minimize the number of training points required, the surrogate is trained online during the optimization. This means that the surrogate is trained on the training data as the data is generated and at the same time predictions are made with the surrogate. At the beginning of the global optimization, all points are computed with the local optimization procedure. Surrogate predictions are made only when the expected error is below a certain threshold, τ. When that point is reached, the cost function evaluations required by the global optimization algorithm are done with the surrogate.

The error of the surrogate predictions is expected to decrease as more training points are used but to increase when the optimization algorithm explores new areas of the input space where no training points have been generated. To detect these increases in the error, the error is updated after a fixed number of individuals predicted with the surrogate, C_n. If the error is computed once on the leg i of the individual j, the next computation of the error will happen in the leg $i + 1$ of the individual $j + C_n$. Higher τ values and lower C_n values reduce the number of numerical evaluations of the cost function required, which decreases the computation time, but also increases the error in the surrogate predictions, and vice versa.

LEG COMPUTATION

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

[†]The operator ∘ indicates function composition.

Several options are available to compute the cost of the legs when LTP is used. The method used to compute $\hat{J}_L^*(\boldsymbol{l})$ determines the minimum error achievable with the surrogate. The ability of the ML surrogate to approximate an arbitrary function is based on the Universal Approximation Theorem (UAT) for ELM.[20] However, this theorem only holds when the values of the function to approximate are unique for each input value. This is true for $J_L^*(\boldsymbol{l})$ as it is a solution of the *Hamilton-Jacobi-Bellman equation*. However, this is not true for a general $\hat{J}_L^*(\boldsymbol{l})$ as the solution can be affected by factors not considered in $\boldsymbol{l}$ such as the effect of integration tolerances and the random initial state. To analyze the effect of this in the surrogate error, we define the *expected prediction error* as

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

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

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

Estimating the actual value of σ_s is very difficult, as it requires to sample the value of $L(\hat{J}_L^{*S}(\boldsymbol{l}), \hat{J}_L^*(\boldsymbol{l}))$ at a large number of points. However, Equation (9) indicates that σ_s value has to be as small as possible in order to minimize the prediction error. This means that the numerical method to compute the cost of the legs should be selected so the influence of factors not included in $\boldsymbol{l}$ is as low as possible. We achieve this by choosing a method whose solution is independent of the initial guess. Therefore, we use a direct method for the leg computation as they make possible to use a deterministic local optimization. Nonetheless, these methods typically require a search trough several initial guesses when used as a standalone methods.[3,21] This makes the result dependent on the initial guess, but we solve this problem by obtaining an initial guess with a lower fidelity method whose solutions are independent of the initial guess. Shape-based methods are commonly used for this purpose.[1,22] Therefore, the overall leg computation process is completely deterministic and not dependent on the initial guess.

Shape-Based Method

Shape-based methods assume an analytical shape for the trajectory of the leg. The thrust required to follow the shape is then computed and the cost function evaluated. These methods are usually the fastest option when LTP is used.[2] However, the trajectories obtained are not optimal as they are restricted to the shape selected. We identify the Spherical[23] shape-based method as the best option to obtain the initial guess for the trajectory. This was based on the following set of requirements for the shape-based method: (1) it shall produce three-dimensional trajectories, (2) it shall be able to fully match the initial and final state, (3) it shall not require a numerical optimization. Shape-based methods provide different solutions depending one the number of revolutions, n_{rev}, selected for the trajectory. Therefore, the solution is not unique unless a n_{rev} value is selected. In this work, we selected the n_{rev} corresponding to the Spherical shape-based solution with the lowest fuel cost of all the possible solutions within a predetermined n_{rev} range.

Direct Method

A direct method allows the transformation of the orbit optimization problem into a non-linear programming problem. These methods are usually considered to be medium fidelity.[2] We use a modified Sims-Flanagan (SF) method,[24] as we decided that an increase in the accuracy of the solution was preferable to shorter computation times. This implementation uses the Sundman transformation, continuous thrust along the segments instead of impulses at their middle points, and Taylor integration. The Sundman transformation is considered beneficial for trajectories with a large difference of radii between initial and final state. Continuous thrust

has the advantage of providing trajectories that are physically feasible, which is not granted with the impulse approximation of the original SF implementation.[25] However, trajectories can no longer be considered Keplerian between impulses, although the increase in computation time is compensated in part by using Taylor integration.

Once the initial guess is obtained with the shape-based method, the thrust values at each segment are optimized using a local optimization algorithm. We selected the Sequential Least SQuares Programming (SLSQP) algorithm from the nlopt package[†]. A SF result is considered valid if the thrust magnitude is below T_{max} during the whole trajectory and the discontinuities in the trajectory are below a given tolerance. In some cases, the algorithm is unable to find a solution that fulfills all theses constraints. If that is the case, $\boldsymbol{l}_L \notin \hat{L}'$, the leg is considered unfeasible, and the transfer is discarded. However, the surrogate provides numerical results even when evaluated at legs where the numerical evaluation fails to converge. One of the main obstacles for the use of the surrogate is that these unfeasible legs are still used by the global optimization algorithm, and the final transfer selected may not be a feasible one. Therefore, the convergence rate of the SF computations influence the optimization results obtained when using the surrogate.

MACHINE LEARNING SURROGATE

As mentioned previously, we identified the ELM as a the most adequate ML method for the creation of the surrogate. This sections presents the details of this method.

Review of Extreme Learning Machines

ELMs[14] are based on ANNs with a single hidden layer. The peculiarity of this method is that the weights between the input later and the hidden layer are initialized randomly when the model is created, and remain frozen for the remaining of the training process. This allows the computation of the weights between the hidden and the output layer by least squares regression. Therefore, the result is always optimal for the training points in terms of Mean Squared Error (MSE). Moreover, the UAT is valid also for the ELM.[20]

The definition of an ELM requires an activation function, $AF : \mathbb{R} \to \mathbb{R}$, which should be infinitely differentiable in any interval,[14] and a number of hidden units, NHU. The training data contains N pairs of observations[‡] $\left(\boldsymbol{l}_{(i)}, \boldsymbol{y}_{(i)}\right) \in \mathbb{R}^{n_I \times 1} \times \mathbb{R}^{n_O \times 1}$, where $\boldsymbol{l}_{(i)}$ is the input vector at point i, $\boldsymbol{y}_{(i)}$ is the output vector at point i, and n_I and n_O are respectively the lengths of the input and target vectors. The training process is the following one.[14] First, the vectors $\boldsymbol{a}_j \in \mathbb{R}^{1 \times n_I} \forall j \in \{1, \ldots, NHU\}$ and $\boldsymbol{b} \in \mathbb{R}^{1 \times NHU}$ are randomly initialized. In our case, each element of the vectors is drawn from a random uniform distribution with range $[-1, 1]$. Then, the *hidden layer output matrix* is computed as

$$\boldsymbol{H} = \begin{bmatrix} AF\left(\boldsymbol{a}_1 \cdot \boldsymbol{l}_{(1)} + b_1\right) & \cdots & AF\left(\boldsymbol{a}_{NHU} \cdot \boldsymbol{l}_{(1)} + b_{NHU}\right) \\ \vdots & \ddots & \vdots \\ AF\left(\boldsymbol{a}_1 \cdot \boldsymbol{l}_{(N)} + b_1\right) & \cdots & AF\left(\boldsymbol{a}_{NHU} \cdot \boldsymbol{l}_{(N)} + b_{NHU}\right) \end{bmatrix},$$

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

$$\hat{\boldsymbol{\beta}} = \left(\boldsymbol{H}^T \boldsymbol{H}\right)^{-1} \boldsymbol{H}^T \boldsymbol{Y}, \qquad \text{where} \qquad \boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_{(1)}^T & \cdots & \boldsymbol{y}_{(N)}^T \end{bmatrix}^T. \tag{10}$$

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

$$\hat{\boldsymbol{y}}' = \begin{bmatrix} AF\left(\boldsymbol{a}_1 \cdot \boldsymbol{l}' + b_1\right) & \cdots & AF\left(\boldsymbol{a}_{NHU} \cdot \boldsymbol{l}' + b_{NHU}\right) \end{bmatrix} \hat{\boldsymbol{\beta}}.$$

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

[‡] In this section, we use a subscript ($\square_i$) to indicate a variable associated with the hidden unit i of the ELM and a subscript enclosed in parenthesis ($\square_{(i)}$) to indicate data from transfer i.

The Online Sequential ELM

The online version of the ELM is the OS-ELM.[16] The computation of the weights can be adapted to the arrival of new training points so the weights always correspond to the MSE-optimal solution for the full set of points trained. This is done by updating $\hat{\boldsymbol{\beta}}$ with a recursive least-squares formula.[16] We consider only the case in which new training points arrive one by one. The total number of training samples is arbitrary, and the first N training samples are selected for initial training. First, the initial weights[†], $\hat{\boldsymbol{\beta}}_{(0)} = \hat{\boldsymbol{\beta}}$, are computed with the N initial training samples as shown in Equation (10). The subsequent pairs of training points are $\left(\boldsymbol{l}_{(k)}, \boldsymbol{y}_{(k)}\right)$, where k indicates the position in which they arrive after the last point used for initial training. The weights are updated as

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

where

$$\boldsymbol{H}_{(k+1)} = \begin{bmatrix} AF\left(\boldsymbol{a}_1 \cdot \boldsymbol{l}_{(k+1)} + b_1\right) & \cdots & AF\left(\boldsymbol{a}_{NHU} \cdot \boldsymbol{l}_{(k+1)} + b_{NHU}\right) \end{bmatrix},$$
$$\boldsymbol{P}_{(k+1)} = \boldsymbol{P}_{(k)} - \boldsymbol{P}_{(k)}\boldsymbol{H}_{(k+1)}^T \left(\boldsymbol{I} + \boldsymbol{H}_{(k+1)}\boldsymbol{P}_{(k)}\boldsymbol{H}_{(k+1)}^T\right)^{-1} \boldsymbol{H}_{(k+1)}\boldsymbol{P}_{(k)}, \quad \text{and} \quad \boldsymbol{P}_{(0)} = \left(\boldsymbol{H}^T\boldsymbol{H}\right)^{-1}.$$

The initial training process, defined in Equation 10, requires the rank of $\boldsymbol{H}$ to be equal to NHU. In order to do achieve this, it is necessary that $N \geq NHU$. However, our tests show that the pDE algorithm tends to generate the training points in groups of very similar input values. This causes the matrix inverse in Equation (10) to be ill-conditioned. The use of this matrix results in very high numerical errors in some cases. However, using $N = 2NHU$ eliminates this problem as the chances of having at least NHU independent samples in $\boldsymbol{H}$ is very high.

This requirement of a minimum number of training points is one of the main limitations of the OS-ELM method. In general, a larger NHU means smaller surrogate error. However, this requires a larger number of points for initial training. This is relevant in our implementation, as generating additional training points has a high computation cost, and the surrogate cannot be used while these points are being generated.

OS-ELM Multi-Agent System

The training process of the OS-ELM is extremely fast compared to the time required to generate each training point. In our tests, the update of an average OS-ELM surrogate took $\sim 0.6\,\mathrm{s}$ per data point, while the computation of each point took $\sim 120\,\mathrm{s}$. This can be exploited by combining the predictions of several OS-ELM with different random initializations. The method used for the combination is know as the ensemble method. We designed an ensemble method based on the ELM-MAS.[15] This method uses a series of child ELMs and a parent ELM. The children work as standard ELMs. However, the predictions of all of them are used as inputs for the parent, whose output is a single corrected prediction for the output value. Therefore, the input size for the parent ELM is $NM \times n_O$, where NM is the number of children. When training, the children are first trained as usual and then the predictions on the training points are used as inputs for the training of the parent.

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

[†]To simplify the notation, we use a subscript enclosed in parenthesis ($\square_{(k)}$) to indicate data from transfer k after the first N transfers in this and the following sections. This is equivalent to writing $\square_{(N+k)}$ using the convention of the previous section.

Error Estimation

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

$$M_{(k)} = \frac{S_{(k)}}{D_{(k)}}, \qquad \text{where} \qquad S_{(k)} = \epsilon_{(k)} + \alpha S_{(k-1)}, \qquad D_{(k)} = 1 + \alpha D_{(k-1)},$$

α is the fading factor, and $\epsilon_{(k)} = \left|\hat{\boldsymbol{y}}_{(k)} - \boldsymbol{y}_{(k)}\right|$ is the prediction error on sample k. Our tests indicated that a value $\alpha = 0.999$ produces results close to the validation error as long as $k > 20$.

TEST SETUP

The objective of this work is to design a method to include a ML surrogate in the optimization of any MGATs with LTP. In this section, we present the procedure we followed to evaluate whether the use of the surrogate results in an improvement of the optimization results. In particular, we focus on the difference in performance with respect to the optimization without the surrogate.

Test Cases

We selected three different cases that represent a variety of past and proposed missions to evaluate the performance of the surrogate. These missions were selected based on three criteria: (1) the expected improvement when using a ML surrogate, (2) the availability of results using similar methods in the literature, (3) the variety of number of legs, propulsion parameters and destinations. The cases matching these criteria were the two first legs of the Dawn mission (Dawn),[3] an Earth-Earth-Jupiter (EEJ) transfer,[21] and an Earth-Venus-Venus-Mercury-Mercury-Mercury (EVVMMM) transfer.[21]

The parameters of each case are selected based on the previous results[3,21] that we consider to use the most similar optimization procedure to the one used in this work. In all cases, we use the same the range of departure dates, the spacecraft parameters, and velocity constraints at the initial and final point of the trajectory that were used for the optimization in the references. The range of numbers of revolutions considered for the initial shape-based solution is set to $n_{rev} \in \{0, 1, 2, 3\}$. The references do not specify this value but none of the example trajectories completes more than 3 revolutions during a single leg. The TOF range allowed for the intermediate legs was not specified in the references either. Instead, we use the ranges in which the shape-based method could always find a solution during preliminary tests. Solutions outside these ranges have a very high cost when a solution can be found, so they are not considered of interest. The numerical values of the parameters described are shown in Table 1. Finally, all cases are computed with 10 segments in each of the SF legs.[21]

Dawn The parameters of this transfer are shown in the first column of Table 1. The original spacecraft[3] had a Solar Electric Propulsion (SEP) system, but we modeled it as a NEP system instead. This case was selected as an example of transfers to bodies whose orbit has a semi-major axis similar to that of the Earth.

EEJ The parameters of this transfer are shown in the second column of Table 1. We selected this case because the low-thrust transcription method used in the reference[21] was also SF, although thrust was approximated as an impulse at the midpoint of the segments and the Sundman transformation was not used.

Table 1. Parameters of the cases tested.

Case	Dawn[3]	EEJ[21]	EVVMMM[21]
Sequence	Earth-Mars-Vesta	Earth×2-Jupiter	Earth-Venus×2-Mercury×3
T_{max} (N)	0.368	2.26	0.34
I_{sp} (s)	2620	6000	3200
m_0 (kg)	2481	20 000	1300
t_0 (MJD2000)	7305 to 8401	7305 to 10 958	7305 to 10 762
TOF^1 (d)	100 to 2000	100 to 1000	100 to 1500
TOF^2 (d)	100 to 2000	1000 to 3000	100 to 1500
TOF^3 to TOF^5 (d)	–	–	100 to 1500
v^0 ($\mathrm{km\,s^{-1}}$)	0 to 3.5	0 to 2	0 to 1.925
v^1 ($\mathrm{km\,s^{-1}}$)	0 to 9	0 to 15	0 to 9
v^2 ($\mathrm{km\,s^{-1}}$)	0	0	0 to 9
v^3 and v^4 ($\mathrm{km\,s^{-1}}$)	–	–	0 to 9
v^5 ($\mathrm{km\,s^{-1}}$)	–	–	0 to 0.5017

Therefore, an indication of the expected performance is available in the reference but results can not be directly compared. This case is an example of a transfer to an outer planet, and the final semi-major axis of the orbit is several times larger than the initial.

EVVMMM The parameters of this transfer are shown in the third column of Table 1. This case is an example of a transfer to an inner planet. In addition, it has a higher number of legs, which serves as a way to evaluate the variations in the performance of the method with the number of legs.

Benchmark

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

The pDE algorithm requires a population with N_I individuals, which evolves in generations, and each individual corresponds to one transfer. The performance measure is the propellant cost of the best individual computed during the whole optimization. When using the surrogate, the optimization algorithm does not report the true cost values, reporting instead the values estimated by the surrogate. To make a fair comparison, the true cost values of the individuals corresponding to the last population of the optimization with the surrogate are computed again with the SF method. To account for that, extra time is reserved at the end of the optimization with the surrogate. We set the run time to $3\,\mathrm{d} = 72\,\mathrm{h}$, but there are small individual variations as the pDE algorithm can only stop between generations. These variations are not considered to have an important effect on the final results as the cost of the best individual only improves in a small proportion of the generations computed. Therefore, the chances of one extra generation changing the result are low.

Parameters Test

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

1. We run the optimization for several N_I values and a fixed random initial state for the pDE, and the most adequate N_I value is selected. Simultaneously, we run the optimization using the surrogate with the same random initial state and N_I values. The initial values $C_n = 4$ and $\tau = 0.06$ were selected by estimating the order of magnitude of the expected errors. The results of these optimizations help define the C_n and τ ranges to be tested in the next step.

2. Several combinations of C_n and τ values are selected based on the previous results to determine their optimal values. In general, a large difference between the final value predicted by the surrogate and the final value computed when reevaluating the final population with SF means C_n is too large. However, small C_n values result in more SF computations, decreasing the speed of the optimization. When it comes to τ, the objective is to select a value that includes most of the points so the surrogate is used during most of the time, but that excludes error peaks. As mentioned before, we hypothesize that these peaks correspond to points in which the optimization algorithm moves into a new area of the input space. The values of C_n and τ that result in a better surrogate performance are used in the next step.

3. The optimization is repeated with the final C_n and τ combination and 6 different random initializations for the pDE. All random initializations are used to obtain results both for the baseline and using the surrogate. Then, we test the hypothesis that the means of the result with the surrogate and the baseline are different with a paired t-test, which is described in the following section. The results obtained with the 6 random initializations of the pDE algorithm are the samples used for the paired t-test.

Paired T-Test

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

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

Probability of Obtaining the Best Result with the Surrogate

The means of the results are not the only relevant information that can be obtained by analyzing the distributions of the results. When designing a mission, it is common to repeat the optimization several times to mitigate the effects of the random initialization of the optimization algorithm.[3] Usually, the only relevant result is usually the best one, as that is the trajectory followed in the end. Therefore, the probability of obtaining with the surrogate the best overall result when the optimization is repeated is also considered a relevant evaluation metric.

Assuming the Cumulative Distribution Function (CDF) of J^*, $F_{J^*}(J)$, and its Probability Density Function (PDF), $f_{J^*}(J)$, are known, the CDF and PDF of the best value after n tests are respectively[28†]

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

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

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

where superscript S refers to results with the surrogate and B to results with the baseline. We compute this probability by assuming J^{*S} and J^{*B} follow normal distributions with the same mean, μ_{J^*}, and standard deviation, σ_{J^*}, as the samples used for the paired t-test. The results give an estimation of the probability of obtaining a better result with the surrogate when the optimization is repeated n times.

OS-ELM-MAS Architecture Parameters

The OS-ELM-MAS model has some parameters that have to be defined before surrogate is created. These are know as architecture parameters, and are the following ones: the number of child OS-ELM and their number of hidden units, the number of hidden units of the parent OS-ELM, and the activation function. Typically, the input and output data is normalized before passing it to the OS-ELM.[16] We consider the normalization procedure as an architecture parameter too. The architecture parameters selection aims to achieve an adequate trade-off between prediction accuracy and lowest predicted value. The lowest predicted value is considered a relevant measure because very low predictions are more likely to be selected by the optimization algorithm.

We selected the parameters by testing both accuracy and lowest predicted value when training in the data from the legs computed during the optimization of the Dawn case with 20 individuals and no surrogate. This training data is not representative of all the test cases, but it accurately represents a realistic situation in which data for architecture selection is limited. The best combination is 16 child OS-ELMs with 128 hidden units each, 64 hidden units in the parent OS-ELM, and the *hyperbolic tangent* as activation function. In addition, all input and output variables are are scaled so their mean is 0 and their standard deviation is 1. The scaling factors are determined with the initial training data.

Computational Implementation

We implemented the method described in Python. We used the *Pykep* package[29] for the SF computations and astrodynamics related operations such as ephemerides lookup, the *Pygmo* package[30] for the optimization tools, and the *TensorFlow-OS-ELM* package[‡] for the OS-ELM implementation. Finally, we implemented the spherical shape-based method using the Scipy tools[§] and following the description in the literature.[23]

[†] In this case, $J^*_{(1)}$ refers to the best result among all the optimizations.

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

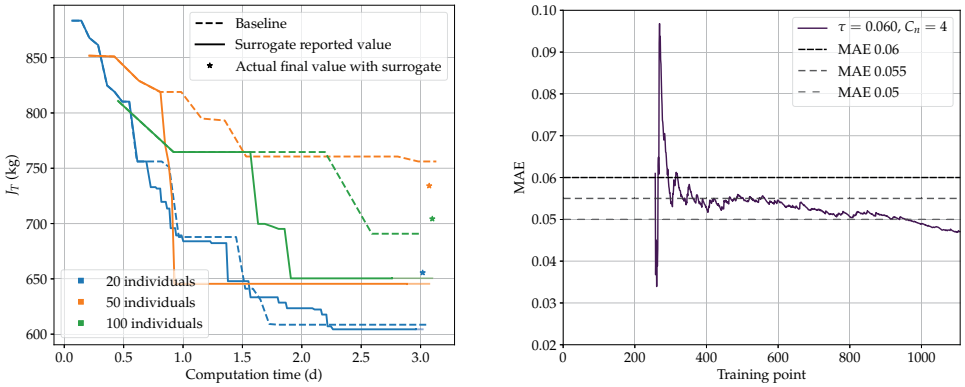
[§] `https://www.scipy.org/` (accessed May 13, 2020).

RESULTS

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

Dawn

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



a) b)

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

The second step was to determine the best C_n and τ. The results presented in Plot a) of Figure 1 indicate that the difference between the value predicted by the surrogate and the actual best result was within a 10% of the total cost. This value was considered to be accurate enough, so the value $C_n = 4$ was also considered adequate. In the end, the values $C_n \in \{3, 4, 6\}$ were also tested to analyze the effect of C_n in the final result.

When it comes to τ, the value selected was higher than the estimated error during most of the optimization process, as shown in Plot b) of Figure 1. This indicates that the threshold was not useful to discriminate most error outliers. However, it left out the highest peaks, so the initial guess was not considered completely wrong. Therefore, the values selected to be tried were $\tau \in \{0.060, 0.055\}$. All the combinations of the C_n and τ values were tested. In addition, a test with $\tau = 0.050$ was also performed to evaluate even lower threshold values. However, this was not expected to provide good results so it was tested only with $C_n = 4$. The final values of the optimization using the surrogate with the parameter combinations previously selected are shown in Table 2.

The test case with $\tau = 0.055$ and $C_n = 4$ reached the best overall result and a lower cost value than the baseline. Therefore, this combination was used on the subsequent tests. The advantage of the optimization with the surrogate with respect to the baseline can be seen in the results of Table 2: the number of J evaluations in a fixed time is much larger with the surrogate. In addition, the best τ and C_n combination achieved a low final MAE and kept a high SF convergence rate.

The third step was to determine the statistical significance of the results. A summary of results of the tests for 6 random initializations of the pDE can be seen in Table 3. From the paired t-test results, we can

Table 2. Relevant results of the optimization with the surrogate and various combinations of parameters in the Dawn case. The final combination is shown in bold.

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

conclude that the mean value of the optimization with the surrogate was higher than the mean of the baseline. Therefore, the surrogate is not recommended for an individual optimization. In addition, the best result we found has a very similar cost to the best result in the literature.[3]

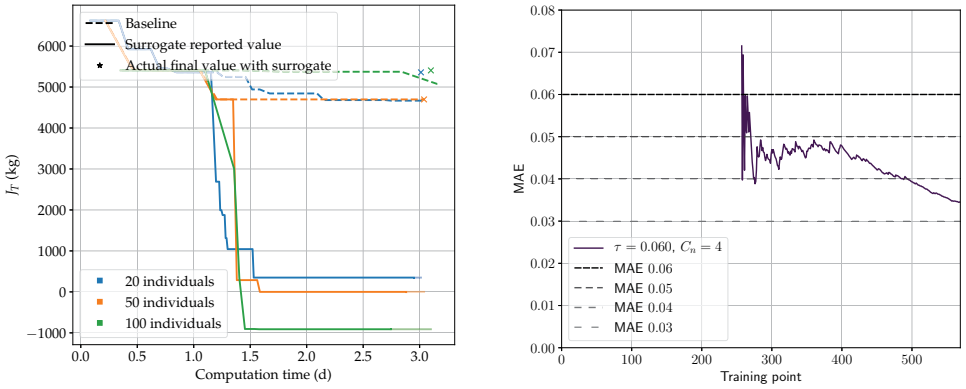
Table 3. Statistics of the results for the Dawn case.

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

[a] The confidence level is to reject the null hypothesis of the t-test (both means are equal). A negative confidence value indicates that the mean of the baseline results is lower (i.e. better) than the mean using the surrogate.

Earth-Earth-Jupiter

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



a) b)

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

The second step was again to determine the best C_n and τ. This case had the particularity of having a low convergence rate when computing the legs with the SF method. This had two important consequences when using the surrogate: the number of training points available was much lower as only converged points could be used, and the chances of the surrogate selecting a valid transfer at the end of the optimization were low. The evolution of the predicted best value in Plot a) of Figure 2 show that the surrogate prediction were unrealistically optimistic. In some cases, the surrogate predicted negative propellant use. This was obviously wrong but those results could not be immediately disregarded as the true cost was higher but it might still be the best solution. The extreme values predicted could be mitigated by using lower C_n values. The values tested were $C_n \in \{1, 2, 3, 4\}$.

When it comes to the τ value, the estimated MAE values were in general lower than in the Dawn case so the threshold values considered were $\tau \in \{0.06, 0.05, 0.04, 0.03\}$. The final values of the optimization using the surrogate with the parameter combinations selected are shown in Table 4. Identical values were obtained for different combinations because the MAE estimation reached values below all the thresholds very fast, so it did no make any difference for the optimization.

The test case with $\tau = 0.03$ and $C_n = 3$ reached the lowest cost, so those values were used in the subsequent tests. However, the result with the surrogate was not better than the result of the baseline. This can be explained by the small increase of J evaluations achieved when using the surrogate and the best combination of parameters. Other combinations resulted in a larger increase in the J evaluations but at the expense of a reduction in the SF convergence rate.

Table 4. Relevant results of the optimization with the surrogate and various combinations of parameters in the EEJ case. The final combination is shown in bold.

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

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

Finally, we evaluated the statistical significance of the results. The results of the test can be seen in Table 5. The results when the surrogate was used were almost identical to the ones of the baseline. We hypothesize that the main reason for this was the low convergence of the SF results. The surrogate had a low number of training points and the τ value needed to be very restrictive to avoid excessively optimistic predictions on invalid points. As a consequence, the surrogate made predictions in very few cases, and had almost no effect on the results. The best result obtained had a higher cost value than the best result in Reference 21. However, results cannot be directly compared as different transcription methods were used in this work and in the reference. Our solution is guaranteed to be feasible, while the one in the reference might not be.[24]

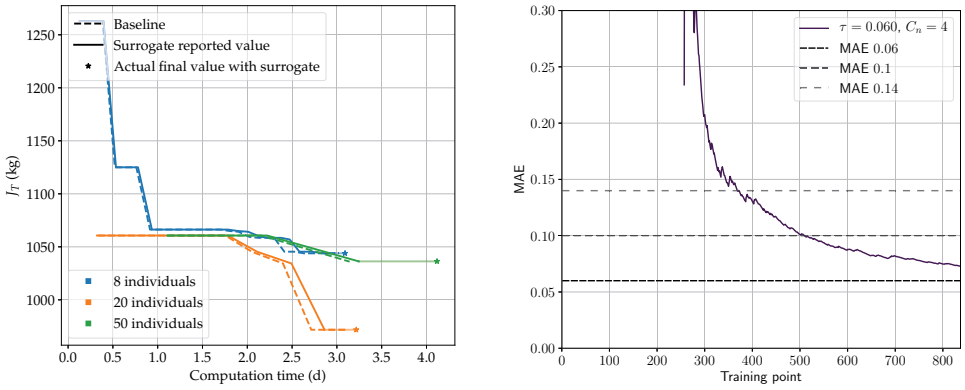
Table 5. Statistics of the results for the EEJ case.

Reference 21	Baseline			Surrogate			Confidence[a]
$J^*_{(1)}$ (kg)	μ_{J^*} (kg)	σ_{J^*} (kg)	$J^*_{(1)}$ (kg)	μ_{J^*} (kg)	σ_{J^*} (kg)	$J^*_{(1)}$ (kg)	
2898	4669	129	4534	4675	135	4496	$-10\,\%$

[a] The confidence level is to reject the null hypothesis of the t-test (both means are equal). A negative confidence value indicates that the mean of the baseline results is lower (i.e. better) than the mean using the surrogate.

Earth-Venus-Venus-Mercury-Mercury-Mercury

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



a) b)

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

The next step was to determine the best C_n and τ values. When the surrogate was used, the error estimation never reached values below $\tau = 0.06$. Therefore, the information relative to the selection of surrogate parameters obtained from this test was limited. The obvious conclusion was that a larger value of τ was required. We decided to use $\tau \in \{0.100, 0.140\}$ for the subsequent tests. The value of C_n was more difficult to determine as the effect of the surrogate predictions could not be evaluated. However, the convergence rate was high, as in the Dawn case, and every leg required a longer computation time when computed with the SF method. In the Dawn case, $C_n = 4$ resulted in the best performance, and higher C_n values mean fewer SF computations per generation. Therefore, the values selected for the tests were $C_n \in \{4, 6, 8\}$.

The final values of the optimization using the surrogate with the parameters combinations selected are shown in Table 6. The test case with $\tau = 0.100$ and $C_n = 4$ reached a lower cost value than the baseline. These results agreed with our expectations: as in the Dawn case, the value $C_n = 4$ seemed to be adequate to estimate the error in real time, and $\tau = 0.100$ was low enough to leave out the higher error cases but still use the surrogate frequently enough to speed up the optimization. In addition, this combination achieved an increase in the number of J evaluations while keeping a high SF convergence rate. Therefore, this combination was used for the subsequent tests.

We evaluated again the statistical significance of the results. The results of the test can be seen in Table 7. We conclude that the mean value of the optimization with the surrogate was lower than the optimization without the surrogate. In this case, the conditions were the most adequate for the use of the surrogate: each

Table 6. Relevant results of the optimization with the surrogate and various combinations of parameters in the EVVMMM case. The final combination is shown in bold.

C_n	**4**		6		8		Baseline
τ	**0.10**	0.14	0.10	0.14	0.10	0.14	
J^* (kg)	**963**	1066	1063	1066	1066	1066	1044
Final MAE	**0.0881**	0.0838	0.0937	0.0876	0.0898	0.0995	–
J evaluations	**728**	1240	336	1648	1568	1592	232
SF evaluations	**698**	662	744	670	685	644	1053
SF conv. rate	**0.838**	0.796	0.817	0.797	0.788	0.828	0.901

trajectory had a higher number of legs, which meant a larger computation time per individual, each leg had a higher computation time than in the previous cases, and the convergence rate of the SF method was very high, so chances of selecting an invalid trajectory when the surrogate was used were low. Nonetheless, the best result obtained had a higher cost value than the best result in the reference.[21] We hypothesize that the reason was the different transcription method used in this work and in the reference. Our solution is guaranteed to be feasible, while the one in the reference might not be feasible.[24]

Table 7. Statistics of the results for the EVVMMM case.

Reference 21	Baseline			Surrogate			Confidence[a]
$J^*_{(1)}$ (kg)	μ_{J^*} (kg)	σ_{J^*} (kg)	$J^*_{(1)}$ (kg)	μ_{J^*} (kg)	σ_{J^*} (kg)	$J^*_{(1)}$ (kg)	
236	1088	68	998	1037	142	880	62 %

[a] The confidence level is to reject the null hypothesis of the t-test (both means are equal). A negative confidence value indicates that the mean of the baseline results is lower (i.e. better) than the mean using the surrogate.

Best Result Statistics

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

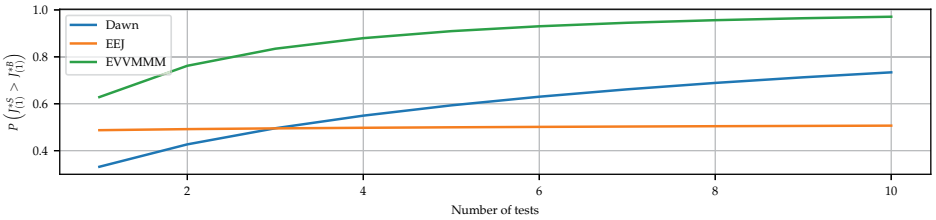


Figure 4. Probability of obtaining a better result when using the surrogate than with the baseline as a function of the number of cases tested.

CONCLUSIONS

The use of the surrogate during the optimization is always expected to be beneficial in the EVVMMM case, decreasing the average propellant cost of the trajectories found at the end of a single optimization by around 12%. This case was the most promising for the use of the surrogate due to the long computation time required for each trajectory. The Dawn case was not expected to be as adequate for the use of the surrogate because the cost function evaluations are faster, but the surrogate is also expected to be beneficial if more than 4 optimizations with different initializations are run for the same problem. Finally, the EEJ case has a very low convergence rate for the SF method so the optimal strategy for the surrogate is to be very conservative with the predictions, resulting in almost identical results to the baseline. This indicates that the surrogate is more useful in cases with a large SF convergence rate and a large computation time per individual.

These results were obtained without adapting the OS-ELM-MAS architecture parameters to each case. In fact, the architecture was selected based on data from the Dawn case and the surrogate showed a better performance in the other two cases. This proved that the OS-ELM-MAS method can be applied to different problems without having to adapt the parameters. This may potentially save an important amount of computation time, as the data to test the parameters does not need to be generated, and repeated training with different parameters combinations can be avoided. However, the tuning of the C_n and τ parameters was required to achieve better results with the surrogate than with the baseline. Nonetheless, there seems to be a relationship between the optimal parameters and the parameters of the transfers used for training. The proof of this is that we managed to find a combination of parameters for which the surrogate had a better performance than the baseline by testing a small number of carefully selected parameter combinations. This show that the surrogate is beneficial if several optimizations are going to be performed for the same problem, as the extra effort of finding the optimal parameters will be rewarded with a more optimal final result.

REFERENCES

[1] T. J. Debban, T. Troy McConaghy, and J. M. Longuski, "Design and Optimization of Low-Thrust Gravity-Assist Trajectories to Selected Planets," *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, No. August, 2002, 10.2514/6.2002-4729.

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

[3] J. A. Englander and B. A. Conway, "Automated Solution of the Low-Thrust Interplanetary Trajectory Problem," *Journal of Guidance, Control, and Dynamics*, Vol. 40, No. 1, 2017, pp. 15–27, 10.2514/1.G002124.

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

[5] B. Dachwald, "Optimization of Very-Low-Thrust Trajectories Using Evolutionary Neurocontrol," *Acta Astronautica*, Vol. 57, No. 2-8, 2005, pp. 175–185, 10.1016/j.actaastro.2005.03.004.

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

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

[8] D. Hennes, D. Izzo, and D. Landau, "Fast Approximators for Optimal Low-Thrust Hops Between Main Belt Asteroids," *EEE Symposium Series on Computational Intelligence*, 2016, 10.1109/SSCI.2016.7850107.

[9] A. Mereta, D. Izzo, and A. Wittig, "Machine Learning of Optimal Low-Thrust Transfers Between Near-Earth Objects," *Lecture Notes in Computer Science*, Vol. 10334 LNCS, 2017, pp. 543–553, 10.1007/978-3-319-59650-1.

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

[11] C. Ampatzis and D. Izzo, "Machine Learning Techniques for Approximation of Objective Functions in Trajectory Optimisation," *Proceedings of the International Joint Conference on Artificial Intelligence 2009 Workshop on Artificial Intelligence in Space*, No. September, 2009.

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

[13] L. Bouwman, Y. Liu, and K. Cowan, "Gaussian Process models for preliminary low-thrust trajectory optimization," *AAS/AIAA Astrodynamics Conference* (I. I. Hussein, K. R. Horneman, C. Scott, and B. W. Hansen, eds.), Portland, Maine, 2019.

[14] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme Learning Machine: Theory and Applications," *Neurocomputing*, Vol. 70, 2006, pp. 489–501, 10.1016/j.neucom.2005.12.126.

[15] C. T. Yaw, S. Y. Wong, K. S. Yap, and C. H. Tan, "Extreme Learning Machine Neural Networks for Multi-Agent System in Power Generation," *International Journal of Engineering and Technology (UAE)*, Vol. 7, No. 4, 2018, pp. 347–353, 10.14419/ijet.v7i4.35.22760.

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

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

[18] M. Bardi and I. Capuzzo-Dolcetta, "Optimal Control Problems with Continuous Value Functions: Restricted State Space," *Optimal Control and Viscosity Solutions of Hamilton-Jacobi-Bellman Equations* (M. Bardi and I. Capuzzo-Dolcetta, eds.), ch. 4, pp. 227–284, Boston, Massachusetts: Birkhäuser Boston, 1997, 10.1007/978-0-8176-4755-1.

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

[20] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal Approximation Using Incremental Constructive Feedforward Networks with Random Hidden Nodes," *IEEE transactions on neural networks*, Vol. 17, jul 2006, pp. 879–892, 10.1109/TNN.2006.875977.

[21] C. H. Yam, D. Izzo, and D. D. Lorenzo, "Low-Thrust Trajectory Design as a Constrained Global Optimization Problem," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, Vol. 225, No. 11, 2011, pp. 1243–1251, 10.1177/0954410011401686.

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

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

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

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

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

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

[28] H. A. David, *Order statistics*. Wiley series in probability and mathematical statistics, New York, New York: Wiley, 2nd ed., 1981.

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

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