SGN – Assignment #1

Daniele Paternoster, 10836125

1 Periodic orbit

Exercise 1

Consider the 3D Earth–Moon Circular Restricted Three-Body Problem with $\mu = 0.012150$. Note that the CRTBP has an integral of motion, that is, the Jacobi constant

$$J(x, y, z, v_x, x_y, v_z) := 2\Omega(x, y, z) - v^2 = C$$

where $\Omega(x, y, z) = \frac{1}{2}(x^2 + y^2) + \frac{1-\mu}{r_1} + \frac{\mu}{r_2} + \frac{1}{2}\mu(1-\mu)$ and $v^2 = v_x^2 + v_y^2 + v_z^2$.

1) Find the coordinates of the five Lagrange points L_i in the rotating, adimensional reference frame with at least 10-digit accuracy and report their Jacobi constant C_i.

Solutions to the 3D CRTBP satisfy the symmetry

$$\mathcal{S} : (x, y, z, \dot{x}, \dot{y}, \dot{z}, t) \to (x, -y, z, -\dot{x}, \dot{y}, -\dot{z}, -t).$$

Thus, a trajectory that crosses perpendicularly the $y = 0$ plane twice is a periodic orbit.

2) Given the initial guess $\mathbf{x}_0 = (x_0, y_0, z_0, v_{x0}, v_{y0}, v_{z0})$, with

$$\begin{aligned} x_0 &= 1.068792441776 \\ y_0 &= 0 \\ z_0 &= 0.071093328515 \\ v_{x0} &= 0 \\ v_{y0} &= 0.319422926485 \\ v_{z0} &= 0 \end{aligned}$$

Find the periodic halo orbit having a Jacobi Constant $C = 3.09$; that is, develop the theoretical framework and implement a differential correction scheme that uses the STM, either approximated through finite differences **or** achieved by integrating the variational equation.
Hint: Consider working on $\varphi(\mathbf{x} + \Delta\mathbf{x}, t + \Delta t)$ and $J(\mathbf{x} + \Delta\mathbf{x})$ and then enforce perpendicular cross of $y = 0$ and Jacobi energy.

The periodic orbits in the CRTBP exist in families. These can be computed by 'continuing' the orbits along one coordinate or one parameter, e.g., the Jacobi energy C. The *numerical continuation* is an iterative process in which the desired variable is *gradually* varied, while the rest of the initial guess is taken from the solution of the previous iteration, thus aiding the convergence process.

3) By gradually decreasing C and using numerical continuation, compute the families of halo orbits until $C = 3.04$.

(8 points)

1.1 Libration points of the Earth-Moon CRTBP

The Lagrange points are the equilibrium points of the CRTBP. They can be found by imposing the stationarity of the potential function $\Omega(x, y, z)$ expressed in the rotating frame. Moreover, they all lie on the z-plane and can be divided into two main categories:

- **Collinear points**: three points (L1, L2, L3) aligned on the x-axis of the rotating frame
- **Triangular points**: two points (L4, L5) symmetric with respect to the x axis of the rotating frame

The triangular points can be found by noting, that imposing both $z = 0$ and $r_1 = r_2 = 1$ in the gradient of $\Omega(x, y, z)$, it yields $\nabla\Omega(x, y, z) = \mathbf{0}$. This means that the triangular points are found by intersection of two circles of unitary radius centered in the position of the two primaries. Since the a-dimensional formulation of the CRTBP requires that the two primaries be at unitary distance, the two triangular Lagrange points L_4 and L_5 are on the vertices of two equilateral triangles with unitary side. Hence:

$$x_{L_4} = x_{L_5} = \frac{x_E + x_M}{2} = \frac{-\mu + (1-\mu)}{2} = \frac{1-2\mu}{2} \tag{1}$$

$$y_{L_4} = +\frac{\sqrt{3}}{2} \qquad y_{L_5} = -\frac{\sqrt{3}}{2} \tag{2}$$

Therefore, the coordinates of both triangular points can be expressed analytically without the need to implement a zero-finding problem. On the other hand, the coordinates of the three collinear points are found by searching the zeros of the derivative of the potential along the x-axis:

$$\frac{\partial U}{\partial x}(x, 0, 0) = x - (1-\mu)\frac{(x+\mu)}{|x+\mu|^3} - \mu\frac{(x+\mu-1)}{|x+\mu-1|^3} \tag{3}$$

The zero-finding problem for 3 cannot be solved analytically, hence a numerical implementation has been set up. From the theory it is known that there are three zeros for the function 3 (corresponding to the abscissa of L1, L2 and L3). The existence of these points can be appreciated graphically in figure Figure 1, in which there are exactly three X-axis crossings. The location of the two primaries (Earth and Moon) represent the position of the two asymptotes of the function. The graph allows to have an initial guess on the location of the collinear libration points, needed for the implementation of the numerical solver. The zero finding problem for the function 3 has been solved in *Matlab®* through the use of the `fzero` built-in function. It allows finding a zero for a single input scalar function, given a certain interval or initial guess and a relative tolerance `tolX` on the step. The algorithms used are *bisection* and *interpolation techniques*. Since this routine can find a single solution at each call, the execution of the process has been repeated three times with the same solver options but different initial intervals for each solution. The intervals have been selected graphically from Figure 1.

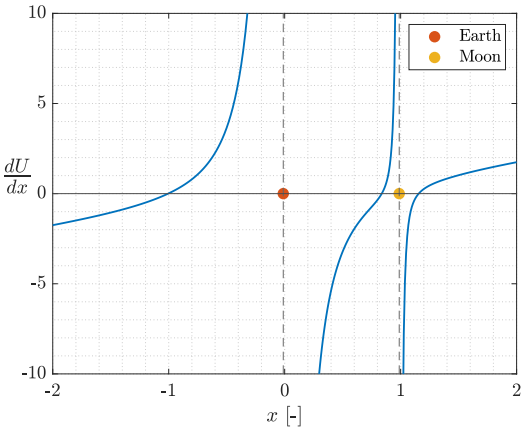

Figure 1: Gradient along x

$$L1 : [0.6 \quad (1-\mu) - 0.05] \qquad L2 : [(1-\mu) + 0.05 \quad 1.5] \qquad L3 : [-2 \quad -\mu - 0.4]$$

The options for the `fzero` solver can be modified by using `optimset`, in this case the default (minimum) value `tolX = 2.2204e-16` has been used, which is the machine-epsilon for double precision floating point numbers. This relative tolerance allows to achieve around 15-digit accuracy, which clearly satisfies the requirement of 10-digit accuracy. The Jacobi constant for each point can be retrieved by applying its definition. The libration points are represented graphically in the adimensional and rotating frame of the CRTBP in Figure 2

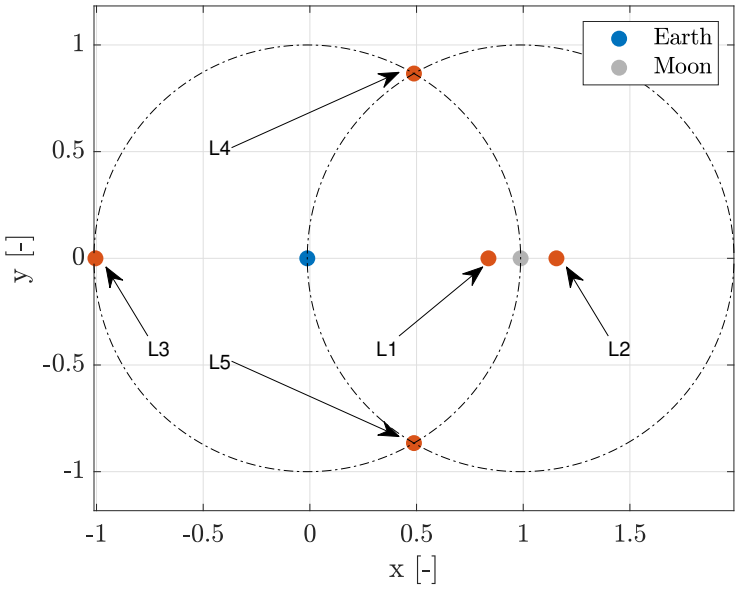

Figure 2: Libration points of the Earth-Moon CRTBP

1.1.1 Numerical results: Libration points coordinates

	L_1	L_2	L_3	L_4	L_5
x	0.8369180073	1.1556799131	−1.0050624018	0.4878500000	0.4878500000
y	0.0000000000	0.0000000000	0.0000000000	+0.8660254038	−0.8660254038
C	3.2003380950	3.1841582164	3.0241489429	3.0000000000	3.0000000000

Table 1: Lagrangian points coordinates and Jacobi constants

1.2 Periodic Halo orbit

The aim of this section is to present the theoretical framework that has been implemented in *Matlab* in order to find a Halo orbit in the Earth-Moon CRTBP, given the Jacobi constant C of the orbit and initial guess $\mathbf{x_0}$ to start the propagation.

Problem Statement:
Given the dynamics $\mathbf{f}(\mathbf{x})$ of the CRTBP for the Earth-Moon system,
find the Halo orbit corresponding to a given Jacobi constant C, provided a sufficiently close initial condition (guess IC) $\mathbf{x_0}$. Where the initial guess is in the form:

$$\mathbf{x_0} = (x_0, 0, z_0, 0, v_{y0}, 0)$$

The request of finding the periodic Halo orbit can be narrowed down to the request of correcting the initial guess through an appropriate scheme, in order to obtain a periodic orbit. In particu-

lar, by exploiting the symmetry property, the periodic orbit is found when the flow $\boldsymbol{\phi}(\mathbf{x_0}, t_0; t)$ intersects twice the (x, z) plane with perpendicular velocity vector.

1.2.1 Differential scheme

The development of the differential scheme can be started by expanding in series the flow of the CRTBP with respect to the intial condition $\mathbf{x_0}$ and the final time of propagation t_f:

$$\begin{aligned}\boldsymbol{\phi}(\mathbf{x_0} + \delta\mathbf{x_0}, t_0; t_f + \delta t) &= \boldsymbol{\phi}(\mathbf{x_0}, t_0; t_f) + \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{x_0}}\delta\mathbf{x_0} + \frac{\partial \boldsymbol{\phi}}{\partial t_f}\delta t_f \\ &= \boldsymbol{\phi}(\mathbf{x_0}, t_0; t_f) + \Phi(\mathbf{x_0}, t_f)\delta\mathbf{x_0} + \mathbf{f}(\mathbf{x_f}, \mathbf{t_f})\delta t_f\end{aligned} \tag{4}$$

Equation 4 describes how the state $\mathbf{x} = (\mathbf{r}, \mathbf{v})$ at time t_f changes if the initial condition is perturbed by δx_0 and the time of propagation itself is perturbed with magnitude δt_f. These two variations are retrieved respectively through the state transition matrix $\Phi(\mathbf{x_0}, t_f)$ (sensitivity of the flow $\boldsymbol{\phi}$ with respect to the initial condition $\mathbf{x_0}$) and the right hand side of the CRTBP dynamics evaluated at final state $\mathbf{f}(\mathbf{x_f}, \mathbf{t_f})$. Equation 4 can be re-arranged by bringing the first term of the right side to the left:

$$\begin{aligned}\boldsymbol{\phi}(\mathbf{x_0} + \delta\mathbf{x_0}, t_0; t_f + \delta t) - \boldsymbol{\phi}(\mathbf{x_0}, t_0; t_f) &= \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{x_0}}\delta\mathbf{x_0} + \frac{\partial \boldsymbol{\phi}}{\partial t_f}\delta t_f \\ \delta\mathbf{x} &= \frac{\partial \boldsymbol{\phi}}{\partial \mathbf{x_0}}\delta\mathbf{x_0} + \frac{\partial \boldsymbol{\phi}}{\partial t_f}\delta t_f\end{aligned} \tag{5}$$

One important property of the solutions of the CRTBP is exploited here: when the orbit intersect two times the (x, z) plane with velocity perpendicular to the same plane, then it is a periodic (Halo) orbit. This fact is used to further develop the single terms in Equation 5:

$$\delta\mathbf{x_0} = \begin{bmatrix} \delta x_0 \\ 0 \\ \delta z_0 \\ 0 \\ \delta v_{y0} \\ 0 \end{bmatrix} \tag{6}$$

With $\delta\mathbf{x_0}$ in the form of Equation 6, only variations on the (x, z) plane position and in the v_{y0} component of the velocity are allowed in order to exploit its property of already being an intersection point with (x, z) plane and having velocity perpendicular to it. Moreover, the desired final state $\mathbf{x_d}$ is the ideal final state that the process must aim to target in order to obtain the correct initial condition $\mathbf{x_0}$ out of the process. Basically, it is the second point of the (x, z) plane intersection which has perpendicular velocity to it:

$$\mathbf{x_d} = \begin{bmatrix} x(\mathbf{x_0} + \delta\mathbf{x_0}, t_f + \delta t_f) \\ 0 \\ z(\mathbf{x_0} + \delta\mathbf{x_0}, t_f + \delta t_f) \\ 0 \\ v_y(\mathbf{x_0} + \delta\mathbf{x_0}, t_f + \delta t_f) \\ 0 \end{bmatrix} \tag{7}$$

The deviation of the propagated state $\phi(\mathbf{x_0}, t_f)$ with respect to the final desired state $\mathbf{x_d}$ is

explicitly calculated in Equation 8.

$$\delta\mathbf{x} = \mathbf{x_d} - \phi(\mathbf{x_0}, t_f) = \begin{bmatrix} x(\mathbf{x_0} + \delta\mathbf{x_0}, t_f + \delta t_f) - x(\mathbf{x_0}, t_f) \\ 0 - y(\mathbf{x_0}, t_f) \\ z(\mathbf{x_0} + \delta\mathbf{x_0}, t_f + \delta t_f) - z(\mathbf{x_0}, t_f) \\ 0 - v_x(\mathbf{x_0}, t_f) \\ v_y(\mathbf{x_0} + \delta\mathbf{x_0}, t_f + \delta t_f) - v_y(\mathbf{x_0}, t_f) \\ 0 - v_z(\mathbf{x_0}, t_f) \end{bmatrix} \tag{8}$$

Ideally, the inversion of the second representation of Equation 5 would allow to obtain a solution for $\delta\mathbf{x}_0$ and δt_f (the propagation time is unknown) which depend on $\delta\mathbf{x}$. Not all of the equations represented by 5 are of interest, in particular the relevant ones are coming from the second, fourth and sixth row. This fact can be appreciated from the explicit form of $\delta\mathbf{x}$ in Equation 8. The quantities before the minus sign of the first, third and fifth row are not known and not of interest since they are not constrained quantities. Moreover, the zeros of the second, fourth and sixth components in Equation 6 allows further simplification. The reduced system of equations coming from Equation 5 is then:

$$\begin{bmatrix} -y(\mathbf{x_0}, t_f) \\ -v_x(\mathbf{x_0}, t_f) \\ -v_z(\mathbf{x_0}, t_f) \end{bmatrix} = \begin{bmatrix} \Phi_{21} & \Phi_{23} & \Phi_{25} \\ \Phi_{41} & \Phi_{43} & \Phi_{45} \\ \Phi_{61} & \Phi_{63} & \Phi_{65} \end{bmatrix} \begin{bmatrix} \delta x_0 \\ \delta z_0 \\ \delta v_{y0} \end{bmatrix} + \begin{bmatrix} v_y(t_f) \\ v_y(t_f) + U_x(tf) \\ U_z(t_f) \end{bmatrix} \delta t_f \tag{9}$$

The unknowns are four (δx_0, δz_0,δv_{y0},δt_f), while there are only three equations. One additional equation is found from the expansion in series of the Jacobi constant formula:

$$\begin{aligned} J(\mathbf{x_0} + \delta\mathbf{x_0}) &= J(\mathbf{x_0}) + \frac{\partial J}{\partial x}\delta\mathbf{x_0} \\ C_{halo} &= J(\mathbf{x_0}) + \frac{\partial J}{\partial x}\delta\mathbf{x_0} \end{aligned} \tag{10}$$

The analyical expression for the gradient of the Jacobi constant is found in the appendix 4.2 The four equations in four unknowns can be gathered into an augmented system:

$$\begin{bmatrix} \Phi_{21} & \Phi_{23} & \Phi_{25} & v_y(t_f) \\ \Phi_{41} & \Phi_{43} & \Phi_{45} & v_y(t_f) + U_x(tf) \\ \Phi_{61} & \Phi_{63} & \Phi_{65} & U_z(t_f) \\ J_x(\mathbf{x_0}) & J_y(\mathbf{x_0}) & J_z(\mathbf{x_0}) & 0 \end{bmatrix} \begin{bmatrix} \delta x_0 \\ \delta z_0 \\ \delta v_{y0} \\ \delta t_f \end{bmatrix} = \begin{bmatrix} -y(\mathbf{x_0}, t_f) \\ -v_x(\mathbf{x_0}, t_f) \\ -v_z(\mathbf{x_0}, t_f) \\ C_{halo} - J(\mathbf{x_0}) \end{bmatrix} \tag{11}$$

The solution of this system ideally would give the solution in one step. Since it is based on first-order approximations of both the dynamics and the Jacobi constant, the process must be iterated. Each solution of the system in Equation 11 will try to approach the correct initial condition for the Halo wit the given Jacobi constant C_{halo}. This process will eventually converge to the solution after n steps, which depends on the tolerances that are set into the numerical implementation of the correction scheme.

1.2.2 Matlab implementation of the differential scheme

The differential correction scheme presented in Equation 11 has been implemented in *Matlab* in order to allow the iteration of the process. The pseudocode 1 presents the main steps that have to be performed. Some details regarding the implementation are listed here below in order to better appreciate how the code has been set up:

- The correction scheme Equation 11 has been iteratively applied with the *while-loop* since it is a first-order correction. The cycle ends when three absolute tolerances are satisfied: tolerance ϵ_J on the Jacobi constant, tolerance ϵ_v on the x and z components of the velocity

vector at t_f and tolerance ϵ_y on the y position component at the end of the propagation. All the tolerances were set to `1e-10` adimensional units, in order to be compatible with the propagation tolerances which are discussed in the next bullet. The tolerance on the Jacobi constant is needed since the augmented desired state targets also the Jacobi constant.

- In order to differentially correct the time of propagation t_f, an initial guess is needed. This initial guess for t_f is obtained at line 4 of the pseudo-code: the guess of the initial condition $\mathbf{x_0}^g$ is propagated with the CRTBP dynamics.
- The integrator used for all the numerical propagations is the in-built `ode78`, typically used when high accuracy is needed in the solution [1]. For the initial guess on the t_f, an *event function* is used in order to stop the propagation when the (x, z) plane crossing occurs.
- The tolerances `AbsTol` and `RelTol` for the propagator are both set to `1e-12`. The solution components at each step are around or below the unit due to the scaling of the problem. The solver will give a solution which is accurate up to the $\approx 11/12^{\text{th}}$ digit.
- The STM is calculated numerically at each step through forward finite differences, with the perturbation on the state defined by the policy: $\max(\sqrt{\epsilon_M}, x(i)\sqrt{\epsilon_m})$.

Algorithm 1 `findHalo`

Require: $\left(\mathbf{x_0^g},\ C_{halo},\ \epsilon_J,\ \epsilon_v\right)$

1: $k = 0$
2: $\mathbf{x_0^k} = \mathbf{x_0^g}$
3: $\delta\mathbf{x_0^k} = \mathbf{0}$; $\delta t_f^k = 0$; $J_{x0} = J(\mathbf{x}_0^k)$;
4: Propagate $\mathbf{x_0^g}$ with CRTBP dynamics until $\boldsymbol{\phi}(\mathbf{x_0^g}, t_f)$ intersect (x, z) plane to find t_f^g
5: **Define:** $t_f^k = t_f^g$
6: **while** $(|C_{halo} - J(\mathbf{x_0}^k)| > \epsilon_J \vee (|v_x| > \epsilon_v) \vee (|v_z| > \epsilon_v) \vee (|y| > \epsilon_y)) \wedge k < N_{max}$ **do**
7: Propagate $\mathbf{x_0^k}$ with CRTBP dynamics $\boldsymbol{\phi}(\mathbf{x_0^k}, t_f^k)$
8: **Calculate**:
9: $\boldsymbol{\Phi}(\mathbf{x_0}^k, t_f^k)$; $\boldsymbol{\phi}(\mathbf{x_0}^k, t_f^k)$; $\mathbf{f}(\mathbf{x_0}^k, t_f^k)$; $J(\mathbf{x_0}^k)$ and $\nabla J(\mathbf{x_0}^k)$
10: **Solve** augmented system defined in 11 with previous quantities:
11: $\delta\mathbf{x_0}^{k+1}$
12: **Update** initial condition and propagation time:
13: $x_0^{k+1} = x_0^k + \delta x_0^k$; $z_0^{k+1} = z_0^k + \delta z_0^k$; $v_{y0}^{k+1} = v_{y0}^k + \delta v_{y0}^k$; $t_f^{k+1} = t_f^k + \delta t_f^k$
14: $k = k + 1$
15: **end while**

The numerical continuation process has been implemented in order to obtain the solution for the Halo orbit with $C = 3.04$. The same `findHalo` algorithm 1 has been used iteratively with decreasing values of Jacobi constant. The solver finds the corresponding initial solution within its tolerances, the solution is then feed again to the solver with the value of the Jacobi constant slightly decreased. The process is repeated until the solution for $C = 3.04$ is reached. The degree of freedom that can be adjusted is how many values of Jacobi constant must be iterated in order to have convergence: too many values would be useless, while few values could not guarantee convergence or eventually converge in more steps. An equi-spaced grid of ten elements between $C = 3.09$ and $C = 3.04$ has been used in this case.

1.2.3 Numerical results: nominal Halo orbit and numerical continuation

In this section the results obtained from the numerical implementation scheme of the differential correction are presented. The results are reported with 10-digit accuracy, compatible with the

tolerances discussed before.

x	y	z
1.0590402077	0.0000000000	0.0739277378

v_x	v_y	v_z
0.0000000000	0.3469245709	0.0000000000

Table 2: Corrected initial state of the halo orbit with $C = 3.09$

x	y	z
1.0125655235	0.0000000000	0.0672339583

v_x	v_y	v_z
0.0000000000	0.5103251959	0.0000000000

Table 3: Corrected initial state of the halo orbit with $C = 3.04$

The following figures show the ten Halo orbits which were found during the process, with the associated Jacobi constant that is described qualitatively by the colored bar. The frame is the a-dimensional rotating frame in which the equations of the CRTBP are defined. The propagation time for which the plots are represented is one full period of the corresponding orbit.

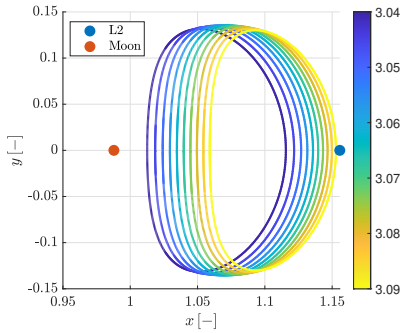

Figure 3: (x, y) plane projection

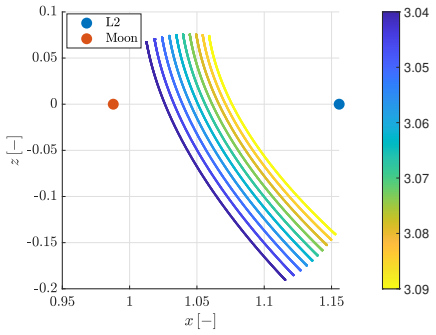

Figure 4: (x, z) plane projection

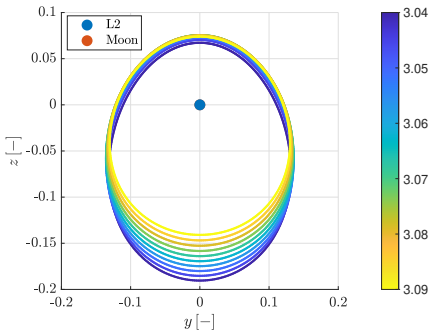

Figure 5: (z, y) plane projection

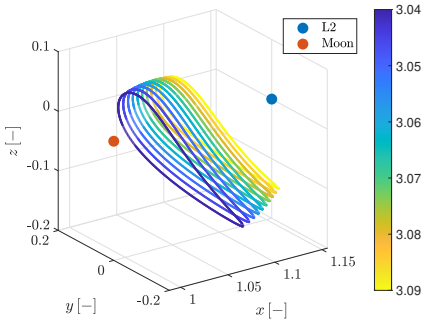

Figure 6: Lateral view

2 Impulsive guidance

Exercise 2

Consider the two-impulse transfer problem stated in Section 3.1 (Topputo, 2013)*.

1) Using the procedure in Section 3.2, produce a first guess solution using $\alpha = 0.2\pi$, $\beta = 1.41$, $\delta = 4$, and $t_i = 2$. Plot the solution in both the rotating frame and Earth-centered inertial frame (see Appendix 1 in (Topputo, 2013)). Consider the parameters listed in Table 4 and extract the radius and gravitational parameters of the Earth and Moon from the provided kernels and use the latter to compute the parameter μ.

Symbol	Value	Units	Meaning
m_s	3.28900541×10^5	-	Scaled mass of the Sun
ρ	3.88811143×10^2	-	Scaled Sun-(Earth+Moon) distance
ω_s	$-9.25195985 \times 10^{-1}$	-	Scaled angular velocity of the Sun
ω_{em}	$2.66186135 \times 10^{-1}$	s^{-1}	Earth-Moon angular velocity
l_{em}	3.84405×10^8	m	Earth-Moon distance
h_i	167	km	Altitude of departure orbit
h_f	100	km	Altitude of arrival orbit
DU	3.84405000×10^5	km	Distance Unit
TU	4.34256461	days	Time Unit
VU	1.02454018	km/s	Velocity Unit

Table 4: Constants to be considered to solve the PBRFBP. The units of distance, time, and velocity are used to map scaled quantities into physical units.

2) Considering the first guess in 1) and using $\{\mathbf{x}_i, t_i, t_f\}$ as variables, solve the problem in Section 3.1 with simple shooting in the following cases

   a) without providing any derivative to the solver, and

   b) by providing the derivatives and by estimating the state transition matrix with variational equations.

3) Considering the first guess solution in 1) and the procedure in Section 3.3, solve the problem with multiple shooting taking $N = 4$ and using the variational equation to compute the Jacobian of the nonlinear equality constraints.

4) Perform an n-body propagation using the solution $\{\mathbf{x}_i, t_i, t_f\}$ obtained in point 2), transformed in Earth-centered inertial frame and into physical units. To move from 2-D to 3-D, assume that the position and velocity components in inertial frame are $r_z(t_i) = 0$ and $v_z(t_i) = 0$. To perform the propagation it is necessary to identify the epoch t_i. This can be done by mapping the relative position of the Earth, Moon and Sun in the PCRTBP to a similar condition in the real world:

   a) Consider the definition of $\theta(t)$ provided in Section 2.2 to compute the angle $\theta_i = \theta(t_i)$. Note that this angle corresponds to the angle between the rotating frame x-axis, aligned to the position vector from the Earth-Moon System Barycenter (EMB) to the Moon, and the Sun direction.

   b) The angle θ ranges between $[0, 2\pi]$ and it covers this domain in approximately the revolution period of the Moon around the Earth.

*F. Topputo, "On optimal two-impulse Earth–Moon transfers in a four-body model", Celestial Mechanics and Dynamical Astronomy, Vol. 117, pp. 279–313, 2013, DOI: 10.1007/s10569-013-9513-8.

c) Solve a zero-finding problem to determine the epoch at which the angle Moon-EMB-Sun is equal to θ_i, considering as starting epoch `2024 Sep 28 00:00:00.000 TDB`. **Hints**: Exploit the SPK kernels to define the orientation of the rotating frame axes in the inertial frame for an epoch t. Consider only the projection of the EMB-Sun position vector onto the so-defined x-y plane to compute the angle (planar motion).

Plot the propagated orbit and compare it to the previously found solutions. (11 points)

2.1 Calculate initial guess

The four parameters $(\alpha, \beta, \delta, t_i)$ define a domain for which the space of initial guess can be mapped. Those parameters are converted into: a Cartesian state expressed in the rotating-frame centered in the Earth-Moon barycenter in a-dimensional units, an initial and final time of propagation:

$$\mathbf{x_0} = \begin{bmatrix} r_{x,0} \\ r_{y,0} \\ v_{x,0} \\ v_{y,0} \end{bmatrix} = \begin{bmatrix} r_0 \cos\alpha - \mu \\ r_0 \sin\alpha \\ -(v_0 - r_0)\sin\alpha \\ (v_0 - r_0)\cos\alpha \end{bmatrix} \tag{12}$$

$r_{x,0}$ $[DU]$	$r_{y,0}$ $[DU]$	$v_{x,0}$ $[VU]$	$v_{y,0}$ $[VU]$
0.001624	0.010008	−6.302738	8.674975

Table 5: Initial guess in Earth-Moon rotating frame.

The initial guess defined in Table 5 can be propagated from t_i to $t_i + \delta$ with the dynamics of the PBRFBP in the rotating frame. Moreover, the same solution can be plotted in both the rotating frame itself and also in a Earth-Centered inertial frame. The Appendix 1 in (Topputo, 2013)[2] allows to define a transformation from the rotating frame centered in the Earth-Moon barycenter to an inertial frame which is centered in the Earth. This last frame has the x-axis pointing to the Moon from the Earth at time $t = 0$.

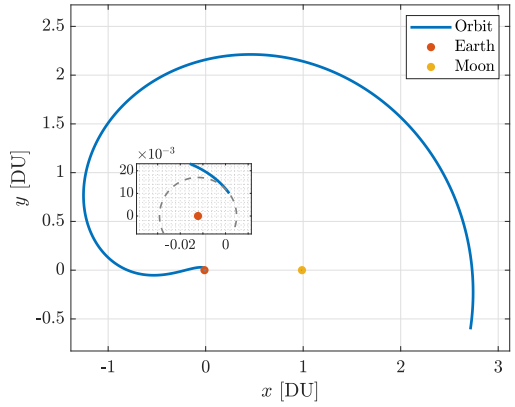

Figure 7: Propagated initial guess $\mathbf{x_0}$ in the rotating a-dimensional frame

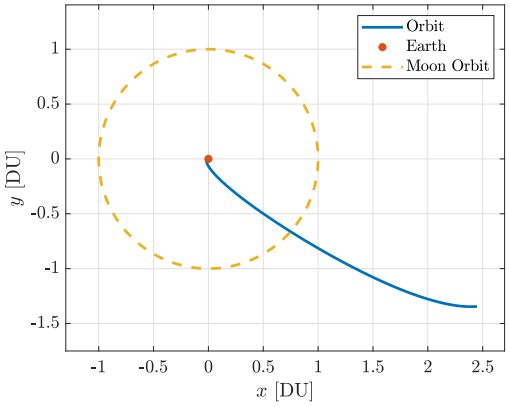

Figure 8: Propagated initial guess $\mathbf{x_0}$ in the inertial a-dimensional frame

From Figure 7 it can be inferred that the initial guess misses the Moon by more than two scaled units DU. On the other hand, the zoom on the same figure qualitatively shows that the initial state is on the expected circular orbit and tangent to it as requested by the constraint function

stated in the paper. In fact, generating the initial condition with the four parameters stated as in (Topputo, 2013)[2] permits to generate states which respect *exactly* the boundary constraint at t_i. The constraint functions will be shown in the following sections.

2.2 Shooting methods

The aim of this section is to show how the optimization problem for the two-impulse Earth-Moon transfer can be mapped into a NLP which can be solved with two methodologies: simple shooting or multiple shooting. The general framework of both methods is:

Problem Statement:
Minimize the cost function $f(\mathbf{y})$ while satisfying the constraints: $\mathbf{c}(\mathbf{y}) = \mathbf{0}$

A shooting method must implemented in this framework since the evaluation of both cost and constraint function needs the propagation of a state into a dynamic field (in this case the PBRFBP). In particular, the initial and final state must satisfy some equality constraints $\mathbf{c}(\mathbf{y})$: the shooting method allows to correct the initial condition (or in our case the NLP vector of variables $\mathbf{y}$) to satisfy the boundary conditions.

2.2.1 Simple shooting

The idea of the simple shooting method applied in the NLP context is to use the Jacobian of the constraint function $\mathbf{c}(\mathbf{y})$ and the gradient of the objective function $f(\mathbf{y})$ to obtain a solution which will satisfy the constraints better during the optimization process. In particular, to target the correct point (or in general a set of points in the position and velocity space) at final time while minimizing the objective function. Such derivatives can be calculated numerically or analytically. The convergence to the solution will be faster with the analytical formulation since the numerical derivatives require more function evaluations. The explicit definitions of the cost and constraint functions with their derivatives are here reported.

Objective function: simple shooting $\quad \mathbf{y} = (\mathbf{x_i}, t_i, t_f) \to (f_s, \nabla f_s)$

$$\begin{aligned} f_s &= \Delta v(\mathbf{x_i}, t_i, t_f) = \Delta v_i(\mathbf{x_i}) + \Delta v_f(\phi(\mathbf{x_i}, t_i; t_f)) = \cdots \\ &= \sqrt{(\dot{x}_i - y_i)^2 + (\dot{y}_i + x_i + \mu)^2} - \sqrt{\frac{1-\mu}{r_i}} + \sqrt{(\dot{x}_f - y_f)^2 + (\dot{y}_f + x_f + \mu - 1)^2} - \sqrt{\frac{\mu}{r_f}} \end{aligned} \tag{13}$$

$$\nabla f_s = \frac{\partial f_s}{\partial \mathbf{y}} = \begin{bmatrix} \dfrac{\partial f_s}{\partial \mathbf{x_i}} & \dfrac{\partial f_s}{\partial t_i} & \dfrac{\partial f_s}{\partial t_f} \end{bmatrix} \tag{14}$$

Equation 14 can be further split, by noting that $\mathbf{x_f} = \phi(\mathbf{x_i}, t_i; t_f)$, replacing it in Δv_f. In this way it is clear the application of the chain rule in Equation 15

$$\frac{\partial f_s}{\partial \mathbf{x_i}} = \frac{\partial \Delta v_i(\mathbf{x_i})}{\partial \mathbf{x_i}} + \frac{\partial \Delta v_f(\mathbf{x_f})}{\partial \mathbf{x_f}} \frac{\partial \mathbf{x_f}}{\partial \mathbf{x_i}} \tag{15}$$

$$\frac{\partial \Delta v_i}{\partial \mathbf{x_i}} = \frac{1}{\sqrt{(\dot{x}_i - y_i)^2 + (\dot{y}_i + x_i + \mu)^2}} \begin{bmatrix} (\dot{y}_i + x_i + \mu) & y_i - \dot{x}_i & \dot{x}_i - y_i & x_i + \dot{y}_i + \mu \end{bmatrix} \tag{16}$$

$$\begin{aligned} \frac{\partial \Delta v_f}{\partial \mathbf{x_f}} \frac{\partial \mathbf{x_f}}{\partial \mathbf{x_i}} = \frac{\partial \Delta v_f}{\partial \mathbf{x_f}} \Phi(\mathbf{x_i}; t_f) = \frac{1}{\sqrt{(\dot{x}_f - y_f)^2 + (\dot{y}_f + x_f + \mu - 1)^2}} \\ \begin{bmatrix} \dot{y}_f + x_f + \mu - 1 & y_f - \dot{x}_f & \dot{x}_f - y_f & x_f + \dot{y}_f + \mu - 1 \end{bmatrix} \Phi(\mathbf{x_i}; t_f) \end{aligned} \tag{17}$$

The term $\Delta v_i(\mathbf{x_i})$ doesn't depend on t_i and neither on t_f. Hence, only the term Δv_f of the cost function depends on t_i and t_f

$$\frac{\partial \Delta v_f}{\partial t_i} = \frac{\partial \Delta v_f}{\partial \mathbf{x_f}} \frac{\partial \Delta \mathbf{x_f}}{\partial t_i} = \frac{1}{\sqrt{(\dot{x}_f - y_f)^2 + (\dot{y}_f + x_f + \mu - 1)^2}} \tag{18}$$

$$\begin{bmatrix} \dot{y}_f + x_f + \mu - 1 & y_f - \dot{x}_f & \dot{x}_f - y_f & x_f + \dot{y}_f + \mu - 1 \end{bmatrix} \cdot (-\Phi(\mathbf{x_i}; t_f)\boldsymbol{f}(\mathbf{x_i}))$$

$$\frac{\partial \Delta v_f}{\partial t_f} = \frac{\partial \Delta v_f}{\partial \mathbf{x_f}} \frac{\partial \Delta \mathbf{x_f}}{\partial t_f} = \frac{1}{\sqrt{(\dot{x}_f - y_f)^2 + (\dot{y}_f + x_f + \mu - 1)^2}} \tag{19}$$

$$\begin{bmatrix} \dot{y}_f + x_f + \mu - 1 & y_f - \dot{x}_f & \dot{x}_f - y_f & x_f + \dot{y}_f + \mu - 1 \end{bmatrix} \cdot \boldsymbol{f}(\mathbf{x_f})$$

Equality constraints: simple shooting $\quad \mathbf{y} = (\mathbf{x_i}, t_i, t_f) \to (\mathbf{c}, \nabla \mathbf{c})$

$$\mathbf{c} = \begin{bmatrix} \boldsymbol{\psi_i}(\mathbf{x_i}) \\ \boldsymbol{\psi_f}(\mathbf{x_f}) \end{bmatrix} \tag{20}$$

$$\boldsymbol{\psi_i}(\mathbf{x_i}) = \begin{bmatrix} \psi_{i,1} \\ \psi_{i,2} \end{bmatrix} = \begin{bmatrix} (x_i + \mu)^2 + y_i^2 - r_i^2 \\ (x_i + \mu)(\dot{x}_i - y_i) + y_i(\dot{y}_i + x_i + \mu) \end{bmatrix} \tag{21}$$

$$\boldsymbol{\psi_f}(\mathbf{x_f}) = \begin{bmatrix} \psi_{f,1} \\ \psi_{f,2} \end{bmatrix} = \begin{bmatrix} (x_f + \mu - 1)^2 + y_f^2 - r_f^2 \\ (x_f + \mu - 1)(\dot{x}_f - y_f) + y_f(\dot{y}_f + x_f + \mu - 1) \end{bmatrix} \tag{22}$$

Where $\boldsymbol{\psi_i}$ and $\boldsymbol{\psi_f}$ express the following conditions:

- At t_i: being on a circular orbit with radius r_i around the Earth and with a injection velocity $\mathbf{v_i}$ which is tangential to the circular orbit itself
- At arrival t_f: being on a circular orbit with radius r_f around the Moon and with a final velocity of the Earth-Moon transfer trajectory $\mathbf{v_f}$ which is tangential to the circular orbit itself.

The Jacobian of the constraint with respect to the NLP variable $\mathbf{y} = (\mathbf{x}_i, t_i, t_f)$ is calculated as:

$$\nabla \mathbf{c} = \begin{bmatrix} \nabla \psi_{i,1} & \nabla \psi_{i,2} & \nabla \psi_{f,1} & \nabla \psi_{f,2} \end{bmatrix} \tag{23}$$

The Jacobian in this case is expressed per columns (each column is the gradient of the i-th component of the constraint with respect to $\mathbf{y}$). The computation of $\nabla \psi_{i,1}$ and $\nabla \psi_{i,2}$ is trivial since the expressions explicitly depend on x_i, y_i, $\dot{x}_i$ and $\dot{y}_i$, and both have no dependence on t_i nor t_f.

$$\frac{\partial \boldsymbol{\psi_i}}{\partial \mathbf{x_i}} = \begin{bmatrix} 2(x_i + \mu) & \dot{x}_i \\ 2y_i & \dot{y}_i \\ 0 & x_i + \mu \\ 0 & y_i \end{bmatrix} \qquad \frac{\partial \boldsymbol{\psi_i}}{\partial t_i} = \frac{\partial \boldsymbol{\psi_i}}{\partial t_f} = \mathbf{0}_{1\times 2} \tag{24}$$

The derivatives of $\psi_{f,1}$ and $\psi_{f,2}$ with respect to $(\mathbf{x_i}, t_i, t_f)$ are instead more complicated since those variables doesn't appear *explicitly* in Equation 22. In order to make them appear, the chain rule is applied to have the derivatives of $\psi_{f,1}$ and $\psi_{f,2}$ with respect to the final state $\mathbf{x_f}$.

$$\frac{\partial \psi_{f,1}}{\partial \mathbf{x_i}} = \frac{\partial \psi_{f,1}}{\partial \mathbf{x_f}} \frac{\partial \mathbf{x_f}}{\partial \mathbf{x_i}} = \Phi(\mathbf{x_i}; t_f)^t \begin{bmatrix} 2(x_f + \mu - 1) \\ 2y_f \\ 0 \\ 0 \end{bmatrix} \tag{25}$$

$$\frac{\partial \psi_{f,1}}{\partial t_i} = \frac{\partial \psi_{f,1}}{\partial \mathbf{x_f}} \frac{\partial \mathbf{x_f}}{\partial t_i} = \begin{bmatrix} 2(x_f + \mu - 1) \\ 2y_f \\ 0 \\ 0 \end{bmatrix} \cdot (-\Phi(\mathbf{x_i}; t_f) \boldsymbol{f}(\mathbf{x_i})) \tag{26}$$

$$\frac{\partial \psi_{f,1}}{\partial t_f} = \frac{\partial \psi_{f,1}}{\partial \mathbf{x_f}} \frac{\partial \mathbf{x_f}}{\partial t_f} = \begin{bmatrix} 2(x_f + \mu - 1) \\ 2y_f \\ 0 \\ 0 \end{bmatrix} \cdot \boldsymbol{f}(\mathbf{x_f}) \tag{27}$$

The transposition of the STM in *Equation* 25 is needed in order to have the column vector as ouput and respect matrix multiplication requirements. The last two equations are scalar quantities (dot products between vectors).

Similarly for $\psi_{f,2}$:

$$\frac{\partial \psi_{f,2}}{\partial \mathbf{x_i}} = \frac{\partial \psi_{f,2}}{\partial \mathbf{x_f}} \frac{\partial \mathbf{x_f}}{\partial \mathbf{x_i}} = \Phi(\mathbf{x_i}; t_f)^t \begin{bmatrix} \dot{x}_f \\ \dot{y}_f \\ x_f + \mu - 1 \\ y_f \end{bmatrix} \tag{28}$$

$$\frac{\partial \psi_{f,2}}{\partial t_i} = \frac{\partial \psi_{f,2}}{\partial \mathbf{x_f}} \frac{\partial \mathbf{x_f}}{\partial t_i} = \begin{bmatrix} \dot{x}_f \\ \dot{y}_f \\ x_f + \mu - 1 \\ y_f \end{bmatrix} \cdot (-\Phi(\mathbf{x_i}; t_f) \boldsymbol{f}(\mathbf{x_i})) \tag{29}$$

$$\frac{\partial \psi_{f,2}}{\partial t_f} = \frac{\partial \psi_{f,2}}{\partial \mathbf{x_f}} \frac{\partial \mathbf{x_f}}{\partial t_f} = \begin{bmatrix} \dot{x}_f \\ \dot{y}_f \\ x_f + \mu - 1 \\ y_f \end{bmatrix} \cdot \boldsymbol{f}(\mathbf{x_f}) \tag{30}$$

The pseudo-code regarding the simple shooting implementation is here presented.

Algorithm 2 `simpleShooting`

Require: ($\mathbf{y_0}$, gradFlag)
Initialize cost and constraint functions
  Get $\mathbf{x_0}$, t_i and t_f from $\mathbf{y}$
  Propagate $\mathbf{x_0}$ from t_i to t_f $\rightarrow$ $\boldsymbol{\phi}(t_f, \mathbf{x_i})$ and $\boldsymbol{\Phi}(t_f, \mathbf{x_i})$
function COST FUNCTION($\mathbf{y}$)
  return f_s and $\boldsymbol{\nabla} f_s$
end function
function CONSTRAINT FUNCTION($\mathbf{y}$)
  return $\mathbf{c}$ and $\boldsymbol{\nabla}\mathbf{c}$
end function
Minimize $f(\mathbf{y})$ while satisfying $\mathbf{c}(\mathbf{y})$ using $\mathbf{y_0}$ as initial guess

The main aspects of the numerical implementation are here reported:

- The propagation of the dynamics of the PBRFBP and the STM (obtained through the variational approach) necessary in the cost and constraints functions, have been performed with `ode78` Runge-Kutta integrator of Matlab. The `RelTol` and `AbsTol` were both set to `1e-12`, this allows to have around 11/12-digit accuracy on the propagated state for the adimensionalized dynamics of the PBRFBP.
- The constrained minimization problem is solved within the *Matlab* environment by using the `fmincon` function. The `active-set` algorithm was chosen for the minimization since

it delivered fast performances. The *constraint tolerance* is the absolute tolerance for the equality constraint violation that can be accepted, it has been set to `1e-10` in order to have restricted constraints violation on the final solution. The a-posteriori check on the solution found is performed also by evaluating its constraints violation of both $\boldsymbol{\psi}_i$ and $\boldsymbol{\psi}_f$.

- The linear inequality constraint $t_i < t_f$ has been expressed with the A and $\mathbf{b}$ formalism as:

$$A = \begin{bmatrix} \mathbf{0}_{1\times 4} & 1 & -1 \end{bmatrix} \qquad b = 0$$
$$A\mathbf{y} = b$$

- The lower and upper bound have also been given to the solver in order to help the optimization process. In particular:

$$x_i \in [-\mu - r_i, -\mu + r_i] \qquad y_i \in [-r_i, r_i]$$
$$\dot{x_i}, \dot{y_i} \in \left[-\sqrt{2}\sqrt{\frac{1-\mu}{r_i}}, +\sqrt{2}\sqrt{\frac{1-\mu}{r_i}}\right]$$
$$t_i \in \left[0, \frac{2\pi}{|\omega_s|}\right] \qquad t_f \in \left[0, \frac{2\pi}{|\omega_s|} + 23\right]$$

  The first row on constraints simply states that the solution starts in a square box of r_i length sides, centered in the Earth. The second row sets a maximum magnitude on the velocity of $2\sqrt{(1-\mu)/r_i}$. The third row expresses a constraint on the time domain, the initial time is defined in a time period which allows to explore the whole set of possible Sun configurations, with a maximumtransfer duration up to 23 TU (around 100days) [2]. Clearly solutions which have t_f near 0 are not feasible due to increased value of cost (recalling $\Delta v \Delta t$ plots [2]). That minimum value has been put due to lack of further information on the final time.

- The calculation of cost $f(\mathbf{y})$ and constraints $\mathbf{c}(\mathbf{y})$ requires the same propagated state at each step of the minimization algorithm (initial condition and time span of propagation are contained in $\mathbf{y}$). In order to perform one propagation during the same optimization step of `fmincon`, the cost and constraint functions are organized in a common *parent* function.

- The first optimization call is done without analytical gradients. The approximation of derivatives for both cost and constraint is the *forward finite differences*, in this case computational efforts is saved against accuracy in derivatives approximation.

Gradients	$r_{x,0}$ $[DU]$	$r_{y,0}$ $[DU]$	$v_{x,0}$ $[VU]$	$v_{y,0}$ $[VU]$	t_i $[TU]$	t_f $[TU]$
False	+0.001517	+0.010154	−6.379292	+8.586913	+2.040	+6.057
True	+0.001620	+0.010013	−6.290741	+8.651415	+2.198	+6.203

Table 6: Simple shooting solutions in the Earth-Moon rotating frame.

In both cases, the solver finds a possible minimum and stops due to the satisfaction of the step size tolerance (exit flag is 4), and constraints are satisfied within their tolerances. The minimization with the analytical gradient is much faster than without it, thus justifying a clear advantage in terms of computational effort (also visible during the minimization process in terms of reduced number of function evaluations at each iteration). It is also interesting to note that the *first-order optimality* is 4 orders of magnitude less for the analytical-gradient case (around 10^1). This indicates that the solution found with the analytical gradients satisfies better the necessary condition for optimality. The constraint violations at the arrival orbit $\boldsymbol{\psi_f(x_{sol})}$ are:

- No gradients: Around 10^{-12} and 10^{-11} for first and second component of $\boldsymbol{\psi_f}$ respectively (i.e. position constraint and tangential constraint).
- With gradients: Around 10^{-14} and 10^{-12} for first and second component of $\boldsymbol{\psi_f}$ respectively (i.e. position constraint and tangential constraint).

The solutions evaluated at the initial constraint $\boldsymbol{\psi_i}$ in both cases reveal much lower values (around 10^{-17}). Given all the previous considerations, both solutions are valid due to their respect of constraints.

It is also interesting to retrieve the total cost and total time of flight for the two transfers, resumed in Table 7:

Gradients	Δv $[km/s]$	Δt $[days]$
False	4.1059	17.443
True	4.1052	17.392

Table 7: Simple shooting solutions.

The graphical representation of both solution is here reported.

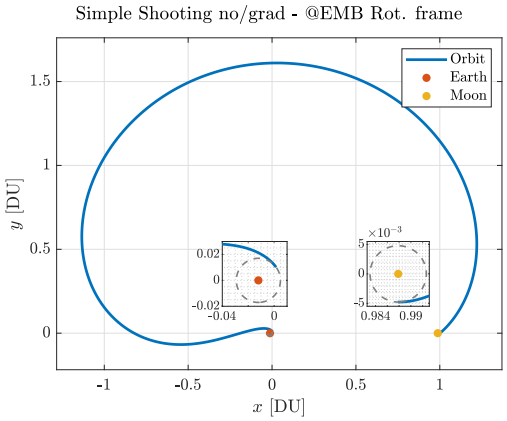

Figure 9: Simple Shooting w/out gradients - synodic frame

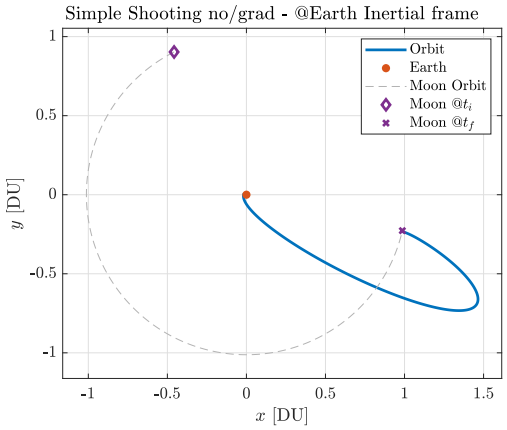

Figure 10: Simple Shooting w/out gradients - inertial frame

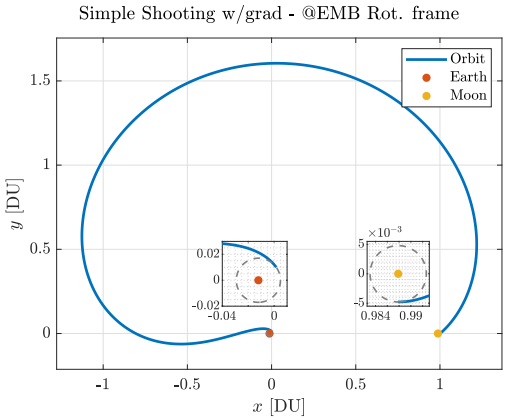

Figure 11: Simple Shooting with gradients - synodic frame

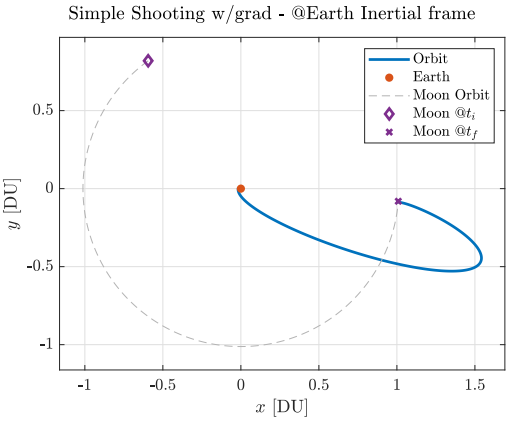

Figure 12: Simple Shooting with gradients - inertial frame

The solutions have qualitatively similar shapes as appreciated by the trajectory in the synodic frame (Figure 9 and Figure 11). However, the difference on t_i (Table 6) and Δt (Table 7) can be appreciated by the plots of the trajectory in the inertial frame: the trajectory in Figure 12 is slightly offset clockwise with respect to the trajectory in Figure 10.

2.2.2 Multiple shooting

The idea of the multiple shooting method is to attempt to solve the same constrained optimization problem stated into a NLP formulation, while trying to reduce the sensitivity of the problem by adding more variables in $\mathbf{y}$. In particular, the issue is that the dynamics is very sensitive to changes on the initial conditions and also the propagation time is high. This means that after a certain amount of time of propagation, the numerical solution is no more reliable. In particular, the STM information which is contained in the analytical derivatives of both $f(\mathbf{y})$ and $\mathbf{c}(\mathbf{y})$ loses of significance. In this case the propagation is divided in $N-1=3$ pieces, thus the NLP variable $\mathbf{y}$ is composed of $4N+2=18$ variables. The construction of the NLP variable $\mathbf{y}$ is summarized as follows:

- Consider the uniform time grid from t_i to t_f : $[t_1 = t_i, t_2, t_3, t_4 = t_f]$.
- Discretize the solution $\mathbf{x}(t)$ of the PBRFBP dynamics over the same time grid to get: $[\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}]$
- Take the NLP variable as $\mathbf{y} = [\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}, t_i, t_f]$. In this way the state at the discretized times is allowed to vary, without depending directly on the propagation.

The same cost function defined for the simple shooting is here represented, however the crucial change of the NLP variable $\mathbf{y}$ will be appreciated when calculating the gradients of $f(\mathbf{y})$ and $\mathbf{c}(\mathbf{y})$.

Objective function: multiple shooting $\quad \mathbf{y} = (\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}, t_i, t_f) \to (f_m, \nabla f_m)$

$$f_m = \Delta v(\mathbf{x_1}, \mathbf{x_4}) = \Delta v_1(\mathbf{x_1}) + \Delta v_f(\mathbf{x_4}) = \cdots$$

$$= \sqrt{(\dot{x}_1 - y_1)^2 + (\dot{y}_1 + x_1 + \mu)^2} - \sqrt{\frac{1-\mu}{r_i}} + \sqrt{(\dot{x}_4 - y_4)^2 + (\dot{y}_4 + x_4 + \mu - 1)^2} - \sqrt{\frac{\mu}{r_f}} \tag{31}$$

$$\nabla f_m = \frac{\partial f_m}{\partial \mathbf{y}} = \begin{bmatrix} \frac{\partial f_m}{\partial \mathbf{x_1}} & \frac{\partial f_m}{\partial \mathbf{x_2}} & \frac{\partial f_m}{\partial \mathbf{x_3}} & \frac{\partial f_m}{\partial \mathbf{x_4}} & \frac{\partial f_m}{\partial t_i} & \frac{\partial f_m}{\partial t_f} \end{bmatrix} = \tag{32}$$

$$= \begin{bmatrix} \frac{\partial f_m}{\partial \mathbf{x_1}} & \mathbf{0}_{1\times 4} & \mathbf{0}_{1\times 4} & \frac{\partial f_m}{\partial \mathbf{x_4}} & 0 & 0 \end{bmatrix}$$

Where:

$$\frac{\partial f_m}{\partial \mathbf{x_1}} = \frac{1}{\sqrt{(\dot{x}_1 - y_1)^2 + (\dot{y}_1 + x_1 + \mu)^2}} \begin{bmatrix} (\dot{y}_1 + x_1 + \mu) & y_1 - \dot{x}_1 & \dot{x}_1 - y_1 & x_1 + \dot{y}_1 + \mu \end{bmatrix} \tag{33}$$

$$\frac{\partial f_m}{\partial \mathbf{x_4}} = \frac{1}{\sqrt{(\dot{x}_4 - y_4)^2 + (\dot{y}_4 + x_4 + \mu - 1)^2}} \tag{34}$$

$$\begin{bmatrix} \dot{y}_4 + x_4 + \mu - 1 & y_4 - \dot{x}_4 & \dot{x}_4 - y_4 & x_4 + \dot{y}_4 + \mu - 1 \end{bmatrix}$$

The key point of both Equation 33 and Equation 34 is that there is no dynamics propagation (the STM doesn't show up here). The sensitivity of the problem to the dynamic propagation is greatly reduced in this way, since the cost function has no need of a propagated dynamics to be performed, the cost is uniquely build up by considering the independent variables $\mathbf{x_1}$ and $\mathbf{x_4}$.

Equality constraints: multiple shooting. $\mathbf{y} = (\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}, \mathbf{x_4}, t_i, t_f) \rightarrow (\mathbf{c}, \mathbf{c_{dis}}, \nabla\mathbf{c}, \nabla\mathbf{c_{dis}})$

$$\mathbf{c} = \begin{bmatrix} \boldsymbol{\zeta_1} = \boldsymbol{\phi}(\mathbf{x_1}, t_1; t_2) - \mathbf{x_2} \\ \boldsymbol{\zeta_2} = \boldsymbol{\phi}(\mathbf{x_2}, t_2; t_3) - \mathbf{x_3} \\ \boldsymbol{\zeta_3} = \boldsymbol{\phi}(\mathbf{x_3}, t_3; t_4) - \mathbf{x_4} \\ \boldsymbol{\psi_i}(\mathbf{x_i}) \\ \boldsymbol{\psi_f}(\mathbf{x_f}) \end{bmatrix} \tag{35}$$

The constraints $\mathbf{c}(\mathbf{y})$ contains also the need to match the full-state between the propagation of $\mathbf{x_{i-1}}$ to t_i and the NLP variable itself $\mathbf{x_i}$, their difference is called *defect* and is represented by the symbol $\boldsymbol{\zeta_{i-1}}$. Forcing the full-state to be the same means that no internal impulsive manoeuvres are imposed: .

$$\nabla\mathbf{c} = \begin{bmatrix} \dfrac{\partial\boldsymbol{\zeta_1}}{\partial\mathbf{y}} \\ \dfrac{\partial\boldsymbol{\zeta_2}}{\partial\mathbf{y}} \\ \dfrac{\partial\boldsymbol{\zeta_3}}{\partial\mathbf{y}} \\ \dfrac{\partial\boldsymbol{\psi_i}}{\partial\mathbf{y}} \\ \dfrac{\partial\boldsymbol{\psi_f}}{\partial\mathbf{y}} \end{bmatrix} \qquad \left.\frac{\partial\boldsymbol{\zeta_i}}{\partial\mathbf{x_j}}\right|_{i=1,2,3} = \begin{cases} \mathbf{0}_{4\times4} & j > i+1 \quad \vee \quad j < i \\ \Phi\left(\mathbf{x_i}; t_{i+1}\right) & j = i \\ -\mathbf{I}_{4\times4} & j = i+1 \end{cases} \qquad j = 1,2,3,4 \tag{36}$$

And also (from [3]):

$$\left.\frac{\partial\boldsymbol{\zeta}_i}{\partial t_1}\right|_{i=1,2,3} = -\boldsymbol{\Phi}\left(\mathbf{x_i}; t_{i+1}\right)\boldsymbol{f}(\mathbf{x_i}, t_i)\frac{\partial t_i}{\partial t_1} + \mathbf{f}(\phi(\mathbf{x_i}; \mathbf{t_{i+1}}))\frac{\partial t_{i+1}}{\partial t_1} = \cdots \tag{37}$$

$$= -\boldsymbol{\Phi}\left(\mathbf{x_i}; t_{i+1}\right)\boldsymbol{f}(\mathbf{x_i}, t_i)\frac{N-i}{N-1} + \mathbf{f}(\phi(\mathbf{x_i}; \mathbf{t_{i+1}}))\frac{N-1-i}{N-1} \qquad i = 1,2,3$$

$$\left.\frac{\partial\boldsymbol{\zeta}_i}{\partial t_4}\right|_{i=1,2,3} = -\boldsymbol{\Phi}\left(\mathbf{x_i}; t_{i+1}\right)\boldsymbol{f}(\mathbf{x_i}, t_i)\frac{\partial t_i}{\partial t_4} + \mathbf{f}(\phi(\mathbf{x_i}; \mathbf{t_{i+1}}))\frac{\partial t_{i+1}}{\partial t_4} = \cdots \tag{38}$$

$$= -\boldsymbol{\Phi}\left(\mathbf{x_i}; t_{i+1}\right)\boldsymbol{f}(\mathbf{x_i}, t_i)\frac{i-1}{N-1} + \mathbf{f}(\phi(\mathbf{x_i}; \mathbf{t_{i+1}}))\frac{i}{N-1} \qquad i = 1,2,3$$

The derivatives of the path constraints $\boldsymbol{\psi_i}(\mathbf{x_1})$ and $\boldsymbol{\psi_f}(\mathbf{x_4})$ depends explicitly on $\mathbf{x_1}$ and $\mathbf{x_4}$ respectively. While no dependence on neither t_i nor t_f. There is no dynamics involved, as in the case of the cost function f_m. Hence:

$$\frac{\partial\boldsymbol{\psi_i}}{\partial t_1} = \frac{\partial\boldsymbol{\psi_i}}{\partial t_4} = \frac{\partial\boldsymbol{\psi_f}}{\partial t_1} = \frac{\partial\boldsymbol{\psi_f}}{\partial t_4} = \mathbf{0}_{2\times1} \tag{39}$$

and

$$\frac{\partial\boldsymbol{\psi_i}}{\partial\mathbf{x_1}} = \begin{bmatrix} 2(x_1+\mu) & 2y_1 & 0 & 0 \\ \dot{x}_1 & \dot{y}_1 & x_1+\mu & y_1 \end{bmatrix} \qquad \frac{\partial\boldsymbol{\psi_f}}{\partial\mathbf{x_4}} = \begin{bmatrix} 2(x_4+\mu-1) & 2y_4 & 0 & 0 \\ \dot{x}_4 & \dot{y}_4 & x_4+\mu-1 & y_4 \end{bmatrix}$$

$$\frac{\partial\boldsymbol{\psi_i}}{\partial\mathbf{x_j}} = \mathbf{0}_{2\times4} \quad j = 2,3,4 \qquad \frac{\partial\boldsymbol{\psi_f}}{\partial\mathbf{x_j}} = \mathbf{0}_{2\times4} \quad j = 1,2,3$$

The implementation on *Matlab* requires the transpose of the just defined Jacobian for the constraint with respect to the formulation here presented.

During the optimization process, the trajectory must not impact neither the Earth nor the Moon, this aspect is handled by the inequality constraints on the mesh points: $\boldsymbol{\eta}_j < 0$, where([3]):

$$\boldsymbol{\eta}_j = \begin{cases} R_{e,ad} - (x_j + \mu)^2 - y_j^2 \\ R_{m,ad} - (x_j + \mu - 1)^2 - y_j^2 \end{cases} \qquad j = 1, 2, 3, 4 \tag{40}$$

Moreover, the final time must be greater than the initial time $t_4 > t_1$. This would be a linear inequality constraint but can be organized in the set of non linear inequality constraints $\mathbf{c_{dis}}(\mathbf{y}) \leq \mathbf{0}$. Its Jacobian $\nabla \mathbf{c_{dis}}(\mathbf{y})$ can be expressed as ([3]):

$$\frac{\partial \mathbf{c_{dis}}}{\partial \mathbf{y}} = \begin{bmatrix} \dfrac{\partial \boldsymbol{\eta_1}}{\partial \mathbf{y}} \\ \dfrac{\partial \boldsymbol{\eta_2}}{\partial \mathbf{y}} \\ \dfrac{\partial \boldsymbol{\eta_3}}{\partial \mathbf{y}} \\ \dfrac{\partial \boldsymbol{\eta_4}}{\partial \mathbf{y}} \\ \dfrac{\partial (t_4 - t_1)}{\partial \mathbf{y}} \end{bmatrix} \qquad \left.\frac{\partial \boldsymbol{\eta_i}}{\partial \mathbf{x_j}}\right|_{i=1\cdots4} = \begin{cases} \mathbf{0}_{2\times4} & i \neq j \\ \boldsymbol{S_i} & i = j \end{cases} \qquad j = 1, \cdots, 4 \tag{41}$$

$$\boldsymbol{S_i} = \begin{bmatrix} -2(x_i + \mu) & -2y_i & 0 & 0 \\ -2(x_1 + \mu - 1) & -2y_i & 0 & 0 \end{bmatrix} \quad i = 1 \cdots 4 \tag{42}$$

$$\frac{\partial (t_4 - t_1)}{\partial \mathbf{y}} = \begin{bmatrix} \mathbf{0}_{1\times16} & 1 & -1 \end{bmatrix} \tag{43}$$

A simplified pseudo-code of the multiple shooting algorithm **with** analytical derivatives of *cost function*, *disequality* and *equality constraints* is represented:

Algorithm 3 `multipleShooting`

Require: $(\mathbf{y_0})$
Minimize $f(\mathbf{y})_m$ while satisfying $\mathbf{c}(\mathbf{y})$ and $\mathbf{c}(\mathbf{y})_{dis}$ using $\mathbf{y_0}$ as initial guess
Initialize cost and constraint functions
function COST FUNCTION($\mathbf{y}$)
    $\mathbf{x_1} \leftarrow \mathbf{y}(1:4)$ and $\mathbf{x_4} \leftarrow \mathbf{y}(13:16)$
    return f_m and $\boldsymbol{\nabla} f_m$
end function
function CONSTRAINT FUNCTION($\mathbf{y}$)
    for $k \leftarrow 1$ to 3 **do**
        Propagation step: $\boldsymbol{\phi}(\mathbf{x_i}, t_i; t_{i+1})$ and $\boldsymbol{\Phi}(\mathbf{x_i}, t_i; t_{i+1}) \leftarrow \mathbf{x_i}$
    end for
    return $\mathbf{c}$, $\mathbf{c}_{dis}$, $\boldsymbol{\nabla}\mathbf{c}$ and $\boldsymbol{\nabla}\mathbf{c}_{diseq}$
end function

The main aspects of the numerical implementation are here reported:

- Unlike the *simple shooting* method, in this case the cost function f_m and its gradient do not need the propagation of the dynamics. As a consequence, there is no need of a *parent* function to share the common propagation.
- The propagation of the dynamics of both PBRFBP dynamics and the STM has been performed with `ode78` with both `AbsTol` and `RelTol` set to `1e-12`.
- The constrained minimization problem is solved within the *Matlab* environment by using the `fmincon` function with the `active-set` algorithm. The *constraint tolerance* of both equality and inequality constraints was set to `1e-10`, compatible with the previously stated tolerances of the propagation.

- In the case of *multiple shooting*, the satisfaction of the *defect* constraints $\zeta_{1,2,3}$ is crucial to guarantee a continuous flow. Roughly speaking, the imposition of 10^{-10} as tolerance on the defects would mean to accept an error of ≈ 4 cm on position and $\approx 10^{-5} cm/s$ on velocities. The *full-state* must be continuous within a certain tolerance, since internal burns are not allowed in this case.
- The correct satisfaction of all the constraints (defects $\boldsymbol{\zeta}_i$, boundary constraint $\boldsymbol{\psi}_i$, $\boldsymbol{\psi}_f$ on $\mathbf{x_1}$ and $\mathbf{x_4}$ and also the inequality constraints $\boldsymbol{\eta}_i$) is checked a-posteriori on the found solution.
- Linear equality and inequality constraints are not applied, the same for lower and upper bounds.
- The multiple shooting algorithm requires an initial guess of the NLP variable y which is composed of four full states evenly spaced on the time grid. These values are calculated by propagating the initial guess retrieved from ($\alpha = 0.2\pi, \beta = \sqrt{2}, t_i = 2, \delta = 4$). The propagation for building the initial guess is performed with `ode78`, both `AbsTol` and `RelTol` are set to `1e-12` and the only values $\mathbf{x}_i$ on the discrete time grid t_1, t_2, t_3 and t_4 are used. Since the NLP variable size increased with respect to the simple shooting case, the converging process is really sensitive on the initial guess and the optimization can take some time.
- Max function evaluations and max number of iterations needed to be increased to let the solver search the optimal solution without stopping earlier.

$r_{x,0}$ $[DU]$	$r_{y,0}$ $[DU]$	$v_{x,0}$ $[VU]$	$v_{y,0}$ $[VU]$	t_i $[TU]$	t_f $[TU]$
-0.019033	-0.015574	9.755916	-4.311381	3.6126	4.7071

Table 8: Multiple shooting solution in the Earth-Moon rotating frame.

In this case the solver takes approximately 6000 iterations and 29500 function evaluations to find the solution which respects the constraint tolerance. The presence of the analytical gradient allows to have few function evaluations per each iteration and more robust derivative calculation with respect to the numerical approximations, however the optimization process with this initial guess takes some minutes (around 10 minutes). The solver stops due to satisfaction of the step tolerance condition (the exit flag is 4). The maximum violation of the constraints is around 10^{-11}, compatible with both required tolerances on the defects (for the continuity of the state) and on the initial/final orbits path constraints. The values for cost and time of flight are in Table 9.

Gradients	Δv $[km/s]$	Δt $[days]$
true	3.9587	4.753

Table 9: Multiple shooting solution

From Table 9 it is clear a slight advantage in terms of cost and a great reduction of time of flight with respect to the simple shooting solutions Table 7, starting from the same parametrized initial guess.

In the *Matlab* code, the multiple shooting solution is in a `.mat` file called `MSHOOT_refined` due to the time needed to be found. It is possible to run the Multiple Shooting solver by changing the initial guess propagation options (in particular decreasing it), to find another solution that takes much less time to be found. This solution respects the constraint (the max violation is always lower than 10^{-10}), but it is very different from the multiple shooting solution presented in this report and more similar to the simple shooting solution presented in the previous section.

The solution trajectory is plotted in both rotating EMB centered and inertial Earth centered reference frames.

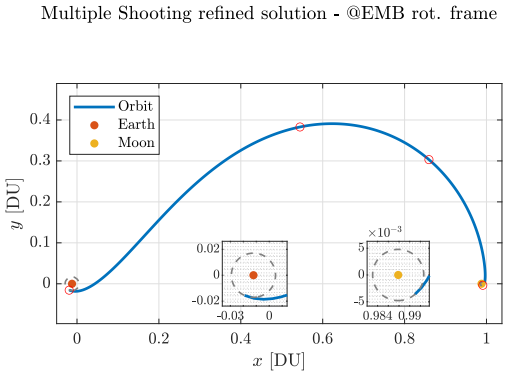

Figure 13: Multiple Shooting - @EMB synodic frame

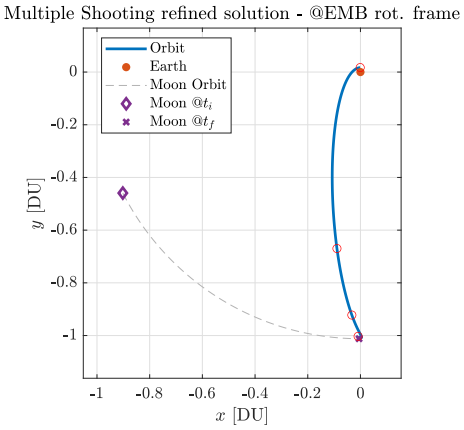

Figure 14: Multiple Shooting - @Earth inertial frame

The four red dots on both trajectories are the points in which the multiple shooting solution is discretized. It is clear, from both Figure 13 and Figure 14, the difference of shape with respect to the simple shooting solutions. The reduction of time of flight Δt is qualitatively visible in both representations of the trajectory. In fact, the simple shooting solutions have a wider shape in the rotating frame representation and they cross the lunar trajectory twice in the inertial frame.

2.3 Comparison of the solutions

The found solutions of both simple and multiple shooting resumed in Table 7 and Table 9, can be put in the general framework developed by (Topputo, 2013 [2]) in order to be better categorized. The Δv-Δt plot is here reported with the found solutions.

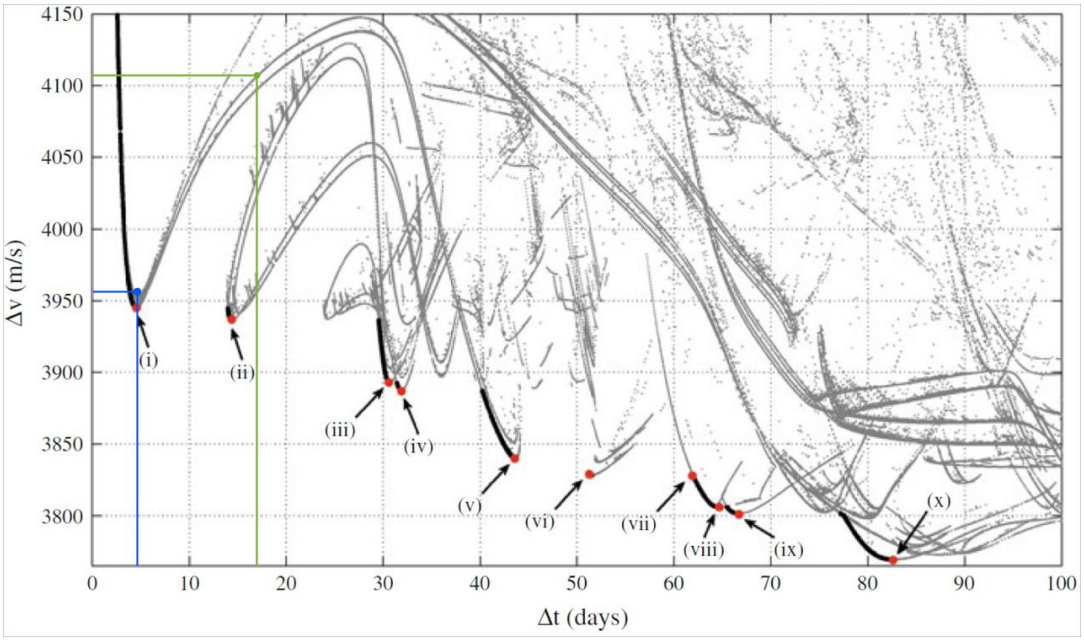

Figure 15: Δv-Δt plot from [2]

- The solution trajectories presented before are found within two different schemes: multiple shooting (blue dot) and simple shooting (green dot). The solutions that populates the Δv-Δt diagram in Figure 15 are all resulting from a constrained optimization within the multiple shooting scheme as described in (Topputo, 2013 [2]). For comparison, also the simple shooting solutions found in this report are introduced.
- It is expected that the solutions presented in this report are not the pareto-optimal found in (Topputo,2013 [2]), since those last are found after that a rich population of feasible solutions is built (i.e. through numerical continuation).
- The solution of the multiple shooting scheme that is represented by Table 9 and the blue dot in Figure 15 is near the pareto-optimal solutions of the family (a), but it is not pareto-efficient. Infact, the sample point (i) which is part of pareto-optimal of the (a) family, is characterized by $\Delta v = 3.9448$km/s and $\Delta t = 4.59$ days ([2]).
- The two solutions of the simple shooting method, represented both by a green dot, are always in the family (a), defined in (Topputo,2013 [2]). In particular on the branch (a+) which defines the prograde orbits (this is confirmed by the plots of the trajectory Figure 10). The solution is minimizing the cost and satisfying the constraints, still it is not pareto-efficient.

2.4 N-body propagation

The N-body propagation requires the knowledge of the initial epoch since the position of the gravitational masses must be known to the *spice* kernels, this aspect is treated in 2.4.1. Then, the solution found in the simple shooting is converted into a state in the real world (see 2.4.2). Lastly, the n-body propagation is performed in Matlab and compared with the trajectory of the same initial condition propagated with the PBRFBP dynamic model in 2.4.3.

2.4.1 Find an epoch in real world

The search of a reference epoch t_i expressed in *ephemeris time* ET_i is carried out as follows:

- Using the adimensional initial time t_i of the reference solution found with simple shooting: calculate the angle $\tilde{\theta}$ between the Moon and Sun directions both from the Earth-Moon-Barycenter, in the synodic frame.

$$\tilde{\theta} = |\omega_s t_i| \in [0, 2\pi) \tag{44}$$

- Define the function $\Delta\theta = \tilde{\theta} - \theta$, where θ is defined as the angle **between** the projection of the Sun vector onto the Earth-Moon motion plane and the $\hat{\mathbf{x}}$ direction of the synodic frame (Moon direction seen from the EMB). The steps to calculate θ are:

  1. **Define the Earth-Moon frame in EMB-Centered ECLIPJ2000**
     This frame is denoted by $[\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}]$. Where $\hat{\mathbf{x}}$ is the Moon direction from the EMB expressed in ECLIPJ2000 frame. The $\hat{\mathbf{z}}$ direction is the cross product between the Moon direction from the EMB expressed in ECLIPJ2000 frame and the Moon velocity expressed in the same frame. The $\hat{\mathbf{y}}$ is obtained as a consequence in order to obtain the right-handed reference frame. This definition corresponds to the osculating Earth-Moon frame, whose axis clearly change in time with respect to the inertial ECLIPJ2000.
  2. **Obtain the transformation matrix $\boldsymbol{R}$** from ECLIPJ2000 to the Earth-Moon

rotating frame (both centered in EMB).

$$\boldsymbol{R} = \begin{bmatrix} \hat{\mathbf{x}} \\ \hat{\mathbf{y}} \\ \hat{\mathbf{z}} \end{bmatrix} \tag{45}$$

3. **Express** the Sun direction in the rotating frame (RF) by using the Sun direction in the EMB-centered ECLIPJ2000 frame and the transformation matrix $\boldsymbol{R}$:

$$\mathbf{r}_{Sun,RF} = \boldsymbol{R}\,\mathbf{r}_{Sun,ECJ2} \tag{46}$$

This vector in general have all three components also in the just-defined Earth-Moon rotating frame. The angle of interest is calculated using the projection on the Earth-Moon motion plane $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$.

4. **Calculate the angle** of the projected vector with the $\hat{\mathbf{x}}$ axis:

$$\theta = \arctan\left(\frac{x_{Sun,RF}}{y_{Sun,RF}}\right) \in [0, 2\pi) \tag{47}$$

- Find the zero of $\Delta\theta\,(ET)$ using `fzero`. Before starting the zero-finding algorithm, the function have been plotted from the initial time `2024 Sep 28 00:00:00.000 TDB` for a period which an entire revolution of the sun in the EMB frame. Due to the wrap function, the angle have a discontinuities. To help the convergence process of `fzero`, the initial guess domain has been restricted accordingly (see the plot in appendix 4.4). The default relative tolerance on `tolX` of $\epsilon_M = 2.2204 \cdot 10^{-16}$ has been used: the solution is in the order of $7 \cdot 10^8$ s in Ephemeris Time, hence the accuracy on the solution found is on the order of 10^{-7} s.

2.4.2 Retrieve the initial condition for the n-body propagation

The simple shooting solution state $\mathbf{x_s}$ expressed in the a-dimensional synodic reference frame of the PBRFBP, is converted into a state in the Earth-Centered Inertial reference frame dimensionalized with DU, VU and TU. The transformation between synodic and inertial Earth-centered reference frame in Appendix 1 of (Topputo,2013)[2] is used. Notice that this transformation is a pure rotation around the z-axis plus a translation, while the real transformation between the Earth-Moon rotating frame and Earth centered ECLIPJ2000 reference frame would be made using *Spice* kernels. The transformation from Earth-centered inertial reference frame to ECLIPJ2000 is simplified by putting both the z-components of position and velocity put to 0 in order to have an initial condition.

2.4.3 N-body propagation: numerical results

The main aspects of the n-body propagation in the *Matlab* framework are here reported:

- The numerical propagation of the n-body dynamics centered at the Earth has been performed with `ode78`. The `RelTol` has been set to `1e-12`. The dynamical model of the n-body propagation is presented in the appendix 4.5
- Since the equations of motion are dimensional, the `AbsTol` for positions have been set to `1e-8` [km]. The maximum value for the position components in this case is approximately `1e6` [km]: in that worst case the accuracy is obtained with the relative tolerance and it is $\approx$ `1e-6` [km]. When the switch to the absolute tolerance happens (when the position components are around or below `1e4` [km]) the accuracy is around `1e-8` km.

- The `AbsTol` for the velocity has been set to `1e-11` [km/s], since the velocity components are always near the unity or smaller (down to 0), the control on the accuracy for the velocity happens almost always with the absolute tolerance.

The obtained solution of the N-body propagation (nBD) is represented graphically and compared with the PBRFBP propagation (PBRFBP) of the same initial guess. The considered frame is the ECLIPJ2000 centered in the Earth (Figure 16 and Figure 17). The PBRFBP is mapped into the ECLIPJ2000 frame by putting to 0 both r_z and v_z. Both qualitative and quantitave trends have been plotted for the comparison.

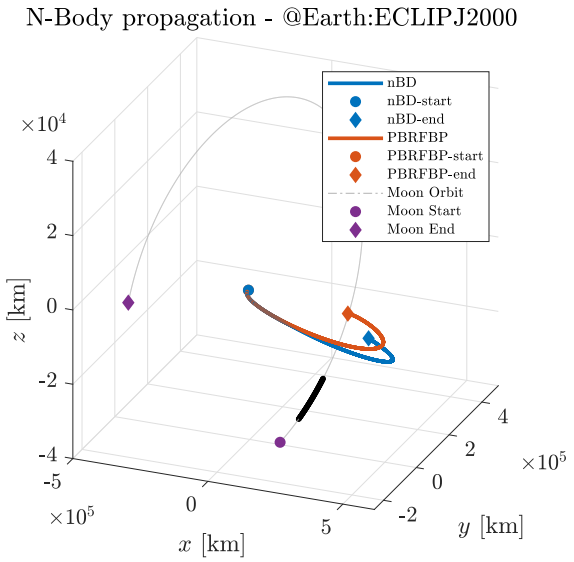

Figure 16: Lateral view

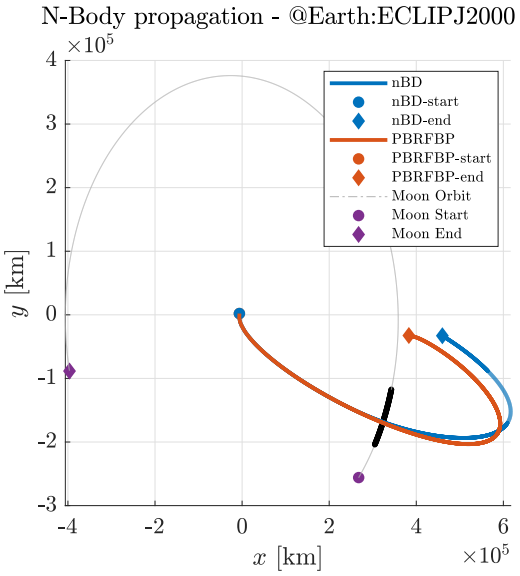

Figure 17: (x, y) plane view

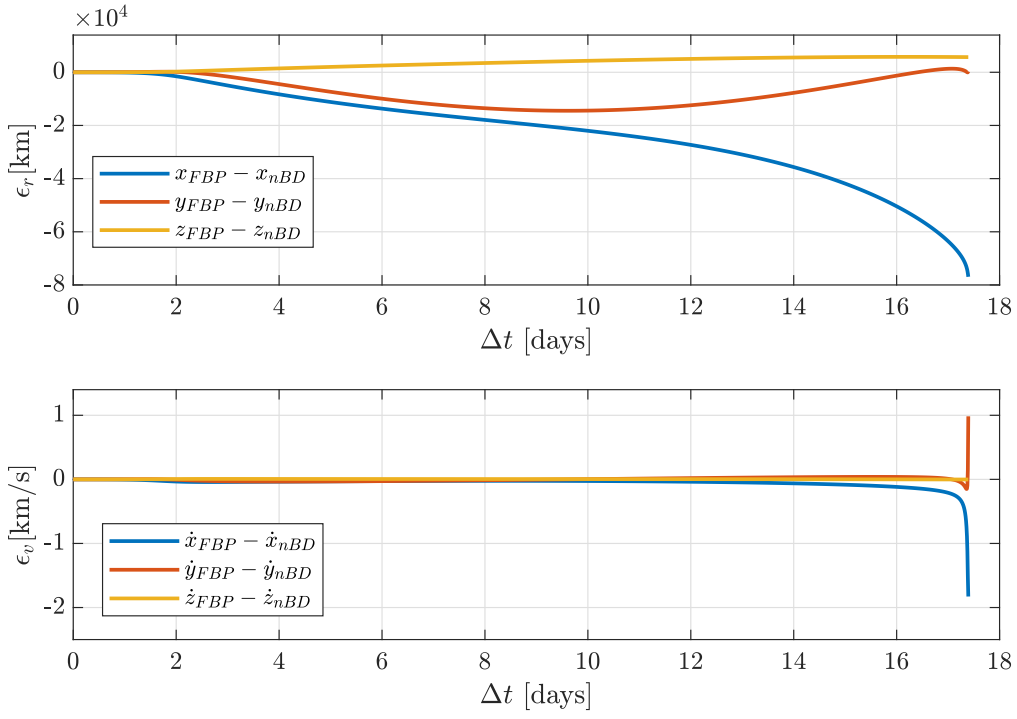

Figure 18: Quantitative evolution - starting epoch t_i = `2024-10-12T21:25:33.894` UTC

Some remarks are reported:

- The N-body propagation in the real world has a different shape with respect to the PBRFBP trajectory. See Figure 16 and Figure 17.
- The N-body propagation develops a z-component in terms of position that can be appreciated quantitatively in Figure 18 and also graphically in Figure 16.
- It is interesting to notice how the deviation of the N-body propagation is not immediate but it develops after a certain time. This aspect is appreciated in Figure 18 in which after approximately two days of propagation (around `2024-OCT-12` UTC) there is a clear deviation of the two propagation methods. Around that epoch, the Moon is tracing the black trajectory in Figure 17: its motion outside the (x, y) plane (which for the PBRFBP model is both the Earth-Moon plane and the Sun motion plane) is responsible for the tilting of the N-body trajectory with respect the planar one of the PBRFBP.
- Regarding the second plot in Figure 18, there is an increasing error in the velocity components x and y at the end of the propagation. This is due to the fact that the trajectory of the PBRFBP in its model is doing a rendezvous with the Moon before the injection in the final circular orbit. In this framework, when the S/C approach the Moon, the absolute value of the velocity increases in norm due to the approach with a gravitational attractor. On the other hand, the real trajectory (from the N-body propagation) is not near the Moon, nor approaching it. As a consequence there is no build up in the velocity components v_x and v_y, this induces a large difference in velocity components between the models.

Symbol	**Calendar epoch** (UTC)
t_i	2024-10-12T21:25:33.894
t_f	2024-10-30T06:50:20.051

$r_{x,0}$ $[km]$	$r_{y,0}$ $[km]$	$r_{z,0}$ $[km]$
-6223.62922245	$+2026.14224923$	0.0

$v_{x,0}$ $[km/s]$	$v_{y,0}$ $[km/s]$	$v_{z,0}$ $[km/s]$
-3.39799462	-10.43749945	0.0

Table 10: Initial epoch, final epoch, and initial state in Earth-centered inertial frame.

3 Continuous guidance

Exercise 3

A low-thrust option is being considered to perform an orbit raising maneuver using a low-thrust propulsion system in Earth orbit. The spacecraft is released on a circular orbit on the equatorial plane at an altitude of 800 km and has to reach an orbit inclined by 0.75 deg on the equatorial plane at 1000 km. This orbital regime is characterized by a large population of resident space objects and debris, whose spatial density q can be expressed as:

$$q(\rho) = \frac{k_1}{k_2 + \left(\dfrac{\rho - \rho_0}{DU}\right)^2}$$

where ρ is the distance from the Earth center. The objective is to design an optimal orbit raising that minimizes the risk of impact, that is to minimize the following objective function

$$F(t) = \int_{t_0}^{t_f} q(\rho(t))\,\mathrm{d}t.$$

The parameters and reference Distance Unit to be considered are provided in Table 11.

Symbol	Value	Units	Meaning
h_i	800	km	Altitude of departure orbit
h_f	1000	km	Altitude of arrival orbit
Δi	0.75	deg	Inclination change
R_e	6378.1366	km	Earth radius
μ	398600.435	$\mathrm{km}^3/\mathrm{s}^2$	Earth gravitational parameter
ρ_0	$750 + R_e$	km	Reference radius for debris flux
k_1	1×10^{-5}	DU^{-1}	Debris spatial density constant 1
k_2	1×10^{-4}	DU^2	Debris spatial density constant 2
m_0	1000	kg	Initial mass
$T_{\max}$	3.000	N	Maximum thrust
I_{sp}	3120	s	Specific impulse
DU	7178.1366	km	Distance Unit
MU	m_0	kg	Mass Unit

Table 11: Problem parameters and constants. The units of time TU and velocity VU can be computed imposing that the scaled gravitational parameter $\overline{\mu} = 1$.

1) Plot the debris spatial density $q(\rho) \in [h_i - 100;\ h_f + 100]$ km and compute the initial state and target orbital state, knowing that: i) the initial and final state are located on the x-axis of the equatorial J2000 reference frame; ii) the rotation of the angle Δi is performed around the x-axis of the equatorial J2000 reference frame (RAAN = 0).

2) Adimensionalize the problem using as reference length $DU = \rho_i = h_i + R_e$ and reference mass $MU = m_0$, imposing that $\mu = 1$. Report all the adimensionalized parameters.

3) Using the PMP, write down the spacecraft equations of motion, the costate dynamics, and the zero-finding problem for the unknowns $\{\boldsymbol{\lambda}_0, t_f\}$ with the appropriate transversality condition. **Hint**: the spacecraft has to reach the target state computed in point 1).

4) Solve the problem considering the data provided in Table 11. To obtain an initial guess for the costate, generate random numbers such that $\lambda_{0,i} \in [-250; +250]$ while $t_f \approx 20\pi$. Report the obtained solution in terms of $\{\boldsymbol{\lambda}_0, t_f\}$ and the error with respect to the target.

Assess your results exploiting the properties of the Hamiltonian in problems that are not time-dependent and time-optimal solution. Plot the evolution of the components of the primer vector $\boldsymbol{\alpha}$ in a NTW reference frame[†].

5) Solve the problem for a lower thrust level $T_{\max} = 2.860$ N. Compare the new solution with the one obtained in the previous point. **Hint**: exploit numerical continuation. (11 points)

3.1 Debris density function, calculation of initial and target state

The debris density function is represented here.

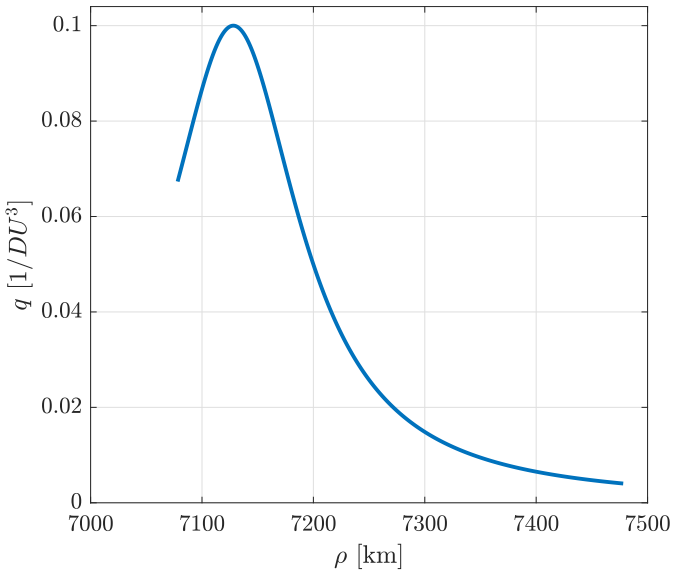

Figure 19: Debris density function q expressed in adimensional units

The maximum of the debris density is at $\rho_0 = 7128.1$ km, slightly before the initial and final orbit radii of interest for the problem that is solved (see Table 12). The values for the cartesian state are obtained as:

$$r_{x,i} = R_e + h_i \qquad r_{x,f} = R_e + h_f$$

$$v_{y,i} = \sqrt{\frac{r_{x,i}^3}{\mu}} \qquad v_{y,f} = \sqrt{\frac{r_{x,f}^3}{\mu}}\cos(i_2) \qquad v_{z,f} = \sqrt{\frac{r_{x,f}^3}{\mu}}\sin(i_2)$$

$r_{x,i}$ $[km]$	$r_{y,i}$ $[km]$	$r_{z,i}$ $[km]$	$v_{x,i}$ $[km/s]$	$v_{y,i}$ $[km/s]$	$v_{z,i}$ $[km/s]$
+7178.136600	0000.000000	0000.000000	0.00000000	+7.45183148	0.00000000

$r_{x,f}$ $[km]$	$r_{y,f}$ $[km]$	$r_{z,f}$ $[km]$	$v_{x,f}$ $[km/s]$	$v_{y,f}$ $[km/s]$	$v_{z,f}$ $[km/s]$
+7378.136600	0000.000000	0000.000000	0.00000000	+7.34950091	+0.09621034

Table 12: Initial and target state in Earth-centered equatorial J2000 intertial frame.

[†]The T-axis is aligned with the velocity, the N-axis is aligned with the angular momentum, while the W-axis is pointing inwards, i.e., towards the Earth.

3.2 Adimensionalization of the problem

The problem is adimensionalized with the following parameters.

$$DU = \rho_i = h_i + R_e \tag{48}$$

$$MU = m_0 \tag{49}$$

The adimensional period, by choosing $\bar{\mu} = 1$, would become (recalling $a = \rho_i$, $\bar{a} = \dfrac{a}{DU} = 1$):

$$\bar{T} = 2\pi\sqrt{\frac{\bar{a}^3}{\bar{\mu}}} = 2\pi \tag{50}$$

Consequently, the time unit TU becomes:

$$TU = \frac{T}{\bar{T}} = \sqrt{\frac{a^3}{\mu}} \tag{51}$$

The velocity unit VU is obtained simply by composition of DU and TU:

$$VU = \frac{DU}{TU} = \sqrt{\frac{\mu}{\rho_i}} \tag{52}$$

From this fundamentals dimensional units, all other composition can be retrieved (Force unit FU and Acceleration unit AcU):

$$FU = \frac{MU \cdot DU}{TU^2} \tag{53}$$

Recalling that the dimensional force must be expressed in $\left[\dfrac{kg \cdot km}{s^2}\right]$

$$AcU = \frac{DU}{TU^2} \tag{54}$$

Recalling that the dimensional acceleration must be expressed in $\left[\dfrac{km}{s^2}\right]$.

The scaling step is important in the context of trajectory optimization problems since it allows to have different quantities with similar scales (typically position and velocity), and consequently helping the convergence process of the numerical optimization algorithms.

3.3 S/C equations of motion and costate dynamics: definition of the zero-finding problem

The objective of this section is to define the mathematical process in order to obtain the formulation of the optimal control problem that minimizes the functional $F(t)$ defined in the exercise statement. The problem can be formalized as:

Problem Statement:

$$\min_{\mathbf{u}\in\mathcal{U},t_f} \int_{t_0}^{t_f} \frac{k_1}{k_2+(\bar{\rho}(t,\mathbf{u})-\bar{\rho}_0)^2}dt \qquad \text{s.t.} \quad \begin{cases} \dot{\mathbf{x}} & = \mathbf{f}(\mathbf{x},\mathbf{u},t) \\ \mathbf{x}(t_0) & = \mathbf{x}_0 \\ x_i(t_f) & = \psi_i(t_f) \quad i=1,\cdots,6 \end{cases} \tag{55}$$

Where the set of admissible controls is defined as:

$$\mathcal{U} = \{\, (u,\hat{\alpha}) \mid u\in[0,1], ||\hat{\alpha}|| = 1 \,\} \tag{56}$$

And the right-hand side of the S/C dynamics is defined as (a-dimensional):

$$\mathbf{f}(\mathbf{x},\mathbf{u}) = \begin{cases} \dot{\mathbf{r}} & = \mathbf{v} \\ \dot{\mathbf{v}} & = -\dfrac{\mathbf{r}}{r^3} + u\dfrac{T_{max}}{m}\hat{\boldsymbol{\alpha}} \\ \dot{m} & = -u\dfrac{T_{max}}{I_{sp}g_0} \end{cases} \tag{57}$$

The stated OCP requires to find both the control action $\mathbf{u}$ and the final time of propagation t_f that minimizes the functional. Since the control action is bounded, the PMP is used to find the necessary conditions of the optimality. To find a solution, the problem is solved by writing down the necessary conditions for the optimality and retrieve a TPBVP which is solved with a zero-finding problem. The procedure is here described, all the equations are adimensionalized according to the quantities defined in subsection 3.2. The first step to perform is to write down the Hamiltonian of the system:

1. **Write down the Hamiltonian of the system**

$$H = q(\rho(t)) + \boldsymbol{\lambda}\cdot\boldsymbol{f} = \frac{k_1}{k_2+(\bar{\rho}(t)-\bar{\rho}_0)^2} + \begin{bmatrix}\boldsymbol{\lambda_r}\\ \boldsymbol{\lambda_v}\\ \lambda_m\end{bmatrix}\cdot\begin{bmatrix}\mathbf{v}\\ -\dfrac{\mathbf{r}}{r^3}+u\dfrac{T_{max}}{m}\hat{\boldsymbol{\alpha}}\\ -u\dfrac{T_{max}}{I_{sp}g_0}\end{bmatrix} = \cdots \tag{58}$$

$$= \frac{k_1}{k_2+(\bar{\rho}(t)-\bar{\rho}_0)^2} + \boldsymbol{\lambda_r}\cdot\mathbf{v} - \frac{\boldsymbol{\lambda_v}\cdot\mathbf{r}}{r^3} + u\frac{T_{max}}{m}\boldsymbol{\lambda_v}\cdot\hat{\boldsymbol{\alpha}} - \lambda_m u\frac{T_{max}}{I_{sp}g_0}$$

Note that $\bar{\rho} = ||\boldsymbol{r}(t)||$ which is part of the solution to be found.

2. **Apply the PMP condition**

$$(u^*,\hat{\alpha}^*) = \arg\min_{(u^*,\hat{\alpha}^*)\in\mathcal{U}} H \tag{59}$$

The quantities $(u^*,\hat{\alpha}^*)$ are found in two steps. Firstly, the vector $\hat{\boldsymbol{\alpha}}^*$ is found by noting that the minimization of H (Equation 58) happens when the direction of the thrust is:

$$\hat{\boldsymbol{\alpha}}^* = -\frac{\boldsymbol{\lambda_v}}{\lambda_v} \tag{60}$$

Then, the magnitude of the control u^* is found by replacing $\hat{\alpha}^*$ in H to find:

$$H = \frac{k_1}{k_2+(\bar{\rho}(t)-\bar{\rho}_0)^2} + \boldsymbol{\lambda_r}\cdot\mathbf{v} - \frac{\boldsymbol{\lambda_v}\cdot\mathbf{r}}{r^3} + \frac{T_{max}}{I_{sp}g_0}u^*\left(-\frac{\lambda_v I_{sp}g_0}{m}-\lambda_m\right) = \cdots \tag{61}$$

$$= \frac{k_1}{k_2+(\bar{\rho}(t)-\bar{\rho}_0)^2} + \boldsymbol{\lambda_r}\cdot\mathbf{v} - \frac{\boldsymbol{\lambda_v}\cdot\mathbf{r}}{r^3} + \frac{T_{max}}{I_{sp}g_0}u^*S_t$$

The Hamiltonian in this form is minimized by the following control policy:

$$u^* = \begin{cases} 0 & S_t > 0 \\ 1 & S_t < 0 \\ \in (0,1) & S_t = 0 \end{cases} \qquad \text{with :} \quad S_t = \left(-\frac{\lambda_v I_{sp} g_0}{m} - \lambda_m \right) \tag{62}$$

3. **Define the E/L equations embedding the PMP condition** The necessary conditions for the optimality are expressed with the E/L equations, which are derived by the Hamiltonian H which already embeds the PMP condition (that would be the third equation $\frac{\partial H}{\partial \mathbf{u}} = \mathbf{0}$ coming from the E/L necessary conditions). The first two equations are respectively the S/C and the costate dynamics, the last equation is the transversality condition and it expresses the fact that the final time t_f is not fixed but still a constraint on the final orbital state is present.

$$\begin{cases} \dot{\mathbf{x}} = \dfrac{\partial H}{\partial \boldsymbol{\lambda}} \\ \dot{\boldsymbol{\lambda}} = -\dfrac{\partial H}{\partial \mathbf{x}} \\ H(t_f) - \boldsymbol{\lambda}(t_f) \cdot \dot{\boldsymbol{\psi}}_f = 0 \end{cases}$$

$$\begin{cases} \dot{\mathbf{r}} = \mathbf{v} \\ \dot{\mathbf{v}} = -\dfrac{\mathbf{r}}{r^3} - u^*(m, \boldsymbol{\lambda_v}, \lambda_m) \dfrac{T_{max}}{m} \dfrac{\boldsymbol{\lambda_v}}{\lambda_v} \\ \dot{m} = u^*(m, \boldsymbol{\lambda_v}, \lambda_m) \dfrac{\boldsymbol{T_{max}}}{I_{sp} g_0} \\ \dot{\boldsymbol{\lambda}}_r = \dfrac{2k_1(\bar{\rho} - \bar{\rho}_0)}{(k_2 + (\bar{\rho} - \bar{\rho}_0)^2)^2} \dfrac{\mathbf{r}}{r} - \dfrac{3}{r^5} (\mathbf{r} \cdot \boldsymbol{\lambda_v}) \mathbf{r} + \dfrac{\boldsymbol{\lambda_v}}{r^3} \\ \dot{\boldsymbol{\lambda}}_v = -\boldsymbol{\lambda}_r \\ \dot{\lambda}_m = -u^*(m, \boldsymbol{\lambda_v}, \lambda_m) \dfrac{\lambda_v T_{max}}{m^2} \end{cases} \qquad \begin{cases} \mathbf{r}(t_0) = \mathbf{r_0} \\ \mathbf{v}(t_0) = \mathbf{v_0} \\ m(t_0) = m_0 \\ \mathbf{r}(t_f) = \mathbf{r_f} \\ \mathbf{v}(t_f) = \mathbf{v_f} \\ \lambda_m(t_f) = 0 \\ H(t_f) = \boldsymbol{\lambda}(t_f) \cdot \dot{\boldsymbol{\psi}}_f \end{cases} \tag{63}$$

The system of differential equations 63 can be further developed by noting that the dynamics of λ_m is such that $\dot{\lambda}_m \leq 0$. This has drastic consequences on the control policy expressed in equation Equation 62. In fact, the sign of the switching function would become $S_t < 0$ always, leading to have $u^* = 1$ for all the transfer. Moreover the transversality condition can be explicitly stated as:

$$H(t_f) = \begin{bmatrix} \boldsymbol{\lambda}_r(t_f) \\ \boldsymbol{\lambda}_v(t_f) \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0_{3\times1} \\ 0_{3\times1} \\ \psi_{m,f} \end{bmatrix} \implies H(t_f) = 0 \qquad \forall \qquad \boldsymbol{\lambda}_r(t_f), \boldsymbol{\lambda}_v(t_f), m_f \tag{64}$$

After these important remarks, the coupled dynamics of the S/C state and the co-state becomes:

$$\begin{cases} \dot{\mathbf{r}} = \mathbf{v} \\ \dot{\mathbf{v}} = -\dfrac{\mathbf{r}}{r^3} - \dfrac{T_{max}}{m} \dfrac{\boldsymbol{\lambda_v}}{\lambda_v} \\ \dot{m} = \dfrac{\boldsymbol{T_{max}}}{I_{sp} g_0} \\ \dot{\boldsymbol{\lambda}}_r = \dfrac{2k_1(\bar{\rho} - \bar{\rho}_0)}{(k_2 + (\bar{\rho} - \bar{\rho}_0)^2)^2} \dfrac{\mathbf{r}}{r} - \dfrac{3}{r^5} (\mathbf{r} \cdot \boldsymbol{\lambda_v}) \mathbf{r} + \dfrac{\boldsymbol{\lambda_v}}{r^3} \\ \dot{\boldsymbol{\lambda}}_v = -\boldsymbol{\lambda}_r \\ \dot{\lambda}_m = \dfrac{\lambda_v T_{max}}{m^2} \end{cases} \qquad \begin{cases} \mathbf{r}(t_0) = \mathbf{r_0} \\ \mathbf{v}(t_0) = \mathbf{v_0} \\ m(t_0) = m_0 \\ \mathbf{r}(t_f) = \mathbf{r_f} \\ \mathbf{v}(t_f) = \mathbf{v_f} \\ \lambda_m(t_f) = 0 \\ H(t_f) = 0 \end{cases} \tag{65}$$

The set of differential equations and boundary conditions of Equation 65 defines a TPBVP. The aim of the next discussion is to solve it: to do so, the set of seven initial conditions for the co-state $\boldsymbol{\lambda}_0$ and the final time t_f must be found. More formally the problem becomes:

Zero-finding Problem:
Find $\boldsymbol{\lambda}_0$ and t_f such that

$$\begin{bmatrix} \mathbf{r}(t_f) \\ \mathbf{v}(t_f) \\ \boldsymbol{\lambda}(t_f) \end{bmatrix} = \phi \left(\begin{bmatrix} \mathbf{r}_0 \\ \mathbf{v}_0 \\ \boldsymbol{\lambda}_0 \end{bmatrix}, t_o; t_f \right)$$

Satisfies:

$$S_f(\boldsymbol{\lambda_0}, t_f) = \begin{cases} \mathbf{r}(t_f) - \mathbf{r_f} = \mathbf{0}_{3\times 1} \\ \mathbf{v}(t_f) - \mathbf{v_f} = \mathbf{0}_{3\times 1} \\ \lambda_m(t_f) = 0 \\ H(t_f) = 0 \end{cases} \tag{66}$$

Where ϕ is the flow of the dynamics represented by the differential equations in 65 and $S_f(\boldsymbol{\lambda_i}, t_f)$ is the shooting function for which the zero must be found.

Basically, the solution of this zero-finding problem will be compatible with the necessary conditions for the optimality of the OCP.

3.4 Solution of the zero-finding problem

The zero-finding problem for the function S_f defined in Equation 66 has been implemented in *Matlab*. In particular:

- The shooting function itself requires the propagation of the coupled state and co-state dynamics (adimensional states). The in-built `ode78` propagator has been selected. The `RelTol` and `AbsTol` tolerances has been set both to `1e-12`.
- The initial conditions $\boldsymbol{\lambda}_0$ has been randomly generated in the domain $[-250; +250]$ apart from $\lambda_{m,0}$ which is varied from $[0; +250]$. Moreover, the initial guess for the final time has been set to 20π (approximately 10 revolutions) with a slight random perturbation of $\pm\pi/2$, in order to allow the solver to explore more possibilities for the optimum research.
- The `levenberg-marquardt` algorithm of `fsolve` has been selected since it delivered fast performances. The `function tolerance` has been set to `1e-7`. This would mean to accept a variation of $\leq 10^{-7}$ DU and $\leq 10^{-7}$ VU on the first three components of the S_f function, which correspond to approximately 0.7m (in the order of the size of a S/C) and a velocity variation of approximately $8 \cdot 10^{-4}$ m/s.
- The research of a feasible solution requires repeating the zero-finding problem multiple times on different random initial guesses. In order to do this, several random generation of initial guesses $(\boldsymbol{\lambda}_0, t_f)$ has been explored. The ones which struggled from the beginning to minimize the residual of the algorithm, have been immediately discarded. Some other have been saved into a batch and further explored by letting the solver proceed with the iterations (in particular `MaxFunctionEvaluations` to `2e4` and `MaxIterations` to `1e3`).
- The criteria adopted in order to accept the final solution is based on the satisfaction of the orbital state constraint at the found solution. When the solver finds $(\boldsymbol{\lambda}_0, t_f)$ such that $\max(\mathbf{r}(t_f) - \mathbf{r}_f)$ is below 1 m (approximately the sice of a S/C) and $\max(\mathbf{v}(t_f) - \mathbf{v}_f)$ is below 10^{-4} m/s , the solution can be considered as feasible. This means a maximum function value at the solution around 10^{-7} DU and 10^{-7} VU for the first three components relative to the position and velocity respectively.

- No analytical derivatives have been given to the solver, hence the choice of using a central finite difference scheme for the derivatives, accepting the trade-off with longer execution times.

The solution features are reported in the following tables:

t_f [mins]	m_f [kg]
1035.1974	993.9120

λ_{0,r_x}	λ_{0,r_y}	λ_{0,r_z}	λ_{0,v_x}	λ_{0,v_y}	λ_{0,v_z}	$\lambda_{0,m}$
-214.9812	-10.3659	$+0.8856$	-10.3929	-214.6105	-112.9454	2.5964

Table 13: Optimal orbit raising transfer solution ($T_{\max} = 3.000$ N).

Error	**Value**	**Units**
$\|\|\mathbf{r}(t_f) - \mathbf{r}_f\|\|$	$2.435529 \cdot 10^{-7}$	km
$\|\|\mathbf{v}(t_f) - \mathbf{v}_f\|\|$	$2.417092 \cdot 10^{-7}$	m/s

Table 14: Final state error with respect to target position and velocity ($T_{\max} = 3.000$ N).

The first element of the Table 14 represent the error done on the position of the S/C at t_f, in the order of 10^{-7} km ($\approx 10^{-10} DU$) which is well below the requested cited criteria. The validity of the found solution is also checked against an important theoretical fact. In particular, the Hamiltonian of this system evaluated at the optimal solution must be 0. In fact, the transversality condition requires $H(t_f) = 0$. Moreover, the dynamics doesn't depend explicitly by time so the Hamiltonian has zero derivative with respect to time $\dot{H} = 0$. The numerical Hamiltonian calculated with a-dimensional units is around 10^{-9} for the whole time span of the solution.
The analysis proceeds by plotting the components of the direction of thrust $\boldsymbol{\alpha}^*$ for the found optimal-solution. The vector is expressed in the NTW frame, defined as: N aligned with the angular momentum, T aligned with the local velocity and W as the cross product of N and T (thus pointing inwards).

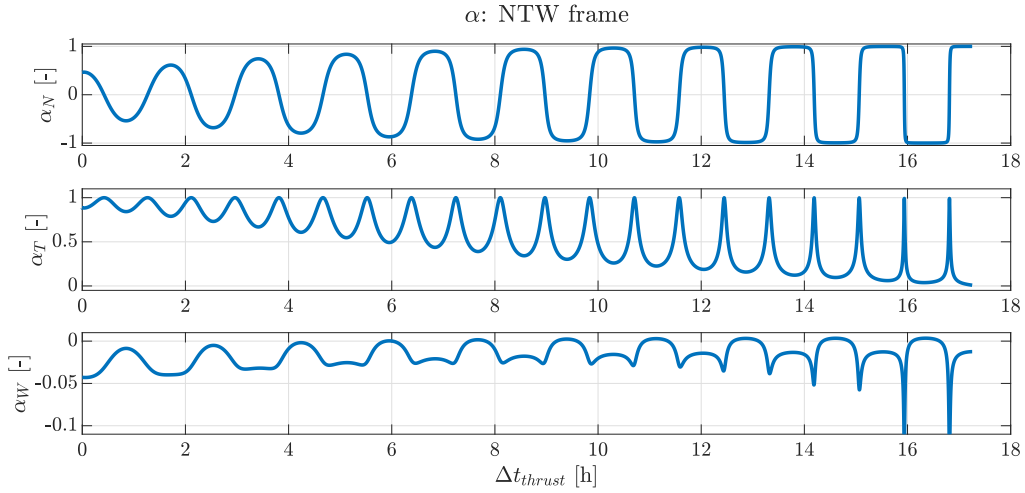

Figure 20: $\boldsymbol{\alpha}^*$ components in the NTW frame

The most relevant aspects that can be appreciated by Figure 20 are the following:

- From all the three plots, it is clear the presence of a periodical behaviour which has approximately the period of the first orbit (around 1h and 40min). The rise of 200 km will increase it up to 1h and 45min, still for this analysis it is useful to take a value of the two just for qualitative aspects. Even though there is a repetition pattern given by the time constant TU, the maximum amplitude of all components $\alpha_{N,T,W}$ changes during the transfer.
- The α_N component in the first plot touches both positive and negative values, moreover the amplitude of the oscillations increase along the transfer. The plot starts at 0 h around a local maximum of the graph, at that instant the S/C is on the ascending node of the orbit (in this case it is on the Vernal line). Approximately every half period ($\approx$ 0.84 h), there is a maximium/minimum of α_N. This component is basically the out-of-orbit component which is responsible for the change of plane. In order to increase the inclination i, without changing too much the RAAN, the maneouvers for the plane change are mostly performed on the nodal line. In particular, at the ascending node the thrust will have a relevant component in the $+N$ direction, while at the descending node the thrust will have a relevant component in the $-N$ direction. The increase of the amplitudes of the peaks of α_N is correlated to the next bullet.
- The α_T component is defined to be along the current velocity of the S/C. From the graph, it is clear that it only touches positive values. This is in line with the rising manoevure that the S/C must perform. Moreover, the mean value of α_T is much higher in the first phases of thrusting. In particular, $\alpha_T \geq 0.5$ for the first 6 h of thrust. It is clear that the solver prefers to perform the rising manoeuvre at the beginning of the transfer while the major contribution to the change plane happens later (when α_N has larger maximum amplitude). This is clear when Figure 19 is recalled in the context of the OCP that has just been solved. Basically, the solutions tends to depart as fast as possible from the initial altitude, which has an higher debris density. This fact is confirmed also by Figure 21 in which the radius of the orbit has a steeper increase in the first phase of the transfer. A

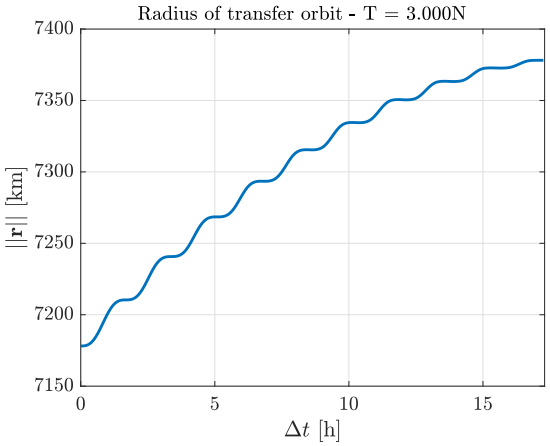

Figure 21: Radius of the transfer orbit - dimensional units

3.5 Numerical continuation of the problem

The solution of the same problem with lower level of thrust 2.860 N can be achieved through numerical continuation. A simplified scheme on how to perform the numerical continuation is here reported:

- Solve the OCP defined by the zero-finding problem of Equation 66 for $T_{max} = 3.000$ N.
- Generate an array of n values of thrust: $T_{max} = 3.000 \rightarrow 2.860$ N

1. Choose $k = 2, \cdots n$
2. Find the zero of $S_{f,k}(\boldsymbol{\lambda_{i,k}}, t_{f,k})$ taking as initial guesses $(\lambda_{i,k-1}, t_{f,k-1})$, where $S_{f,k}$ is the zero finding problem defined for the level of thrust:

$$T_{max,k} = 3.000 - \frac{0.140}{n-1}(k-1) \qquad [N]$$

3. Repeat from 2)

- Verify the convergence of the solver at each step of the numerical continuation like: exitflag (greater than 0) and S_f value at the solution. This last step is crucial in order to evaluate if the found solution is acceptable.

The zero-finding problem is solved with the same options for propagation and `fsolve` used and explained for the solution of the same problem for $T_{max} = 3.000$ N in subsection 3.4, expect for the algorithm which in this case is the default one since it showed faster performances.

t_f [mins]	m_f $[kg]$
1030.9760	994.2198

λ_{0,r_x}	λ_{0,r_y}	λ_{0,r_z}	λ_{0,v_x}	λ_{0,v_y}	λ_{0,v_z}	$\lambda_{0,m}$
−593.1529	−11.5904	2.2973	−11.9293	−592.7890	−920.3698	17.3814

Table 15: Optimal orbit raising transfer solution ($T_{\max} = 2.860$ N).

Error	**Value**	**Units**
$\|\|\mathbf{r}(t_f) - \mathbf{r}_f\|\|$	$6.584697 \cdot 10^{-8}$	km
$\|\|\mathbf{v}(t_f) - \mathbf{v}_f\|\|$	$6.889367 \cdot 10^{-8}$	m/s

Table 16: Final state error with respect to target position and velocity ($T_{\max} = 2.860$ N).

The solution with 2.860N of thrust in Table 15 says that there is a slight decrease in total flight time compared with the solution 3N of thrust. Since the specific impulse is the same, the mass consumption is also lower for the solution with the 2.860N of thrust. This fact is also confirmed by the higher final mass in the last solution found (see Table 15). The solution can be analyzed also by plotting the profile in time of the direction of the thrust $\boldsymbol{\alpha}$ in the NTW frame shown in Figure 22 and comparing it with the one found for 3.000N of thrust Figure 20.

- The new profile in Figure 22 shows that the N component of the $\boldsymbol{\alpha}$ vector is starting with a larger value compared to the one seen before. There is still a slight increase of the maximum amplitude in this component during the whole trhusting arc. Moreover, the peaks and valleys of α_N happen always at the ascending or descending node in order to obtain larger changes of inclination.
- The component T of $\boldsymbol{\alpha}$ (along the tangential direction) has a more regular behaviour with respect to the case of 3.000N. Its value oscillates with a maximum of 1 (when all other components are 0) in correspondence of the antinodal points (defined as the points which are 90-degrees shifted from the ascending or descending node). The tangential component α_T and the inward components α_W are responsible for the enlargement of the orbit (increase of Keplerian energy).
- In this case the solution under the point of view of the components of $\boldsymbol{\alpha}$ reveals that the change of plane and altitude rising manoevures are not performed in distinct phases (as it was visible in Figure 20 for the 3.000N case), instead they seems to be more blended along the whole transfer. This aspect can be appreciated by the more regular behaviour

of the $\boldsymbol{\alpha}$ profile (Figure 22) and the plot of the radius of the transfer orbit for the new case (Figure 23) which doesn't have a steeper increase at the beginning (as it was for the T = 3.000N case).

- It is true that in the case of 2.860N of thrust the transfer takes slightly less time, but it is also true that in this lower-maximum thrust scenario, the S/C passes more time at lower altitudes where the debris has higher density. The best performance between the two cases of thrust in term of probability of debris impact must consider the integral in time of the debris density which contains also the $\rho(t)$ profile in time and not just the total mission time t_f.

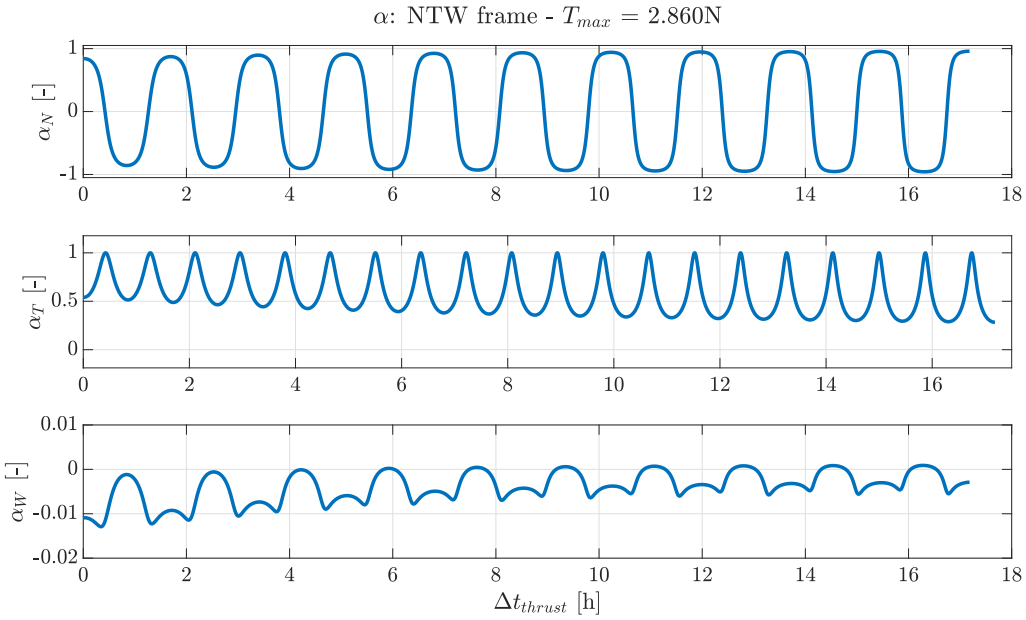

Figure 22: $\boldsymbol{\alpha}^*$ components in the NTW frame

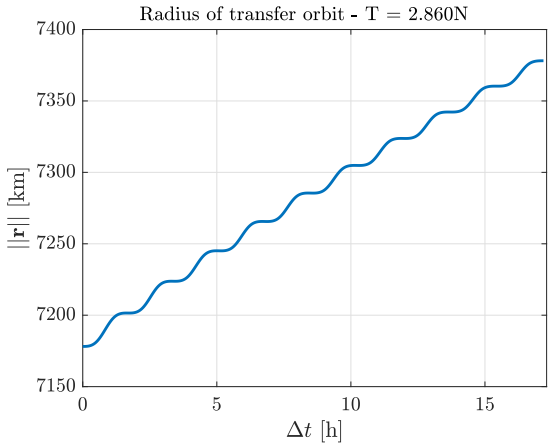

Figure 23: Radius of the transfer orbit - dimensional units

4 Appendix

4.1 CRTBP dynamics

The CRTBP for the Earth-Moon system dynamics written in the rotating EMB centered frame is (adimensional):

$$\begin{bmatrix}\dot{\mathbf{r}}\\ \dot{\mathbf{v}}\end{bmatrix} = \begin{bmatrix} \dot{x}\\ \dot{y}\\ \dot{z}\\ 2v_y + \Omega_x\\ -2v_x + \Omega_y\\ \Omega_z \end{bmatrix} \tag{67}$$

Where the potential is defined as:

$$\Omega(\mathbf{r}) = \frac{1}{2}(x^2+y^2) + \frac{1-\mu}{r_1} + \frac{\mu}{r_2} + \frac{1}{2}\mu(1-\mu) \tag{68}$$

With:

$$r_1 = \sqrt{(x+\mu)^2 + y^2 + z^2} \tag{69}$$

$$r_2 = \sqrt{(x+\mu-1)^2 + y^2 + z^2} \tag{70}$$

The gradient of the potential is:

$$\nabla\Omega = \begin{bmatrix}\Omega_x\\ \Omega_y\\ \Omega_z\end{bmatrix} = \begin{bmatrix} x - \dfrac{(1-\mu)}{r_1^3}(x+\mu) - \dfrac{\mu}{r_2^3}(x+\mu-1)\\ y - \dfrac{(1-\mu)}{r_1^3}y - \dfrac{\mu}{r_2^3}y\\ -\dfrac{(1-\mu)}{r_1^3}z - \dfrac{\mu}{r_2^3}z \end{bmatrix} \tag{71}$$

4.2 Jacobi constant gradient

The gradient of the Jacobi constant can be written as:

$$\nabla J = \begin{bmatrix} 2x + \dfrac{(2\mu+2x)(\mu-1)}{\left[(\mu+x)^2+y^2+z^2\right]^{3/2}} - \dfrac{\mu(2\mu+2x-2)}{\left[(\mu+x-1)^2+y^2+z^2\right]^{3/2}}\\ 2y - \dfrac{2\mu y}{\left[(\mu+x-1)^2+y^2+z^2\right]^{3/2}} + \dfrac{2y(\mu-1)}{\left[(\mu+x)^2+y^2+z^2\right]^{3/2}}\\ \dfrac{2z(\mu-1)}{\left[(\mu+x)^2+y^2+z^2\right]^{3/2}} - \dfrac{2\mu z}{\left[(\mu+x-1)^2+y^2+z^2\right]^{3/2}}\\ -2v_x\\ -2v_y\\ -2v_z \end{bmatrix} \tag{72}$$

4.3 PBRFBP dynamics

The PBRFBP adimensional dynamics for the Earth-Moon in the rotating EMB centered frame (the same frame of the CRTBP) is:

$$\begin{bmatrix}\dot{\mathbf{r}}\\ \dot{\mathbf{v}}\end{bmatrix} = \begin{bmatrix} \dot{x}\\ \dot{y}\\ 2v_y + \Omega_{4,x}\\ -2v_x + \Omega_{4,y} \end{bmatrix} \tag{73}$$

Where the potential Ω_4 is the the PBRFBP potential written as:

$$\Omega_4 = \Omega_3 + \frac{m_s}{r_3} - \frac{m_2}{\rho^2}(x\cos(\omega_s t) + y\sin(\omega_s t)) \tag{74}$$

Where Ω_3 is the potential of the CRTBP and r_3 is the distance of the S/C from the Sun:

$$r_3 = \sqrt{(x - \rho\cos(\omega_s t))^2 + (y - \rho sin(\omega_s t))^2} \tag{75}$$

The dynamics of the STM for the PBRFBP is given by the variational equations

$$\begin{bmatrix} \dot{\mathbf{x}}(t) \\ \dot{\mathbf{\Phi}}(t_0, t) \end{bmatrix} = \begin{cases} \boldsymbol{f}_{FBP}(\mathbf{x}, t) \\ \boldsymbol{A}(t)\mathbf{\Phi}(t_0, t) \end{cases} \qquad \begin{cases} \mathbf{x_0}(t_0) = \mathbf{x}_0 \\ \Phi(t_0, t_0) = \boldsymbol{I}_{4\times 4} \end{cases}$$

The dynamics is coupled because the evaluation of the matrix $\boldsymbol{A(t)}$ is defined on the reference trajectory descrbied by the PBRFBP dynamics (first row). In fact:

$$\boldsymbol{A}(t) = \frac{\partial \boldsymbol{f}}{\partial \mathbf{x}}\bigg|_{x^*} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \Omega_{4,xx} & \Omega_{4,xy} & 0 & 2 \\ \Omega_{4,yx} & \Omega_{4,yy} & -2 & 0 \end{bmatrix}$$

Where the bottom-left 4×4 square sub matrix is the Hessian of the PBRFBP potential, calculated with the symbolic *Matlab* toolbox.

4.4 $\Delta\theta$ plot

The plot of $\Delta\theta$ has been represented over the time window from `2024 SEP 27 23:58:50.817 UTC` to `2024 OCT 27 12:40:25.938 UTC` (the initial epoch that was given in TBD has been converted to UTC). The time span is approximately 30 days, which is the period rotation of the Sun in the rotating frame of the PBRFBP of the Earth-Moon system.

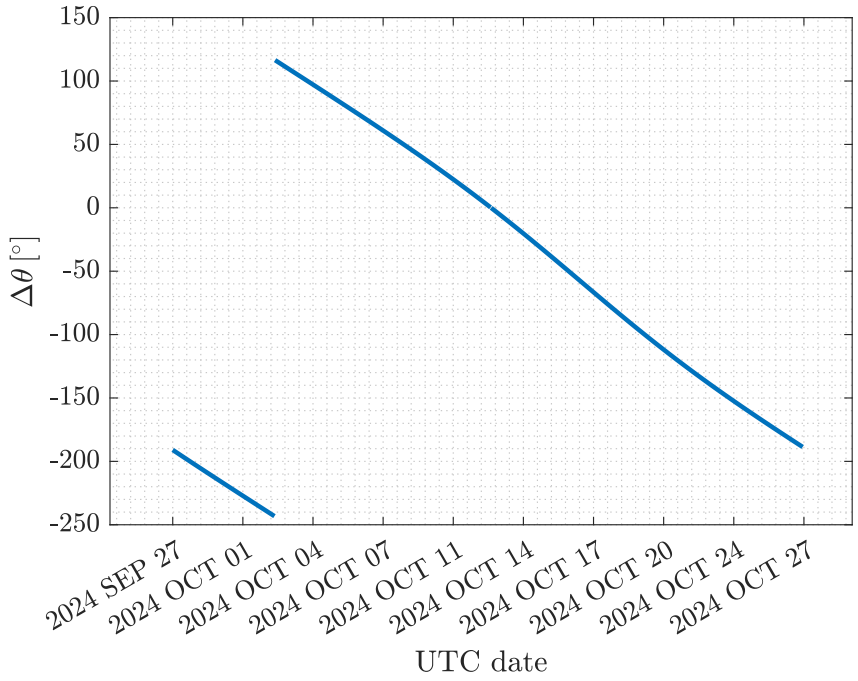

Figure 24: $\Delta\theta$ time profile

After this plot, due to the discontinuity at around "2024 OCT 2", the time window for the research the zero has been shrunk in a more reasonable time window, in particular from `2024 OCT 06 15:18:50.817 UTC` to `2024 OCT 23 23:58:50.817 UTC`.

4.5 N-body propagation equation

This section presents the equations of motions of a S/C in a n-body environment, considering as integration center the Earth. The vectors are expressed in the ECLIPJ2000 frame.

$$\ddot{\mathbf{r}} = -GM_E \frac{\mathbf{r}}{||\mathbf{r}||^3} - \sum_i^{N_{th,bod}} GM_i \left(\frac{\mathbf{d}_i}{||\mathbf{d}_i||^3} + \frac{\boldsymbol{\rho_i}}{||\boldsymbol{\rho_i}||^3} \right) \tag{76}$$

Where $\mathbf{r}$ is the distance of the S/C from the Earth in the ECLIPJ2000 frame, GM_E is the GM constant for the Earth. On the other hand, the other accelerations, coming from the other bodies are accounted in the sum term. In particular $\boldsymbol{\rho}_i$ is the distance of the i-th body from the Earth center expressed in the ECLIPJ2000 frame while $\mathbf{d}_i$ is the distance of the S/C from the i-th body expressed in the ECLIPJ2000 frame. The implementation of Equation 76 in *Matlab* can suffer numerical instabilities hence the actual form which is used in the numerical integration is:

$$\ddot{\mathbf{r}} = -GM_E \frac{\mathbf{r}}{||\mathbf{r}||^3} - \sum_i^{N_{th,bod}} GM_i \frac{1}{||\mathbf{d}_i||^3} \left(\mathbf{r} + \boldsymbol{\rho}_i f(q_i) \right) \tag{77}$$

With

$$f(q_i) = \frac{q_i(3 + 3q_i + q_i^2)}{1 + (1 + q_i)^{3/2}}$$

$$q_i = \frac{\mathbf{r} \cdot (\mathbf{r} - 2\boldsymbol{\rho}_i)}{\boldsymbol{\rho}_i \cdot \boldsymbol{\rho}_i}$$

The n-bodies considered, apart from the Earth are listed in order as: Moon, Sun, Venus, Mars Barycenter, Mercury, Jupiter Barycenter, Saturn Barycenter, Uranus barycenter, Neptune barycenter, Pluto barycenter.

References

[1] *Matlab-Choose an ODE solver*. URL: `https://it.mathworks.com/help/matlab/math/choose-an-ode-solver.html#bu3n5rf-1`.

[2] F. Topputo. *On optimal two-impulse Earth–Moon transfers in a four-body model*. URL: `https://doi.org/10.1007/s10569-013-9513-8`.

[3] T. Yanao K.Oshima F. Topputo. *Low-energy transfers to the Moon with long transfer time*. URL: `https://doi.org/10.1007/s10569-019-9883-7`.