ASEN 6060 - Final Project

Giovanni Fereoli

Tuesday 7th May, 2024; 16:18

Dynamical systems theory for libration point mission trajectory design

Motivation Following the success of NASA's ARTEMIS mission, the European Space Agency (ESA) aims to better understand how the strong solar wind at the moon's distance shapes the Earth's magnetosphere and how the moon's tiny magnetic field interacts with the solar wind. ESA's new mission will use simultaneous measurements of particles, electric and magnetic fields from two locations to provide insight into how energetic particle acceleration occurs near the moon's orbit, in the distant magnetosphere. Two spacecraft on opposite sides of the Moon will measure the Moon's magnetic field to determine its regional influence on solar wind particles. For ESA's measurement objectives on both sides of the Moon, deploying two spacecraft orbiting the L_1 and L_2 Lagrange points emerges as an optimal solution. Ideally, a bounded motion around these Lagrangian points would entail Lissajous orbits. However, for simplicity's sake, this preliminary analysis solely considers a planar variant known as a Lyapunov orbit. Both spacecraft embark on their journey from Earth, traverse to a Lyapunov orbit at L_1 or L_2, contingent upon the mission phase design, and then one of them transitions to the Lyapunov orbit at the other Lagrangian point. Consequently, this initial analysis focuses solely on transfers from L_1 to L_2. As the spacecraft are likely to be CubeSats, these transfers entail minimal specific requirements beyond fuel optimization. However, no optimization in this regard is conducted at this stage.

Dynamical Model The CR3BP[1] models the trajectory of a spacecraft, considered to have negligible mass, under the gravitational influence of the Earth and the Moon, which have masses M_1 and M_2 respectively. These bodies orbit their common center of mass in circular paths. The motion is analyzed in a rotating reference frame centered at this barycenter with axes $\{\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}\}$, where $\hat{\mathbf{x}}$ points from Earth towards the Moon, $\hat{\mathbf{z}}$ aligns with the orbital angular momentum of the system, and $\hat{\mathbf{y}}$ forms a right-handed coordinate system. Dimensionless variables are used, scaled by characteristic length (l^*), mass (m^*), and time (t^*) units, where l^* is the fixed distance between Earth and Moon, m^* is the combined mass of Earth and Moon, and t^* sets the non-dimensional orbital period at 2π. The spacecraft state in this

[1] Ross, Shane & Koon, Wang & Lo, Martin & Marsden, Jerrold. (2022). Dynamical Systems, the Three-Body Problem, and Space Mission Design.

frame is expressed as $\mathbf{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T$, leading to the corresponding equations of motion:

$$\begin{aligned}
\ddot{x} &= 2\dot{y} + x - (1-\mu)\frac{x+\mu}{r_1^3} - \mu\frac{x+\mu-1}{r_2^3} \\
\ddot{y} &= -2\dot{x} + y - (1-\mu)\frac{y}{r_1^3} - \mu\frac{y}{r_2^3} \\
\ddot{z} &= (1-\mu)\frac{z}{r_1^3} - \mu\frac{z}{r_2^3}
\end{aligned} \tag{1}$$

$$\mathbf{r}_1 = [x+\mu,\ y,\ z] \tag{2}$$

$$\mathbf{r}_2 = [x+\mu-1,\ y,\ z] \tag{3}$$

The vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ represent the locations of the spacecraft P_3 relative to the primary bodies P_1 and P_2. The mass ratio $\mu = M_2/(M_2+M_2)$ governs the dynamics completely and, for the Earth-Moon system, μ is 0.012150584269542. Moreover, this autonomous dynamical system possesses a conserved quantity, known as the Jacobi constant, which is:

$$C_J = \left(x^2+y^2\right) + \frac{2\,(1-\mu)}{r_1} + \frac{2\mu}{r_2} - \left(\dot{x}^2+\dot{y}^2+\dot{z}^2\right) \tag{4}$$

This quantity offers insights into allowable regions of motion, as well as heuristics for maneuver and trajectory design, which will be utilized later in this preliminary design.

The integration scheme selected for each propagation is **ode113**, which is a variable-step variable-order Adams-Bashforth-Moulton solver with orders ranging from 1 to 13. The chosen tolerances are $RelTol = 2.22045 \cdot 10^{-14}$ and $AbsTol = 2.22045 \cdot 10^{-16}$.

Orbit Selection As outlined in the Motivations section, two Lyapunov orbits are chosen around L_1 and L_2 to position two spacecraft capable of measuring the local electromagnetic field at two distinct locations. The selection of these specific Lyapunov orbits adheres to project requirements with respect to ΔC_J, aiming to minimize it while still meeting the necessary criteria of $\Delta C_J > 0.01$. This approach ensures orbits with energy levels as close as possible to minimize propellant usage. Using the numerical framework developed throughout the course, these orbits have been generated. By linearizing the equations of motion of the Earth-Moon system's Circular Restricted Three-Body Problem (CR3BP) around the L_1 and L_2 Lagrangian points, the initial deviations from the equilibrium state are obtained by removing the non-oscillatory modes. Once this initial guess is acquired, it can be corrected using a **Generalized Variable-Time Single Shooting** method. This method transforms Two-Point Boundary Value Problems (TPBVPs) into Initial Value Problems (IVPs), adjusting the initial condition $\mathbf{x}_0$ of the trajectory so that after a specified integration period P, a designated constraint vector $\mathbf{F}$ (that is, the enforcement of state continuity) converges to zero within a prescribed tolerance tol (set at $tol = 10^{-10}$ for this analysis). This iterative process, known as Newton's method, incrementally improves the initial conditions for a periodic orbit, represented as a free variable vector in the single shooting framework $\mathbf{V} = [\mathbf{x}_0,\ P]^T$.Following this, the Natural Parameter Continuation method is utilized as a subsequent approach to generate the entire set of L_1 and L_2 Lyapunov orbits. Specifically, in this context, the orbits are continued by varying the parameter that represents

the x-coordinate. From this family, the desired orbits are selected for the subsequent step of this trajectory optimization project. The continuation scheme follows a similar iterative approach, where orbits are corrected, continued on the basis of a chosen natural parameter, and then corrected again iteratively.

The initial conditions for the selected L_1 Lyapunov orbit in the rotating frame are as follows:

$$\mathbf{x}_{0,L_1} = \begin{bmatrix} 0.821950426219030 \\ 0 \\ 0 \\ 0 \\ 0.141479662833491 \\ 0 \end{bmatrix} \tag{5}$$

The selected orbit has an orbital period of $P_{L_1} = 2.757108054159905$, and a Jacobi constant of $C_{J,L_1} = 3.170724284915385$. Similarly, the initial conditions for the selected L_2 Lyapunov orbit in the rotating frame are as follows:

$$\mathbf{x}_{0,L_2} = \begin{bmatrix} 1.175773196736922 \\ 0 \\ 0 \\ 0 \\ -0.119977116007445 \\ 0 \end{bmatrix} \tag{6}$$

Here, the orbital period is $P_{L_2} = 3.396688765837098$, and the Jacobi constant C_{J,L_2} is $C_{J,L_2} = 3.160514921065930$. These trajectories are further depicted in Figure 1, accompanied by the Zero-Velocity-Curve[2] (ZVC) relative to the lowest Jacobi constant dynamical structure.

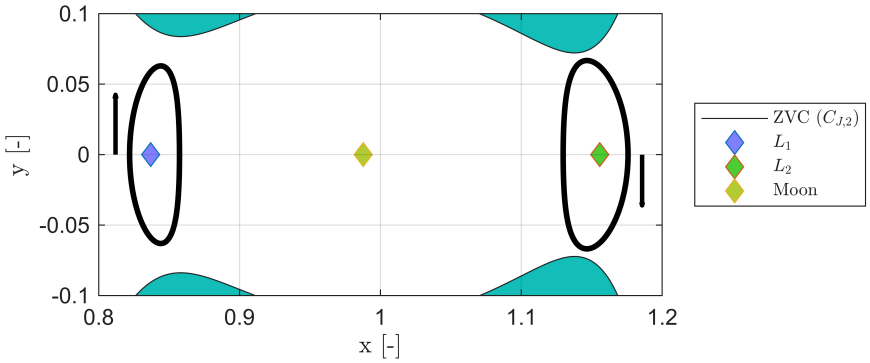

Figure 1: Lyapunov orbits around the L_1 and L_2 Lagrangian points in the Earth-Moon system are under consideration for transfer design.

[2]The Zero-Velocity-Curve (ZVC) is a set of points in the three-dimensional space that divide the phase space into regions with real or imaginary velocity magnitudes. The ZVC indicates the areas where the spacecraft does not have enough energy to fly, according to its initial condition. This forbidden region bounds the spacecraft's motion, as it would have a velocity magnitude $v \in \mathbb{C}$, which is not feasible.

Initial Guess Construction Crafting initial guesses for spacecraft trajectories in cislunar space and multi-body systems poses a challenge, necessitating the development of a systematic, swift, and robust approach. The complexity of the solution space and the quality of the initial guess play a crucial role in determining the feasibility of finding a solution. One method employed for initial guess construction in multi-body systems is based on Poincaré mapping. Initially, fundamental solutions are computed from low-fidelity models like the CR3BP. These solutions are then manually analyzed using Poincaré maps[3], illustrating their intersections with a hyperplane. In an autonomous differential equation system like $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$, a Poincaré map represents the continuous trajectory $\mathbf{x}(\mathbf{x}_0,\ t)$ resulting from the integration of an initial condition $\mathbf{x}_0$. This map transforms the trajectory into a sequence of discrete states, which typically occur at intersections with a designated section surface denoted as Σ, which intersects the flow. Each intersection, identified as the i^{th} return to the map, is labeled $P^i(\mathbf{x}_0)$.

In this scenario, to derive an initial transfer guess between L_1 and L_2 Lyapunov orbits, their respective invariant hyperbolic manifolds are utilized. Within dynamical systems, trajectories along stable manifolds gradually converge towards a periodic orbit in forward time, while those following unstable manifolds converge towards the orbit in backward time. Approximating stable or unstable half-manifolds involves a numerical computation. Initially, a periodic orbit is discretized into states. In each state x_{PO}, a magnitude perturbation d is introduced along the direction of a stable (or unstable) eigenvector $\mathbf{v}_{s/u}$ from the corresponding monodromy matrix, normalized by the magnitudes of its components. Subsequently, the perturbed state $\mathbf{x}_{PO} \pm d\mathbf{v}_{s/u}$ propagates backward (or forward) in time to trace a trajectory along the stable (or unstable) half-manifold. This numerical process iterates for selected states along the periodic orbit, providing a discrete approximation of the desired global half-manifold within a specified time interval. After considering these factors, let's delve into how the initial guesses for the transfers were actually obtained. Beginning with the initial conditions of the Lyapunov orbits provided in the previous section, two Poincaré maps are examined: for transfers from L_1 to L_2 (referred to as 1A and 1B), the map corresponds to L_1's unstable manifold and L_2's stable manifold, whereas for transfers from L_2 to L_1 (designated as 2A and 2B), it pertains to L_1's stable manifold and L_2's unstable manifold. Both cases exclusively consider manifolds directed towards the Moon. The initial conditions for the manifolds involve 600 states in the periodic orbits, with a step in the unstable and stable eigenspaces of $d = 2.601456815816858 \cdot 10^{-7}$. The integration period for the manifolds is selected to ensure up to two positive crossings of the surface $\Sigma : x = 1 - \mu$, thus obtaining the one-sided Poincarè maps concerning $\dot{x} > 0$, illustrating $(y, \dot{y})$ coordinates. Restricting the choice to a maximum of two crossings (indicating a specific number of moon revolutions) and filtering events with $\dot{x} > 0$ (encompassing transfers with a manifold connection in $y < 0$) only captures a portion of potential transfer guesses. However, at this point in the investigation, this simplification is deemed acceptable. The initial conditions for stable and unstable manifolds are integrated for a non-dimensional time period of $P = 10$, employing a MATLAB `event` function configured to record crossings. Once the number of crossings exceeds two during integration throughout the period P, it is capped at this upper threshold.

[3]Whittington, Paige Alana (2022). Multi-Body Trajectory Design in the Earth-Moon Region Utilizing Poincare Maps. Purdue University Graduate School. Thesis.

The resulting maps are shown in Figures 2 and 3, providing adequate resolution for a comprehensive analysis. In this investigation, the departure and arrival orbits exhibit distinct C_J, resulting in varying energy levels across the states relative to the different manifolds within the Poincarè maps. Upon analyzing these maps, one can observe where the closed curves created by the manifolds intersect. The intersection points are useful for formulating initial guesses for maneuver-enabled transfers. At these points, where the manifolds meet at the same spatial coordinates, impulsive maneuvers can be applied to align the velocity coordinates. Such maneuvers are crucial for connecting trajectories between orbits that do not completely align in phase space, yet have the same spatial coordinates. In particular, these intersection points are not typical heteroclinic connections, since there is no clear trajectory in the state space that connects the beginning and end orbits. The variation in energy levels causes the states to differ only in the $\dot{x}$-coordinate at these intersection points.

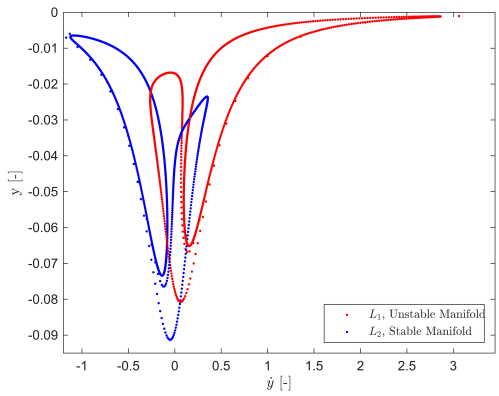

Figure 2: Poincarè map depicting the intersection of the unstable manifold of the L_1 Lyapunov orbit and the stable manifold of the L_2 Lyapunov orbit with the Σ plane.

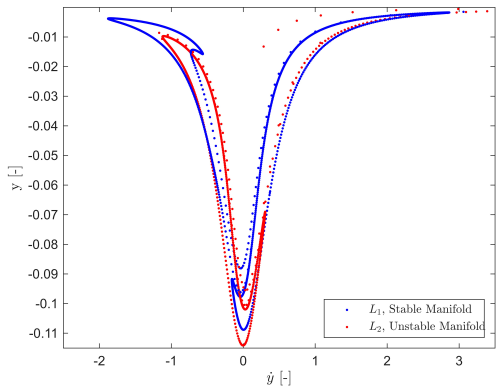

Figure 3: Poincarè map depicting the intersection of the unstable manifold of the L_2 Lyapunov orbit and the stable manifold of the L_1 Lyapunov orbit with the Σ plane.

After generating these maps, the intersection points could have been visually located. However, due to the complexity of these maps and to enhance the quality of the initial guesses, a dedicated algorithm is employed to identify intersections among the event states across the different manifolds. Throughout the map creation process, the initial conditions and the event states of the manifolds are documented. The algorithm analyzes each event state within the Poincarè map of the different manifolds, precisely measuring their distances to pinpoint overlaps (with a Euclidean distance of 10^{-3} signaling a detection). Once an intersection is detected, all pertinent information useful for initial guess generation is displayed on the terminal. This method improves the effectiveness of intersection detection and management. Several intersections were detected in the previous Poincarè maps, with two intersections from the first map and an additional two from the second map chosen as initial guesses for the four transfers required. Figures 4, 5, 6, and 7 represent these initial transfer guesses in the rotating frame. Each trajectory shown features blue and red arcs representing the two distinct half-manifolds, while the thick black lines depict the segments of the trajectory the spacecraft would traverse on the departure and final orbits to reach the designated positions set as boundary conditions for the transfer from or to the initial condition given above for the respective Lyapunov orbit.

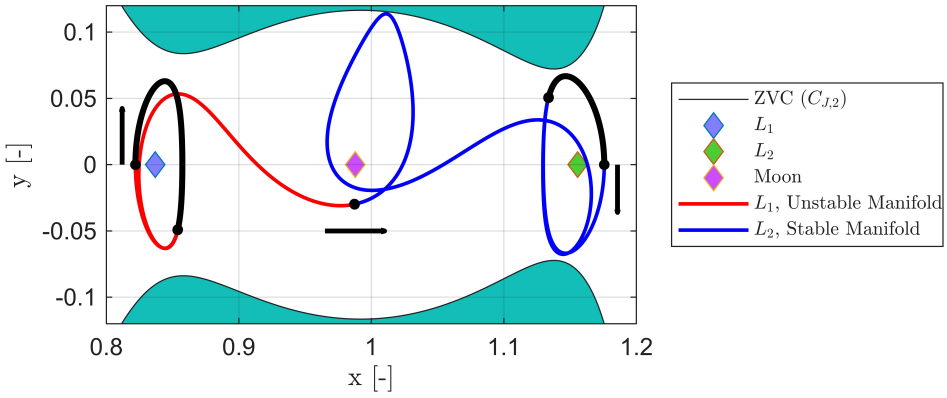

Figure 4: Initial guess for the correction of the transfer 1A.

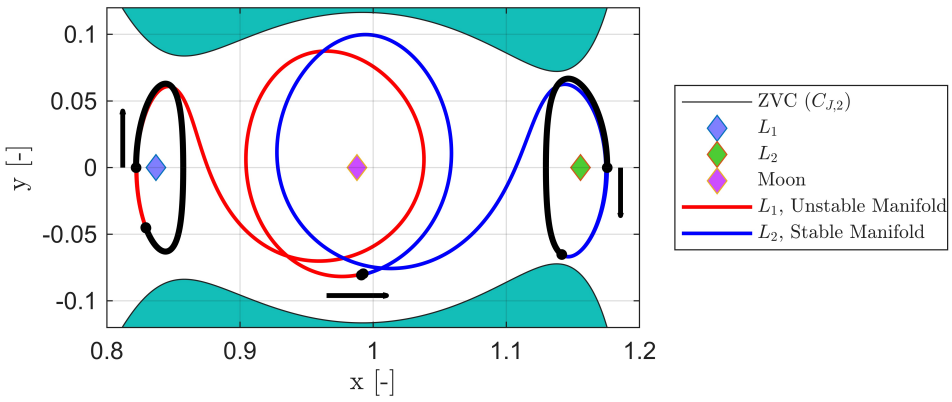

Figure 5: Initial guess for the correction of the transfer 1B.

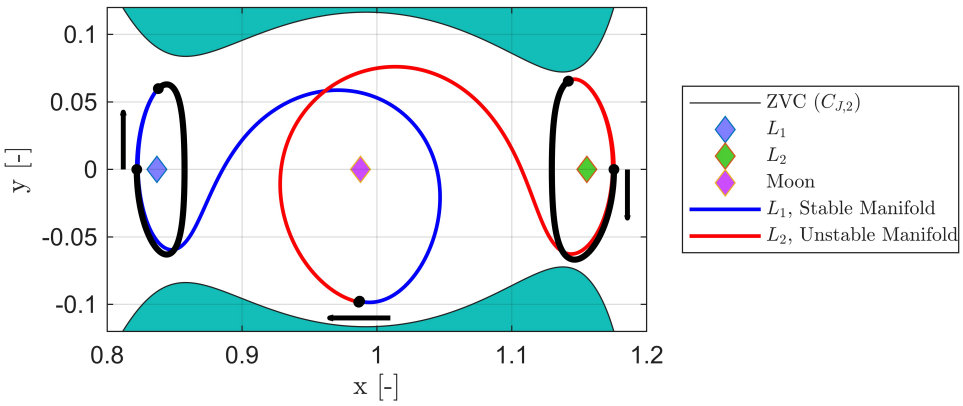

Figure 6: Initial guess for the correction of the transfer 2A.

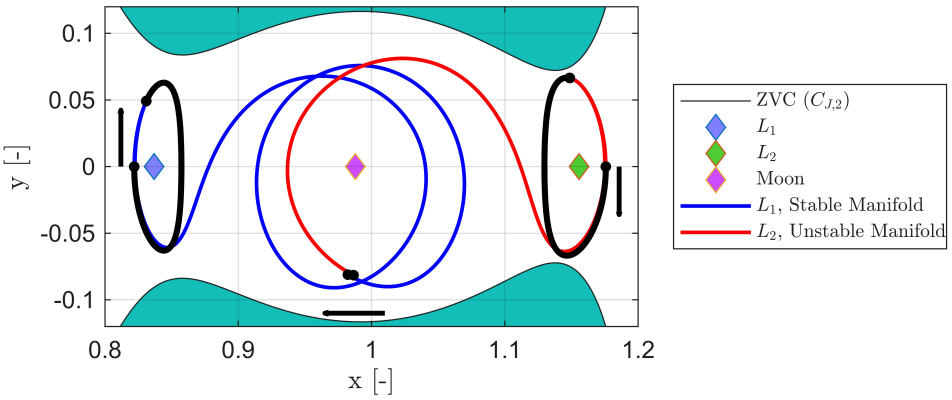

Figure 7: Initial guess for the correction of the transfer 2B.

The respective times of flight are: $ToF_{1A} = 54.5134\ days$, $ToF_{1B} = 62.6973\ days$, $ToF_{2A} = 55.3488\ days$, and $ToF_{1A} = 62.0165\ days$. The time of flight appears to exceed the typical durations found in the literature[4]. However, this variance can be attributed to the parameters utilized in the computation of the manifolds. Specifically, a very small step d from the central eigenspace to the stable and unstable ones. Consequently, the manifolds complete a few revolutions around the periodic orbits before departing or arriving at them.

As mentioned earlier, the plots above provide a visual representation of the initial transfer guesses. They depict the numerically computed segments of the manifolds that serve as the basis for the refinement of the trajectory. In fact, the actual trajectory guess used for the correction only relates to the manifold trajectories; initial and final conditions on the departure and arrival orbits act as boundary constraints.

Initially, a simpler approach with only two free nodes (i.e., three boundary nodes) and two arcs was considered to combine the manifold trajectories while adhering to the boundary constraints. However, this approach proved to be sensitive to the given initial conditions and lacked robustness during the analysis. Therefore, the decision was made to employ four free nodes (i.e., five boundary nodes) and four arcs for better reliability. Each manifold has been cut into two segments. Each node's constraints focus solely on the positional elements of the state vector to enhance flexibility, granting each node impulsive maneuver capability. The nodes comprise: a first free node at the beginning of the first segment of the unstable manifold, which also establishes a constraint on the departure orbit and aids in formulating a boundary condition; a second free node positioned within the unstable manifold at $P_{u,m}/2$, serving as the initial condition for its second segment; a third free node located at the beginning of the first segment of the stable manifold, which will be restricted to the end of the unstable manifold; and a fourth free node positioned midway through the stable manifold at $P_{s,m}/2$, initiating its second segment. In its conclusion, this final segment imposes the boundary condition on the arrival orbit. This formulation demonstrates significant flexibility and the ability to correct a broad spectrum of initial guesses, as additional nodes improve the distribution of the correction error more effectively.

[4]Smith, T. R., & Bosanac, N. (2023). Motion Primitive Approach to Spacecraft Trajectory Design in a Multi-body System. The Journal of the Astronautical Sciences, Vol. 70, Issue 5.

Multiple-Shooting Formulation Once initial transfer guesses are acquired, a multiple shooting correction scheme can refine these guesses to yield the final transfer trajectory. Shooting methods[5] solve Two-Point Boundary Value Problems (TPBVPs) by iteratively adjusting a vector of free variables $\mathbf{V}$ until a constraint vector $\mathbf{F}$ converges to zero within a prescribed tolerance *tol* (in this case, *tol* is set to 10^{-10}). A multiple shooting formulation divides the overall solution interval into several smaller intervals to significantly enhance the distribution of non-linearity and improve numerical stability compared to single shooting methods. As mentioned above, each manifold segment in this analysis has been split into two segments, resulting in a total of four segments and four free nodes. The vector of free variables encodes the state of each free node along with the propagation time required for each arc. Therefore, it is defined as $\mathbf{V} = \begin{bmatrix}\mathbf{x}_{0,1},\ P_1,\ \mathbf{x}_{0,2},\ P_2,\ \mathbf{x}_{0,3},\ P_3,\ \mathbf{x}_{0,4},\ P_4\end{bmatrix}^T \in \mathbb{R}^{n=28}$. It is important to note that each period P_i represents the propagation time of each arc's initial condition $\mathbf{x}_{0,i}$. To obtain the total time of flight, all periods should be summed.

In order to calculate a continuous trajectory, it is necessary to ensure that a specific set of continuity constraints is met at the boundary nodes. The vector of free variables must be adjusted to satisfy the constraint vector:

$$\mathbf{F}(\mathbf{V}) = \begin{bmatrix} \mathbf{r}_{0,1} - \mathbf{r}_{0,des} \\ \mathbf{r}_{f,1} - \mathbf{r}_{0,2} \\ \mathbf{r}_{f,2} - \mathbf{r}_{0,3} \\ \mathbf{r}_{f,3} - \mathbf{r}_{0,4} \\ \mathbf{r}_{f,4} - \mathbf{r}_{f,des} \end{bmatrix} \in \mathbb{R}^{m=15} \tag{7}$$

where $\mathbf{r}_{f,i}$ are the position components of the state vector $\mathbf{x}_{0,i}$ integrated for the duration of P_i. The constraint vector consists of the boundary conditions in the position of the spacecraft, with the first and last components representing the desired position of the spacecraft to correctly depart from the departure orbit at $\mathbf{r}_{0,des}$ and inject it into the arrival orbit at $\mathbf{r}_{f,des}$. The remaining three components, $\mathbf{r}_{f,i+1} - \mathbf{r}_{0,i}$, ensure the continuity of the transfer arcs at the remaining boundary nodes. Each boundary node serves as a location for an impulsive maneuver that inherently restricts only the position. This characteristic enhances the flexibility and robustness of corrections. The iterative updating process of the free variable vector utilizes Newton's method. Essentially, this method addresses a root-finding problem that aims to find a solution $\mathbf{V}^*$ such that $\mathbf{F}(\mathbf{V}^*) \approx \mathbf{0}$. The update equation, derived from a first-order Taylor expansion and setting the left-hand side to zero, is expressed as $\mathbf{V}_{k+1} = \mathbf{V}_k - \mathbf{DF}^+(\mathbf{V}_k)\,\mathbf{F}(\mathbf{V}_k)$, where $\mathbf{DF}^+(\mathbf{V}_k)$ denotes the pseudo-inverse (Moore-Penrose inverse, providing a minimum-norm solution) of $\mathbf{DF}(\mathbf{V}_k)$. Given that the free variable vector encompasses more components than the constraint vector, $\mathbf{DF}$ forms a matrix with more columns than rows, resulting in an infinite number of solutions for the linear system. The matrix $\mathbf{DF}$, sized $m \times n$, encompasses the partial derivatives of each constraint concerning each free variable and is defined as follows:

$$\mathbf{DF} = \begin{bmatrix} \begin{bmatrix}\mathbf{I}_{3x3} & \mathbf{0}_{3x3}\end{bmatrix} & \mathbf{0}_{3x1} & \mathbf{0}_{3x6} & \mathbf{0}_{3x1} & \mathbf{0}_{3x6} & \mathbf{0}_{3x1} & \mathbf{0}_{3x6} & \mathbf{0}_{3x1} \\ \mathbf{\Phi}_1(P_1,\ 0)_{:3,:} & \dot{\mathbf{r}}_{f,1} & \begin{bmatrix}-\mathbf{I}_{3x3} & \mathbf{0}_{3x3}\end{bmatrix} & \mathbf{0}_{3x1} & \mathbf{0}_{3x6} & \mathbf{0}_{3x1} & \mathbf{0}_{3x6} & \mathbf{0}_{3x1} \\ \mathbf{0}_{3x6} & \mathbf{0}_{3x1} & \mathbf{\Phi}_2(P_2,\ 0)_{:3,:} & \dot{\mathbf{r}}_{f,2} & \begin{bmatrix}-\mathbf{I}_{3x3} & \mathbf{0}_{3x3}\end{bmatrix} & \mathbf{0}_{3x1} & \mathbf{0}_{3x6} & \mathbf{0}_{3x1} \\ \mathbf{0}_{3x6} & \mathbf{0}_{3x1} & \mathbf{0}_{3x6} & \mathbf{0}_{3x1} & \mathbf{\Phi}_3(P_3,\ 0)_{:3,:} & \dot{\mathbf{r}}_{f,3} & \begin{bmatrix}-\mathbf{I}_{3x3} & \mathbf{0}_{3x3}\end{bmatrix} & \mathbf{0}_{3x1} \\ \mathbf{0}_{3x6} & \mathbf{0}_{3x1} & \mathbf{0}_{3x6} & \mathbf{0}_{3x1} & \mathbf{0}_{3x6} & \mathbf{0}_{3x1} & \mathbf{\Phi}_4(P_4,\ 0)_{:3,:} & \dot{\mathbf{r}}_{f,4} \end{bmatrix} \tag{8}$$

[5]Topputo, Francesco. (2013). On optimal two-impulse Earth–Moon transfers in a four-body model. Celestial Mechanics and Dynamical Astronomy. 117.

The State Transition Matrix (STM), denoted as $\mathbf{\Phi}_i$, embodies the derivative of the state vector $\mathbf{x}_i$ with respect to its initial conditions $\mathbf{x}_{0,i}$, formulated as $\mathbf{\Phi}_i\left(t,\ t_0\right) = \partial \mathbf{x}_i\left(t\right) / \partial \mathbf{x}_i\left(t_0\right)$, and evolves according to $\dot{\mathbf{\Phi}}_i\left(t,\ t_0\right) = \mathbf{A}_i\left(t\right) \mathbf{\Phi}_i\left(t,\ t_0\right)$. Here, $\mathbf{A}_i\left(t\right) = \partial \mathbf{f}\left(\mathbf{x}_i\right) / \partial \mathbf{x}\left(t\right)$ signifies the Jacobian matrix of the CR3BP equations of motion, obtained using variational equations. The initial conditions of the STM are consistently established as $\mathbf{\Phi}\left(t_0,\ t_0\right) = \mathbf{I}_{6x6}$ for each arc. Furthermore, STMs are marked with the subscript $(:3,:)$ to signify their restriction to rows pertaining exclusively to positional components. The elements $\dot{\mathbf{r}}_{f,i}$ denote the velocity component $\mathbf{v}_{f,i}$ of the state vectors $\mathbf{x}_{f,i}$. However, they are expressed as the time derivative of position to provide clearer insight into their origin when examining the constraint vector and the free variable vector.

The algorithm iterations end when the correction scheme converges, indicated by the norm of the constraint vector $\mathbf{F}$ dropping below a specified threshold tol, set at 10^{-10} in this instance, or when divergence occurs (using a threshold of 10^3), triggering the generation of an error message. The anticipated theoretical quadratic convergence of Newton's method will be validated in the following section. As mentioned previously, the transfer has been segmented into four arcs, resulting in four free nodes and five boundary nodes. This division aims to distribute the nodal error more evenly along the trajectory, leading to a more robust and numerically efficient correction scheme. Without adequate segmentation, the problem could potentially become numerically ill-conditioned, resulting in a magnitude of $|\mathbf{DF}\ \mathbf{DF}^T| \approx 10^{14}$, for instance. This segmentation significantly enhances the algorithm's robustness against imperfect initial guesses. It is important to reiterate that each arc's boundary condition, which confines only positional components, also signifies an impulsive maneuver.

Transfer Results and Analysis Utilizing the aforementioned algorithm and the initial trajectory guesses outlined in the previous section, all corrected transfers are shown in the xy-plane of the rotating frame in Figures 8, 9, 10, and 11. Few numerical information on these transfers, taking into account the total required impulse and flight duration, are documented in Table 1. To gain a deeper understanding of the correction process, Figures 12, 13, 14, and 15 illustrate the norms of vector constraints at each correction step for each transfer. The effectiveness of the algorithm and the accuracy of the initial guesses are validated by reducing $|\mathbf{F}_k|$ below the predetermined threshold, which typically occurs in approximately 5 iterations. Moreover, the correction strategy appears to be accurately implemented, as evidenced by the visual alignment of the figures with theoretical expectations, demonstrating the expected quadratic convergence pattern of Newton's method. The graphs show an almost monotonic decreasing trend in the norm of the constraint vector, providing further validation of the implementation accuracy.

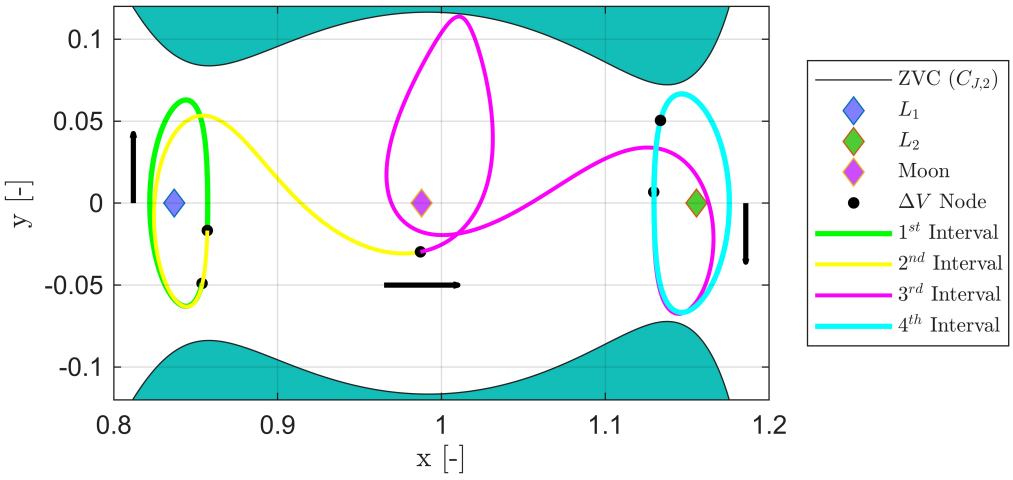

Figure 8: Transfer 1A corrected with the multiple shooting algorithm.

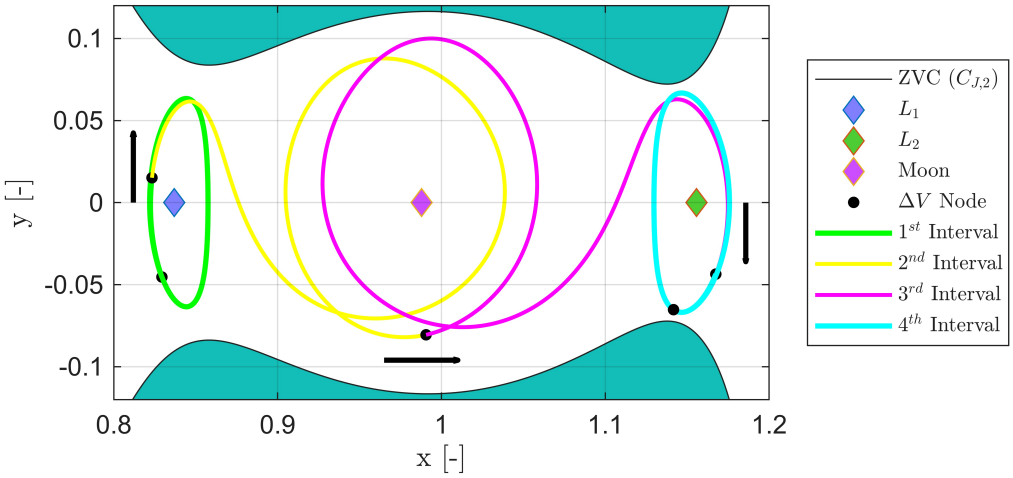

Figure 9: Transfer 1B corrected with the multiple shooting algorithm.

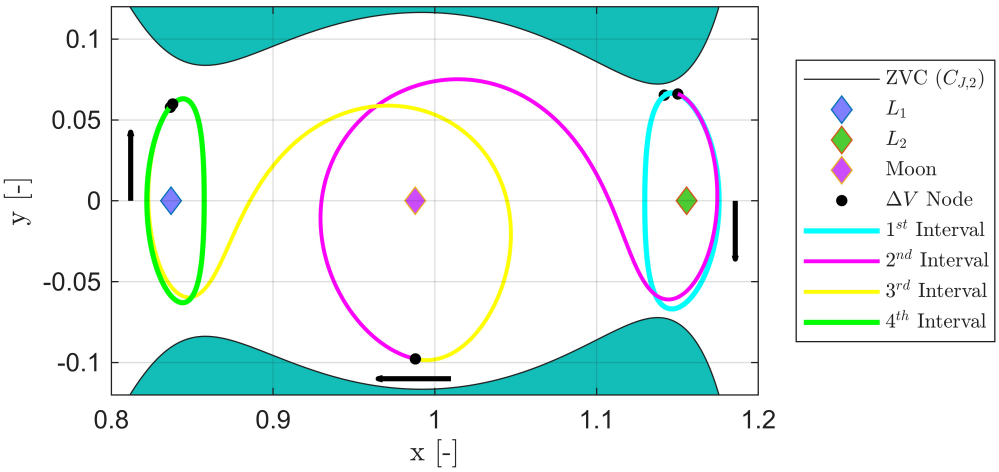

Figure 10: Transfer 2A corrected with the multiple shooting algorithm.

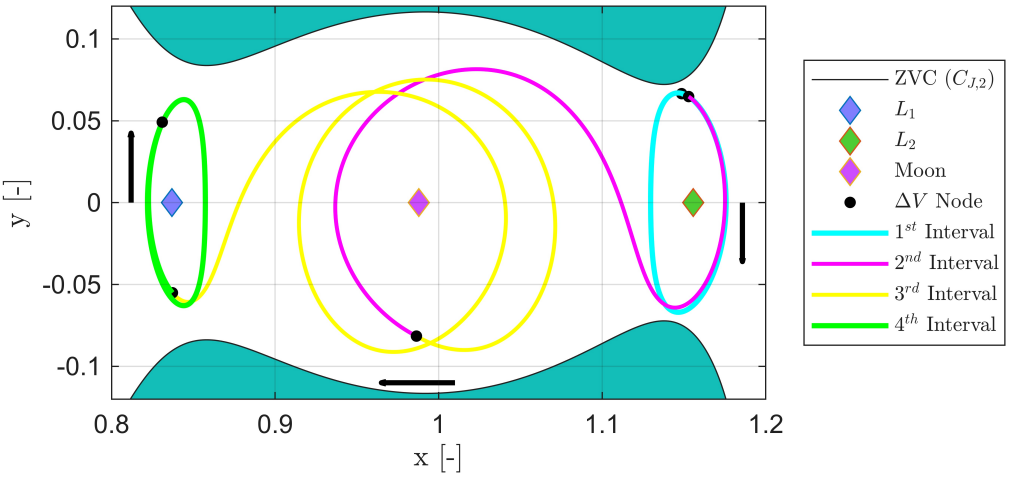

Figure 11: Transfer 2B corrected with the multiple shooting algorithm.

Table 1: Results of the corrected transfers.

Transfer	ΔV [m/s]	ToF [days]
1A	216.9912	54.5115
1B	175.6904	62.6981
2A	188.1754	55.3488
2B	146.4061	62.0165

The corrected transfers are virtually visually indistinguishable from their initial guesses, as evidenced by the plots and results, particularly the time-of-flights, in the table. As a result, they were not plotted together as they would simply overlap. This emphasizes the accuracy of the initial guesses made.

Transfers 1A and 1B involve spacecraft transitioning from a L_1 Lyapunov orbit to a L_2 Lyapunov orbit. In Transfer 1A, the spacecraft completes one counter-clockwise revolution around the moon after entering the stable manifold of the L_2 Lyapunov orbit. In contrast, in Transfer 1B, the spacecraft completes two revolutions: one along the unstable manifold of the L_1 Lyapunov orbit and one along the stable manifold of the L_2 Lyapunov orbit. Similar scenarios occur for Transfers 2A and 2B, which involve the spacecraft transitioning from the L_2 Lyapunov orbit to the L_1 Lyapunov orbit. In Transfer 2A, the spacecraft completes a clockwise revolution, evenly split between the unstable and stable manifolds. Transfer 2B involves two clockwise revolutions entirely within the stable manifold of the L_1 Lyapunov orbit. In all cases, the transfers smoothly depart and inject into the departure and arrival orbits, following their trajectories for a few revolutions and seamlessly aligning with them.

According to Table 1, all four transfers exhibit remarkably similar maneuver ΔV requirements and flight times. Each transfer entails approximately $\Delta V \approx 175\ m/s$ and $ToF \approx 57.5\ days$. Interestingly, transfers involving two revolutions around the Moon, regardless of whether they depart or arrive at a L_1 or L_2 Lyapunov orbit, demonstrate lower fuel consumption, although with slightly longer flight times. However, for the mission scenario highlighted at the beginning of this preliminary analysis, all transfers have fuel requirements and flight time that have been found to be reasonable and feasible.

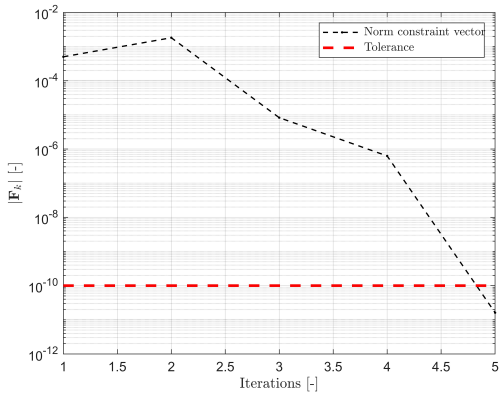

Figure 12: Transfer 1A correction with the multiple shooting algorithm.

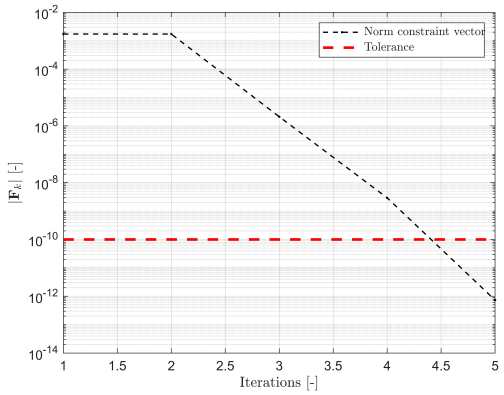

Figure 13: Transfer 1B correction with the multiple shooting algorithm.

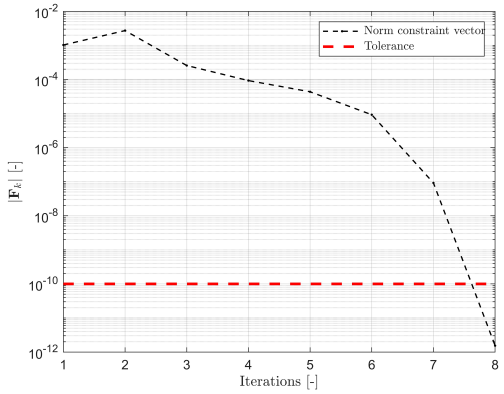

Figure 14: Transfer 2A correction with the multiple shooting algorithm.

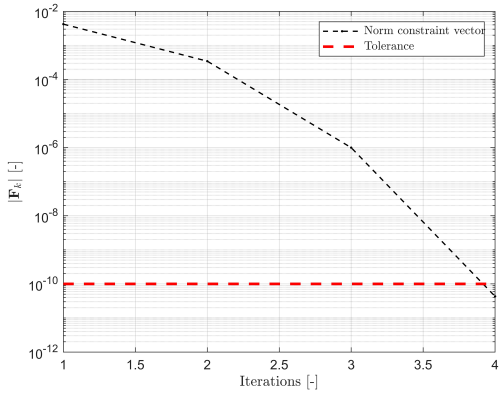

Figure 15: Transfer 2B correction with the multiple shooting algorithm.

Recommendation for Future Analysis To enhance the current analysis, several suggestions can be made for future work in this direction. Firstly, it is advisable to formulate the problem as a proper Optimal Control Problem (OCP), aiming to minimize propellant usage through a well-defined objective function in addition to the constraints outlined in this study. Solving this OCP as a Non-Linear Programming (NLP) problem using more advanced Direct methods like Transcription and Collocation could yield more robust solutions that also minimize propellant consumption. In addition, there may be instances where additional constraints need to be incorporated into the solution, such as exclusion and inclusion cones, among others. These constraints are highly mission-specific and would require further analysis.

However, before proceeding with the OCP approach, it is crucial to conduct a more thorough analysis of the solution space concerning the initial guesses, possibly employing different techniques. To enhance the completeness of the analysis while maintaining the current Poincarè mapping approach, several strategies can be explored. These include increasing the number of points, considering maps triggered when $\dot{x} < 0$, exploring trajectories with more revolutions around the Moon, etc.

In terms of dynamical accuracy, the solution could be enhanced by integrating the Sun's fourth-body effect using a Bicircular Restricted Four-Body Problem (BRFBP) model, along with incorporating Solar Radiation Pressure (SRP) into the equations of motion. This approach would lead to more realistic results.

Finally, as discussed in the Motivation section, alternative orbits around L_1 and L_2 may be advantageous. Other departure and arrival orbits, such as the Lissajous one, may provide greater stability and thus improve efficiency in terms of station-keeping fuel consumption. As a result, this research could be expanded to investigate initial and final orbits beyond the Lyapunov ones and compare various options, taking into account not only this specific transfer but also the overall mission requirements.

Bibliography

1. Ross, Shane & Koon, Wang & Lo, Martin & Marsden, Jerrold. (2022). Dynamical Systems, the Three-Body Problem, and Space Mission Design.

2. Whittington, Paige Alana (2022). Multi-Body Trajectory Design in the Earth-Moon Region Utilizing Poincare Maps. Purdue University Graduate School. Thesis.

3. Smith, T. R., & Bosanac, N. (2023). Motion Primitive Approach to Spacecraft Trajectory Design in a Multi-body System. The Journal of the Astronautical Sciences, Vol. 70, Issue 5.

4. Topputo, Francesco. (2013). On optimal two-impulse Earth–Moon transfers in a four-body model. Celestial Mechanics and Dynamical Astronomy. 117.

Project Script: Transfer 1A and 1B

```
%% Advanced Astrodynamics (ASEN6060) - Project, First Half
% Instructor:   Prof. Bosanac
% Student:      Giovanni Fereoli
% Student ID:   111040314

%% Data
clear; clc; close;

% Data
GM_earth = 398600.435436; % km^3/s^2
GM_moon = 4902.800066;
GM_sun = 132712440041.93938;
G_tilde = 6.67408*1e-20; % km^3/kg*s^2
l_star = 384400;  % km
t_star = 3.751902619518436 * 1e5; % s

% Computations mass ratios
mu_se = GM_earth / (GM_sun + GM_earth); % Mass Ratio
mu_em = GM_moon / (GM_earth + GM_moon);

%% Plot Lyapunov Orbits
clc; close;

% Initial conditions
[xL1, xL2, xL3, xL4, xL5] = EquilibriumPoints(mu_em);
x0_1 = [0.821950426219030, 0, 0, 0, 0.141479662833491, 0];
x0_2 = [1.175773196736922, 0, 0,  0, -0.119977116007445, 0];
P_1 = 2.757108054159905;
P_2 = 3.396688765837098;

% Correction scheme
tol = 1e-10;
[x0_1, P_1, normF_iter_1] = general_corrector(x0_1',...
    P_1, mu_em, tol);
[x0_2, P_2, normF_iter_2] = general_corrector(x0_2',...
    P_2, mu_em, tol);

% Check Jacobi Constant
C_1 = JacobiConstant(x0_1, mu_em);
C_2 = JacobiConstant(x0_2, mu_em);

% Plot orbits
options = odeset('RelTol', 2.24*1e-14, 'AbsTol', 2.24*1e-15);
```

```
[~, xx1] = ode113(@(t,X) CRTBP(t, X, mu_em), [0 P_1], x0_1,
   options);
[~, xx2] = ode113(@(t,X) CRTBP(t, X, mu_em), [0 P_2], x0_2,
   options);

%Plot trajectory
gca = figure(1);
ZVCxy(mu_em, C_2);
hold on;
plot3(xL1(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.5 0.5 1]);
plot3(xL2(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.3 0.8 0.2]);
plot3(1-mu_em, 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.7 0.8 0.2]);
plot3(xx1(:,1), xx1(:,2), xx1(:,3),'k','LineWidth', 2.5);
plot3(xx2(:,1), xx2(:,2), xx2(:,3),'k','LineWidth', 2.5);
quiver3(xx1(1,1) - 0.01, xx1(1,2), xx1(1,3),...
    xx1(1,4) / 3, xx1(1,5) / 3, xx1(1,6) / 3,...
    'k', 'LineWidth', 2);
quiver3(xx2(1,1) + 0.01, xx2(1,2), xx2(1,3),...
    xx2(1,4) / 3, xx2(1,5) / 3, xx2(1,6) / 3,...
    'k', 'LineWidth', 2);
xlabel('x [-]', 'interpreter','latex');
ylabel('y [-]', 'interpreter','latex');
zlabel('z [-]', 'interpreter','latex');
grid on;
axis equal;
xlim([0.8, 1.2]);
ylim([-0.1, 0.1]);
legend('ZVC ($C_{J,2}$)$', '$L_1$', '$L_2$', 'Moon', '
   interpreter', 'latex',...
    'Location', 'eastoutside', 'FontSize', 8);
view(2);

% Export graphics
exportgraphics(gca, 'Project1_orbits.pdf', 'ContentType','image
   ',...
    'Resolution', 1000);

%% Plot Manifolds
clc; close;

%Initialization
options = odeset('RelTol', 2.24*1e-14, 'AbsTol', 2.24*1e-15);
```

```
l_star_em = 384400;
d = 0.1 / l_star_em;
disc = 600;
T_man_1 = 5.5;
T_man_2 = 7;

% Generate initial conditions manifolds
[~, X0_man_unstab_pos_1] = ic_manifolds_pos(x0_1, P_1, d, disc,
   ...
    mu_em);
[X0_man_stab_neg_2, ~] = ic_manifolds_neg(x0_2, P_2, d, disc,
   ...
    mu_em);

% Integration with STM
[~, xx_unstab_pos_1] = ode113(@(t,xM) STMexact_CRTBP(t, xM,
   mu_em), ...
    [0 T_man_1], [X0_man_unstab_pos_1(:, 1); reshape(eye(6)
       ,[],1)], options);
[~, xx_stab_neg_2] = ode113(@(t,xM) STMexact_CRTBP(t, xM, mu_em
   ), ...
    [0 -T_man_2], [X0_man_stab_neg_2(:, 1); reshape(eye(6)
       ,[],1)], options);

%Plot trajectory
gca2 = figure(2);
plot3(xx_stab_neg_2(:,1), xx_stab_neg_2(:,2),xx_stab_neg_2(:,3)
   ,'b',...
    'LineWidth', 0.5);
hold on;
plot3(xx_unstab_pos_1(:,1), xx_unstab_pos_1(:,2),
   xx_unstab_pos_1(:,3),'r',...
    'LineWidth', 0.5);
plot3(xL1(1), 0, 0, 'd', 'MarkerSize', 6, 'MarkerFaceColor',
   [0.5 0.5 1]);
plot3(xL2(1), 0, 0, 'd', 'MarkerSize', 6, 'MarkerFaceColor',
   [0.2 0.5 1]);
plot3(1-mu_em, 0, 0, 'd', 'MarkerSize', 6, 'MarkerFaceColor',
   [0.8 0.3 1]);
for i=2:disc
    % Integration with STM - positive manifolds
    [~, xx_unstab_pos_1] = ode113(@(t,xM) STMexact_CRTBP(t, xM,
       mu_em),...
        [0 T_man_1], [X0_man_unstab_pos_1(:, i); reshape(eye(6)
           ,[],1)], options);
```

```
    [~, xx_stab_neg_2] = ode113(@(t,xM) STMexact_CRTBP(t, xM,
       mu_em),...
        [0 -T_man_2], [X0_man_stab_neg_2(:, i); reshape(eye(6)
           ,[],1)], options);

    % Plot
    plot3(xx_stab_neg_2(:,1), xx_stab_neg_2(:,2),xx_stab_neg_2
       (:,3),'b',...
        'LineWidth', 0.5);
    plot3(xx_unstab_pos_1(:,1), xx_unstab_pos_1(:,2),
       xx_unstab_pos_1(:,3),'r',...
        'LineWidth', 0.5);
    disp(i);
end
plot3(xx1(:,1), xx1(:,2), xx1(:,3),'k','LineWidth', 2.5);
plot3(xx2(:,1), xx2(:,2), xx2(:,3),'k','LineWidth', 2.5);
plot3(xL1(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.5 0.5 1]);
plot3(xL2(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.3 0.8 0.2]);
plot3(1-mu_em, 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.8 0.3 1]);
xlabel('x [-]', 'interpreter','latex');
ylabel('y [-]', 'interpreter','latex');
zlabel('z [-]', 'interpreter','latex');
grid on;
axis equal;
view(2);
legend('Stable Manifold', 'Unstable Manifold',...
    '$L_1$', '$L_2$', 'Moon', 'interpreter', 'latex', 'Location
       ',...
    'eastoutside', 'FontSize', 8);

% Export graphics
exportgraphics(gca2, 'Project1_manifolds.pdf', 'ContentType','
   image',...
    'Resolution', 1000);

%% Plot Poincare' map
clc; close;

% Generate initial conditions manifolds
options_Moon_for = odeset('RelTol', 2.24*1e-14, 'AbsTol',
   2.24*1e-15,...
    'Events', @event_Moon_for);
```

```
options_Moon_back = odeset('RelTol', 2.24*1e-14, 'AbsTol',
   2.24*1e-15,...
    'Events', @event_Moon_back);
T_man = 10;

% Initialize to record crossings
num_crossing = zeros(2, disc-1);

% Initialize cell arrays to store variables for each i
xe1_cell = cell(disc-1, 1);
xe2_cell = cell(disc-1, 1);
te1_cell = cell(disc-1, 1);
te2_cell = cell(disc-1, 1);
ie1_cell = cell(disc-1, 1);
ie2_cell = cell(disc-1, 1);

% Plot Poincare'
gca3 = figure(3);
for i=2:disc
   % Integration with STM - positive manifolds
   [~, xx_unstab_pos_1, te1, xe1, ie1] = ode113(@(t, X) CRTBP(t
      , X, mu_em),...
       [0 T_man], X0_man_unstab_pos_1(:, i), options_Moon_for);
   [~, xx_stab_neg_2, te2, xe2, ie2] = ode113(@(t, X) CRTBP(t,
      X, mu_em),...
       [0 -T_man], X0_man_stab_neg_2(:, i), options_Moon_back);

   % Retain only the first two elements
   te1 = te1(1:min(2, length(ie1)));
   xe1 = xe1(1:min(2, length(ie1)),:);
   ie1 = ie1(1:min(2, length(ie1)));
   te2 = te2(1:min(2, length(ie2)));
   xe2 = xe2(1:min(2, length(ie2)),:);
   ie2 = ie2(1:min(2, length(ie2)));

   % Record number of crossings
   num_crossing(1, i-1) = length(ie1);
   num_crossing(2, i-1) = length(ie2);

   % Save variables for each i
   xe1_cell{i-1} = xe1;
   xe2_cell{i-1} = xe2;
   te1_cell{i-1} = te1;
   te2_cell{i-1} = te2;
   ie1_cell{i-1} = ie1;
```

```
    ie2_cell{i-1} = ie2;

    % Plot
    plot(xe1(:,5), xe1(:, 2), '.r', 'MarkerSize', 6);
    hold on;
    plot(xe2(:,5), xe2(:,2), '.b', 'MarkerSize', 6);
    disp(i);
end
xlabel('$\dot{y}$ [-]', 'interpreter','latex');
ylabel('y [-]', 'interpreter','latex');
xlim([-1.2, 3.45]);
ylim([-0.095, 0]);
legend('$L_1$, Unstable Manifold', '$L_2$, Stable Manifold', '
    interpreter', 'latex',...
     'Location', 'southeast');

% Export graphics
exportgraphics(gca3, 'Project1_Poincare.pdf', 'ContentType','
    image',...
     'Resolution', 1000);

%% Find Crossings Poincare'

% Initialize
intersection_Poincare = cell(disc-1, disc-1, 4);
intersection_ICs = cell(disc-1, disc-1, 2);
threshold = 1e-3;
counter = 0;

% Fill crossings matrix
for i = 1:disc-1
    for j = 1:disc-1
        for k = 1:size(xe1_cell{i}, 1)
            for w = 1:size(xe2_cell{j}, 1)
                if norm(xe1_cell{i}(k, [2,5]) - xe2_cell{j}(w,
                    [2,5])) < threshold
                    intersection_Poincare{i, j, 1} = xe1_cell{i
                        }(k, :);
                    intersection_Poincare{i, j, 2} = xe2_cell{j
                        }(w, :);
                    intersection_Poincare{i, j, 3} = te1_cell{i
                        }(k);
                    intersection_Poincare{i, j, 4} = te2_cell{j
                        }(w);
                    counter = counter +1;
```

```
                    disp(counter);
                end
            end
        end
    end
end

% Print not-zero elements
for i = 1:size(intersection_Poincare, 1)
    for j = 1:size(intersection_Poincare, 2)
        % Check if the current element is not empty
        if ~isempty(intersection_Poincare{i, j, 1})
            % Print the indices of non-empty elements
            disp(['Non-empty element found at index (', ...
                num2str(i), ',' ...
                '', num2str(j), ')']);
            intersection_ICs{i, j, 1} = X0_man_unstab_pos_1(:, i)';
            intersection_ICs{i, j, 2} = X0_man_stab_neg_2(:, j)';
            intersection_ICs{i, j, 3} = P_1 * (i-1) / disc;
            intersection_ICs{i, j, 4} = P_2 * (j-1) / disc;
        end
    end
end

%% First Transfer

% Initialize Poincare' Intersection
row = 383; col = 379;
% row = 523; col = 176;

% Initial guesses
x0_L1 = x0_1;
P_L1 = intersection_ICs{row, col, 3};
x0_man_L1 = intersection_ICs{row, col, 1};
P_man_L1 = intersection_Poincare{row, col, 3};
x0_L2 = x0_2;
P_L2 = intersection_ICs{row, col, 4};
x0_man_L2 = intersection_ICs{row, col, 2};
P_man_L2 = intersection_Poincare{row, col, 4};
disc = 10000;

% Plot orbits
options = odeset('RelTol', 2.24*1e-14, 'AbsTol', 2.24*1e-15);
```

```
[~, xx1] = ode113(@(t,X) CRTBP(t, X, mu_em), linspace(0, P_L1,
   disc), x0_L1, options);
[~, xx2] = ode113(@(t,X) CRTBP(t, X, mu_em), linspace(0,
   P_man_L1, disc), x0_man_L1, options);
[~, xx3] = ode113(@(t,X) CRTBP(t, X, mu_em), linspace(0,
   P_man_L2, disc), x0_man_L2, options);
[~, xx4] = ode113(@(t,X) CRTBP(t, X, mu_em), linspace(0, (P_L2-
   P_2), disc), x0_L2, options);
fprintf('The transfer guess needs a ToF of %s [days].\n',...
    num2str((abs(P_man_L1) + abs(P_man_L2)) * t_star / 86400));

%Plot trajectory
gca4 = figure(4);
ZVCxy(mu_em, C_2);
hold on;
plot3(xL1(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.5 0.5 1]);
plot3(xL2(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.3 0.8 0.2]);
plot3(1-mu_em, 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.8 0.3 1]);
plot3(xx2(:,1), xx2(:,2), xx2(:,3),'r','LineWidth', 1.5);
plot3(xx3(:,1), xx3(:,2), xx3(:,3),'b','LineWidth', 1.5);
plot3(xx4(:,1), xx4(:,2), xx4(:,3),'k','LineWidth', 2.5);
plot3(xx1(:,1), xx1(:,2), xx1(:,3),'k','LineWidth', 2.5);
plot3(xx1(1,1), xx1(1,2), xx1(1,3), '.k', 'MarkerSize', 15);
plot3(xx1(end,1), xx1(end,2), xx1(end,3), '.k', 'MarkerSize',
   15);
plot3(xx2(1,1), xx2(1,2), xx2(1,3), '.k', 'MarkerSize', 15);
plot3(xx2(end,1), xx2(end,2), xx2(end,3), '.k', 'MarkerSize',
   15);
plot3(xx3(1,1), xx3(1,2), xx3(1,3), '.k', 'MarkerSize', 15);
plot3(xx3(end,1), xx3(end,2), xx3(end,3), '.k', 'MarkerSize',
   15);
plot3(xx4(1,1), xx4(1,2), xx4(1,3), '.k', 'MarkerSize', 15);
plot3(xx4(end,1), xx4(end,2), xx4(end,3), '.k', 'MarkerSize',
   15);
quiver3(xx1(1,1) - 0.01, xx1(1,2), xx1(1,3),...
    xx1(1,4) / 3, xx1(1,5) / 3, xx1(1,6) / 3,...
    'k', 'LineWidth', 2);
quiver3(xx4(1,1) + 0.01, xx4(1,2), xx4(1,3),...
    xx4(1,4) / 3, xx4(1,5) / 3, xx4(1,6) / 3,...
    'k', 'LineWidth', 2);
quiver3(0.965, -0.05, 0,...
    0.05, 0,  0,...
```

```
    'k', 'LineWidth', 2);
xlabel('x [-]', 'interpreter','latex');
ylabel('y [-]', 'interpreter','latex');
zlabel('z [-]', 'interpreter','latex');
grid on;
axis equal;
xlim([0.8, 1.2]);
ylim([-0.12, 0.12]);
legend('ZVC ($C_{J,2})$', '$L_1$', '$L_2$', 'Moon',...
    '$L_1$, Unstable Manifold', '$L_2$, Stable Manifold',...
    'interpreter', 'latex', 'Location', 'eastoutside', '
      FontSize', 8);
view(2);

% Reverse few integrations due stable manifold for NEXT section
x0_man_L2 = xx3(end, :);
P_man_L2 = - P_man_L2;
x0_L2 = xx4(end, :); % extra: P_L2 = (P_2-P_L2);
xxf_des = xx4;
xx0_des = xx1;

% Export graphics
exportgraphics(gca4, 'Project1_FirstTransferGuess.pdf', '
   ContentType','image',...
    'Resolution', 1000);

%% First Transfer Correction

% Initialization and initial guess correction
% OSS: the second manifold has been 'reversed'!
V_guess = [x0_man_L1, P_man_L1/2, xx2(end/2,:), P_man_L1/2,...
    x0_man_L2, P_man_L2/2,  xx3(end/2, :), P_man_L2/2];
X0_des = xx1(end, :)';
Xf_des = x0_L2';
tol = 1e-10;

% Correction
[V_corr, normF_iter] = transfer_ms(V_guess, X0_des, Xf_des,
   mu_em, tol);

% Integration orbits
options = odeset('RelTol', 2.24*1e-14, 'AbsTol', 2.24*1e-15);
[~, xx1_corr] = ode113(@(t,X) CRTBP(t, X, mu_em), [0 V_corr(7)
   ], V_corr(1:6), options);
[~, xx2_corr] = ode113(@(t,X) CRTBP(t, X, mu_em), [0 V_corr(14)
```

```
   ], V_corr(8:13), options);
[~, xx3_corr] = ode113(@(t,X) CRTBP(t, X, mu_em), [0 V_corr(21)
   ], V_corr(15:20), options);
[~, xx4_corr] = ode113(@(t,X) CRTBP(t, X, mu_em), [0 V_corr(28)
   ], V_corr(22:27), options);

% Plot orbits
gca5 = figure(5);
ZVCxy(mu_em, C_2);
hold on;
plot3(xL1(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.5 0.5 1]);
plot3(xL2(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.3 0.8 0.2]);
plot3(1-mu_em, 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.8 0.3 1]);
plot3(xx1_corr(1,1), xx1_corr(1,2), xx1_corr(1,3), '.k', '
   MarkerSize', 15);
plot3(xx1_corr(:,1), xx1_corr(:,2), xx1_corr(:,3),'g','
   LineWidth', 2);
plot3(xx1_corr(end,1), xx1_corr(end,2), xx1_corr(end,3), '.k',
   'MarkerSize', 15);
plot3(xx2_corr(1,1), xx2_corr(1,2), xx2_corr(1,3), '.k', '
   MarkerSize', 15);
plot3(xx2_corr(:,1), xx2_corr(:,2), xx2_corr(:,3),'y','
   LineWidth', 1.5);
plot3(xx2_corr(end,1), xx2_corr(end,2), xx2_corr(end,3), '.k',
   'MarkerSize', 15);
plot3(xx3_corr(1,1), xx3_corr(1,2), xx3_corr(1,3), '.k', '
   MarkerSize', 15);
plot3(xx3_corr(:,1), xx3_corr(:,2), xx3_corr(:,3),'m','
   LineWidth', 1.5);
plot3(xx3_corr(end,1), xx3_corr(end,2), xx3_corr(end,3), '.k',
   'MarkerSize', 15);
plot3(xx4_corr(1,1), xx4_corr(1,2), xx4_corr(1,3), '.k', '
   MarkerSize', 15);
plot3(xx4_corr(:,1), xx4_corr(:,2), xx4_corr(:,3),'c','
   LineWidth', 2);
plot3(xx4_corr(end,1), xx4_corr(end,2), xx4_corr(end,3), '.k',
   'MarkerSize', 15);
quiver3(xx1(1,1) - 0.01, xx1(1,2), xx1(1,3),...
    xx1(1,4) / 3, xx1(1,5) / 3, xx1(1,6) / 3,...
    'k', 'LineWidth', 2);
quiver3(xx4(1,1) + 0.01, xx4(1,2), xx4(1,3),...
    xx4(1,4) / 3, xx4(1,5) / 3, xx4(1,6) / 3,...
```

```
    'k', 'LineWidth', 2);
quiver3(0.965, -0.05, 0,...
    0.05, 0,  0,...
    'k', 'LineWidth', 2);
xlabel('x [-]', 'interpreter','latex');
ylabel('y [-]', 'interpreter','latex');
zlabel('z [-]', 'interpreter','latex');
grid on;
axis equal;
xlim([0.8, 1.2]);
ylim([-0.12, 0.12]);
legend('ZVC ($C_{J,2})$', '$L_1$', '$L_2$', 'Moon', '$\Delta V$
    Node', '$1^{st}$ Interval',...
    '', '', '$2^{nd}$ Interval', '', '', '$3^{rd}$ Interval', '
       ', '', '$4^{th}$ Interval',...
    'interpreter', 'latex', 'Location', 'eastoutside', '
       FontSize', 8);
view(2);

% Export graphics
exportgraphics(gca5, 'Project1_FirstTransferCorrected.pdf', '
   ContentType','image',...
    'Resolution', 1000);

% Plot correction performance
iter = 1:length(normF_iter);
gca6 = figure(6);
semilogy(iter, normF_iter, 'k.--', 'LineWidth', 1);
hold on;
semilogy(iter, tol * ones(length(normF_iter), 1), 'r--',...
    'LineWidth', 2);
xlabel('Iterations [-]', 'interpreter','latex');
ylabel('$|\mathbf{F}_k|$ [-]', 'interpreter','latex');
grid on;
real axis;
xlim([1, length(normF_iter)]);
ylim([0.1 * normF_iter(end), normF_iter(1)]);
legend('Norm constraint vector', 'Tolerance', 'Interpreter', '
   latex');

% Export graphics
exportgraphics(gca6, 'Project1_FirstTransferCorrection.pdf', '
   ContentType','image',...
    'Resolution', 1000);
```

```
% After plotting you have the flows, compute TCMs and ToF
TCM1 = xx1_corr(1, 4:6) - xx0_des(end, 4:6);
TCM2 = xx2_corr(1, 4:6) - xx1_corr(end, 4:6);
TCM3 = xx3_corr(1, 4:6) - xx2_corr(end, 4:6);
TCM4 = xx3_corr(end, 4:6) - xx4_corr(1, 4:6);
TCM5 = xx4_corr(end, 4:6) - xxf_des(1, 4:6);
TCMtot_norm_A = norm(TCM1) + norm(TCM2) + norm(TCM3) + norm(
   TCM4) + norm(TCM5);
ToF_A = abs(V_corr(7)) + abs(V_corr(14)) + abs(V_corr(21)) +
   abs(V_corr(28));
fprintf('The transfer needs a DV of %s [m/s].\n',...
    num2str(TCMtot_norm_A * l_star / t_star * 1000));
fprintf('The transfer needs a ToF of %s [days].\n',...
    num2str(ToF_A * t_star / 86400));

%% Second Transfer

% Initialization
% row = 383; col = 379;
row = 523; col = 176;

% Initial guesses
x0_L1 = x0_1;
P_L1 = intersection_ICs{row, col, 3};
x0_man_L1 = intersection_ICs{row, col, 1};
P_man_L1 = intersection_Poincare{row, col, 3};
x0_L2 = x0_2;
P_L2 = intersection_ICs{row, col, 4};
x0_man_L2 = intersection_ICs{row, col, 2};
P_man_L2 = intersection_Poincare{row, col, 4};
disc = 10000;

% Plot orbits
options = odeset('RelTol', 2.24*1e-14, 'AbsTol', 2.24*1e-15);
[~, xx1] = ode113(@(t,X) CRTBP(t, X, mu_em), linspace(0, P_L1,
   disc), x0_L1, options);
[~, xx2] = ode113(@(t,X) CRTBP(t, X, mu_em), linspace(0,
   P_man_L1, disc), x0_man_L1, options);
[~, xx3] = ode113(@(t,X) CRTBP(t, X, mu_em), linspace(0,
   P_man_L2, disc), x0_man_L2, options);
[~, xx4] = ode113(@(t,X) CRTBP(t, X, mu_em), linspace(0, (P_L2-
   P_2), disc), x0_L2, options);
fprintf('The transfer guess needs a ToF of %s [days].\n',...
    num2str((abs(P_man_L1) + abs(P_man_L2)) * t_star / 86400));
```

```
%Plot trajectory
gca7 = figure(7);
ZVCxy(mu_em, C_2);
hold on;
plot3(xL1(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.5 0.5 1]);
plot3(xL2(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.3 0.8 0.2]);
plot3(1-mu_em, 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.8 0.3 1]);
plot3(xx2(:,1), xx2(:,2), xx2(:,3),'r','LineWidth', 1.5);
plot3(xx3(:,1), xx3(:,2), xx3(:,3),'b','LineWidth', 1.5);
plot3(xx4(:,1), xx4(:,2), xx4(:,3),'k','LineWidth', 2.5);
plot3(xx1(:,1), xx1(:,2), xx1(:,3),'k','LineWidth', 2.5);
plot3(xx1(1,1), xx1(1,2), xx1(1,3), '.k', 'MarkerSize', 15);
plot3(xx1(end,1), xx1(end,2), xx1(end,3), '.k', 'MarkerSize',
   15);
plot3(xx2(1,1), xx2(1,2), xx2(1,3), '.k', 'MarkerSize', 15);
plot3(xx2(end,1), xx2(end,2), xx2(end,3), '.k', 'MarkerSize',
   15);
plot3(xx3(1,1), xx3(1,2), xx3(1,3), '.k', 'MarkerSize', 15);
plot3(xx3(end,1), xx3(end,2), xx3(end,3), '.k', 'MarkerSize',
   15);
plot3(xx4(1,1), xx4(1,2), xx4(1,3), '.k', 'MarkerSize', 15);
plot3(xx4(end,1), xx4(end,2), xx4(end,3), '.k', 'MarkerSize',
   15);
quiver3(xx1(1,1) - 0.01, xx1(1,2), xx1(1,3),...
    xx1(1,4) / 3, xx1(1,5) / 3, xx1(1,6) / 3,...
    'k', 'LineWidth', 2);
quiver3(xx4(1,1) + 0.01, xx4(1,2), xx4(1,3),...
    xx4(1,4) / 3, xx4(1,5) / 3, xx4(1,6) / 3,...
    'k', 'LineWidth', 2);
quiver3(0.965, -0.096, 0,...
    0.05, 0,  0,...
    'k', 'LineWidth', 2);
xlabel('x [-]', 'interpreter','latex');
ylabel('y [-]', 'interpreter','latex');
zlabel('z [-]', 'interpreter','latex');
grid on;
axis equal;
xlim([0.8, 1.2]);
ylim([-0.12, 0.12]);
legend('ZVC ($C_{J,2}$)$', '$L_1$', '$L_2$', 'Moon',...
    '$L_1$, Unstable Manifold', '$L_2$, Stable Manifold',...
    'interpreter', 'latex', 'Location', 'eastoutside', '
```

```
       FontSize', 8);
view(2);

% Reverse few integrations due stable manifold for NEXT section
x0_man_L2 = xx3(end, :);
P_man_L2 = - P_man_L2;
x0_L2 = xx4(end, :);
P_L2 = (P_2-P_L2);
xxf_des = xx4;
xx0_des = xx1;

% Export graphics
exportgraphics(gca7, 'Project1_SecondTransferGuess.pdf', '
   ContentType','image',...
    'Resolution', 1000);

%% Second Transfer Correction

% Initialization and initial guess correction
% OSS: the second manifold has been 'reversed'!
V_guess = [x0_man_L1, P_man_L1/2, xx2(end/2,:), P_man_L1/2,...
    x0_man_L2, P_man_L2/2,  xx3(end/2, :), P_man_L2/2];
X0_des = xx1(end, :)';
Xf_des = x0_L2';
tol = 1e-10;

% Correction
[V_corr, normF_iter] = transfer_ms(V_guess, X0_des, Xf_des,
   mu_em, tol);

% Integration orbits
options = odeset('RelTol', 2.24*1e-14, 'AbsTol', 2.24*1e-15);
[~, xx1_corr] = ode113(@(t,X) CRTBP(t, X, mu_em), [0 V_corr(7)
   ], V_corr(1:6), options);
[~, xx2_corr] = ode113(@(t,X) CRTBP(t, X, mu_em), [0 V_corr(14)
   ], V_corr(8:13), options);
[~, xx3_corr] = ode113(@(t,X) CRTBP(t, X, mu_em), [0 V_corr(21)
   ], V_corr(15:20), options);
[~, xx4_corr] = ode113(@(t,X) CRTBP(t, X, mu_em), [0 V_corr(28)
   ], V_corr(22:27), options);

% Plot orbits
gca8 = figure(8);
ZVCxy(mu_em, C_2);
hold on;
```

```
plot3(xL1(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.5 0.5 1]);
plot3(xL2(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.3 0.8 0.2]);
plot3(1-mu_em, 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.8 0.3 1]);
plot3(xx1_corr(1,1), xx1_corr(1,2), xx1_corr(1,3), '.k', '
   MarkerSize', 15);
plot3(xx1_corr(:,1), xx1_corr(:,2), xx1_corr(:,3),'g','
   LineWidth', 2);
plot3(xx1_corr(end,1), xx1_corr(end,2), xx1_corr(end,3), '.k',
   'MarkerSize', 15);
plot3(xx2_corr(1,1), xx2_corr(1,2), xx2_corr(1,3), '.k', '
   MarkerSize', 15);
plot3(xx2_corr(:,1), xx2_corr(:,2), xx2_corr(:,3),'y','
   LineWidth', 1.5);
plot3(xx2_corr(end,1), xx2_corr(end,2), xx2_corr(end,3), '.k',
   'MarkerSize', 15);
plot3(xx3_corr(1,1), xx3_corr(1,2), xx3_corr(1,3), '.k', '
   MarkerSize', 15);
plot3(xx3_corr(:,1), xx3_corr(:,2), xx3_corr(:,3),'m','
   LineWidth', 1.5);
plot3(xx3_corr(end,1), xx3_corr(end,2), xx3_corr(end,3), '.k',
   'MarkerSize', 15);
plot3(xx4_corr(1,1), xx4_corr(1,2), xx4_corr(1,3), '.k', '
   MarkerSize', 15);
plot3(xx4_corr(:,1), xx4_corr(:,2), xx4_corr(:,3),'c','
   LineWidth', 2);
plot3(xx4_corr(end,1), xx4_corr(end,2), xx4_corr(end,3), '.k',
   'MarkerSize', 15);
quiver3(xx1(1,1) - 0.01, xx1(1,2), xx1(1,3),...
    xx1(1,4) / 3, xx1(1,5) / 3, xx1(1,6) / 3,...
    'k', 'LineWidth', 2);
quiver3(xx4(1,1) + 0.01, xx4(1,2), xx4(1,3),...
    xx4(1,4) / 3, xx4(1,5) / 3, xx4(1,6) / 3,...
    'k', 'LineWidth', 2);
quiver3(0.965, -0.096, 0,...
    0.05, 0,  0,...
    'k', 'LineWidth', 2);
xlabel('x [-]', 'interpreter','latex');
ylabel('y [-]', 'interpreter','latex');
zlabel('z [-]', 'interpreter','latex');
grid on;
axis equal;
xlim([0.8, 1.2]);
```

```
ylim([-0.12, 0.12]);
legend('ZVC ($C_{J,2}$)$', '$L_1$', '$L_2$', 'Moon', '$\Delta V$
    Node', '$1^{st}$ Interval',...
    '', '', '$2^{nd}$ Interval', '', '', '$3^{rd}$ Interval', '
       ', '', '$4^{th}$ Interval',...
    'interpreter', 'latex', 'Location', 'eastoutside', '
       FontSize', 8);
view(2);

% Export graphics
exportgraphics(gca8, 'Project1_SecondTransferCorrected.pdf', '
   ContentType','image',...
    'Resolution', 1000);

% Plot correction performance
iter = 1:length(normF_iter);
gca9 = figure(9);
semilogy(iter, normF_iter, 'k.--', 'LineWidth', 1);
hold on;
semilogy(iter, tol * ones(length(normF_iter), 1), 'r--',...
    'LineWidth', 2);
xlabel('Iterations [-]', 'interpreter','latex');
ylabel('$|\mathbf{F}_k|$ [-]', 'interpreter','latex');
grid on;
real axis;
xlim([1, length(normF_iter)]);
ylim([0.1 * normF_iter(end), normF_iter(1)]);
legend('Norm constraint vector', 'Tolerance', 'Interpreter', '
   latex');

% Export graphics
exportgraphics(gca9, 'Project1_SecondTransferCorrection.pdf', '
   ContentType','image',...
    'Resolution', 1000);

% After plotting you have the flows, compute TCMs
TCM1 = xx1_corr(1, 4:6) - xx0_des(end, 4:6);
TCM2 = xx2_corr(1, 4:6) - xx1_corr(end, 4:6);
TCM3 = xx3_corr(1, 4:6) - xx2_corr(end, 4:6);
TCM4 = xx3_corr(end, 4:6) - xx4_corr(1, 4:6);
TCM5 = xx4_corr(end, 4:6) - xxf_des(1, 4:6);
TCMtot_norm_B = norm(TCM1) + norm(TCM2) + norm(TCM3) + norm(
   TCM4) + norm(TCM5);
ToF_B = abs(V_corr(7)) + abs(V_corr(14)) + abs(V_corr(21)) +
   abs(V_corr(28));
```

```
fprintf('The transfer need a DV of %s [m/s].\n',...
    num2str(TCMtot_norm_B * l_star / t_star * 1000));
fprintf('The transfer need a ToF of %s [days].\n',...
    num2str(ToF_B * t_star / 86400));

%% Functions

% Compute ZVC in the xy-plane
function ZVCxy(mu, C)
    % Initialization
    x_zvc = -1.5:0.001:1.55;
    y_zvc = -1.5:0.001:1.55;
    Z_zvc = zeros(length(y_zvc), length(x_zvc));

    % ZVS computations
    for i=1:length(x_zvc)
        for j=1:length(y_zvc)
            Z_zvc(j, i)= (x_zvc(i)^2+y_zvc(j)^2) + ...
                2*(1-mu)./sqrt((x_zvc(i)+mu)^2+y_zvc(j)^2) +
                   ...
                2*mu./sqrt((x_zvc(i)-1+mu)^2+y_zvc(j)^2);
        end
    end

    % Plot
    contourf(x_zvc, y_zvc, -Z_zvc,[-C -C]);
end

% Compute Jacobi Constant in the xy-plane
function C = JacobiConstant(X, mu)
    % Initialization
    x = X(1);
    y = X(2);
    z = X(3);
    xdot = X(4);
    ydot = X(5);
    zdot = X(6);

    % Jacobi Constant Computation
    C = (x^2+y^2) + 2*(1-mu)/sqrt((x+mu)^2+y^2+z^2) + ...
        2*mu/sqrt((x-1+mu)^2+y^2+z^2) - sqrt(xdot^2+ydot^2+zdot
           ^2)^2;
end

% CR3BP Equations of Motions
```

```
function dXdt = CRTBP(~, X, mu)
    %Initialize
    dXdt = zeros(6,1);

    x = X(1);
    y = X(2);
    z = X(3);
    xdot = X(4);
    ydot = X(5);
    zdot = X(6);

    % CRTBP dynamics
    r1_norm = sqrt((x+mu)^2+y^2+z^2);
    r2_norm = sqrt((x+mu-1)^2+y^2+z^2);

    dXdt(1:3) = [xdot; ydot; zdot];
    dXdt(4:6) = [2*ydot+x-(1-mu)*(x+mu)/r1_norm^3-mu*(x+mu-1)/
       r2_norm^3;...
         -2*xdot+y-(1-mu)*y/r1_norm^3-mu*y/r2_norm^3;...
         -(1-mu)*z/r1_norm^3-mu*z/r2_norm^3];
end

% Find equilibrium points
function [xL1, xL2, xL3, xL4, xL5] = EquilibriumPoints(mu)
    % Collinear points
    % Position primaries along x
    xxP1 = -mu;
    xxP2 = 1-mu;

    % Gradient of U* in x
    f = @(x) x-(1-mu)*(x+mu)/(abs(x+mu))^3-mu*(x+mu-1)/(abs(x+
       mu-1))^3;

    %Inital guesses
    z = (mu/3)^(1/3);
    xxL10 = xxP2 - (z-(1/3)*z^2-(1/9)*z^3+(58/81)*z^4);
    xxL20 = xxP2 + (z+(1/3)*z^2-(1/9)*z^3+(50/81)*z^4);
    xxL30 = xxP1 - (1-(7/12)*mu-(1127/20736)*mu^3-(7889/248832)
       *mu^4);

    %Zeros computation
    options = optimoptions('fsolve', 'Display', 'none', 'TolFun
       ', 1e-15);
    xL1 = [fsolve(f, xxL10, options), 0, 0];
    xL2 = [fsolve(f, xxL20, options), 0, 0];
```

```
    xL3 = [fsolve(f, xxL30, options), 0, 0];

    % Triangular points
    xL4 = [0.5 - mu, sqrt(3) / 2, 0];
    xL5 = [0.5 - mu, - sqrt(3) / 2, 0];
end

% General initial guess correction algorithm
function [X0_corr, P_corr, normF_iter] = general_corrector(X0,
   P, mu, tol)
    % Initialization
    options = odeset('RelTol', 2.24*1e-14, 'AbsTol', 2.24*1e
       -15);
    F = ones(6,1);
    V = [X0; P];
    normF_iter = zeros(100, 1);
    iter = 1;

    while norm(F) > tol
        %Integration state and STM
        [~, xx] = ode113(@(t,xM) STMexact_CRTBP(t, xM, mu), [0
           V(end)],...
        [V(1:6); reshape(eye(6), [], 1)], options);

        %Reshape final STM
        STMf=(reshape(xx(end, 7:42), 6, 6))';

        %Jacobian of F
        xMdotf = STMexact_CRTBP(V(end), xx(end, :), mu);
        DF = [STMf - eye(6), xMdotf(1:6)];

        % Update F
        F = xx(end,1:6)- xx(1,1:6);
        normF_iter(iter) = norm(F);

        % Check if norm of F exceeds tolerance
        if norm(F) > 1e3
            fprintf('Error: Norm of F diverging. \n');
            break;  % Exit the loop if norm(F) exceeds
               tolerance
        end

        % Correction
        dV = - lsqminnorm(DF, F', tol);
        V = V + dV;
```

```
        iter = iter + 1;
    end

    % Final correction
    X0_corr = V(1:6);
    P_corr = V(7);
    normF_iter = normF_iter(1:iter-1);

    % Final print
    disp('Correction terminated successfully.');
    fprintf('         Current function value: %.14f\n',
       normF_iter(end));
    fprintf('         Iterations: %d\n', iter-1);
end

% C3RBP ODEs: state and STM
function dxMdt = STMexact_CRTBP(~,xM,mu)
    %Initialize ODE
    dxMdt = zeros(42,1);

    % Unpack state
    x = xM(1);
    y = xM(2);
    z = xM(3);
    xdot = xM(4);
    ydot = xM(5);
    zdot = xM(6);

    % Unpack STM
    M = (reshape(xM(7:42),6,6))';   %From state to STM

    % CR3TBP miscellaneous
    % P3 distance from primaries
    r1_norm = sqrt((x+mu)^2+y^2+z^2);
    r2_norm = sqrt((x+mu-1)^2+y^2+z^2);

    % Variational equations
    df4dx = 1-(1-mu)/r1_norm^3+3*(1-mu)*(x+mu)^2/r1_norm^5-mu/
       r2_norm^3+...
        3*mu*(x+mu-1)^2/r2_norm^5;
    df4dy = 3*(1-mu)*(x+mu)*y/r1_norm^5+3*mu*(x+mu-1)*y/r2_norm
       ^5;
    df4dz = 3*(1-mu)*(x+mu)*z/r1_norm^5+3*mu*(x+mu-1)*z/r2_norm
       ^5;
    df5dy = 1-(1-mu)/r1_norm^3+3*(1-mu)*y^2/r1_norm^5-mu/
```

```
        r2_norm^3+...
         3*mu*y^2/r2_norm^5;
    df5dz = 3*(1-mu)*y*z/r1_norm^5+3*mu*y*z/r2_norm^5;
    df6dz = -(1-mu)/r1_norm^3+3*(1-mu)*z^2/r1_norm^5-mu/r2_norm
        ^3+...
         3*mu*z^2/r2_norm^5;

    % Jacobian
    A = [0, 0, 0, 1, 0, 0;...
         0, 0, 0, 0, 1, 0;...
         0, 0, 0, 0, 0, 1;...
         df4dx, df4dy, df4dz, 0, 2, 0;...
         df4dy, df5dy, df5dz, -2, 0, 0;...
         df4dz, df5dz, df6dz, 0, 0, 0];

    % CR3TBP dynamics
    % CR3BP dynamics: position and velocity
    dxMdt(1:3) = [xdot; ydot; zdot];
    dxMdt(4:6) = [2*ydot+x-(1-mu)*(x+mu)/r1_norm^3-mu*(x+mu-1)/
        r2_norm^3;...
         -2*xdot+y-(1-mu)*y/r1_norm^3-mu*y/r2_norm^3;...
         -(1-mu)*z/r1_norm^3-mu*z/r2_norm^3];

    % CR3BP dynamics: STM
    dMdt = A*M;
    dxMdt(7:12) = dMdt(1,1:6)';
    dxMdt(13:18) = dMdt(2,1:6)';
    dxMdt(19:24) = dMdt(3,1:6)';
    dxMdt(25:30) = dMdt(4,1:6)';
    dxMdt(31:36) = dMdt(5,1:6)';
    dxMdt(37:42) = dMdt(6,1:6)';
end

% Events Moon
function [value, isterminal, direction] = event_Moon_for(~, X,
    ~)
    % Data
    GM_earth = 398600.435436; % km^3/s^2
    GM_moon = 4902.800066;

    % Computations mass ratios
    mu_em = GM_moon / (GM_earth + GM_moon);

    % Event
    value = X(1) - (1-mu_em);    % x = 1-mu
```

```
    isterminal = 0;
    direction = 1; % all directions
end

function [value, isterminal, direction] = event_Moon_back(~, X,
    ~)
    % Data
    GM_earth = 398600.435436; % km^3/s^2
    GM_moon = 4902.800066;

    % Computations mass ratios
    mu_em = GM_moon / (GM_earth + GM_moon);

    % Event
    value = X(1) - (1-mu_em);   % x = 1-mu
    isterminal = 0;
    direction = -1; % all directions
end

% Positive half-manifold
function [X0_man_stab, X0_man_unstab] = ic_manifolds_pos(x0, P,
    d, disc, mu)
    % Initialization
    options = odeset('RelTol', 2.24*1e-14, 'AbsTol', 2.24*1e
       -15);
    [~, xx] = ode113(@(t,xM) STMexact_CRTBP(t, xM, mu), ...
        linspace(0, P, disc), [x0; reshape(eye(6),[],1)],
           options);
    X0_man_unstab = zeros(6, disc);
    X0_man_stab = zeros(6, disc);

    % Generate x0 Monodromy Matrix
    M = (reshape(xx(end, 7:42), 6, 6))';

    % Analize monodromy matrix
    [eig_vec, ~] = eig(M);

    % Generate initial directions
    v_unstab_manifold = eig_vec(:,1);  % HP: always first and
       second!
    v_stab_manifold = eig_vec(:,2);

    % Initial IC's manifolds ICs
    X0_man_unstab(:, 1)  = x0 + d * v_unstab_manifold / norm(x0
       (1:3));
```

```
    X0_man_stab(:, 1) =  x0 + d * v_stab_manifold / norm(x0
       (1:3));

    % Loop for all manifold' ICs
    for i = 2:disc
        % Generate state and STM iteration
        STM_iter = (reshape(xx(i, 7:42), 6, 6))';
        xx_iter = xx(i, 1:6)';

        % Initial IC's manifolds ICs
        X0_man_unstab(:, i)  = xx_iter + d * STM_iter *
           v_unstab_manifold...
            / norm(STM_iter * v_unstab_manifold);
        X0_man_stab(:, i) =  xx_iter + d * STM_iter *
           v_stab_manifold...
            / norm(STM_iter * v_stab_manifold);
    end
end

% Negative half-manifold
function [X0_man_stab, X0_man_unstab] = ic_manifolds_neg(x0, P,
    d, disc, mu)
    % Initialization
    options = odeset('RelTol', 2.24*1e-14, 'AbsTol', 2.24*1e
       -15);
    [~, xx] = ode113(@(t,xM) STMexact_CRTBP(t, xM, mu), ...
        linspace(0, P, disc), [x0; reshape(eye(6),[],1)],
           options);
    X0_man_unstab = zeros(6, disc);
    X0_man_stab = zeros(6, disc);

    % Generate x0 Monodromy Matrix
    M = (reshape(xx(end, 7:42), 6, 6))';

    % Analize monodromy matrix
    [eig_vec, ~] = eig(M);

    % Generate initial directions
    v_unstab_manifold = - eig_vec(:,1);  % HP: always first and
        second!
    v_stab_manifold = - eig_vec(:,2);

    % Initial IC's manifolds ICs
    X0_man_unstab(:, 1)  = x0 + d * v_unstab_manifold / norm(x0
       (1:3));
```

```
    X0_man_stab(:, 1) =  x0 + d * v_stab_manifold / norm(x0
       (1:3));

    % Loop for all manifold' ICs
    for i = 2:disc
        % Generate state and STM iteration
        STM_iter = (reshape(xx(i, 7:42), 6, 6))';
        xx_iter = xx(i, 1:6)';

        % Initial IC's manifolds ICs
        X0_man_unstab(:, i)  = xx_iter + d * STM_iter *
           v_unstab_manifold...
            / norm(STM_iter * v_unstab_manifold);
        X0_man_stab(:, i) =  xx_iter + d * STM_iter *
           v_stab_manifold...
            / norm(STM_iter * v_stab_manifold);
    end
end

% Multiple-Shooting Correction algorithm
function [V_corr, normF_iter] = transfer_ms(V_guess, X1, X4, mu
   , tol)

    % Initialization
    V = V_guess;
    [F, ~] = nonlcons_ms(V, mu, X1, X4);
    normF_iter = zeros(100, 1);
    iter = 1;

    while norm(F) > tol
        % Jacobian of F, update F
        [F, DF] = nonlcons_ms(V, mu, X1, X4);
        normF_iter(iter) = norm(F);
        disp(normF_iter(iter))

        % Check if norm of F exceeds tolerance
        if norm(F) > 1e3
            fprintf('Error: Norm of F diverging. \n');
            break;  % Exit the loop if norm(F) exceeds
               tolerance
        end

        % Correction
        dV = - DF' * ((DF*DF') \ F);
        V = V + dV';
```

```
        iter = iter + 1;
        disp(iter)
    end

    % Final correction
    V_corr = V;
    normF_iter = normF_iter(1:iter-1);

    % Final print
    disp('Correction terminated successfully.');
    fprintf('         Current function value: %.14f\n',
       normF_iter(end));
    fprintf('         Iterations: %d\n', iter-1);
end

% Constraint vector and gradient for Multiple-Shooting
function [F, DF] = nonlcons_ms(V, mu, X0_des, Xf_des)
    %Upacking state
    X1 = V(1:6)';
    T1 = V(7);
    X2 = V(8:13)';
    T2 = V(14);
    X3 = V(15:20)';
    T3 = V(21);
    X4 = V(22:27)';
    T4 = V(28);

    %Integrations
    options = odeset('RelTol', 2.24*1e-14, 'AbsTol', 2.24*1e
       -16);

    %Flow 1
    [~, xxM1] = ode113(@(t, xM) STMexact_CRTBP(t, xM, mu), [0
       T1],...
            [X1; reshape(eye(6), [], 1)],...
            options);
    Xf1 = (xxM1(end, 1:6))';
    rhs_f1 = CRTBP(T1, Xf1, mu);
    STMf1 = (reshape(xxM1(end, 7:42), 6, 6))';

    %Flow 2
    [~, xxM2] = ode113(@(t, xM) STMexact_CRTBP(t, xM, mu), [0
       T2],...
            [X2; reshape(eye(6), [], 1)],...
            options);
```

```
Xf2 = (xxM2(end, 1:6))';
rhs_f2 = CRTBP(T2, Xf2, mu);
STMf2 = (reshape(xxM2(end, 7:42), 6, 6))';   %From
   equations to STM

%Flow 3
[~, xxM3] = ode113(@(t, xM) STMexact_CRTBP(t, xM, mu), [0
   T3],...
        [X3; reshape(eye(6), [], 1)],...
        options);
Xf3 = (xxM3(end, 1:6))';
rhs_f3 = CRTBP(T3, Xf3, mu);
STMf3 = (reshape(xxM3(end, 7:42), 6, 6))';   %From
   equations to STM

%Flow 4
[~, xxM4] = ode113(@(t, xM) STMexact_CRTBP(t, xM, mu), [0
   T4],...
        [X4; reshape(eye(6), [], 1)],...
        options);
Xf4 = (xxM4(end, 1:6))';
rhs_f4 = CRTBP(T4, Xf4, mu);
STMf4 = (reshape(xxM4(end, 7:42), 6, 6))';   %From
   equations to STM

% Constraint Vector (i.e., positions on nodes)
F = [X1(1:3) - X0_des(1:3); ...
    Xf1(1:3) - X2(1:3); ...
    Xf2(1:3) - X3(1:3); ...
    Xf3(1:3) - X4(1:3); ...
    Xf4(1:3) - Xf_des(1:3)];

%Composition of DF
DFrow1 = [eye(3), zeros(3,3), zeros(3,1), zeros(3,6), zeros
   (3,1),...
    zeros(3,6), zeros(3,1), zeros(3,6), zeros(3,1)];
DFrow2 = [STMf1(1:3, :), rhs_f1(1:3), -eye(3), zeros(3,3),
   zeros(3,1),...
    zeros(3,6), zeros(3,1), zeros(3,6), zeros(3,1)];
DFrow3 = [zeros(3,6), zeros(3,1), STMf2(1:3, :), rhs_f2
   (1:3),...
    -eye(3), zeros(3,3), zeros(3,1), zeros(3,6), zeros(3,1)
       ];
DFrow4 = [zeros(3,6), zeros(3,1), zeros(3,6), zeros(3,1),
   ...
```

```
        STMf3(1:3, :), rhs_f3(1:3), -eye(3), zeros(3,3), zeros
           (3,1)];
    DFrow5 = [zeros(3,6), zeros(3,1), zeros(3,6), zeros(3,1),
       ...
        zeros(3,6), zeros(3,1), STMf4(1:3, :), rhs_f4(1:3)];

    DF = [DFrow1; DFrow2; DFrow3; DFrow4; DFrow5];

end

% EXTRA CODE to make checks (Solving NLP)
%
% options=optimoptions('fsolve', 'Algorithm', 'Levenberg-
   Marquardt', ...
%     'Display','iter', 'StepTolerance', 0.2 * 1e-6, ...
%     'MaxIterations', 1000, 'SpecifyObjectiveGradient', true);
% [valid,err] = checkGradients(@(V) obj_ss(V, mu_em, X0_des,
   Xf_des),...
%     V_guess, Display="on")
% V_corr = fsolve(@(V) obj_ss(V, mu_em, X0_des, Xf_des),
   V_guess, options);
%
% function [F, DF] = obj_ss(V, mu, X0_des, Xf_des)
%    %Upacking state
%     X1 = V(1:6)';
%     T1 = V(7);
%     X2 = V(8:13)';
%     T2 = V(14);
%     X3 = V(15:20)';
%     T3 = V(21);
%     X4 = V(22:27)';
%     T4 = V(28);
%
%     %Integrations
%     options = odeset('RelTol', 2.24*1e-14, 'AbsTol', 2.24*1e
   -16);
%
%     %Flow 1
%     [~, xxM1] = ode113(@(t, xM) STMexact_CRTBP(t, xM, mu), [0
    T1],...
%             [X1; reshape(eye(6), [], 1)],...
%             options);
%     Xf1 = (xxM1(end, 1:6))';
%     rhs_f1 = CRTBP(T1, Xf1, mu);
%     STMf1 = (reshape(xxM1(end, 7:42), 6, 6))';
```

```
%
%     %Flow 2
%     [~, xxM2] = ode113(@(t, xM) STMexact_CRTBP(t, xM, mu), [0
    T2],...
%                 [X2; reshape(eye(6), [], 1)],...
%                 options);
%     Xf2 = (xxM2(end, 1:6))';
%     rhs_f2 = CRTBP(T2, Xf2, mu);
%     STMf2 = (reshape(xxM2(end, 7:42), 6, 6))';   %From
   equations to STM
%
%     %Flow 3
%     [~, xxM3] = ode113(@(t, xM) STMexact_CRTBP(t, xM, mu), [0
    T3],...
%                 [X3; reshape(eye(6), [], 1)],...
%                 options);
%     Xf3 = (xxM3(end, 1:6))';
%     rhs_f3 = CRTBP(T3, Xf3, mu);
%     STMf3 = (reshape(xxM3(end, 7:42), 6, 6))';   %From
   equations to STM
%
%     %Flow 4
%     [~, xxM4] = ode113(@(t, xM) STMexact_CRTBP(t, xM, mu), [0
    T4],...
%                 [X4; reshape(eye(6), [], 1)],...
%                 options);
%     Xf4 = (xxM4(end, 1:6))';
%     rhs_f4 = CRTBP(T4, Xf4, mu);
%     STMf4 = (reshape(xxM4(end, 7:42), 6, 6))';   %From
   equations to STM
%
%     % Constraint Vector (i.e., positions on nodes)
%     F = [X1(1:3) - X0_des(1:3); ...
%         Xf1(1:3) - X2(1:3); ...
%         Xf2(1:3) - X3(1:3); ...
%         Xf3(1:3) - X4(1:3); ...
%         Xf4(1:3) - Xf_des(1:3)];
%
%     if nargout > 1
%         %Composition of DF
%         DFrow1 = [eye(3), zeros(3,3), zeros(3,1), zeros(3,6),
    zeros(3,1),...
%                 zeros(3,6), zeros(3,1), zeros(3,6), zeros(3,1)];
%         DFrow2 = [STMf1(1:3, :), rhs_f1(1:3), -eye(3), zeros
   (3,3), zeros(3,1),...
```

```
%             zeros(3,6), zeros(3,1), zeros(3,6), zeros(3,1)];
%         DFrow3 = [zeros(3,6), zeros(3,1), STMf2(1:3, :),
   rhs_f2(1:3),...
%             -eye(3), zeros(3,3), zeros(3,1), zeros(3,6),
   zeros(3,1)];
%         DFrow4 = [zeros(3,6), zeros(3,1), zeros(3,6), zeros
   (3,1),...
%             STMf3(1:3, :), rhs_f3(1:3), -eye(3), zeros(3,3),
   zeros(3,1)];
%         DFrow5 = [zeros(3,6), zeros(3,1), zeros(3,6), zeros
   (3,1),...
%             zeros(3,6), zeros(3,1), STMf4(1:3, :), rhs_f4
   (1:3)];
%
%         DF = [DFrow1; DFrow2; DFrow3; DFrow4; DFrow5];
%     end
%
% end
```

Project Script: Transfer 2A and 2B

```
%% Advanced Astrodynamics (ASEN6060) - Project, Second Half
% Instructor:   Prof. Bosanac
% Student:      Giovanni Fereoli
% Student ID:   111040314

%% Data
clear; clc; close;

% Data
GM_earth = 398600.435436; % km^3/s^2
GM_moon = 4902.800066;
GM_sun = 132712440041.93938;
G_tilde = 6.67408*1e-20; % km^3/kg*s^2
l_star = 384400;
t_star = 3.751902619518436 * 1e5;

% Computations mass ratios
mu_se = GM_earth / (GM_sun + GM_earth); % Mass Ratio
mu_em = GM_moon / (GM_earth + GM_moon);

%% Plot Lyapunov Orbits
clc; close;

% Initial conditions
[xL1, xL2, xL3, xL4, xL5] = EquilibriumPoints(mu_em);
x0_1 = [0.821950426219030, 0, 0, 0, 0.141479662833491, 0];
x0_2 = [1.175773196736922, 0, 0,  0, -0.119977116007445, 0];
P_1 = 2.757108054159905;
P_2 = 3.396688765837098;

% Correction scheme
tol = 1e-10;
[x0_1, P_1, normF_iter_1] = general_corrector(x0_1',...
    P_1, mu_em, tol);
[x0_2, P_2, normF_iter_2] = general_corrector(x0_2',...
    P_2, mu_em, tol);

% Check Jacobi Constant
C_1 = JacobiConstant(x0_1, mu_em);
C_2 = JacobiConstant(x0_2, mu_em);

% Plot orbits
options = odeset('RelTol', 2.24*1e-14, 'AbsTol', 2.24*1e-15);
```

```
[~, xx1] = ode113(@(t,X) CRTBP(t, X, mu_em), [0 P_1], x0_1,
   options);
[~, xx2] = ode113(@(t,X) CRTBP(t, X, mu_em), [0 P_2], x0_2,
   options);

%Plot trajectory
gca = figure(1);
ZVCxy(mu_em, C_2);
hold on;
plot3(xL1(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.5 0.5 1]);
plot3(xL2(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.3 0.8 0.2]);
plot3(1-mu_em, 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.7 0.8 0.2]);
plot3(xx1(:,1), xx1(:,2), xx1(:,3),'k','LineWidth', 2.5);
plot3(xx2(:,1), xx2(:,2), xx2(:,3),'k','LineWidth', 2.5);
quiver3(xx1(1,1) - 0.01, xx1(1,2), xx1(1,3),...
    xx1(1,4) / 3, xx1(1,5) / 3, xx1(1,6) / 3,...
    'k', 'LineWidth', 2);
quiver3(xx2(1,1) + 0.01, xx2(1,2), xx2(1,3),...
    xx2(1,4) / 3, xx2(1,5) / 3, xx2(1,6) / 3,...
    'k', 'LineWidth', 2);
xlabel('x [-]', 'interpreter','latex');
ylabel('y [-]', 'interpreter','latex');
zlabel('z [-]', 'interpreter','latex');
grid on;
axis equal;
xlim([0.8, 1.2]);
ylim([-0.1, 0.1]);
legend('ZVC ($C_{J,2}$)$', '$L_1$', '$L_2$', 'Moon', '
   interpreter', 'latex',...
    'Location', 'eastoutside', 'FontSize', 8);
view(2);

% Export graphics
exportgraphics(gca, 'Project2_orbits.pdf', 'ContentType','image
   ',...
    'Resolution', 1000);

%% Plot Manifolds
clc; close;

%Initialization
options = odeset('RelTol', 2.24*1e-14, 'AbsTol', 2.24*1e-15);
```

```
l_star_em = 384400;
d = 0.1 / l_star_em;
disc = 600;
T_man_1 = 6;
T_man_2 = 8;

% Generate initial conditions manifolds
[X0_man_stab_neg_1, ~] = ic_manifolds_neg(x0_1, P_1, d, disc,
   ...
    mu_em);
[~, X0_man_unstab_neg_2] = ic_manifolds_neg(x0_2, P_2, d, disc,
   ...
    mu_em);

% Integration with STM
[~, xx_stab_neg_1] = ode113(@(t,xM) STMexact_CRTBP(t, xM, mu_em
   ), ...
    [0 -T_man_1], [X0_man_stab_neg_1(:, 1); reshape(eye(6)
       ,[],1)], options);
[~, xx_unstab_neg_2] = ode113(@(t,xM) STMexact_CRTBP(t, xM,
   mu_em), ...
    [0 T_man_2], [X0_man_unstab_neg_2(:, 1); reshape(eye(6)
       ,[],1)], options);

%Plot trajectory
gca2 = figure(2);
plot3(xx_unstab_neg_2(:,1), xx_unstab_neg_2(:,2),
   xx_unstab_neg_2(:,3),'r',...
    'LineWidth', 0.5);
hold on;
plot3(xx_stab_neg_1(:,1), xx_stab_neg_1(:,2),xx_stab_neg_1(:,3)
   ,'b',...
    'LineWidth', 0.5);
plot3(xL1(1), 0, 0, 'd', 'MarkerSize', 6, 'MarkerFaceColor',
   [0.5 0.5 1]);
plot3(xL2(1), 0, 0, 'd', 'MarkerSize', 6, 'MarkerFaceColor',
   [0.2 0.5 1]);
plot3(1-mu_em, 0, 0, 'd', 'MarkerSize', 6, 'MarkerFaceColor',
   [0.8 0.3 1]);
for i=2:disc
    % Integration with STM - positive manifolds
    [~, xx_stab_neg_1] = ode113(@(t,xM) STMexact_CRTBP(t, xM,
       mu_em),...
        [0 -T_man_1], [X0_man_stab_neg_1(:, i); reshape(eye(6)
           ,[],1)], options);
```

```
    [~, xx_unstab_neg_2] = ode113(@(t,xM) STMexact_CRTBP(t, xM,
        mu_em),...
        [0 T_man_2], [X0_man_unstab_neg_2(:, i); reshape(eye(6)
           ,[],1)], options);

    % Plot
    plot3(xx_unstab_neg_2(:,1), xx_unstab_neg_2(:,2),
       xx_unstab_neg_2(:,3),'r',...
        'LineWidth', 0.5);
    plot3(xx_stab_neg_1(:,1), xx_stab_neg_1(:,2),xx_stab_neg_1
       (:,3),'b',...
        'LineWidth', 0.5);
    disp(i);
end
plot3(xx1(:,1), xx1(:,2), xx1(:,3),'k','LineWidth', 3);
plot3(xx2(:,1), xx2(:,2), xx2(:,3),'k','LineWidth', 3);
xlabel('x [-]', 'interpreter','latex');
ylabel('y [-]', 'interpreter','latex');
zlabel('z [-]', 'interpreter','latex');
grid on;
axis equal;
view(2);
legend('Unstable Manifold', 'Stable Manifold',...
    '$L_1$', '$L_2$', 'Moon', 'interpreter', 'latex', 'Location
       ',...
    'eastoutside', 'FontSize', 8);

% Export graphics
exportgraphics(gca2, 'Project2_manifolds.pdf', 'ContentType','
   image',...
    'Resolution', 1000);

%% Plot Poincare' map

% Generate initial conditions manifolds
options_Moon_for = odeset('RelTol', 2.24*1e-14, 'AbsTol',
   2.24*1e-15,...
    'Events', @event_Moon_for);
options_Moon_back = odeset('RelTol', 2.24*1e-14, 'AbsTol',
   2.24*1e-15,...
    'Events', @event_Moon_back);
T_man = 10;

% Initialize to record crossings
num_crossing = zeros(2, disc-1);
```

```
% Initialize cell arrays to store variables for each i
xe1_cell = cell(disc-1, 1);
xe2_cell = cell(disc-1, 1);
te1_cell = cell(disc-1, 1);
te2_cell = cell(disc-1, 1);
ie1_cell = cell(disc-1, 1);
ie2_cell = cell(disc-1, 1);

% Plot Poincare'
gca3 = figure(3);
for i=2:disc
    % Integration with STM - positive manifolds
    [~, xx_stab_neg_1, te1, xe1, ie1] = ode113(@(t, X) CRTBP(t,
        X, mu_em),...
         [0 -T_man], X0_man_stab_neg_1(:, i), options_Moon_back)
            ;
    [~, xx_unstab_neg_2, te2, xe2, ie2] = ode113(@(t, X) CRTBP(
       t, X, mu_em),...
         [0 T_man], X0_man_unstab_neg_2(:, i), options_Moon_for)
            ;

   % Retain only the first two elements
   te1 = te1(1:min(2, length(ie1)));
   xe1 = xe1(1:min(2, length(ie1)),:);
   ie1 = ie1(1:min(2, length(ie1)));
   te2 = te2(1:min(2, length(ie2)));
   xe2 = xe2(1:min(2, length(ie2)),:);
   ie2 = ie2(1:min(2, length(ie2)));

   % Record number of crossings
   num_crossing(1, i-1) = length(ie1);
   num_crossing(2, i-1) = length(ie2);

   % Save variables for each i
   xe1_cell{i-1} = xe1;
   xe2_cell{i-1} = xe2;
   te1_cell{i-1} = te1;
   te2_cell{i-1} = te2;
   ie1_cell{i-1} = ie1;
   ie2_cell{i-1} = ie2;

   % Plot
   plot(xe1(:,5), xe1(:, 2), '.b', 'MarkerSize', 6);
   hold on;
```

```
    plot(xe2(:,5), xe2(:,2), '.r', 'MarkerSize', 6);
    disp(i);
end
xlabel('$\dot{y}$ [-]', 'interpreter','latex');
ylabel('y [-]', 'interpreter','latex');
xlim([-2.5, 3.5]);
ylim([-0.115, 0]);
legend('$L_1$, Stable Manifold', '$L_2$, Unstable Manifold', '
    interpreter', 'latex',...
     'Location', 'southeast');

% Export graphics
exportgraphics(gca3, 'Project2_Poincare.pdf', 'ContentType','
    image',...
     'Resolution', 1000);

%% Find Crossings Poincare'

% Initialize
intersection_Poincare = cell(disc-1, disc-1, 4);
intersection_ICs = cell(disc-1, disc-1, 2);
threshold = 1e-3;
counter = 0;

% Fill crossings matrix
for i = 1:disc-1
    for j = 1:disc-1
        for k = 1:size(xe1_cell{i}, 1)
            for w = 1:size(xe2_cell{j}, 1)
                if norm(xe1_cell{i}(k, [2,5]) - xe2_cell{j}(w,
                    [2,5])) < threshold
                    intersection_Poincare{i, j, 1} = xe1_cell{i
                        }(k, :);
                    intersection_Poincare{i, j, 2} = xe2_cell{j
                        }(w, :);
                    intersection_Poincare{i, j, 3} = te1_cell{i
                        }(k);
                    intersection_Poincare{i, j, 4} = te2_cell{j
                        }(w);
                    counter = counter +1;
                    disp(counter);
                end
            end
        end
    end
```

```
end

% Print not-zero elements
for i = 1:size(intersection_Poincare, 1)
    for j = 1:size(intersection_Poincare, 2)
            % Check if the current element is not empty
            if ~isempty(intersection_Poincare{i, j, 1})
                % Print the indices of non-empty elements
                disp(['Non-empty element found at index (',
                   num2str(i), ',' ...
                     '', num2str(j), ')']);
                intersection_ICs{i, j, 1} = X0_man_stab_neg_1
                   (:, i)';
                intersection_ICs{i, j, 2} = X0_man_unstab_neg_2
                   (:, j)';
                intersection_ICs{i, j, 3} = P_1 * (i-1) / disc;
                intersection_ICs{i, j, 4} = P_2 * (j-1) / disc;
            end
    end
end

%% First Transfer

% Initialization
row = 123; col = 427;
% row = 88; col = 455;

% Initial guesses
x0_L1 = x0_1;
P_L1 = intersection_ICs{row, col, 3};
x0_man_L1 = intersection_ICs{row, col, 1};
P_man_L1 = intersection_Poincare{row, col, 3};
x0_L2 = x0_2;
P_L2 = intersection_ICs{row, col, 4};
x0_man_L2 = intersection_ICs{row, col, 2};
P_man_L2 = intersection_Poincare{row, col, 4};
disc = 10000;

% Plot orbits
options = odeset('RelTol', 2.24*1e-14, 'AbsTol', 2.24*1e-15);
[~, xx1] = ode113(@(t,X) CRTBP(t, X, mu_em), linspace(0, (P_L1-
   P_1), disc), x0_L1, options);
[~, xx2] = ode113(@(t,X) CRTBP(t, X, mu_em), linspace(0,
   P_man_L1, disc), x0_man_L1, options);
[~, xx3] = ode113(@(t,X) CRTBP(t, X, mu_em), linspace(0,
```

```
   P_man_L2, disc), x0_man_L2, options);
[~, xx4] = ode113(@(t,X) CRTBP(t, X, mu_em), linspace(0, P_L2,
   disc), x0_L2, options);
fprintf('The transfer guess needs a ToF of %s [days].\n',...
    num2str((abs(P_man_L1) + abs(P_man_L2)) * t_star / 86400));

%Plot trajectory
gca4 = figure(4);
ZVCxy(mu_em, C_2);
hold on;
plot3(xL1(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.5 0.5 1]);
plot3(xL2(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.3 0.8 0.2]);
plot3(1-mu_em, 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.8 0.3 1]);
plot3(xx2(:,1), xx2(:,2), xx2(:,3),'b','LineWidth', 1.5);
plot3(xx3(:,1), xx3(:,2), xx3(:,3),'r','LineWidth', 1.5);
plot3(xx4(:,1), xx4(:,2), xx4(:,3),'k','LineWidth', 2.5);
plot3(xx1(:,1), xx1(:,2), xx1(:,3),'k','LineWidth', 2.5);
plot3(xx1(1,1), xx1(1,2), xx1(1,3), '.k', 'MarkerSize', 15);
plot3(xx1(end,1), xx1(end,2), xx1(end,3), '.k', 'MarkerSize',
   15);
plot3(xx2(1,1), xx2(1,2), xx2(1,3), '.k', 'MarkerSize', 15);
plot3(xx2(end,1), xx2(end,2), xx2(end,3), '.k', 'MarkerSize',
   15);
plot3(xx3(1,1), xx3(1,2), xx3(1,3), '.k', 'MarkerSize', 15);
plot3(xx3(end,1), xx3(end,2), xx3(end,3), '.k', 'MarkerSize',
   15);
plot3(xx4(1,1), xx4(1,2), xx4(1,3), '.k', 'MarkerSize', 15);
plot3(xx4(end,1), xx4(end,2), xx4(end,3), '.k', 'MarkerSize',
   15);
quiver3(xx1(1,1) - 0.01, xx1(1,2), xx1(1,3),...
    xx1(1,4) / 3, xx1(1,5) / 3, xx1(1,6) / 3,...
    'k', 'LineWidth', 2);
quiver3(xx4(1,1) + 0.01, xx4(1,2), xx4(1,3),...
    xx4(1,4) / 3, xx4(1,5) / 3, xx4(1,6) / 3,...
    'k', 'LineWidth', 2);
quiver3(1.01, -0.11, 0,...
    -0.05, 0,  0,...
    'k', 'LineWidth', 2);
xlabel('x [-]', 'interpreter','latex');
ylabel('y [-]', 'interpreter','latex');
zlabel('z [-]', 'interpreter','latex');
grid on;
```

```
axis equal;
xlim([0.8, 1.2]);
ylim([-0.12, 0.12]);
legend('ZVC ($C_{J,2})$', '$L_1$', '$L_2$', 'Moon',...
    '$L_1$, Stable Manifold', '$L_2$, Unstable Manifold',...
    'interpreter', 'latex', 'Location', 'eastoutside', '
      FontSize', 8);
view(2);

% Reverse few ICs for NEXT section
x0_man_L2 = xx3(end, :);
P_man_L2 = - P_man_L2;
x0_L2 = xx4(end, :); % extra: P_L1 = (P_1-P_L1);
xxf_des = xx4;
xx0_des = xx1;

% Export graphics
exportgraphics(gca4, 'Project2_FirstTransferGuess.pdf', '
   ContentType','image',...
    'Resolution', 1000);

%% First Transfer Correction

% Initialization and initial guess correction
% OSS: the second manifold has been 'reversed'!
V_guess = [x0_man_L1, P_man_L1/2, xx2(end/2,:), P_man_L1/2,...
    x0_man_L2, P_man_L2/2,  xx3(end/2, :), P_man_L2/2];
X0_des = xx1(end, :)';
Xf_des = x0_L2';
tol = 1e-10;

% Correction
[V_corr, normF_iter] = transfer_ms(V_guess, X0_des, Xf_des,
   mu_em, tol);

% Integration orbits
options = odeset('RelTol', 2.24*1e-14, 'AbsTol', 2.24*1e-15);
[~, xx1_corr] = ode113(@(t,X) CRTBP(t, X, mu_em), [0 V_corr(7)
   ], V_corr(1:6), options);
[~, xx2_corr] = ode113(@(t,X) CRTBP(t, X, mu_em), [0 V_corr(14)
   ], V_corr(8:13), options);
[~, xx3_corr] = ode113(@(t,X) CRTBP(t, X, mu_em), [0 V_corr(21)
   ], V_corr(15:20), options);
[~, xx4_corr] = ode113(@(t,X) CRTBP(t, X, mu_em), [0 V_corr(28)
   ], V_corr(22:27), options);
```

```
% Plot orbits
gca5 = figure(5);
ZVCxy(mu_em, C_2);
hold on;
plot3(xL1(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.5 0.5 1]);
plot3(xL2(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.3 0.8 0.2]);
plot3(1-mu_em, 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.8 0.3 1]);
plot3(xx4_corr(end,1), xx4_corr(end,2), xx4_corr(end,3), '.k',
   'MarkerSize', 15);
plot3(xx4_corr(:,1), xx4_corr(:,2), xx4_corr(:,3),'c','
   LineWidth', 2);
plot3(xx4_corr(1,1), xx4_corr(1,2), xx4_corr(1,3), '.k', '
   MarkerSize', 15);
plot3(xx3_corr(end,1), xx3_corr(end,2), xx3_corr(end,3), '.k',
   'MarkerSize', 15);
plot3(xx3_corr(:,1), xx3_corr(:,2), xx3_corr(:,3),'m','
   LineWidth', 1.5);
plot3(xx3_corr(1,1), xx3_corr(1,2), xx3_corr(1,3), '.k', '
   MarkerSize', 15);
plot3(xx2_corr(:,1), xx2_corr(:,2), xx2_corr(:,3),'y','
   LineWidth', 1.5);
plot3(xx2_corr(end,1), xx2_corr(end,2), xx2_corr(end,3), '.k',
   'MarkerSize', 15);
plot3(xx2_corr(1,1), xx2_corr(1,2), xx2_corr(1,3), '.k', '
   MarkerSize', 15);
plot3(xx1_corr(end,1), xx1_corr(end,2), xx1_corr(end,3), '.k',
   'MarkerSize', 15);
plot3(xx1_corr(:,1), xx1_corr(:,2), xx1_corr(:,3),'g','
   LineWidth', 2);
plot3(xx1_corr(1,1), xx1_corr(1,2), xx1_corr(1,3), '.k', '
   MarkerSize', 15);
quiver3(xx1(1,1) - 0.01, xx1(1,2), xx1(1,3),...
    xx1(1,4) / 3, xx1(1,5) / 3, xx1(1,6) / 3,...
    'k', 'LineWidth', 2);
quiver3(xx4(1,1) + 0.01, xx4(1,2), xx4(1,3),...
    xx4(1,4) / 3, xx4(1,5) / 3, xx4(1,6) / 3,...
    'k', 'LineWidth', 2);
quiver3(1.01, -0.11, 0,...
    -0.05, 0,  0,...
    'k', 'LineWidth', 2);
xlabel('x [-]', 'interpreter','latex');
```

```
ylabel('y [-]', 'interpreter','latex');
zlabel('z [-]', 'interpreter','latex');
grid on;
axis equal;
xlim([0.8, 1.2]);
ylim([-0.12, 0.12]);
legend('ZVC ($C_{J,2}$)$', '$L_1$', '$L_2$', 'Moon', '$\Delta V$
    Node', '$1^{st}$ Interval',...
    '', '', '$2^{nd}$ Interval', '', '$3^{rd}$ Interval', '', '
       ', '', '$4^{th}$ Interval',...
    'interpreter', 'latex', 'Location', 'eastoutside', '
       FontSize', 8);
view(2);

% Export graphics
exportgraphics(gca5, 'Project2_FirstTransferCorrected.pdf', '
   ContentType','image',...
    'Resolution', 1000);

% Plot correction performance
iter = 1:length(normF_iter);
gca6 = figure(6);
semilogy(iter, normF_iter, 'k.--', 'LineWidth', 1);
hold on;
semilogy(iter, tol * ones(length(normF_iter), 1), 'r--',...
    'LineWidth', 2);
xlabel('Iterations [-]', 'interpreter','latex');
ylabel('$|\mathbf{F}_k|$ [-]', 'interpreter','latex');
grid on;
real axis;
xlim([1, length(normF_iter)]);
ylim([0.1 * normF_iter(end), normF_iter(1)]);
legend('Norm constraint vector', 'Tolerance', 'Interpreter', '
   latex');

% Export graphics
exportgraphics(gca6, 'Project2_FirstTransferCorrection.pdf', '
   ContentType','image',...
    'Resolution', 1000);

% After plotting you have the flows, compute TCMs
TCM1 = xx1_corr(1, 4:6) - xx0_des(end, 4:6);
TCM2 = xx2_corr(1, 4:6) - xx1_corr(end, 4:6);
TCM3 = xx3_corr(1, 4:6) - xx2_corr(end, 4:6);
TCM4 = xx3_corr(end, 4:6) - xx4_corr(1, 4:6);
```

```
TCM5 = xx4_corr(end, 4:6) - xxf_des(1, 4:6);
TCMtot_norm_A = norm(TCM1) + norm(TCM2) + norm(TCM3) + norm(
   TCM4) + norm(TCM5);
ToF_A = abs(V_corr(7)) + abs(V_corr(14)) + abs(V_corr(21)) +
   abs(V_corr(28));
fprintf('The transfer need a DV of %s [m/s].\n',...
    num2str(TCMtot_norm_A * l_star / t_star * 1000));
fprintf('The transfer need a ToF of %s [days].\n',...
    num2str(ToF_A * t_star / 86400));

%% Second Transfer

% Initialization
% row = 123; col = 427;
row = 88; col = 455;

% Initial guesses
x0_L1 = x0_1;
P_L1 = intersection_ICs{row, col, 3};
x0_man_L1 = intersection_ICs{row, col, 1};
P_man_L1 = intersection_Poincare{row, col, 3};
x0_L2 = x0_2;
P_L2 = intersection_ICs{row, col, 4};
x0_man_L2 = intersection_ICs{row, col, 2};
P_man_L2 = intersection_Poincare{row, col, 4};
disc = 10000;

% Plot orbits
options = odeset('RelTol', 2.24*1e-14, 'AbsTol', 2.24*1e-15);
[~, xx1] = ode113(@(t,X) CRTBP(t, X, mu_em), linspace(0, (P_L1-
   P_1), disc), x0_L1, options);
[~, xx2] = ode113(@(t,X) CRTBP(t, X, mu_em), linspace(0,
   P_man_L1, disc), x0_man_L1, options);
[~, xx3] = ode113(@(t,X) CRTBP(t, X, mu_em), linspace(0,
   P_man_L2, disc), x0_man_L2, options);
[~, xx4] = ode113(@(t,X) CRTBP(t, X, mu_em), linspace(0, P_L2,
   disc), x0_L2, options);
fprintf('The transfer guess needs a ToF of %s [days].\n',...
    num2str((abs(P_man_L1) + abs(P_man_L2)) * t_star / 86400));

%Plot trajectory
gca7 = figure(7);
ZVCxy(mu_em, C_2);
hold on;
plot3(xL1(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
```

```
   [0.5 0.5 1]);
plot3(xL2(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.3 0.8 0.2]);
plot3(1-mu_em, 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.8 0.3 1]);
plot3(xx2(:,1), xx2(:,2), xx2(:,3),'b','LineWidth', 1.5);
plot3(xx3(:,1), xx3(:,2), xx3(:,3),'r','LineWidth', 1.5);
plot3(xx4(:,1), xx4(:,2), xx4(:,3),'k','LineWidth', 2.5);
plot3(xx1(:,1), xx1(:,2), xx1(:,3),'k','LineWidth', 2.5);
plot3(xx1(1,1), xx1(1,2), xx1(1,3), '.k', 'MarkerSize', 15);
plot3(xx1(end,1), xx1(end,2), xx1(end,3), '.k', 'MarkerSize',
   15);
plot3(xx2(1,1), xx2(1,2), xx2(1,3), '.k', 'MarkerSize', 15);
plot3(xx2(end,1), xx2(end,2), xx2(end,3), '.k', 'MarkerSize',
   15);
plot3(xx3(1,1), xx3(1,2), xx3(1,3), '.k', 'MarkerSize', 15);
plot3(xx3(end,1), xx3(end,2), xx3(end,3), '.k', 'MarkerSize',
   15);
plot3(xx4(1,1), xx4(1,2), xx4(1,3), '.k', 'MarkerSize', 15);
plot3(xx4(end,1), xx4(end,2), xx4(end,3), '.k', 'MarkerSize',
   15);
quiver3(xx1(1,1) - 0.01, xx1(1,2), xx1(1,3),...
    xx1(1,4) / 3, xx1(1,5) / 3, xx1(1,6) / 3,...
    'k', 'LineWidth', 2);
quiver3(xx4(1,1) + 0.01, xx4(1,2), xx4(1,3),...
    xx4(1,4) / 3, xx4(1,5) / 3, xx4(1,6) / 3,...
    'k', 'LineWidth', 2);
quiver3(1.01, -0.11, 0,...
    -0.05, 0,  0,...
    'k', 'LineWidth', 2);
xlabel('x [-]', 'interpreter','latex');
ylabel('y [-]', 'interpreter','latex');
zlabel('z [-]', 'interpreter','latex');
grid on;
axis equal;
xlim([0.8, 1.2]);
ylim([-0.12, 0.12]);
legend('ZVC ($C_{J,2})$', '$L_1$', '$L_2$', 'Moon',...
    '$L_1$, Stable Manifold', '$L_2$, Unstable Manifold',...
    'interpreter', 'latex', 'Location', 'eastoutside', '
      FontSize', 8);
view(2);

% Reverse few ICs for NEXT section
x0_man_L2 = xx3(end, :);
```

```
P_man_L2 = - P_man_L2;
x0_L2 = xx4(end, :); % extra: P_L1 = (P_1-P_L1);
xxf_des = xx4;
xx0_des = xx1;

% Export graphics
exportgraphics(gca7, 'Project2_SecondTransferGuess.pdf', '
   ContentType','image',...
    'Resolution', 1000);

%% Second Transfer Correction

% Initialization and initial guess correction
% OSS: the second manifold has been 'reversed'!
V_guess = [x0_man_L1, P_man_L1/2, xx2(end/2,:), P_man_L1/2,...
    x0_man_L2, P_man_L2/2,  xx3(end/2, :), P_man_L2/2];
X0_des = xx1(end, :)';
Xf_des = x0_L2';
tol = 1e-10;

% Correction
[V_corr, normF_iter] = transfer_ms(V_guess, X0_des, Xf_des,
   mu_em, tol);

% Integration orbits
options = odeset('RelTol', 2.24*1e-14, 'AbsTol', 2.24*1e-15);
[~, xx1_corr] = ode113(@(t,X) CRTBP(t, X, mu_em), [0 V_corr(7)
   ], V_corr(1:6), options);
[~, xx2_corr] = ode113(@(t,X) CRTBP(t, X, mu_em), [0 V_corr(14)
   ], V_corr(8:13), options);
[~, xx3_corr] = ode113(@(t,X) CRTBP(t, X, mu_em), [0 V_corr(21)
   ], V_corr(15:20), options);
[~, xx4_corr] = ode113(@(t,X) CRTBP(t, X, mu_em), [0 V_corr(28)
   ], V_corr(22:27), options);

% Plot orbits
gca8 = figure(8);
ZVCxy(mu_em, C_2);
hold on;
plot3(xL1(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.5 0.5 1]);
plot3(xL2(1), 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.3 0.8 0.2]);
plot3(1-mu_em, 0, 0, 'd', 'MarkerSize', 8, 'MarkerFaceColor',
   [0.8 0.3 1]);
```

```
plot3(xx4_corr(end,1), xx4_corr(end,2), xx4_corr(end,3), '.k',
   'MarkerSize', 15);
plot3(xx4_corr(:,1), xx4_corr(:,2), xx4_corr(:,3),'c','
   LineWidth', 2);
plot3(xx4_corr(1,1), xx4_corr(1,2), xx4_corr(1,3), '.k', '
   MarkerSize', 15);
plot3(xx3_corr(end,1), xx3_corr(end,2), xx3_corr(end,3), '.k',
   'MarkerSize', 15);
plot3(xx3_corr(:,1), xx3_corr(:,2), xx3_corr(:,3),'m','
   LineWidth', 1.5);
plot3(xx3_corr(1,1), xx3_corr(1,2), xx3_corr(1,3), '.k', '
   MarkerSize', 15);
plot3(xx2_corr(:,1), xx2_corr(:,2), xx2_corr(:,3),'y','
   LineWidth', 1.5);
plot3(xx2_corr(end,1), xx2_corr(end,2), xx2_corr(end,3), '.k',
   'MarkerSize', 15);
plot3(xx2_corr(1,1), xx2_corr(1,2), xx2_corr(1,3), '.k', '
   MarkerSize', 15);
plot3(xx1_corr(end,1), xx1_corr(end,2), xx1_corr(end,3), '.k',
   'MarkerSize', 15);
plot3(xx1_corr(:,1), xx1_corr(:,2), xx1_corr(:,3),'g','
   LineWidth', 2);
plot3(xx1_corr(1,1), xx1_corr(1,2), xx1_corr(1,3), '.k', '
   MarkerSize', 15);
quiver3(xx1(1,1) - 0.01, xx1(1,2), xx1(1,3),...
    xx1(1,4) / 3, xx1(1,5) / 3, xx1(1,6) / 3,...
    'k', 'LineWidth', 2);
quiver3(xx4(1,1) + 0.01, xx4(1,2), xx4(1,3),...
    xx4(1,4) / 3, xx4(1,5) / 3, xx4(1,6) / 3,...
    'k', 'LineWidth', 2);
quiver3(1.01, -0.11, 0,...
    -0.05, 0,  0,...
    'k', 'LineWidth', 2);
xlabel('x [-]', 'interpreter','latex');
ylabel('y [-]', 'interpreter','latex');
zlabel('z [-]', 'interpreter','latex');
grid on;
axis equal;
xlim([0.8, 1.2]);
ylim([-0.12, 0.12]);
legend('ZVC ($C_{J,2}$)$', '$L_1$', '$L_2$', 'Moon', '$\Delta V$
    Node', '$1^{st}$ Interval',...
    '', '', '$2^{nd}$ Interval', '', '$3^{rd}$ Interval', '', '
       ', '', '$4^{th}$ Interval',...
    'interpreter', 'latex', 'Location', 'eastoutside', '
```

```
       FontSize', 8);
view(2);

% Export graphics
exportgraphics(gca8, 'Project2_SecondTransferCorrected.pdf', '
   ContentType','image',...
    'Resolution', 1000);

% Plot correction performance
iter = 1:length(normF_iter);
gca9 = figure(9);
semilogy(iter, normF_iter, 'k.--', 'LineWidth', 1);
hold on;
semilogy(iter, tol * ones(length(normF_iter), 1), 'r--',...
    'LineWidth', 2);
xlabel('Iterations [-]', 'interpreter','latex');
ylabel('$|\mathbf{F}_k|$ [-]', 'interpreter','latex');
grid on;
real axis;
xlim([1, length(normF_iter)]);
ylim([0.1 * normF_iter(end), normF_iter(1)]);
legend('Norm constraint vector', 'Tolerance', 'Interpreter', '
   latex');

% Export graphics
exportgraphics(gca9, 'Project2_SecondTransferCorrection.pdf', '
   ContentType','image',...
    'Resolution', 1000);

% After plotting you have the flows, compute TCMs
TCM1 = xx1_corr(1, 4:6) - xx0_des(end, 4:6);
TCM2 = xx2_corr(1, 4:6) - xx1_corr(end, 4:6);
TCM3 = xx3_corr(1, 4:6) - xx2_corr(end, 4:6);
TCM4 = xx3_corr(end, 4:6) - xx4_corr(1, 4:6);
TCM5 = xx4_corr(end, 4:6) - xxf_des(1, 4:6);
TCMtot_norm_B = norm(TCM1) + norm(TCM2) + norm(TCM3) + norm(
   TCM4) + norm(TCM5);
ToF_B = abs(V_corr(7)) + abs(V_corr(14)) + abs(V_corr(21)) +
   abs(V_corr(28));
fprintf('The transfer need a DV of %s [m/s].\n',...
    num2str(TCMtot_norm_B * l_star / t_star * 1000));
fprintf('The transfer need a ToF of %s [days].\n',...
    num2str(ToF_B * t_star / 86400));

%% Functions
```

```
% Compute ZVC in the xy-plane
function ZVCxy(mu, C)
    % Initialization
    x_zvc = -1.5:0.001:1.55;
    y_zvc = -1.5:0.001:1.55;
    Z_zvc = zeros(length(y_zvc), length(x_zvc));

    % ZVS computations
    for i=1:length(x_zvc)
        for j=1:length(y_zvc)
            Z_zvc(j, i)= (x_zvc(i)^2+y_zvc(j)^2) + ...
                2*(1-mu)./sqrt((x_zvc(i)+mu)^2+y_zvc(j)^2) +
                   ...
                2*mu./sqrt((x_zvc(i)-1+mu)^2+y_zvc(j)^2);
        end
    end

    % Plot
    contourf(x_zvc, y_zvc, -Z_zvc,[-C -C]);
end

% Compute Jacobi Constant in the xy-plane
function C = JacobiConstant(X, mu)
    % Initialization
    x = X(1);
    y = X(2);
    z = X(3);
    xdot = X(4);
    ydot = X(5);
    zdot = X(6);

    % Jacobi Constant Computation
    C = (x^2+y^2) + 2*(1-mu)/sqrt((x+mu)^2+y^2+z^2) + ...
        2*mu/sqrt((x-1+mu)^2+y^2+z^2) - sqrt(xdot^2+ydot^2+zdot
           ^2)^2;
end

% CR3BP Equations of Motions
function dXdt = CRTBP(~, X, mu)
    %Initialize
    dXdt = zeros(6,1);

    x = X(1);
    y = X(2);
```

```
    z = X(3);
    xdot = X(4);
    ydot = X(5);
    zdot = X(6);

    % CRTBP dynamics
    r1_norm = sqrt((x+mu)^2+y^2+z^2);
    r2_norm = sqrt((x+mu-1)^2+y^2+z^2);

    dXdt(1:3) = [xdot; ydot; zdot];
    dXdt(4:6) = [2*ydot+x-(1-mu)*(x+mu)/r1_norm^3-mu*(x+mu-1)/
       r2_norm^3;...
        -2*xdot+y-(1-mu)*y/r1_norm^3-mu*y/r2_norm^3;...
        -(1-mu)*z/r1_norm^3-mu*z/r2_norm^3];
end

% Find equilibrium points
function [xL1, xL2, xL3, xL4, xL5] = EquilibriumPoints(mu)
    % Collinear points
    % Position primaries along x
    xxP1 = -mu;
    xxP2 = 1-mu;

    % Gradient of U* in x
    f = @(x) x-(1-mu)*(x+mu)/(abs(x+mu))^3-mu*(x+mu-1)/(abs(x+
       mu-1))^3;

    %Inital guesses
    z = (mu/3)^(1/3);
    xxL10 = xxP2 - (z-(1/3)*z^2-(1/9)*z^3+(58/81)*z^4);
    xxL20 = xxP2 + (z+(1/3)*z^2-(1/9)*z^3+(50/81)*z^4);
    xxL30 = xxP1 - (1-(7/12)*mu-(1127/20736)*mu^3-(7889/248832)
       *mu^4);

    %Zeros computation
    options = optimoptions('fsolve', 'Display', 'none', 'TolFun
       ', 1e-15);
    xL1 = [fsolve(f, xxL10, options), 0, 0];
    xL2 = [fsolve(f, xxL20, options), 0, 0];
    xL3 = [fsolve(f, xxL30, options), 0, 0];

    % Triangular points
    xL4 = [0.5 - mu, sqrt(3) / 2, 0];
    xL5 = [0.5 - mu, - sqrt(3) / 2, 0];
end
```

```
% General initial guess correction algorithm
function [X0_corr, P_corr, normF_iter] = general_corrector(X0,
   P, mu, tol)
% Initialization
options = odeset('RelTol', 2.24*1e-14, 'AbsTol', 2.24*1e-15);
F = ones(6,1);
V = [X0; P];
normF_iter = zeros(100, 1);
iter = 1;

while norm(F) > tol
    %Integration state and STM
    [~, xx] = ode113(@(t,xM) STMexact_CRTBP(t, xM, mu), [0 V(
       end)],...
    [V(1:6); reshape(eye(6), [], 1)], options);

    %Reshape final STM
    STMf=(reshape(xx(end, 7:42), 6, 6))';

    %Jacobian of F
    xMdotf = STMexact_CRTBP(V(end), xx(end, :), mu);
    DF = [STMf - eye(6), xMdotf(1:6)];

    % Update F
    F = xx(end,1:6)- xx(1,1:6);
    normF_iter(iter) = norm(F);

    % Check if norm of F exceeds tolerance
    if norm(F) > 1e3
        fprintf('Error: Norm of F diverging. \n');
        break;  % Exit the loop if norm(F) exceeds tolerance
    end

    % Correction
    dV = - lsqminnorm(DF, F', tol);
    V = V + dV;
    iter = iter + 1;
end

% Final correction
X0_corr = V(1:6);
P_corr = V(7);
normF_iter = normF_iter(1:iter-1);
```

```
% Final print
disp('Correction terminated successfully.');
fprintf('         Current function value: %.14f\n', normF_iter(
   end));
fprintf('         Iterations: %d\n', iter-1);
end

% C3RBP ODEs: state and STM
function dxMdt = STMexact_CRTBP(~,xM,mu)
%Initialize ODE
dxMdt = zeros(42,1);

% Unpack state
x = xM(1);
y = xM(2);
z = xM(3);
xdot = xM(4);
ydot = xM(5);
zdot = xM(6);

% Unpack STM
M = (reshape(xM(7:42),6,6))';   %From state to STM

% CR3TBP miscellaneous
% P3 distance from primaries
r1_norm = sqrt((x+mu)^2+y^2+z^2);
r2_norm = sqrt((x+mu-1)^2+y^2+z^2);

% Variational equations
df4dx = 1-(1-mu)/r1_norm^3+3*(1-mu)*(x+mu)^2/r1_norm^5-mu/
   r2_norm^3+...
    3*mu*(x+mu-1)^2/r2_norm^5;
df4dy = 3*(1-mu)*(x+mu)*y/r1_norm^5+3*mu*(x+mu-1)*y/r2_norm^5;
df4dz = 3*(1-mu)*(x+mu)*z/r1_norm^5+3*mu*(x+mu-1)*z/r2_norm^5;
df5dy = 1-(1-mu)/r1_norm^3+3*(1-mu)*y^2/r1_norm^5-mu/r2_norm^3+
   ...
    3*mu*y^2/r2_norm^5;
df5dz = 3*(1-mu)*y*z/r1_norm^5+3*mu*y*z/r2_norm^5;
df6dz = -(1-mu)/r1_norm^3+3*(1-mu)*z^2/r1_norm^5-mu/r2_norm^3+
   ...
    3*mu*z^2/r2_norm^5;

% Jacobian
A = [0, 0, 0, 1, 0, 0;...
    0, 0, 0, 0, 1, 0;...
```

```
    0, 0, 0, 0, 0, 1;...
    df4dx, df4dy, df4dz, 0, 2, 0;...
    df4dy, df5dy, df5dz, -2, 0, 0;...
    df4dz, df5dz, df6dz, 0, 0, 0];

% CR3TBP dynamics
% CR3BP dynamics: position and velocity
dxMdt(1:3) = [xdot; ydot; zdot];
dxMdt(4:6) = [2*ydot+x-(1-mu)*(x+mu)/r1_norm^3-mu*(x+mu-1)/
   r2_norm^3;...
    -2*xdot+y-(1-mu)*y/r1_norm^3-mu*y/r2_norm^3;...
    -(1-mu)*z/r1_norm^3-mu*z/r2_norm^3];

% CR3BP dynamics: STM
dMdt = A*M;
dxMdt(7:12) = dMdt(1,1:6)';
dxMdt(13:18) = dMdt(2,1:6)';
dxMdt(19:24) = dMdt(3,1:6)';
dxMdt(25:30) = dMdt(4,1:6)';
dxMdt(31:36) = dMdt(5,1:6)';
dxMdt(37:42) = dMdt(6,1:6)';

end

% Events Moon
function [value, isterminal, direction] = event_Moon_for(~, X,
   ~)
% Data
GM_earth = 398600.435436; % km^3/s^2
GM_moon = 4902.800066;

% Computations mass ratios
mu_em = GM_moon / (GM_earth + GM_moon);

% Event
value = X(1) - (1-mu_em);   % x = 1-mu
isterminal = 0;
direction = 1; % all directions
end

function [value, isterminal, direction] = event_Moon_back(~, X,
    ~)
% Data
GM_earth = 398600.435436; % km^3/s^2
GM_moon = 4902.800066;
```

```
% Computations mass ratios
mu_em = GM_moon / (GM_earth + GM_moon);

% Event
value = X(1) - (1-mu_em);   % x = 1-mu
isterminal = 0;
direction = -1; % all directions
end

% % Positive half-manifold
% function [X0_man_stab, X0_man_unstab] = ic_manifolds_pos(x0,
   P, d, disc, mu)
% % Initialization
% options = odeset('RelTol', 2.24*1e-14, 'AbsTol', 2.24*1e-15);
% [~, xx] = ode113(@(t,xM) STMexact_CRTBP(t, xM, mu), ...
%     linspace(0, P, disc), [x0; reshape(eye(6),[],1)], options
   );
% X0_man_unstab = zeros(6, disc);
% X0_man_stab = zeros(6, disc);
%
% % Generate x0 Monodromy Matrix
% M = (reshape(xx(end, 7:42), 6, 6))';
%
% % Analize monodromy matrix
% [eig_vec, ~] = eig(M);
%
% % Generate initial directions
% v_unstab_manifold = eig_vec(:,1);  % HP: always first and
   second!
% v_stab_manifold = eig_vec(:,2);
%
% % Initial IC's manifolds ICs
% X0_man_unstab(:, 1)  = x0 + d * v_unstab_manifold / norm(x0
   (1:3));
% X0_man_stab(:, 1) =  x0 + d * v_stab_manifold / norm(x0(1:3))
   ;
%
% % Loop for all manifold' ICs
% for i = 2:disc
%     % Generate state and STM iteration
%     STM_iter = (reshape(xx(i, 7:42), 6, 6))';
%     xx_iter = xx(i, 1:6)';
%
%     % Initial IC's manifolds ICs
```

```
%     X0_man_unstab(:, i)  = xx_iter + d * STM_iter *
   v_unstab_manifold...
%         / norm(STM_iter * v_unstab_manifold);
%     X0_man_stab(:, i) =  xx_iter + d * STM_iter *
   v_stab_manifold...
%         / norm(STM_iter * v_stab_manifold);
% end
% end

% Negative half-manifold
function [X0_man_stab, X0_man_unstab] = ic_manifolds_neg(x0, P,
    d, disc, mu)
% Initialization
options = odeset('RelTol', 2.24*1e-14, 'AbsTol', 2.24*1e-15);
[~, xx] = ode113(@(t,xM) STMexact_CRTBP(t, xM, mu), ...
    linspace(0, P, disc), [x0; reshape(eye(6),[],1)], options);
X0_man_unstab = zeros(6, disc);
X0_man_stab = zeros(6, disc);

% Generate x0 Monodromy Matrix
M = (reshape(xx(end, 7:42), 6, 6))';

% Analize monodromy matrix
[eig_vec, ~] = eig(M);

% Generate initial directions
v_unstab_manifold = - eig_vec(:,1);  % HP: always first and
   second!
v_stab_manifold = - eig_vec(:,2);

% Initial IC's manifolds ICs
X0_man_unstab(:, 1)  = x0 + d * v_unstab_manifold / norm(x0
   (1:3));
X0_man_stab(:, 1) =  x0 + d * v_stab_manifold / norm(x0(1:3));

% Loop for all manifold' ICs
for i = 2:disc
    % Generate state and STM iteration
    STM_iter = (reshape(xx(i, 7:42), 6, 6))';
    xx_iter = xx(i, 1:6)';

    % Initial IC's manifolds ICs
    X0_man_unstab(:, i)  = xx_iter + d * STM_iter *
       v_unstab_manifold...
        / norm(STM_iter * v_unstab_manifold);
```

```
    X0_man_stab(:, i) =  xx_iter + d * STM_iter *
       v_stab_manifold...
        / norm(STM_iter * v_stab_manifold);
end
end

% Multiple-Shooting Correction algorithm
function [V_corr, normF_iter] = transfer_ms(V_guess, X1, X4, mu
   , tol)

    % Initialization
    V = V_guess;
    [F, ~] = nonlcons_ms(V, mu, X1, X4);
    normF_iter = zeros(100, 1);
    iter = 1;

    while norm(F) > tol
        % Jacobian of F, update F
        [F, DF] = nonlcons_ms(V, mu, X1, X4);
        normF_iter(iter) = norm(F);
        disp(normF_iter(iter))

        % Check if norm of F exceeds tolerance
        if norm(F) > 1e3
            fprintf('Error: Norm of F diverging. \n');
            break;  % Exit the loop if norm(F) exceeds
               tolerance
        end

        % Correction
        dV = - DF' * ((DF*DF') \ F);
        V = V + dV';
        iter = iter + 1;
        disp(iter)
    end

    % Final correction
    V_corr = V;
    normF_iter = normF_iter(1:iter-1);

    % Final print
    disp('Correction terminated successfully.');
    fprintf('         Current function value: %.14f\n',
       normF_iter(end));
    fprintf('         Iterations: %d\n', iter-1);
```

```
end

% Constraint vector and gradient for Multiple-Shooting
function [F, DF] = nonlcons_ms(V, mu, X0_des, Xf_des)
    %Upacking state
    X1 = V(1:6)';
    T1 = V(7);
    X2 = V(8:13)';
    T2 = V(14);
    X3 = V(15:20)';
    T3 = V(21);
    X4 = V(22:27)';
    T4 = V(28);

    %Integrations
    options = odeset('RelTol', 2.24*1e-14, 'AbsTol', 2.24*1e
       -16);

    %Flow 1
    [~, xxM1] = ode113(@(t, xM) STMexact_CRTBP(t, xM, mu), [0
       T1],...
            [X1; reshape(eye(6), [], 1)],...
            options);
    Xf1 = (xxM1(end, 1:6))';
    rhs_f1 = CRTBP(T1, Xf1, mu);
    STMf1 = (reshape(xxM1(end, 7:42), 6, 6))';

    %Flow 2
    [~, xxM2] = ode113(@(t, xM) STMexact_CRTBP(t, xM, mu), [0
       T2],...
            [X2; reshape(eye(6), [], 1)],...
            options);
    Xf2 = (xxM2(end, 1:6))';
    rhs_f2 = CRTBP(T2, Xf2, mu);
    STMf2 = (reshape(xxM2(end, 7:42), 6, 6))';   %From
       equations to STM

    %Flow 3
    [~, xxM3] = ode113(@(t, xM) STMexact_CRTBP(t, xM, mu), [0
       T3],...
            [X3; reshape(eye(6), [], 1)],...
            options);
    Xf3 = (xxM3(end, 1:6))';
    rhs_f3 = CRTBP(T3, Xf3, mu);
    STMf3 = (reshape(xxM3(end, 7:42), 6, 6))';   %From
```

```
    equations to STM

%Flow 4
[~, xxM4] = ode113(@(t, xM) STMexact_CRTBP(t, xM, mu), [0
    T4],...
         [X4; reshape(eye(6), [], 1)],...
         options);
Xf4 = (xxM4(end, 1:6))';
rhs_f4 = CRTBP(T4, Xf4, mu);
STMf4 = (reshape(xxM4(end, 7:42), 6, 6))';   %From
    equations to STM

% Constraint Vector (i.e., positions on nodes)
F = [X1(1:3) - X0_des(1:3); ...
     Xf1(1:3) - X2(1:3); ...
     Xf2(1:3) - X3(1:3); ...
     Xf3(1:3) - X4(1:3); ...
     Xf4(1:3) - Xf_des(1:3)];

%Composition of DF
DFrow1 = [eye(3), zeros(3,3), zeros(3,1), zeros(3,6), zeros
    (3,1),...
     zeros(3,6), zeros(3,1), zeros(3,6), zeros(3,1)];
DFrow2 = [STMf1(1:3, :), rhs_f1(1:3), -eye(3), zeros(3,3),
    zeros(3,1),...
     zeros(3,6), zeros(3,1), zeros(3,6), zeros(3,1)];
DFrow3 = [zeros(3,6), zeros(3,1), STMf2(1:3, :), rhs_f2
    (1:3),...
     -eye(3), zeros(3,3), zeros(3,1), zeros(3,6), zeros(3,1)
        ];
DFrow4 = [zeros(3,6), zeros(3,1), zeros(3,6), zeros(3,1),
    ...
     STMf3(1:3, :), rhs_f3(1:3), -eye(3), zeros(3,3), zeros
        (3,1)];
DFrow5 = [zeros(3,6), zeros(3,1), zeros(3,6), zeros(3,1),
    ...
     zeros(3,6), zeros(3,1), STMf4(1:3, :), rhs_f4(1:3)];

DF = [DFrow1; DFrow2; DFrow3; DFrow4; DFrow5];

end
```