Direct Optimization for Approximated Low-Thrust Interplanetary Trajectories

Rohan Patel *

June 5, 2022

Abstract

The Sims-Flanagan method of approximating low-thrust trajectories is implemented with the direct optimization model to find optimal interplanetary transfers. This method approximates continuous low-thrust arcs as a series of impulsive maneuvers separated by either fixed or variable time steps. Non-linear constraints are placed on the optimization method to ensure that the impulsive maneuver magnitude and segment time satisfy the low-thrust constraint described by Sims and Flanagan. Additional constraints are implemented for mission design parameters such as launch energy, final mass, and available thrust. It is able to find energy or mass optimal transfers between bodies. This paper covers the development of the optimizer, compares its results to an existing implementation of the Sims-Flanagan model, and is tested for interplanetary trajectories. The current implementation uses the two-body dynamical model and is able to read ephemeris data. Example cases to Venus, Mars, and Ceres are demonstrated.

1 Introduction

Since the successes of Deep Space 1, the viability and technological development of low-thrust propulsion systems for interplanetary trajectories has significantly increased. These trajectories can offer a considerable mass delivered advantage over the conventional high-thrust chemical propulsion systems. Previously prohibitively costly high-thrust trajectories can be reformulated to use low-thrust to increase the total ΔV and mass delivered[15]. For its advantages, optimization of low-thrust trajectories come at a cost in their computational effort and methods. Because low-thrust trajectories have continuous thrusting arcs, finding optimal solutions for these transfers is inherently difficult and resource intensive[11]. Performing long-duration low-thrust burns poise additional scheduling and hardware reliability considerations, though the benefits can outweigh the costs.

John Sims and Steve Flanagan at the NASA Jet Propulsion Laboratory pioneered a method to approximate these arcs in their paper "Preliminary Design of Low-Thrust Interplanetary Missions" in 1997[8]. This solution is derived from a series of impulsive ΔVs along with coasting segments, and is constrained by a maximum segment ΔV magnitude. The method and its implementation are described and implemented in a direct optimization problem in this paper.

Both direct and indirect methods exist to optimize low thrust trajectories. Indirect methods, or known as optimal control methods, are derived from variational calculus and are highly sensitive to the initial guess for the states and adjoints[10]. Programs such as VARITOP, SEPTOP[9], and MYSTIC[13] use this for continuous solutions in high-fidelity dynamical models. While powerful and comprehensive, these methods are slow and time consuming, especially during the rapid prototyping phase of trajectory design. Therefore, the alternate formulation of low-thrust trajectory design problems is done with the direct method discussed by Sims and Flanagan. The current state of the art tools are MALTO and GALLOP, and the Advanced Concepts Team at ESA has actively worked to improve computational methods and dynamical models within this problem framework[15]. NSTOP is the current in-house implementation of this method at CU Boulder[12]. The direct method is a non-linear programming problem (NLP) and has its cost minimized, subject to constraints through an

*Masters Student, Aerospace Engineering Department, ASEN6020 Spring 2022

optimal decision vector. This method is more robust to the initial state, but the decision vector can become considerably large as more intermediate thrusting arcs are included.

In this paper, the direct method of optimization is utilized to create low-thrust interplanetary trajectories using the Sims-Flanagan approximation. The paper is broken into three main sections: 1) Problem Formulation, 2) Example Cases, and 3) Future Work.

2 Problem Formulation

2.1 Sims-Flanagan Low-Thrust Model

The Sims-Flanagan method to approximate low thrust trajectories is designed to reformulate a continuous time problem into a discrete non-linear programming one. This is done by approximating the contributions of a continuous thrusting arc as a single impulsive maneuver followed by a coasting period[9, 10, 8]. An arbitrary number of segments (n) can be selected to characterize the low-thrust trajectory. The decision vector ($\vec{X}$) contains the maneuver components for each segment as well as the maneuver time.

$$\vec{X} = \begin{bmatrix} \Delta v_{1x} \\ \Delta v_{1y} \\ \Delta v_{1z} \\ \vdots \\ \Delta v_{nx} \\ \Delta v_{ny} \\ \Delta v_{nz} \\ t_1 \\ \vdots \\ t_n \end{bmatrix}_{(3n+n)\times 1} \tag{1}$$

Eq. 1 shows a variable-time decision vector. The first set of elements in this vector corresponds to the Cartesian coordinates of each segments maneuver. Polar coordinates, or any set of coordinates for that matter, can be used here. The next set of elements are the encounter times for each segment. In a variable-time setup, the optimizer is able to change the propagation time between maneuvers to better exploit the dynamics of the system. A fixed time optimizer will omit all but one time value from $\vec{X}$. This value will correspond to the time of flight (TOF) to the arrival body, and the optimizer will propagate each segment forward by the TOF divided by the number of segments. Variable-time optimization adds $n-1$ more elements to the decision vector which will increase computation times. A direct optimization algorithm is allowed to vary this independent vector such that the cost is minimized and the constraints on the function are met. The following constraint on the impulsive maneuver formulated for the decision vector must be enforced by the optimizer to meet Sims and Flanagan's low-thrust criteria:

$$0 \geq \frac{|\vec{\Delta v}_i|}{dt_i} - \frac{T_{max}}{m} \qquad i = 1, ..., n \tag{2}$$

where Δv_i is the the impulsive maneuver vector, dt_i is the segment time length, T_{max} is the maximum thrust, and m is the spacecraft mass. At each segment (i), this inequality constraint must be satisfied for a proper low thrust approximation. An equality constraint is introduced to match the post-propagated decision vector state with the desired final state. Either position, or position and velocity are good candidates for this equality constraint. The latter is useful for a zero ΔV capture to the arrival body (V_∞=0). Additionally, constraints on the launch energy (C3), flight time, mass, and thrust can be enforced to model mission scenarios.

2.2 Direct Optimization Solver

2.2.1 MATLAB Toolboxes

For this problem, the MATLAB Optimization Toolbox was utilized for direct optimization. Available through the university's software suite, this extension package for the base install of MATLAB allows

users to formulate optimization problems in a variety of methods and includes black-box solvers for linear programming (LP), quadratic programming (QP), nonlinear programming (NLP), and other methods[5]. Modeling, problem-based optimization, or solver-based optimization schemes can be setup by the user to satisfy their requirements. The Sims-Flanagan method of direct optimization was setup as a nonlinear programming, problem-based optimization setup. The specifics on its implementation are discussed in the following subsections.

In order to expedite solution computation times, MATLAB's Parallel Computing Toolbox was utilized. In conjunction with the Optimization Toolbox, this extension automatically configures the optimization problem to work across a multi-core processor and MATLAB calls this feature "parallel-pools"[6]. For this paper, we use a computer equipped with an Advanced Micro-Devices (AMD) Ryzen 7 5800X 8-Core processor which has a total of 16 logical compute cores. Figure 1 consists of two screen-captures of the Windows Task Manager to show CPU utilization across these cores. It demonstrates how MATLAB automatically offloads computation to all the available logical cores on the processor (right sub-figure) versus only using a single thread (left sub-figure) for an example optimization problem. The CPU utilization is increased from an average of 10 to 50 percent in this process. An increase in the system memory was also noticed when using parallel-pools for computation, though this only rose by a few percent. The example Earth-Mars transfer problem, discussed further in this paper, took on average 25 seconds to compute using parallel-pools, and around 80 seconds to compute without it. In testing it was evident that using the Parallel Computing Toolbox with the Optimization Toolbox, on a modern-day processor, can yield significant reductions in computation times.

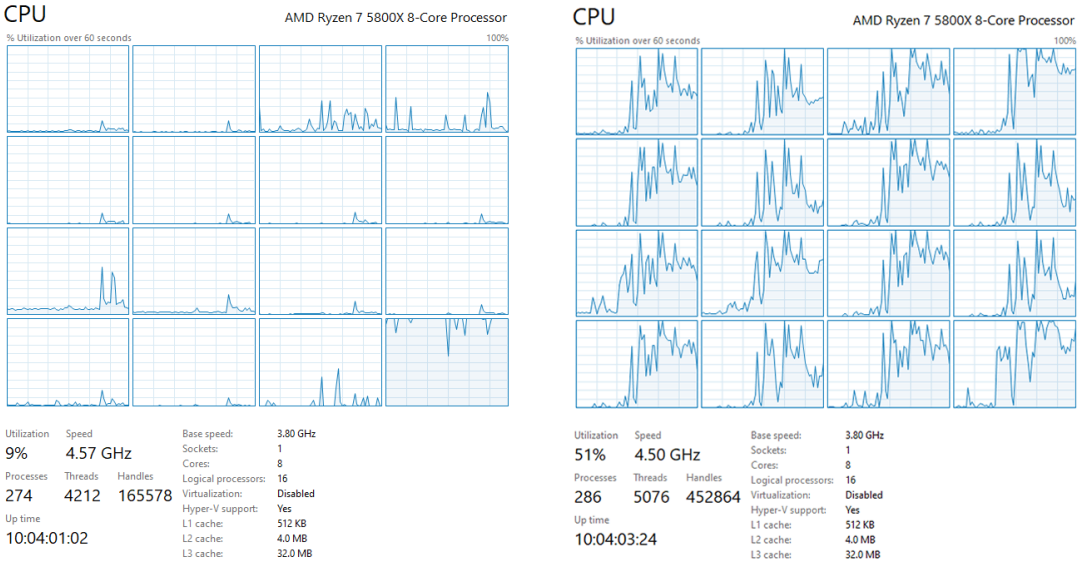

Figure 1: CPU load per logical core during optimization without (left) and with parallel computing (right).

It is worth noting that all compute times presented in this paper, except for the multi-start subsection results, include printing each iteration to a table in MATLAB's command window and plotting the cost graphically. This process is time intensive and considerably slows down each convergence attempt. However, it was essential to see how the algorithm performs during the development phase and thus computation time results discussed here include this step. To put this speed difference into context, the Earth-Mars transfer discussed in the first example took between 25 and 30 seconds to compute with the graphics displays turned on. Without it, the compute time dropped to an average of 6 to 15 seconds. A significant advantage is achieved in turning these aids off, and should be done so for batch optimization runs.

2.2.2 FMINCON Solver and Options

The MATLAB Optimization Problem framework relies on FMINCON to find the minimum of constrained nonlinear multi-variable function. This method is gradient-based that requires the objective

and constraint functions to be both continuous and have continuous first derivatives[3]. The minimization problem is shown in Eq. 3. J is the cost function, $\vec{c}$ and $c\vec{e}q$ and the non-linear inequality and equality constraints. $[A]$, $[Aeq]$, $\vec{b}$, and $b\vec{e}q$ are the linear inequality and equality constraint matrices and vectors. Finally, $\vec{ub}$ and $\vec{lb}$ are upper and lower bounds on the decision vector. For this problem formulation, linear equality and inequality constraints are not required.

$$\min_{\vec{X}} J(\vec{X}) \text{ such that } \begin{cases} c(\vec{X}) \leq \vec{0} \\ ceq(\vec{X}) = \vec{0} \\ [A] \cdot \vec{X} \leq \vec{b} \\ [Aeq] \cdot \vec{X} = b\vec{e}q \\ \vec{lb} \leq \vec{X} \leq \vec{ub} \end{cases} \tag{3}$$

FMINCON is configured to use the Interior-Point Algorithm which solves a series of approximate minimization problems. This is accomplished by taking either of two step methods - a direct or conjugate step based on solving the KKT equations or using a trust region respectively[3]. For the context of this paper, specifics on this algorithm won't be covered but its configuration to solve the low-thrust transfer problem will be discussed below. FMINCON is set to use the central finite difference method to approximate derivatives, and uses the BFGS method to approximate the Hessian function. The optimizer itself has several inputs that can be tuned by the user. In this problem, the number of iterations and function calls are limited to 600 and 60,000 respectively. The constraint function evaluation tolerance is set at 1e-6 and norm of the step tolerance is limited to 1e-12. In testing, these values proved to be useful in terminating the optimization if a solution was not reached, and also allowed the optimizer to take small steps if needed to refine the solution.

Feasibility and first-order optimality are computed by FMINCON at each iteration to determine if the optimization is required to continue. The feasibility evaluates how well the constraints are satisfied, and the first-order optimality reports the necessary condition for direct optimization. The later measure does not cover the sufficiency conditions, and thus the optimization can prematurely terminate at a local minimum[4]. Values for feasibility and optimality should be investigated by quantifiable numbers. For example, for feasibility, the final mass, position mismatch at the terminal segment, and thrust usage should be verified post-optimization to ensure constraints are met. For optimality, multiple solutions can be performed to verify that the cost is minimized. A method to find a global solution is discussed in the Multiple Start subsection.

One of the key advantages of utilizing the MATLAB Optimization toolbox problem formulation over interfacing with FMINCON directly is the ability to switch methods of computing derivatives and changing optimization criteria. For example, automatic differentiation can be employed over the finite differences method for the objective and constraint derivatives simply by modifying the input options in the problem setup[2]. Further, constraints and cost functions can be combined, switched, or removed by modifying input properties. This convenience makes rapidly prototyping and exploring how optimization parameters change the computation time and solution easy to do and understand.

2.3 Non-Dimensionalization

Before the transfer problem is sent for optimization, the dimensionalized inputs are converted into non-dimensionalized (ND) quantities for ease in gradient traversing by the optimizer. This includes the full state, thrust, ISP, mass, and time of flight. Non-dimensionalization of the position, length (L), is done by taking the reference state at Earth and dividing the position vector by 1 AU (149597870.691 km). ND time (T) is given as the the position divided by the circular velocity of Earth. The following expressions summarize the key variables converted to ND units:

$$L = \frac{\vec{r}_0}{1AU}, \qquad T = \frac{\vec{r}_0}{\sqrt{\frac{\mu}{|\vec{r}_0|}}} \tag{4}$$

where $\vec{r}_0$ is the departure position vector from Earth. From the length and time conversions, subsequent ND units for velocity, acceleration, thrust, mass, and Isp, can be derived. Converting back to a dimensionalized quantity is as simple as multiplying the ND quantity by the conversion factor.

2.4 Dynamical Model

In this implementation of the Sims-Flanagan method, the following simple two-body dynamical system is considered:

$$\ddot{\vec{r}} = \frac{-\mu}{r^3}\vec{r} \tag{5}$$

where $\vec{r}$ is the vector between the massive central body and the relatively insignificant spacecraft mass, and μ is the gravitational parameter of the central body. There are various methods to propagate the two-body problem including: Kepler's equations, Kepler's universal variable, the Taylor integration method, and numerical integration with either fixed or variable time steps. For this formulation, Kepler's equations of an ellipse and hyperbola seen below:

$$M_e = n_m(dt_p) = E - e\sin E \qquad\qquad M_h = \sqrt{\frac{\mu}{a^3}}(dt_p) = e\sinh H - H \tag{6}$$

were used to propagate the solution depending on the computed eccentricity given the state vector. M_e and M_h are the elliptical and hyperbolic mean anomalies, dt_p is the time past periapsis, n_m is the mean motion, E and H are the hyperbolic and eccentric anomalies, and e is the eccentricity. This method, though not utilizing the universal variable formulation of Kepler's problem, is still a relatively fast and convenient method of two-body propagation. In testing, numerically integrating the trajectory using ODE45 took over 450 seconds to compute an Earth-Mars transfer. By comparison, the current algorithm uses less than 30 seconds to find the same solution. The possibility of a fixed step integrator like the Runge-Kutta 4th order is viable if the dynamical model needs to include perturbations.

Planetary states are derived by two methods: 1) Meeus algorithm or 2) SPICE data. The Meeus ephemeris algorithm is a fast and reasonably accurate method of finding planetary states given a date in Julian centuries. It utilizes a polynomial lookup and curve-fit function to derive planetary Keplerian elements. These elements can then be converted to a Cartesian inertial state. While fast, this algorithm does not include small-bodies. Therefore, a second method is also implemented in this model. The JPL Navigation and Ancillary Information Facility (NAIF) SPICE system offers high-fidelity ephemerides for planets and small-bodies[1]. This data is imported via *.bsp* ephemeris files through the CSPICE MICE Toolkit library for MATLAB available through the NAIF website.

2.5 Cost Functions

The optimization routine has two primary cost functions that were evaluated throughout this work. MATLAB's optimization setup allows for easy switching between these functions for prototyping and analysis. Eq. 7 represents a minimum energy cost function where each segment maneuver norm is squared and the sum total makes up the cost. This function demonstrated a significantly better and more reliable convergence over the minimum sum of the maneuver magnitudes cost function.

$$J_1 = \sum_{i=2}^{n} |\Delta v_i|^2 \tag{7}$$

$$J_2 = -m_f \tag{8}$$

The second cost function considered is the maximum final mass function as shown in Eq. 8. The time to find an optimal solution is roughly halved compared to the maneuver sum and final mass functions. For the examples shown in the next section, the minimum energy cost function is used.

2.6 Constraints Setup

A modified cost function to include the constraints is not considered in this work. Therefore, constraints are handled separately via an external function that is then evaluated by the solver. The following constraints satisfy the criteria established by Sims and Flanagan as well as include mission design constraints. Eq. 9 shows the optimization based constraints. The non-linear equality constraint, $c\vec{e}q$, ensures that the component differences between the final propagated state from $\vec{X}$ and the desired body state vector is zero within tolerance. The non-linear inequality constraint $\vec{c}$ is the low thrust constraint

[1] JPL NAIF Toolkit and Ephemerides Website": https://naif.jpl.nasa.gov/naif/toolkit.html.

as described in Eq. 2. Finally, for optimizer stability, another non-linear inequality constraint is added to ensure that each segment time is at least greater than 3600 seconds. This time value is arbitrary and is converted to ND units before being enforced.

$$\vec{ceq} = \vec{0}, \qquad \vec{c} \leq \vec{0}, \qquad \vec{dt} \geq 100s \tag{9}$$

The next set of constraints, seen in Eq. 10 outline the spacecraft specific constraints. The final mass must be greater than or equal to the minimum mass set by the user. The launch energy (C3) must be less than or equal to the maximum launch energy. And finally, the difference between the available thrust at the radial distance from the sun and the thrust required by the trajectory should be greater than or equal to zero.

$$m \geq m_{min.}, \qquad C3 \leq C3_{max}, \qquad \vec{Tr} \geq \vec{0} \tag{10}$$

Algorithm 1 outlines how these constraints are modeled within the FMINCON evaluation function. It begins by separating the parts of the decision variable into the maneuvers and the times. The arrival body's state is then found. For each segment, the state is updated and then propagated. The resulting maneuver and time is then evaluated for the mass lost, thrust, and these are saved for evaluation of the non-linear constraints. If it is the first segment, the launch energy is computed instead. Finally, the post segments propagated state is subtracted by the desired state to create the equality constraint.

Algorithm 1 Constraints Evaluation Pseudo-code Algorithm

Unpack ND constants

$\vec{\Delta v}$ = decision vector $\vec{x}$ velocity variables for each segment

if variable-time **then**
  $\vec{dt}$ = time variables for each segment
  Total Time = dim. $\Sigma|\vec{dt}|$
else
  Extract dt from $\vec{x}$
  Total Time = dim. dt t_{Guess}
end if
Query state and convert to ND of arrival body using Total Time $\vec{x_f}$.

$\vec{x} = \vec{x}_i$, $m = m_i$ ▷ Initialize Full State at Departure Body
for i=1:n **do**
  $\vec{x} = \vec{x} + \vec{\Delta v}_i$
  $\vec{x} = keplerpropagation(\vec{x}, dt_i, mu{=}1)$
  if i=1 **then**
    $C3 = |\Delta v|^2$; ▷ Launch Energy
  else
    $r = |\vec{x}(1:3)|$
    $T_{max} = (\frac{1}{r^2})T_{max1AU}$ ▷ Max Thrust Available
    $m = m \exp(\frac{-|\vec{\Delta v}_i|}{Isp * g_0})$ ▷ Mass Decrement
    $T = \vec{\Delta v}_i \frac{m}{dt_i}$ ▷ Thrust Required
    $c_i = \frac{|\vec{\Delta v}_i|}{dt_i} - \frac{T_{max}}{m}$ ▷ Low-Thrust Approximation
    $Tr_i = T - T_{max}$ ▷ Thrust within max. avail.
  end if
end for

$ceq = \vec{x} - \vec{x_f}$ ▷ Final State Vector Constraint

2.7 Multiple Start

Finding global optima is inherently difficult for a multi-dimensional problem as such. Therefore, several strategies exist to attempt to find these solutions. One commonly employed method is basin hoping[12]. This method iterates on the solution by randomly perturbing the previous local minimum solution[1, 7]. Basin hoping is limited by computational resources, meaning it can run for an arbitrary number of restarts. Its implementation into this algorithm is considered by sequentially running the optimizer p number of times where each iteration is k. A perturbation vector is added to the local optimal decision vector $\vec{X}^*_{k-1}$ and this is shown in Eq. 11 where $\vec{X}_0$ is the first iteration's initial guess decision vector. The MATLAB random number function is utilized to create either positive or negative scaling terms between 0 and 1.

$$\vec{X}_{0,k} = \vec{X}^*_{k-1} + \frac{\vec{X}_0}{10} rand(-1, 1) \tag{11}$$

At the time of developing this algorithm, the perturbation added to the next initial guess is arbitrary. Further research needs to be done in order to refine this guess, but due to project time constraints, the following function is used. Figure 2 shows an example case of multiple start being applied to the

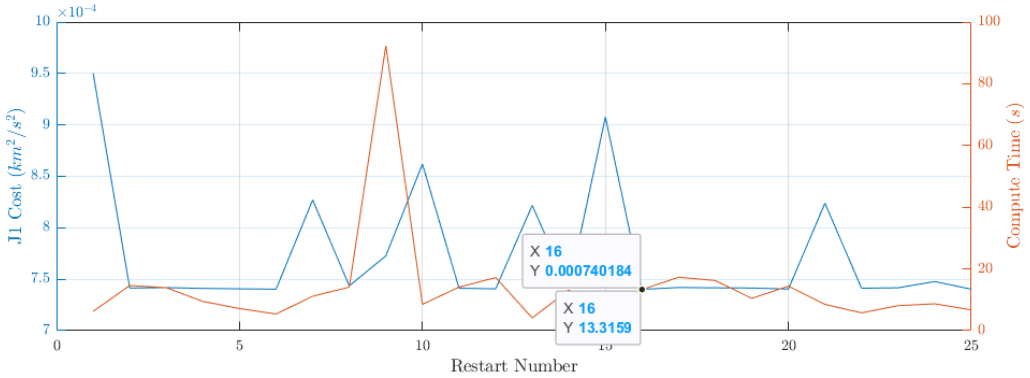

Figure 2: Earth-Mars transfer with multi-start of 25 attempts. Restart number 16 has the lowest cost from the set of solutions. Note that computation times are faster due to suppressing real-time graphical and tabular iteration outputs.

fixed-segment time Earth-Mars transfer for 25 iterations. Due to the arbitrary nature of the perturbations, results and their respective computation times will vary. A string of unfortunate initial guess perturbations can lead to a lack of convergence. This is seen in restart number 9 as the computation time exceeded 80 seconds and the optimizer was prematurely terminated due to reaching the maximum function calls. If a solution is unattainable, the multi-start algorithm takes the solution from the previously converged solution and applies another random perturbation to it. Within this example, restart number 16 had the lowest cost. The cost is compared to the previous iteration and if it is lower, $\vec{X}^*_k$ is saved. The method of basin hoping is commonly employed in very high-dimensional topological problems, and offers a way to find a 'global' solution which would inherently be better than only using completely arbitrary initial guesses once.

3 Example Cases

3.1 Variable Segments, Fixed Time

In this example, we will investigate the optimization routine's performance for a fixed segment time solution between two roughly circular, co-planar orbits. For this case, a transfer between Earth and Mars is considered which departs on a fixed date of July 01, 2020. The maximum thrust available to the spacecraft at 1AU is 0.35 newtons, the Isp is fixed at 3000 seconds, and the guess time for the transfer is 500 days. The maximum launch energy is capped at 12 km^2/s^2, the spacecraft has a initial mass of 3000 kg, and minimum final mass is 2000 kg. A similar problem setup is created in MALTO (discussed in the introduction), and is used as a comparison to see how the optimization routine compares to the state of the art Sims-Flanagan optimizer. It is important to note that there will be slight modeling differences resulting in variation in the solutions between the two algorithms. This is a preliminary examination of the optimizer to see if its performance is appropriate. Figure 3 shows a 15 segment E-M transfer for both algorithms where the left subfigure is the current optimizer and the right is MALTO. Red and blue vectors indicate the anti-maneuver direction (resulting motion of the spacecraft). The blue vector is the launch maneuver and the red vectors indicate each impulsive segment. The left subfigure trajectory alternates between green and cyan colors to indicate segments. Both trajectories have similar structures in their maneuver placements, launch energy, and flight times. It is important to note that subtle differences are present due to MALTO's default optimization cost function being to maximize final mass, and not minimize energy. Figure 4 quantifies the trajectory with information

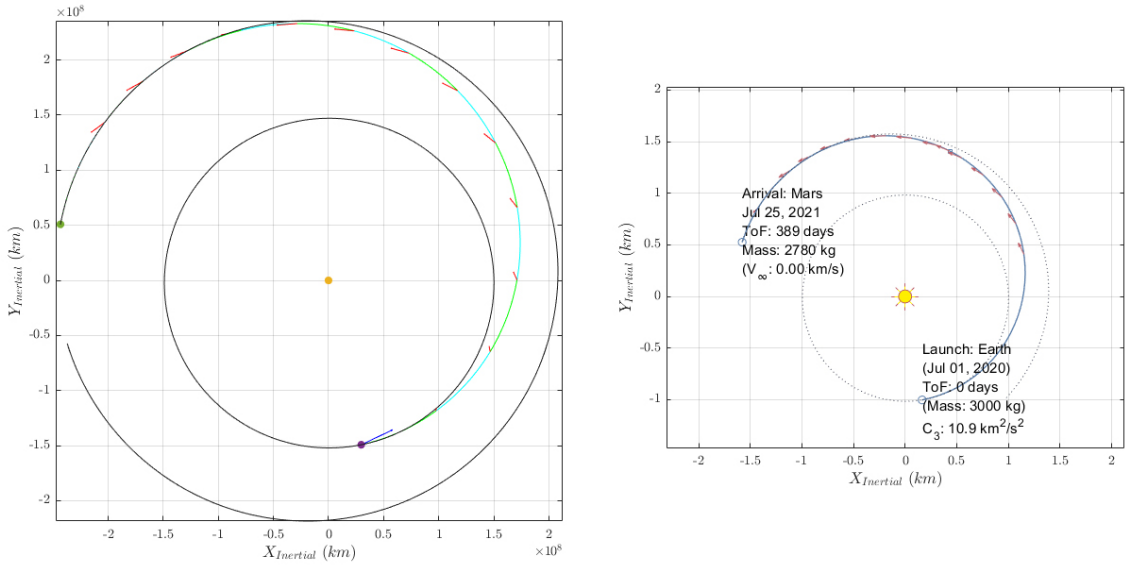

Figure 3: Earth-Mars rendezvous transfer using 15 variable-time segments. The left subfigure is the resulting trajectory from the optimizer, and the right subfigure is a MALTO optimized trajectory with similar initial inputs. Note that exact modeling between these two is not targeted, and this comparison is done to sanity-check the results from the optimizer.

on the ΔV, mass, thrust, and low-thrust constraint for each segment. Recall that by the problem formulation, the first segment ΔV, shown as a blue, is the launch ΔV and is not factored into the cost function. Therefore, the segments will always start at number 2 for these plots. It is evident that the low-thrust constraint is not violated, the final mass is above the minimum, and the thrust is within the maximum bound. The trajectory uses a total low-thrust ΔV of 2.799 km/s and has a departure C3 of 10.889 km^2/s^2, and a flight time of 400 days. MALTO was fixed to depart on the same day with a maximum C3 of 10.90 km^2/s^2, and has a maximum thrust (unaffected by solar panel power) of 0.35 N. Comparing this result to a MALTO Earth-Mars solution, the trajectories are similar and the transfer TOF for the MALTO solution is 389 days. The thrust profile for the MALTO solution appears to be always on full thrust or off. The variation of this versus the optimizer's computed thrust versus segments is likely due to the cost functions used (minimum energy versus maximum mass). Overall,

this comparison demonstrates that the optimizer's solutions are comparable to existing Sims-Flanagan implementations of low thrust approximation.

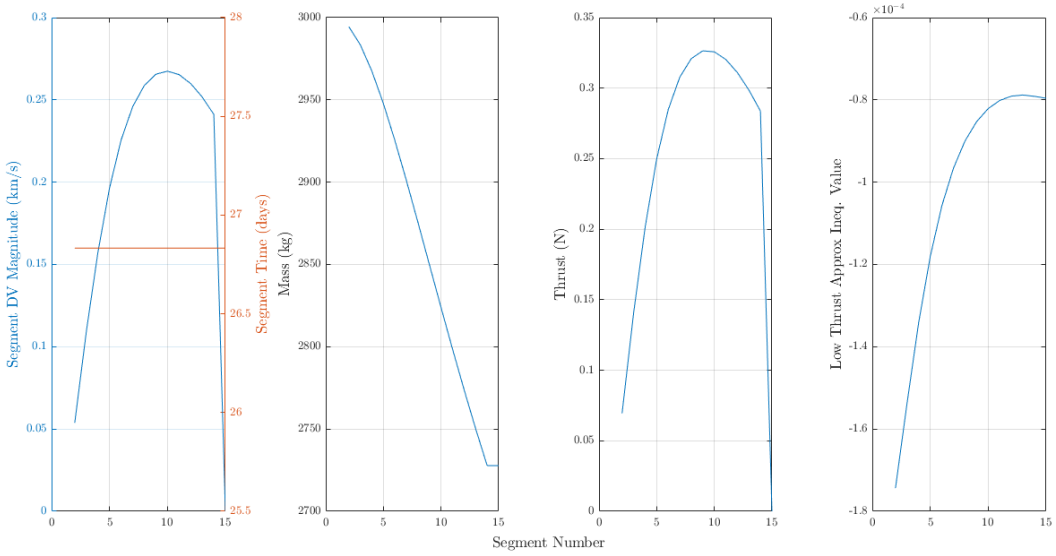

Figure 4: Earth-Mars transfer ΔV, segment time, mass, thrust, and low-thrust constraint.

Analysis of the optimizer's performance can be extended to the number of segments used. Generally speaking, as the number of segments increases, the computation time will also increase. This is due to the decision vector length increasing by 3 variables for every additional fixed time additional segment. This number increases to 4 if we factor in variable time. It is expected that this method of approximating a continuous thrust arc with a small number of impulsive maneuvers will lead to solution granularity and thus inaccurate results. The sensitivity of the solution to the number of segments is tested on the Earth-Mars case and its results are shown in Table 1. Segments are varied from 5 to

Num. of Segments	Launch C3 (km^2/s^2)	Low-Thrust ΔV (km/s)	TOF ($days$)	FMINCON		Computation Time* (s)
				Feasibility	Optimality	
5	10.89458	2.765689	461.5336	4.63E-12	8.36E-07	5.694074
6	10.89704	2.778467	444.8491	6.99E-12	2.56E-07	6.445096
7	10.89866	2.785615	433.4766	3.37E-12	2.56E-07	6.903601
8	10.89812	2.790016	425.2299	4.78E-12	6.36E-07	8.378106
9	10.89675	2.792905	418.9821	5.39E-10	3.85E-07	8.75188
10	10.89519	2.794896	414.0897	1.43E-11	4.87E-07	8.897127
12	10.89232	2.797362	406.9336	6.56E-12	3.60E-07	13.62838
15	10.88891	2.79924	400.0017	9.75E-12	2.56E-07	17.68791
21	10.88433	2.800712	392.3692	5.94E-12	5.45E-07	27.0693
25	10.57685	2.802386	393.0869	9.99E-15	4.63E-05	32.22581
29	10.88052	2.801335	375.4584	2.58E-11	4.89E-07	64.72807
31	10.50601	2.804116	387.4568	4.45E-13	1.48E-03	44.69821
35	10.46057	2.805302	384.6884	1.65E-13	2.09E-03	75.93914

Table 1: Earth-Mars fixed-time transfers for varying number of segments. The granularity of having less than 10 segments is noticeable in the total low-thrust ΔV. A plateauing of the total low-thrust ΔV is noticed as the number of segments increases past 10. This and the computation time should be taken into consideration when optimizing solutions. *Computation time includes displaying each iteration graphically and in MATLAB's command window. Speeds are considerably faster when these real-time outputs are suppressed.

35 and the resulting parameters of the trajectory and optimization are presented. It is evident that there is a dependence on the number of segments and the solved trajectory. This is counter-intuitive as the optimal trajectory, general speaking, should not be dependent on the optimization inputs. A possible explanation for this behavior is due to the algorithm not being able to use multi-start in this test to find the best solution within a certain number of trials. Our method to find "the optimal

trajectory" is to find many candidate local optimal solutions and find the one with the lowest cost. Without including this, there is variation that can creep into the results. From Table 1, a plateauing is seen in the total ΔV as the number of segments is increased starting at roughly 10 segments. The launch energy remains similar for the solutions with the lowest first-order optimality. The flight time decreases as the number of points increases. A likely explanation for this is the optimizer's efforts to stay within the low-thrust approximation given smaller impulsive maneuver magnitudes. All solutions reach Mars with sub-10 kilometer position differences and sub-1e-08 kilometers per second velocity differences. These results are found in post processing given the solution's feasibility. It is important to note that tuning parameters and extensive testing of this algorithm have not be performed yet due to a lack of time. If time permits, additional testing should be conducted to more rigorously find a good number of segments that provides a good trade-off between computation time and solution accuracy.

3.2 Variable Time

The optimization algorithm has the ability to modify the decision vector to incorporate variable time solutions. In this example, we investigate this functionality and how it affects the solution by modeling a Earth to Venus transfer with 15 segments. The time of flight guess is 530 days and a maximum of 1.00 newtons of thrust is available at 1 AU. The mass and Isp are identical to the previous example. Figure 5 shows the trajectory departing from Earth and arriving at Venus with a zero incoming V_∞ (states of the spacecraft and body are identical at final time). An important takeaway with this example is how the optimizer places each impulsive maneuver by controlling the time between segments. The solver realizes that a greater energy change is attainable when the maneuvers are placed closer to perihelion and so the impulsive maneuver model is exploited in this manner. Long coasting arcs are observed

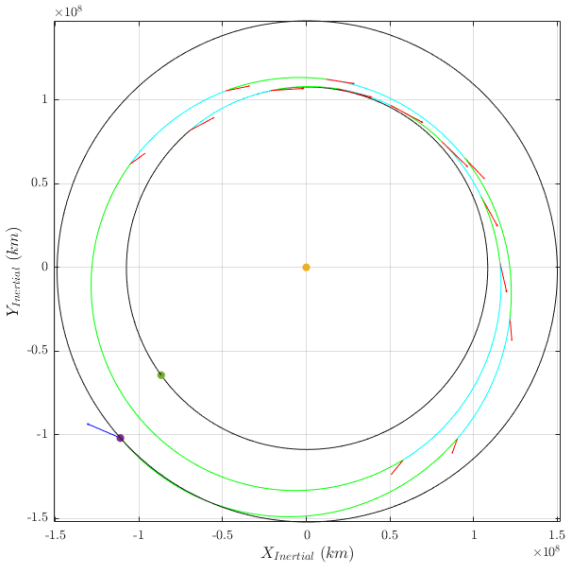

Figure 5: Earth-Venus rendezvous transfer using 15 variable-time segments. Red and blue vectors indicate the anti-maneuver direction (resulting motion of the spacecraft). The blue vector is the launch maneuver and the red vectors indicate each impulsive segment. The trajectory alternates between green and cyan colors to indicate segments.

both in Figure 5 and Figure 6's left-most sub-figure. The roughly 120 day time between segments 7 and 8 correspond to the coasting arc seen in the top figure. In testing, it is worth noting that the optimizer will utilize more thrust magnitude per segment for variable time transfers as opposed to fixed time. In this example, the minimum energy cost is used. This is measured as the sum of each segment's ΔV squared. During post-processing, the sum total of the impulsive ΔV of each segment is considered. The fixed time transfer, not shown in the figures, had a total of 4.307 km/s while the variable time had a total of 3.719 km/s. However, the fixed time solution had a peak thrust magnitude of 0.420 newtons

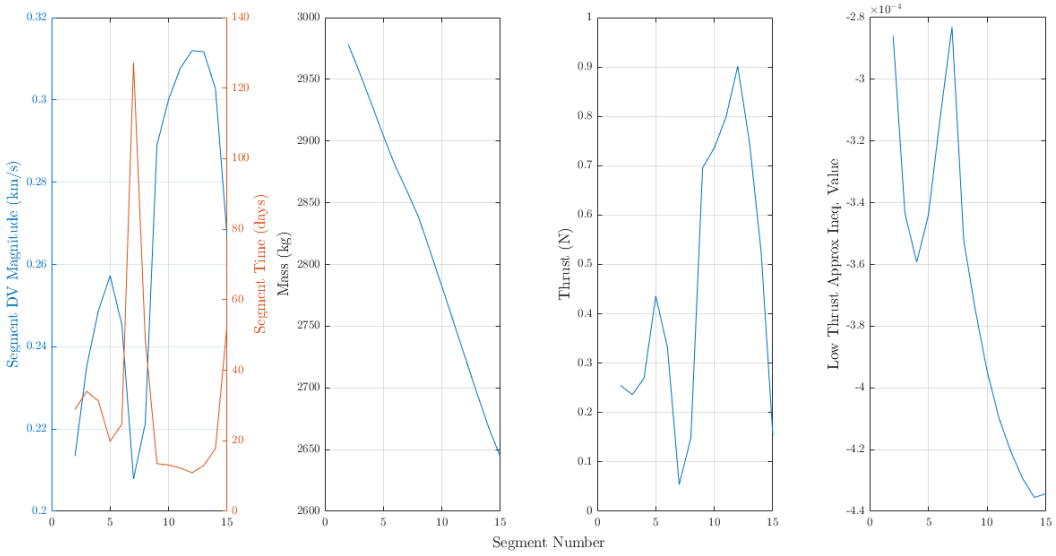

Figure 6: Earth-Venus transfer ΔV, segment time, mass, thrust, and low-thrust constraint.

while the variable time had a peak of 0.900 newtons. The solver is limited by the fixed time placement of each impulsive maneuver which means it, overall, cannot make the greatest change in orbital state when compared to the variable time optimization. For the variable time case, the optimizer is able to exploit the dynamical system's properties (such as burning at periapsis for the greatest energy change), in order to effectively reduce the total impulsive maneuver cost. Both conditions still satisfy the low-thrust approximation proposed by Sims and Flanagan, but yield considerably different results which is worth noting during the trajectory design exploration phase.

3.3 Out of the Ecliptic, Multi-Revolution Transfers

For this test case, a direct transfer from Earth to rendezvous at Ceres is considered. Ceres is chosen to demonstrate the algorithm's ability to read SPICE ephemerides which would theoretically allow this optimization routine to work for any body with an ephemeris. The previous cases have been of bodies that are roughly within the ecliptic, and so Ceres is a great candidate to test the optimizer's ability to find solutions to inclined bodies. It has a heliocentric inertial inclination of 10.6 degrees. Figure

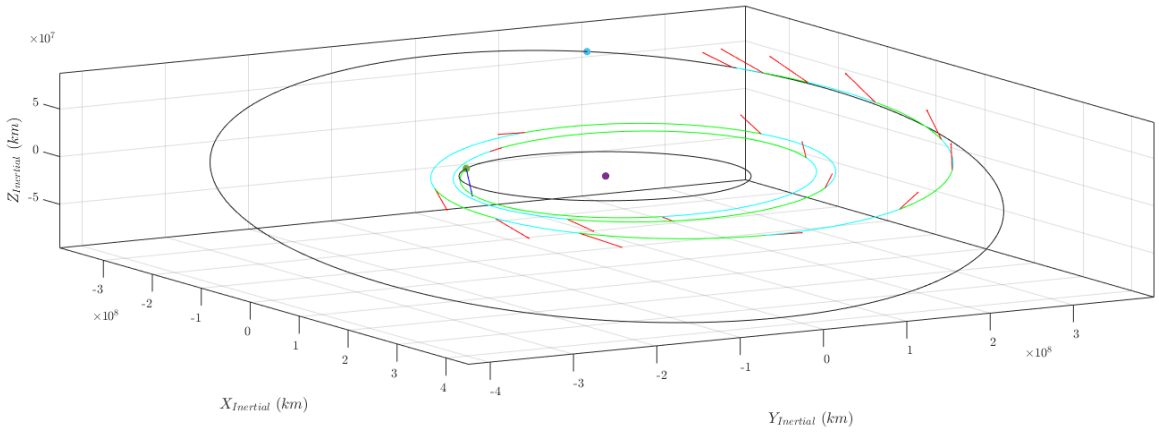

Figure 7: Variable-time transfer between Earth and Ceres showing the inclination differences between the two bodies.

7 is the resulting trajectory as seen from a three-dimensional view. The optimization routine is able to find solutions as such with a little bit of trial and error in the initial guesses. The guess TOF was

gradually increased from 300 days to 1000 days per run of the optimization (feeding back the results of the previous run into the next optimization). Then, the initial guess maneuver magnitude was increased from 0.001 to 0.5 km/s to guide the optimizer in the direction of using more ΔV to find the solution. Without these changes, the optimizer would either violate the thrust and(or) mass constraints to attempt a 0-revolution transfer to Ceres. With these considerations in mind, the optimizer is able to find trajectories to SPICE ephemeris non-planetary bodies.

As with the previous example, a test to see how the optimizer behaves to fixed and variable time segments is also performed. In this test, the fixed-time and variable-time optimizations were continuously restarted by updating the initial guess of the next run with the final state of the previous solution. This process was repeated until the total low-thrust ΔV difference between runs was negligible. The fixed-time and variable-time solutions required 6 and 8 optimizations respectively to reach this point. The final results are presented in Figure 8 and is summarized in Table 2. It is evident from these results that the variable time optimization has a lower total ΔV and this is due to better placement of maneuvers along the trajectory (as discussed in the previous subsection). It also used nearly three times more thrust (0.3 N to 0.9 N) but has a final mass nearly 100 kg more than the fixed-time solution (2200 kg versus 2100 kg). Similar the results seen in the Earth-Venus example, the variable-time optimization has a lower total ΔV as seen in the table. The feasibility and first-order optimality from FMINCON indicate that both trajectories are reaching Ceres essentially perfectly (state errors are negligible), but there is likely the ability to further optimize the variable-time solution.

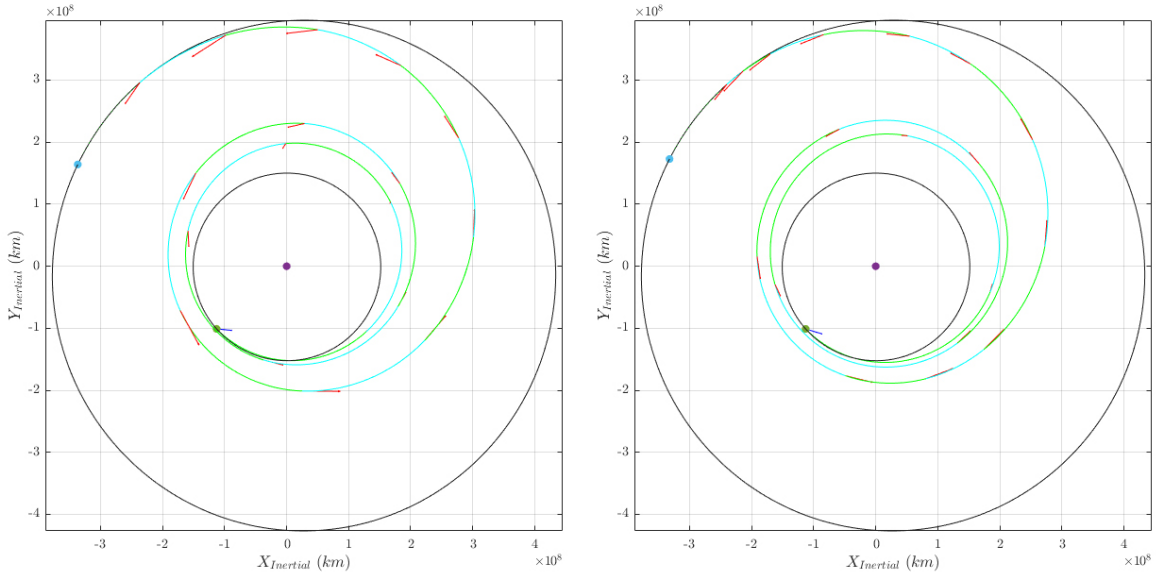

Figure 8: 19 segment transfer between Earth and Ceres. Left subfigure is fixed-time and right subfigure is variable-time segments.

Parameter	Units	Fixed-Time	Variable-Time
Launch C3	(km^2/s^2)	14.78315	13.82194
ΔV Total	(km/s)	10.59539	9.686545
TOF	$(days)$	1899.972	1894.048
Δx	(km)	-8.55337	-7.86636
Δy	(km)	-54.1665	-49.8108
Δz	(km)	-2.14264	-2.83858
ΔVx	(km/s)	1.99E-06	1.84E-06
ΔVy	(km/s)	-1.70E-06	-1.69E-06
ΔVz	(km/s)	-3.51E-07	-3.04E-07
Feasibility		1.63E-09	4.38E-13
Optimality		6.99E-07	3.57E-04

Table 2: Earth-Ceres transfer optimal solutions for fixed-time and variable-time segments.

4 Future Work

All the intended topics have have been addressed. Additional ones were explored and couldn't be completed due to time. Therefore, it is worth mentioning these parts that have been left as work-in-progress as of writing this paper.

4.1 Flybys and Multi-shooting

For interplanetary trajectories, gravity assists are key to reduce ΔV costs. The inclusion of flybys opens up a much broader application of this algorithm. The instantaneous flyby assumption can be used for a multi-part low-thrust transfer optimization. This can be accomplished by breaking up the total trajectory into forwards and backwards propagating nodes by using multiple shooting[15]. Figure 9 demonstrates a work-in-progress model of the multiple shooting initial guess generation. Lambert

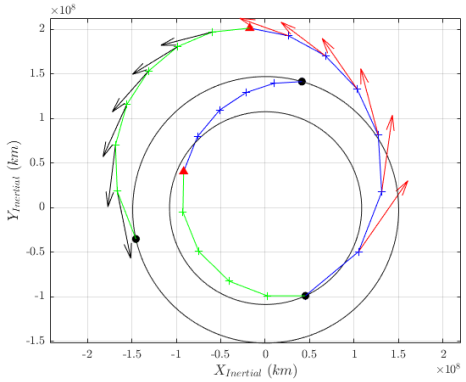

Figure 9: Initial guess setup for multiple-shooting and flyby implementation. Velocity unit vectors are only shown on the second leg (Venus-Earth).

arcs between bodies are fit using the guess TOF and trajectory is then broken into the forwards and backwards shooting segments. A match-point (shown as a red triangle) would need to be coincident to satisfy the non-linear equality constraint. The decision vector in this example will need to include a guess encounter time for each body so its state is known. Forwards and backwards arcs, shown in blue and green respectively, would be propagated from each body. Because the flyby velocity is known, the appropriate flyby assumptions can be used to find the required energy gain contribution by the body to patch the trajectory to the next low-thrust arc.

4.2 Penalty Method

It is worth mentioning that solution speed can be considerably increased by reformulating the problem's costs and constraints using the penalty method. It works by solving many sub-optimization problems, where each problem has increasing penalties applied to unmet constraints. These constraints are refactored to be directly in the cost function and so the original optimization becomes a series of many unconstrained problems[14]. Constraint terms are multiplied by an increasing coefficient as well as a scaling coefficient to tune the cost function performance.

$$\min \Phi_k(\vec{X}) = J(\vec{X}) + \sigma_k \sum_{i \in I} \max(0, c_i(\vec{X}))^2 \tag{12}$$

Eq. 12 describes the unconstrained minimization problem which uses the augmented cost function. J is the original cost function and the summation term is the contribution due to the constraints. The per iteration multiplying term is σ_k which increasingly penalizes the optimizer for violating the constraint. Because the constraint term is squared, the only way to incur no additional cost is to satisfy the constraint. The penalty method is currently utilized by MALTO[10], and this formulation of the problem can help make the optimizer more robust to initial guesses and reduce computation time.

5 Conclusions

In this paper, the Sims-Flanagan approximation for low-thrust trajectories is applied to find optimal interplanetary transfers. Its integration with MATLAB's optimization tools is discussed along with formulations of the costs, constraints, dynamics, and a method to find 'global' solutions. Various example cases demonstrate the functionality of this optimizer and how it compares to the existing state of the art Sims-Flanagan based program, MALTO. The optimizer is able to find solutions with mission design constraints to bodies of varying inclinations and states relative to each other. The computational speed is increased using parallel processing, and a significant speed advantage remains by disabling graphical outputs for each iteration. In all, the Sims-Flanagan method of approximating low-thrust has been successfully implemented to find interplanetary trajectories. Certainly, improvements to the project exist and will hopefully be completed in future versions of this project.

References

[1] J. Brownlee. *Basin Hopping Optimization in Python*. 2021.

[2] MathWorks. *Automatic Differentiation Background*. [Online; accessed 04-May-2022]. 2022.

[3] MathWorks. *Constrained Nonlinear Optimization Algorithms*. [Online; accessed 04-May-2022]. 2022.

[4] MathWorks. *First-Order Optimality Measure*. [Online; accessed 04-May-2022]. 2022.

[5] MathWorks. *Optimization Toolbox*. 2022.

[6] MathWorks. *Parallel Computing Toolbox*. 2022.

[7] SciPy. *Basin Hoping*. [Online; accessed 04-May-2022]. 2022.

[8] J. Sims and S. Flanagan. "Preliminary Design of Low-Thrust Interplanetary Missions". In: 1997.

[9] J. Sims and S. Flanagan. "Preliminary Design of Low-Thrust Interplanetary Missions". In: vol. 99-1373. Aug. 1999.

[10] J. Sims et al. "Implementation of a Low-Thrust Trajectory Optimization Algorithm for Preliminary Design". In: *Collection of Technical Papers - AIAA/AAS Astrodynamics Specialist Conference, 2006* 3 (Aug. 2006). DOI: 10.2514/6.2006-6746.

[11] Francesco Topputo and Chen Zhang. "Survey of Direct Transcription for Low-Thrust Space Trajectory Optimization with Applications". In: *Abstract and Applied Analysis* 2014 (June 2014), pp. 1–15. DOI: 10.1155/2014/851720.

[12] CK Venigalla. *Numerical Methods for (Spacecraft Trajectory) Optimization*. Accessed: 2020–04-25. 2022.

[13] G Whiffen. "Mystic : implementation of the Static Dynamic Optimal Control Algorithm for high-fidelity, low-thrust trajectory design". In: *Pasadena, CA : Jet Propulsion Laboratory, National Aeronautics and Space Administration, 2006* (Aug. 2006).

[14] Wikipedia. *Penalty method — Wikipedia, The Free Encyclopedia*. [Online; accessed 04-May-2022]. 2022.

[15] Chit Hong Yam, Dario Izzo, and Francesco Biscani. "Towards a High Fidelity Direct Transcription Method for Optimisation of Low-Thrust Trajectories". In: (2010). DOI: 10.48550/arxiv.1004.4539.