Spacecraft Trajectory Optimization

João F. L. Seabra
joao.seabra@ist.utl.pt

Instituto Superior Técnico, Lisboa, Portugal

July 2015

Abstract

The problem of optimizing spacecraft interplanetary trajectories is considered in this work; an implementation of a two-step method to solve the problem is presented. The trajectories are set in an inverse-square field with a low-thrust propulsion spacecraft, and gravity-assisted flybys are permitted. The intention is to solve global optimization problems, with large launch windows and complex objective functions, such as the problems from the GTOC competition; the need to find a global optimum in a large phase space led to the implementation of a direct transcription method, with initial guesses provided by a shape-based method using exponential sinusoids to approximate an optimal continuous thrust trajectory. The PyGMO library developed by ESA is used; it provides efficient algorithms for optimization. The program developed in this work is validated against a known solution of an Earth to Mars mass optimized trajectory. The program is also tested in multiple gravity assists transfers; finally, a solution to the first GTOC problem is presented. The method implemented has proven successful in exploring the solution phase space.

Keywords: Trajectory Optimization, Low-thrust Trajectories, Global Optimization, GTOC, Direct Transcription

1. Introduction

The problem of spacecraft trajectory optimization can be stated as the determination of a trajectory that satisfies mission-dependent path constraints while minimizing a given performance index [1]. The objective of this work is to implement techniques for global trajectory optimization, in the case of interplanetary transfers.

The first propulsion systems used were chemical rockets. This system can provide high thrust accelerations; however, chemical rockets are limited in terms of the impulse produced. Spacecrafts using this type of propulsion ignite the rockets for a very small period of time compared to the total time of flight; hence the propulsion can be approximately modelled as impulsive. The optimization of such trajectories is usually achieved by solving Lambert's problem [2]; it consists of finding the trajectory between initial and final positions and a given time of flight, with no thrust applied. In contrast with chemical rockets, electric propulsion systems, such as nuclear electric propulsion (NEP) and solar sail electric propulsion (SEP), can achieve higher specific impulse and hence are significantly more efficient. The drawback is that the force produced is very small, making it impossible to use this kind of propulsion in impulsive manoeuvres; hence it is common practice to refer such systems as low-thrust or continuous thrust propulsion, since the thrust is used along a significant fraction of the interplanetary phase of the trajectory. In this case complications arise in the design of the trajectories; the propulsion is now a continuous function over time and the optimization is more complicated than the impulsive thrust case [3], where the optimization could be transformed into a parameter adjustment over a fixed number of impulses. Currently, the use of low-thrust propulsion systems is decisive in the design of interplanetary flights; the high efficiency in terms of propellant makes it possible to reach regions beyond the chemical rockets capability.

1.1. *Global Trajectory Optimization Competition*

One motivation of this thesis was the Global Trajectory Optimization Competition – GTOC [4]. This event was created by Dario Izzo of the European Space Agency $\left(\text{ESA}\right)$, with the aim of generating a series of interesting problems regarding solutions to global trajectory optimization. The winner of each competition is the host of the next GTOC, presenting a new problem to be solved.

The problem is carefully defined so that it is global; the objective function tends to be unusual, the number of local minima high, and the basin of attraction of the global minimum is taken into account in the formulation of the problem so that

global search methods are needed to successfully achieve an optimum mission design. Also one of the purposes of the competition is to present original problems; consequently the experience of the competitor and the knowledge of standard trajectories solutions are intended to be of little value against the numerical methods implemented.

The nature of the various competition problems entails the following characteristics for a method to be successful:

- Automated global search of the state space
- Numerical robustness
- Algorithmic versatility

The automated search is obvious due to the global nature of the problem: the existence of many local minima requires a wide search for the optimum; the numerical robustness while evaluating trajectories is a key value so that the automated search is feasible; finally, it is convenient that the algorithm is versatile, because the attributes of the problems change with the GTOC edition, so the required changes on the algorithm must be made with ease.

2. Trajectory Optimization

The trajectory optimization problem is usually solved via numerical methods; there is no known analytical solution, except for some very specific problems. Currently the most used approaches are based in direct and indirect methods, both needing an initial guess for the trajectory. Stochastic methods, such as genetic algorithms and neural networks, have the ability of working without an initial guess; however, these methods suffer from not using gradient information of the problem [5]. The stochastic methods are mostly used jointly with direct or indirect methods, usually to optimally chose discrete parameters, such as flyby sequences, types of legs and other combinatorial schemes.

Some approaches provide initial guesses for the optimization process; this is needed to search the state space systematically. This is usually done by making some sort of approximation in calculating the trajectory and its corresponding fitness function.

2.1. *The Optimal Control Problem*

The optimal control problem can be stated as finding the optimal control history that satisfy the system dynamics, while satisfying the constraints. The dynamics of the system are defined by a set of first order differential equations,

$$\dot{\vec{X}} = f(\vec{X}, \vec{u}, t) \tag{1}$$

where $\vec{X}$ is the state vector and $\vec{u}$ is the control vector.

The initial conditions are generically given by

$$\psi_{0l} \leq \psi_0(\vec{X}(t_0), t_0) \leq \psi_{0u} \tag{2a}$$

$$\psi_{fl} \leq \psi_f(\vec{X}(t_f), t_f) \leq \psi_{fu} \tag{2b}$$

and the objective is to minimize the cost function [2]:

$$J = \Phi(\vec{X}(t_f), t_f) + \int_{t_0}^{t_f} L(\vec{X}, \vec{u}, t)\mathrm{d}t. \tag{3}$$

where t_0 and t_f are the initial and final times, Φ is a function of the final state vector and L is evaluated along the trajectory, function of both the instantaneous state and control vectors, and integrated over the entire trajectory. The control variables to be optimized

$$\vec{u} = \begin{bmatrix} u_1(t) \\ u_2(t) \\ \vdots \\ u_n(t) \end{bmatrix}, \tag{4}$$

must lie within the allowed physical bounds

$$\vec{u}_l \leq \vec{u} \leq \vec{u}_u, \tag{5}$$

and the state vector also has to satisfy its bounds

$$\vec{X}_l \leq \vec{X} \leq \vec{X}_u. \tag{6}$$

There may be equality or inequality constraints imposed on the trajectory; they can be event-type constraints, evaluated at a particular instant t_e,

$$C = f_1(\vec{X}(t_e), \vec{u}, t_e), \tag{7a}$$

where f_1 is some general function of the state $\vec{X}$ and control $\vec{u}$ at the instant t_e, and C is some specific value; over the trajectory,

$$C = f_1(\vec{X}(t_f), \vec{u}, t_f) + \int_{t_0}^{t_f} f_2(\vec{X}(t), \vec{u}, t)\mathrm{d}t, \tag{7b}$$

where f_2 is some general function of the same variables but now over the entire trajectory; or they can be a path constraint

$$C(\vec{X}, \vec{u}, t) \leq f_1(\vec{X}, \vec{u}, t), \tag{7c}$$

where the functions C and f_1 are specified over the trajectory.

Having defined the optimal control problem, a method must be applied to minimize the cost function J given by equation (3). The trajectory optimization methods that attempt to solve the problem fall into three categories [5]: direct, indirect, and stochastic methods.

The approach to minimize the cost function J is different in each method; in the direct methods the state and control variables are adjusted directly in order to reach a solution, while in the indirect methods the minimum is sought by solving analytically derived necessary conditions for optimality.

2.2. *Direct Methods*

To directly adjust the dynamic variables (state and control), the control variables have to be parametrized. Each direct method parametrizes the control variables differently, but all use non-linear programming (NLP) to achieve a minimum or maximum solution. Non-linear programming tries to achieve the minimum of the cost function by calculating increments in the vector of control parameters [2], while observing all constraints.

The most widely used numerical methods are single shooting, multiple shooting and collocation or transcription. The single shooting method is the simplest method of the three; the initial state vector values are assigned and the trajectory is propagated considering the parametrized control initial guess, then non-linear programming is applied to solve the constrained optimization problem. In multiple shooting, the trajectory is divided in time, and in each segment the single shooting method is applied; the initial values at each division are optimization parameters, and the continuity at the junction of segments is treated as an equality constraint. This is again solved by non-linear programming.

The transcription or collocation method parametrizes the dynamic variables at grid points. Hence the trajectory is discretized in time, and at each grid point the value and derivatives of the state variable are defined. At intermediate points of the grid, the state and its derivatives are determined by piecewise polynomials. The constraints are defined as the difference between the calculated and interpolated values at coincident times, that must be iteratively reduced to zero.

2.3. *Indirect Methods*

The indirect approach is based on the results derived from the Calculus of Variations and the Pontryagin's minimum principle; the result is a set of conditions for optimality that define a two-point boundary value problem. Multiple shooting methods are usually used to solve the indirect formulation [5].

2.4. *Other Methods*

Evolutionary algorithms are numerical optimizers that find an optimal set of discrete parameters that characterize the problem solution. One advantage of the evolutionary algorithms is that they do not require initial solutions – they are randomly generated within the problem bounds at the start of the optimization process; other advantage is that they are more likely attracted to a global optimum, contrary to the direct and indirect methods that converge to the local optimum dependant on the initial guess. The drawback is that there is no guarantee that a minimum has been found; also, there are no necessary conditions for optimality with this approach.

A great advantage of using genetic algorithms is that they are easy to implement[5], but they are best used together with a more accurate method – direct or indirect, so that the trajectory is feasible. the genetic algorithm has proven successful in generating initial guesses for direct methods[6].

2.5. *Method Selection*

The optimization method selected was the direct transcription method. The direct methods in general are numerically more robust than the indirect methods [5]; there is also less sensitivity in convergence when choosing similar initial guesses [3].

In terms of flexibility of the algorithm, the direct transcription method is particularly more easily changed when the problem structure changes [6]; indirect methods need a new analytical derivation of the necessary conditions. This is crucial for the choosing of the direct transcription method, because future problems to be solved from the GTOC are global and may require different physical models, new path and terminal constraints or new objective functions.

The main drawback of using a direct method is that by using an optimal control formulation that is simpler that the one derived from variational calculus, the solution obtained is an approximation to the real optimum solution [2].

2.6. *Generating Initial Guesses*

Generating a good initial guess is one of the most essential steps in finding a good optimal solution. It is important to note that, although there are no good or bad solutions but only true optimal ones, a good optimal solution is understood as a trajectory that satisfies all the constraints and produces a cost function J that is lower (or higher when considering maximization) than all the previously known trajectories for a specific problem. This is the case of global trajectory optimization; the huge dimension space of the trajectories makes it impossible to assess if a given trajectory produces or not a global minimum.

The high dimension of the problem – usually caused by large launch windows and numerous possible sequences of events – demands a wide search on the state space, so that the probability of finding the global optimum increases. The search is commonly performed by automating the generation of initial guesses.

The global search methods generally use some sort of approximation for the trajectories. The cause for the approximation is that, for wide searches to be feasible, there should be a rapid computation of the trajectories. Two common practices are to exhaustively search the approximate state space or use an evolutionary algorithm to perform

the stochastic search; it is noted that the evolutionary algorithm still needs an approximate manner to evaluate each trajectory, or computational times become infeasible [6].

From the two common approaches for generating an initial guess – globally search an approximate state space or use an evolutionary algorithm–, the first was implemented – the planar shape-based method. It performs an exhaustive search on the approximate state space. The shape used to generate a low-thrust trajectory is the exponential sinusoid of Petropoulos et al [7], that uses four constants to define its shape in the low-thrust plane, parametrized by the polar coordinate θ.

3. Implementation

The two-step approach implemented to solve the global trajectory optimization problem is depicted in the diagram of figure 1. The user first sets as input the sequence of flybys, the launch window and information related to the type of legs; from this input the shape-based method finds thousands of possible trajectories that intersect the flyby bodies; the trajectories found with the shape-based method are then transcribed in the third block *traj2input* to the format needed for the local optimization method.

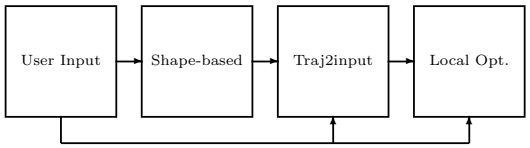

Figure 1: Global Optimization Method Diagram.

3.1. Shape-based Method

The first step of the optimization performs a broad search of low-thrust trajectories with gravity-assists, where a exponential sinusoid shape is used to represent the powered portion of flight. The program runs through a user-defined launch window and sequence of flyby bodies and exhaustively finds the trajectories that intersect them. The leg type (leg is the trajectory between two bodies), is also specified a priori: the legs can be either thrust, coast (no thrust) or mixed. In the case of mixed legs the switch radius must be defined; the switch radius defines the point from where the thrust is turned off for the rest of the leg.

The two-dimensional exponential sinusoid is given in polar coordinates (r, θ) by the following equation,

$$r = k_0 e^{k_1 \sin(k_2\theta+\phi)} \tag{8}$$

where k_0,k_1,k_2 and ϕ are constants that define the trajectory. The constant k_0 is a scaling factor and ϕ defines the orientation of the spiral in the low-thrust plane. The constant k_1 is called the dynamic range parameter and controls the ratio of the apoapsis to the periapsis; k_2 is the winding parameter and controls the number of revolutions around the central body. Figures 2, 3 and 4 present three exponential spirals; the effect of reducing the winding parameter – increasing the number of revolutions – can be seen in figure 3, and the effect of increasing the dynamic range parameter can be seen in figure 4.

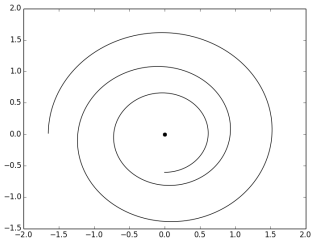

Figure 2: Exponential sinusoid spiral, with $k_0 = 1$,$k_1 = 1/2$,$k_2 = 2/11$,$\phi = -\pi/2$

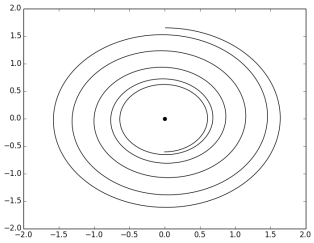

Figure 3: Exponential sinusoid spiral, with $k_2 = 1/11$

By considering the polar equations of motion given

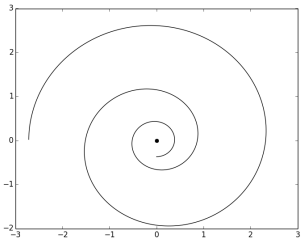

Figure 4: Exponential sinusoid spiral, with $k_1 = 1$

by

$$\ddot{r} - r\dot{\theta}^2 + \frac{\mu}{r^2} = \tau \sin\alpha \tag{9a}$$

$$\frac{1}{r}\frac{\mathrm{d}}{\mathrm{d}t}(r^2\dot{\theta}) = \tau\cos\alpha, \tag{9b}$$

and assuming a tangential thrust, both the angular rate $\dot{\theta}$ and the normalized thrust acceleration a are determined for every point on the shape [7]:

$$\dot{\theta}^2 = \frac{\mu/r^3}{\tan^2\gamma + k_1k_2^2s + 1} \tag{10}$$

$$a = \frac{(-1)^n\tan\gamma}{2\cos\gamma}\left[\frac{1}{\tan^2\gamma + k_1k_2^2s + 1} - \right. \\ \left. - \frac{k_2^2(1-2k_1s)}{(\tan^2\gamma + k_1k_2^2s + 1)^2}\right] \tag{11}$$

where the normalized thrust acceleration is

$$a \equiv \tau/(\mu/r^2), \tag{12}$$

$$s \equiv \sin(k_2\theta + \phi) \tag{13}$$

and the flight path angle is given by

$$\tan\gamma = k_1k_2\cos(k_2\theta + \phi) \tag{14}$$

The thrust angle is given by

$$\alpha = \gamma + n\pi, \quad n = 0, 1 \tag{15}$$

where n is chosen so that the thrust acceleration (11) is positive.

The program steps through the launch window defined by the user, and for each departure time t_0 the trajectories that reach the next body in the sequence are computed. The hyperbolic velocity v_∞, in case of a launch, is defined by the user and may point in any direction; in the case of a flyby the hyperbolic velocity turn angle is constrained by altitude. The program steps through the range of possible turn angles for the outgoing $\vec{v}_\infty$. For each initial relative velocity there corresponds a heliocentric flight-path angle and speed [7], so both $\tan\gamma$ and $\dot{\theta}$ are known, thus the quantity $k_1k_2^2s$ can be obtained from equation (10), and together with the trigonometric identity $\sin^2\theta + \cos^2\theta \equiv 1$ and the flight-path angle equation (14), results the constraint relationship [7]

$$k_1^2k_2^4 - k_2^2\tan^2\gamma - k_1k_2^2s = 0. \tag{16}$$

This equation relates the undetermined shape parameters k_1 and k_2; the free parameter to search for intersections is taken as k_2.

For a departure time t_0, the intersections of the spacecraft with the planet are searched in function of the constant k_2. The other three constants – k_0, k_1, ϕ – are determined so that the spiral starts at the position of the departure planet $\rho(\theta_0) = r_{sc}(\theta_0)$, and with the same angular rate as the determined from the sum of the departure hyperbolic velocity with the planet velocity $\vec{v}_{sc} = \vec{v}_p + \vec{v}_\infty$, and also so that the constraint given by equation (16) is satisfied.

Calculating Intersections

For each turn angle, the k_2 values that yield intersections of the shape with the target body are found; having determined the point where the intersection occurs, the specific value of k_2 that results in a correct time of flight must be calculated.

The intersection of the exponential sinusoid shape r with the conic shape orbit of a body ρ is found through a Newton's method, solving for the roots of the following function

$$d(\theta) = 1/\rho(\theta) - 1/r(\theta); \tag{17}$$

the difference of the inverse of the radii is used to speed up the computations [7].

After computing the intersection point, the transfer time for the spacecraft to reach it must be calculated. For a spacecraft pursuing a exponential sinusoid, the time of flight has to be numerically calculated. The time of transfer tof is obtained by numerical integration, from the $\dot{\theta}$ equation (10):

$$tof = \int_{\theta_i}^{\theta_f}\dot{\theta}^{-1}\mathrm{d}\theta = \int_{\theta_i}^{\theta_f}\sqrt{\frac{\tan^2\gamma + k_1k_2^2s + 1}{\mu/r^3}}\mathrm{d}\theta. \tag{18}$$

Now the planet position at the transfer time may be calculated by solving the Kepler's equation. Knowing both the spacecraft and planet position, the miss angle – heliocentric angle between the spacecraft and the planet – can be calculated, and a search is made over k_2 to find the zero miss angle, that is, the k_2 that produces a flyby is searched.

The program then finds sequences of intersections. When a trajectory that reaches the next body is found, then the gravity-assist equations are used to provide the range of turn angles for the $\vec{v_\infty}$ after the flyby. When a trajectory finds its way through the sequence up to the arrival planet, that trajectory is stored; each solution is then evaluated by the amount of propellant used. As the shape is completely specified by the four constants k_0, k_1, k_2 and ϕ, there are no great memory restrictions when allocating many solutions. For example, 616 trajectories found for a Earth-Mars-Ceres mission with two thrust arcs take up to approximately $1.37MB$ in disk space.

Next is presented an example of the input format, representing a multiple gravity-assist trajectory from Earth to Uranus, with intermediate flybys at Jupiter and Saturn:

Input format example_mga.txt

```
#mission
sequence:ejsu
leg:ttt
v_inf:5.0e3

#data
window:3000,4000,1
angle_step:0.2
```

The legs are all thrust, the departure hyperbolic velocity magnitude is $5.0e3\ m/s$; the trajectory obtained with maximum mass at arrival is depicted in figure 5.

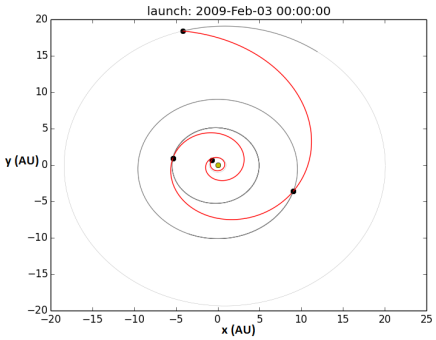

Figure 5: Earth-Jupiter-Saturn-Uranus Shape-based trajectory with maximum mass at arrival

Each trajectory leg is then divided into N segments, and the ΔV of each segment is obtained integrating the thrust profile. This will serve as the input of the direct transcription method.

3.2. Local Optimization

The local optimizer consists of a direct transcription method, created by Sims and Flanagan [3]. The optimal control problem of low-thrust multiple gravity-assists trajectories is transcribed to a non-linear programming problem.

The transcription is made by dividing the trajectory into legs that begin and end at control nodes, and each node is usually associated with a celestial body. On each leg there is a match point; the trajectory is propagated forward from the initial control node to the match point and backward from the final node to the match point.

The low-thrust is modelled as a series of impulses; the $\Delta\vec{V}$ impulses are applied along each of the N segments that divide each leg. The resulting trajectory structure is represented in figure 6.

Both the forward and backward propagation between control nodes and impulses is made according to a two body model, that is, the trajectories are keplerian, and the gravity assists are modelled as an instantaneous turn of the relative velocity $\vec{v}_\infty$.

This formulation leads to a constrained non-

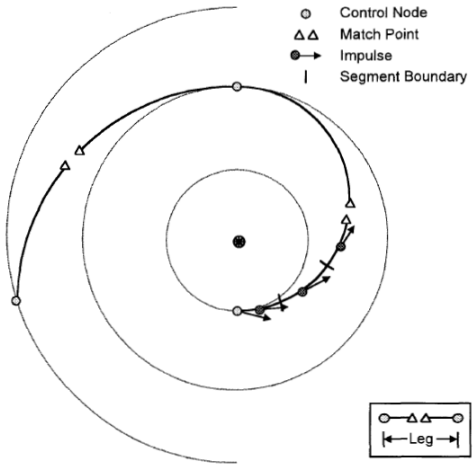

Figure 6: Structure of the transcribed trajectory

linear optimization problem [3]. The variables of optimization are the state $(\vec{r}, \vec{v}, m)$ and corresponding epoch at each control node, the $\Delta\vec{V}$ impulses, and the flyby turn angle at each intermediate flyby. The state at the control node is typically a dependent variable because the position is usually set to be equal to the flyby planet, unless the control node is set to be a fixed point in space, in which case the variable is independent.

The equality constraint of the optimization is the continuity of the spacecraft state on the match point. To ensure that the $\Delta\vec{V}_i$ on each segment is feasible, an inequality constraint is placed on every segment for the magnitude of the impulse $|\Delta\vec{V}_i|$. Any objective function or other constraints can be placed on the trajectory, depending on the problem, provided that they can be expressed by the variables of the trajectory.

3.2.1 Local Optimization Input

To optimize the trajectories with the transcription model, the solutions obtained from the global search must be transformed to the format needed: the legs between flyby bodies have to be segmented in time and the ΔV accumulated over the segment has to be calculated to serve as an input for the optimization.

For each leg, the time of flight tof has already been calculated in the shape base method; so the duration of each segment is $t_{seg} = tof/N$, where N is the number of segments defined by the user for each leg. As the trajectories obtained are all in function of the polar coordinate θ, the various θ_i, corresponding to each segment have to be calculated by quadrature. The ΔV for each segment can be obtained by numerically integrating the specific thrust:

$$|\Delta\vec{V}_i| = \int_{t_i}^{t_{i+1}} \tau \mathrm{d}t = \int_{\theta_i}^{\theta_{i+1}} \tau(\theta)\dot{\theta}^{-1}\mathrm{d}\theta. \qquad (19)$$

The mass on the end node is obtained by

$$m_f = m_i \exp(\Delta V_{leg}/(I_{sp} g_0)), \tag{20}$$

where ΔV_{leg} is the change in velocity accumulated over the entire leg.

In the direct transcription method used, instead of the $\Delta\vec{V}_i$, the engine throttles are used equivalently as the variable for optimization. The throttles, defined as $\vec{\eta} = \vec{T}/T_{max}$, are easily obtained from the ΔV for each segment i of the leg:

$$\vec{\eta}_i = \Delta\vec{V}_i \frac{m_i}{T_{max}} \frac{N}{tof}. \tag{21}$$

Now, to constrain the maximum allowed change in velocity over a segment, it is only necessary to satisfy the N constraints for each leg:

$$|\vec{\eta}_i| \leq 1. \tag{22}$$

3.3. *PyGMO Library*

PyGMO is an open source project intended to be used together to script parallel optimization tasks for use in astrodynamics [8]. The library is written in the C++ language and is exposed to Python, so that introducing new problems and algorithms is done with ease.

PyGMO employs a paradigm called the generalized island model, for the parallelization of optimization algorithms. The solutions for a given problem are represented as individuals; a group of individuals is called a population, and a population over which an algorithm acts to improve the solutions is called an island. Finally, various islands grouped together in an archipelago can be formed in a defined topology and share their solutions.

The user defines the problems by coding a python class (or C++ class wrapped for python) where the virtual methods associated with computing the objective function J and the constraints c need to be provided.

The algorithm that acts on the island can be coded by the user; but it was preferred to use the existing implemented non-linear algorithms in the PyGMO library, already known to be efficient.

The island where the individual lives is then initialized by assigning to it the problem and the algorithm. The initialization of the island automatically generates a random population sized by the user; in my case a population of one individual was used for the optimization. The individual decision vector vector obtained by the global shape-based method is then attributed to the individual, and the algorithm is performed on the island.

4. Results

4.1. Validation

To validate the program developed, the results were compared with a known near-optimal trajectory from Earth to Mars. The trajectory is given by Wall and Conway [9]; the mission characteristics are given in table 1. Although the mission consists of a simple transfer leg, the test is suitable for the global search method implemented, since the launch window has a considerable span of 30 years; thus the shape-based method must be decisive in determining an initial guess around the optimal launch date.

Mission Characteristics	
Leg	Earth-Mars
$\lvert\vec{v_\infty}\rvert$ at departure	$0.8935 \times 10^3 m/s$
Launch Window	[2000-Jan-01, 2029-Dec-31]
Initial Mass m_i	18000 kg
Engine I_{sp}	4110 s
Maximum Thrust T_{max}	2.775 N

Table 1: Earth-Mars Mission characteristics for validation.

The best trajectory found by Wall and Conway is described in table 2.

Reference optimal solution	
Launch date	2001-Jan-19
Time of flight	600 days
Arrival date	2001-Aug-23
Mass at arrival	14696 kg

Table 2: Optimal trajectory for validation.

Running through the launch window with a step of 1 Julian Day, the shape-based search with the input given in table 3 found $76,256$ trajectories; the best spiral trajectories are described in table 4; it can be already seen before applying the local optimization that the best trajectories are near the optimal one provided by Conway et al, both in departure date and final mass.

Transfer Characteristics	
Leg	Earth-Mars
Leg Type	Thrust
$\lvert\vec{v_\infty}\rvert$ at departure	8.935×10^2 m/s
Launch Window	[2000-Jan-01, 2000-Dec-31]
Launch Window Step	1 Julian Days

Table 3: Validation global shape-based search input.

The direct transcription method was applied on the best initial guesses, producing third dimensional locally-optimal trajectories. The description of each changed trajectory is presented in table 5, and the best found trajectory is depicted in figure 8.

Best shape-based trajectories		
Trajectory	Final Mass $[kg]$	Launch Date
1	14814.5	2001-Jan-25
2	14813.9	2001-Jan-25
3	14804.8	2001-Jan-19
4	14800.4	2001-Jan-26
5	14786.8	2000-Jan-26
6	14780.2	2001-Nov-08
7	14775.0	2001-Jan-19
8	14774.7	2001-Jan-18

Table 4: Best shape-based trajectories for validation.

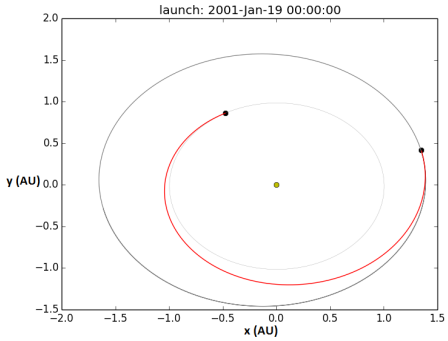

Figure 7: Earth-Mars shape-based trajectory with higher final mass.

Best locally optimized trajectories		
Trajectory id	Final Mass $[kg]$	Launch Date
7	14710.0	2001-Jan-18
6	14708.2	2001-Nov-08
8	14708.0	2001-Jan-18
4	14672.0	2001-Jan-26
3	14643.2	2001-Jan-19
2	14641.5	2001-Jan-25
5	14623.2	2000-Jan-26
1	13844.2	2001-Jan-14

Table 5: Best locally optimized trajectories for validation - Earth-Mars transfer.

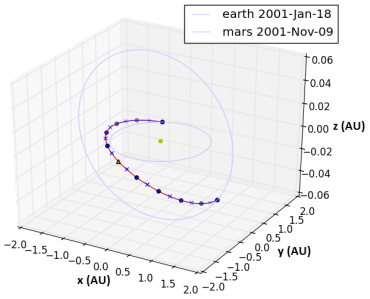

Figure 8: Earth-Mars locally optimized trajectory for validation.

4.2. First GTOC Solution

The first GTOC problem objective is to impact an asteroid and maximize the resulting change of the asteroid's orbit major axis. The following expression is to be maximized:

$$m_{sc}(\vec{v}_{sc/rel} \cdot \vec{v}_{ast})|_{t=t_f}, \tag{23}$$

which induces a maximum change in the target asteroid's major axis of the orbit after collision. $\vec{v}_{sc/rel}$ is the spacecraft's velocity relative to the asteroid and $\vec{v}_{ast}$ is the asteroid heliocentric velocity, m_{sc} the spacecraft's mass, and t_f is the moment of impact.

The mission characteristics are specified in table 6.

GTOC1	
Initial Mass m_i	$2000kg$
Engine I_{sp}	$2500\ s$
Maximum Thrust T_{max}	$0.04\ N$
Maximum $\lvert\vec{v_\infty}\rvert$ at departure	$2.5 \times 10^3\ m/s$
Launch Window	$2010 - Jan - 01$ to $2030 - Jan - 01$
Maximum time-of-flight	30 years
Minimum heliocentric distance	$0.2AU$

Table 6: GTOC1 problem characteristics.

The target asteroid is a NEA (Near Earth Asteroid) designated by $TW229$ and has the following characteristics:

Asteroid TW229 Characteristics	
semi-major axis	$2.5897261AU$
eccentricity	0.2734625
inclination	6.40734
argument of periapsis	264.78691
right ascension	128.34711
mean anomaly at 53600 MJD	320.47955

Table 7: NEA TW229 keplerian elements.

The inclusion of this objective function in the program was easy, since all the information in (23) is readily available: the spacecraft mass at arrival is part of the decision vector, as well as the velocity at the arrival. The velocity of the asteroid is known as a function of time, and the time is also available in the decision vector, summing the initial epoch with the time of flight to collision.

The decision vector used for the GTOC problem is given by

$$\begin{aligned} D \equiv [&t_0, \\ &tof_1, m_{f1}, \vec{v}_\infty|_{dep}, \vec{v}_\infty|_{flyby_{0i}}, \\ &tof_2, m_{f2}, \vec{v}_\infty|_{flyby_{0f}}, \vec{v}_\infty|_{flyby_{1i}}, \\ &\vdots \\ &tof_n, m_{fn}, \vec{v}_\infty|_{flyby_{nf}}, \vec{v}_\infty|_{arr}, \\ &\vec{\eta}_{leg_1}, \vec{\eta}_{leg_2}, \ldots, \vec{\eta}_{leg_n}] \end{aligned} \tag{24}$$

and represents a generic multiple gravity-assist trajectory with n legs; each leg is represented by its time-of-flight, mass at the end point, velocities at the start and end of the leg and the engine throttles.

To satisfy the mission requirement that the spacecraft's distance to the Sun would not be less that 0.2 AU, an inequality constraint was added to the problem:

$$c_{ineq} = 0.2AU - \sqrt{x(t)^2 + y(t)^2 + z(t)^2} < 0. \quad (25)$$

Earth-NEA

To first solve the problem, a direct transfer from Earth to the asteroid was considered. The shape-based method was performed stepping through the launch window with a step of one julian day. The initial-guesses were sorted by the objective function and served as an input to the local optimizer, with parameters described in table 10.

Optimization Parameters	
Number of Segments N_{seg}	35
Initial Mass m_i	2000 kg
Engine I_{sp}	2500 s
Maximum Thrust T_{max}	0.04 N

Table 8: GTOC1 Earth-NEA optimization parameters.

The best trajectory found is depicted in figure 9, with its characteristics in table 9.

Earth-Asteroid Best Solution	
Departure Date	$2021-10-15$
Arrival Date	$2038-12-29$
Objective Function J	9.0978×10^{10}

Table 9: GTOC1 Earth-NEA solution.

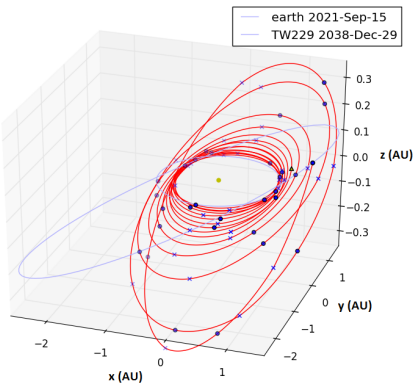

Figure 9: Earth-NEA trajectory.

Earth-Earth-NEA

From the sequence of flybys considered (Earth-Venus-NEA, Earth-Mars-NEA, Earth-Earth-NEA), the best solution found was obtained with a Earth-Earth-NEA trajectory. The shape-based method was again performed with a step of one julian day; the parameters for the local optimizer are shown in table 10.

Optimization Parameters	
Number of Segments N_{seg}	5 - 5
Initial Mass m_i	2000 kg
Engine I_{sp}	2500 s
Maximum Thrust T_{max}	0.04 N

Table 10: GTOC1 Earth-Earth-NEA optimization parameters.

The best solution is depicted in figure 10 and described in table 11.

Earth-Earth-Asteroid Best Solution	
Departure Date	$2021-01-28$
Earth Flyby	$2029-01-18$
Arrival Date	$2030-02-21$
Objective Function J	12.6291×10^{10}

Table 11: GTOC1 Earth-Earth-NEA solution.

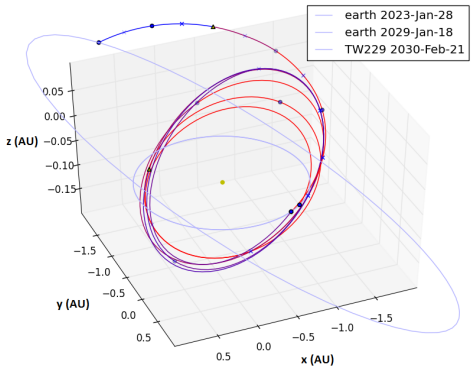

Figure 10: Earth-Earth-NEA trajectory.

Remarks

The best score from the competitor teams was of $J = 185 \times 10^{10}$ and the worse of $J = 0.89 \times 10^{10}$. The winner team was NASA; near my score were teams such as DLR (Germany Aerospace Centre) and Alcatel Alenia Space. The huge leap in the objective function score mostly comes from long flyby sequences used by the best teams (up to 15 flyby planets); not only the sequences are difficult to

construct (too many options), but they are hugely costly in computation time.

The main achievement from attempting to solve the first GTOC problem is identifying the lack of tools available; although the two-step method implemented is a fundamental tool on the design process of low-thrust trajectories, much more is needed to tackle the GTOC problems. The main features that would need to be added are mainly heuristic tools to assess at a first instance what combinations of events are worth exploring in the shape-based method.

5. Conclusions

The problem of spacecraft trajectory optimization considered – that of interplanetary transfers using low-thrust propulsion – was tackled with a two-step approach that globally searches the solution space. The optimization procedure satisfies the intended characteristics: it is robust, versatile and global. The robustness is assured through the direct transcription method; the direct method for local optimization assures that for most initial guesses – even if randomly generated – a feasible solution is obtained. The direct transcription also allows that changes in the formulation of the problem, such as new complex performance indexes, are easily introduced. The global aspect of the two-step approach is obtained by the shape-base method, that exhaustively searches the approximate solution space, generating initial guesses to the local optimizer.

Results show that the implemented method works and has a reasonable performance; the PyGMO library has proven to be an adequate tool for global optimization. The main limitation found was the computational means; increasing the number of flybys in a sequence squared the already costly time of computation. The need for heuristics that can guide the search for a good solution was identified; this would save immensely the time spent in the global search.

Future work on this subject should emphasize on automatically selecting the most promising sequences of flybys; the sequence of flyby bodies is usually selected based on the astrodynamics experience of the mission designer. Due to the discrete nature of the problem, a stochastic algorithm should be used in the selection of the sequence; the problem would consist of finding a suitable fitness function to evaluate each sequence and an efficient way to construct them. Then the multi-processing architecture of the PyGMO library could be used to parallelize the optimization algorithms.

References

[1] Bruce Conway, editor. *Spacecraft Trajectory Optimization*. Cambridge University Press, 2010.

[2] Stephen Kemble. *Interplanetary Missions Analysis and Design*. Praxis Publishing Ltd, 1^{st} edition, 2006.

[3] J. A. Sims and Flanagan S. N. Preliminary Design of Low-Thrust Interplanetary Missions. In *AAS/AIAA Astrodynamics Specialist Conference*, volume AAS 99-338, Girdwood, Alaska, August 1999.

[4] Dario Izzo. 1st ACT Global Trajectory Optimisation Competition : Problem Description and Summary of the Results. *Acta Astronautica*, 61(9):731–734, 2007.

[5] J. T. Betts. Survey of Numerical Methods for Trajectory Optimization. *Journal of Guidance, Control, and Dynamics*, 21(2):193–207, March–April 1998.

[6] Bradley J. Wall and Bruce A. Conway. Genetic Algorithms Applied to the Solution of Hybrid Optimal Control Problems in Astrodynamics. *Journal of Global Optimisation*, 44(4):493–508, August 2009.

[7] Anastassios E. Petropoulos and James M. Longuski. Shape-Based Algorithm for Automated Design. *Journal of Spacecraft and Rockets*, 41(5):787–796, September–October 2004.

[8] Dario Izzo. PyGMO and PyKEP: Open Source Tools for Massively Parallel Optimization in Astrodynamics. In *Proceedings of the International Conference on Astrodynamics Tools and Techniques*, Noordwijk, Netherlands, 29 May – 1 June 2012.

[9] Bradley Wall and Bruce A. Conway. Near-Optimal Low-Thrust Earth Mars Trajectories via a Genetic Algorithm. *Journal of Guidance, Control, and Dynamics*, 28(5):1027–1031, September–October 2005.