Getting closer to our cosmic neighbourhood. Study and optimization of transfer orbits to the L4 and L5 Lagrange Points of the Earth-Sun system.

Document:

Final Thesis

Author:

Urruticoechea Puig, Aitor

Director - Codirector:

Rovira, Adrià - Farrés, Ariadna

Study Plan:

Engineering Degree in Aerospace Technologies (GrETA)

Examination Session:

Autum 2022

It is good to have an end to journey toward,

but it is the journey that matters in the end

Ursula K. Le Guin, *The Left Hand of Darkness*

Abstract

[ENGLISH]

The Circular-Restricted Three-Body Problem (CR3BP) describes the motion of a small celestial body in the gravitational field of two larger bodies. The present study focuses on the application of the CR3BP to the Sun-Earth system. To that end, the governing and defining equations for the CR3BP are described in-depth; and key periodic orbits around the Lagrange Points L1, L2, L4, and L5 are mapped. These last two points, located at the vertices of an equilateral triangle in the rotating frame of reference, are particularly useful for observation and exploration due to their stability. Thus, the L4 and L5 points are the final objective of the present study. The study focuses on optimizing transfer trajectories from the previously mapped orbits around L1 and L2 to orbits around L4 and L5, respectively. The procedures utilized significantly reduce the required delta-V budget to perform these transfers to approximately 3 km/s, a value lower than the minimum requirements for interplanetary transfers departing from Earth. This finding has the potential to open up new opportunities for asteroid science and solar observation based at the equilateral Lagrange Points of the Sun-Earth system; ultimately contributing to advancing the current era of scientific research in the field of astrodynamics.

[CATALÀ - CATALAN]

El problema restringit circular dels tres cossos (CR3BP) descriu el moviment d'un cos celeste petit en el camp gravitatori de dos cossos més grans. L'estudi actual se centra en l'aplicació del CR3BP al sistema Sol-Terra. Amb aquesta finalitat, les equacions que regeixen i defineixen el CR3BP es descriuen en profunditat; i òrbites periòdiques clau al voltant dels punts de Lagrange L1, L2, L4 i L5 es mapegen. Aquests dos últims punts, localitzats en els vèrtexs d'un triangle equilàter en el marc rotatori de referència, són particularment útils per a l'observació i l'exploració a causa de la seva estabilitat. Per tant, els punts L4 i L5 són l'objectiu final de l'estudi actual. L'estudi se centra a optimitzar les trajectòries de transferència de les òrbites prèviament assignades al voltant de L1 i L2 a les òrbites al voltant de L4 i L5, respectivament. Els procediments utilitzats redueixen significativament el pressupost necessari de delta-V per realitzar aquestes transferències a aproximadament 3 km/s, un valor inferior als requisits mínims per a les transferències interplanetàries que surten de la Terra. Tot plegat té el potencial d'obrir noves oportunitats per a la ciència d'asteroides i l'observació solar basada en els Punts de Lagrange equilàters del sistema Sol-Terra; en última instància contribuint així a avançar l'era actual de la investigació científica en el camp de l'astrodinàmica.

[CASTELLANO - SPANISH]

El problema circular restringido de los tres cuerpos (CR3BP) describe el movimiento de un pequeño cuerpo celeste en el campo gravitatorio de dos cuerpos mayores. El presente estudio se centra en la aplicación del CR3BP al sistema Sol-Tierra. Para ello, se describen en profundidad las ecuaciones que gobiernan y definen del CR3BP y se trazan órbitas periódicas clave alrededor de los puntos de Lagrange L1, L2, L4 y L5. Estos dos últimos puntos, situados en los vértices de un triángulo equilátero en el marco de referencia de rotación, son especialmente útiles para la observación y exploración debido a su estabilidad. Así pues, los puntos L4 y L5 constituyen el objetivo final del presente estudio. El estudio se centra en la optimización de las trayectorias de transferencia desde las órbitas previamente trazadas en torno a L1 y L2 a órbitas en torno a L4 y L5, respectivamente. Los procedimientos utilizados reducen significativamente el presupuesto delta-V necesario para realizar estas transferencias a aproximadamente 3 km/s, un valor inferior a los requisitos mínimos para las transferencias interplanetarias que parten de la Tierra. Este hallazgo tiene el potencial de abrir nuevas oportunidades para la ciencia de los asteroides y la observación solar basada en los puntos equiláteros de Lagrange del sistema Sol-Tierra; contribuyendo en última instancia a avanzar la era actual de la investigación científica en el campo de la astrodinámica.

Index

Figure List

Table List

Acronyms

CR3BP Circular-Restricted Three-Body Problem. 5, 6, 8, 9, 11, 12, 15–21, 26, 28, 32, 38, 42, 44, 55, 64, 100, V

GMAT General Mission Analysis Tool. 55, 60

Jc Jacobi constant. 20, 24, 44, 64, VII

JPL Jet Propulsion Laboratory. 16, 32, 37, 59, 62, 65, 100, VII

L1 Lagrange Point 1 - between the two main primaries -. 2, 5, 7–9, 19, 20, 22–24, 29, 31–35, 42–49, 53–55, 62, 63, 65, 77, 80, 89, 94, V–VII

L2 Lagrange Point 2 - on the far side of the smaller primary -. 2, 5–9, 19, 20, 22–24, 29, 31–34, 36, 42–44, 49–51, 53–55, 60, 62, 63, 65, 94, V–VII

L3 Lagrange Point 3 - on the orbit of the smaller primary, opposite to it -. 5, 20, 22–24, 29, 31, 37, 44

L4 Lagrange Point 4 - on the orbit of the smaller primary, ahead of it -. 2, 5, 7–9, 19, 20, 22–26, 31, 37, 39, 40, 42–49, 53–56, 58, 62, 63, 86, 89, 94, V–VII

L5 Lagrange Point 5 - on the orbit of the smaller primary, behind it -. 2, 5, 7–9, 19, 20, 22–25, 31, 37, 39, 41–45, 49–51, 53–56, 58, 62, 63, 86, 94, V–VII

LEO Low Earth Orbit. 7, 8, 11, 54, 55

NASA National Aeronautics and Space Administration. 16, 32, 37, 54, 55, 58–60, 62, 65, 100

NDU Non-Dimensional Units. 15, 18, 19, 23–25, 27, 33–36, 39–41, 43, 46, 48, 49, 53–55, V–VII

RF Reference Frame. 13, 14, 16, 17, 20, 21, 25, V

STM State Transition Matrix. 42, 43, 55, 66, 89

Part #01

General Overview

1.1 Aim of the Project

The aim of this thesis is to propose new routes to key Lagrange Points [1, 2] of the Earth-Sun system using as model the circular-restricted three-body problem. These orbital paths are meant to transfer a spacecraft from Earth's vicinity to L4 and L5 respectively inexpensively using the intrinsic instabilities of the system, which lead to mathematical objects known as manifolds (Figure 1.1). L4 and L5 are two points of remarkable interest due to their stability and their potential as scientific hotspots. In particular, for privileged observation of the Sun [3] and for their chance to be a host of Trojan asteroids [4, 5].

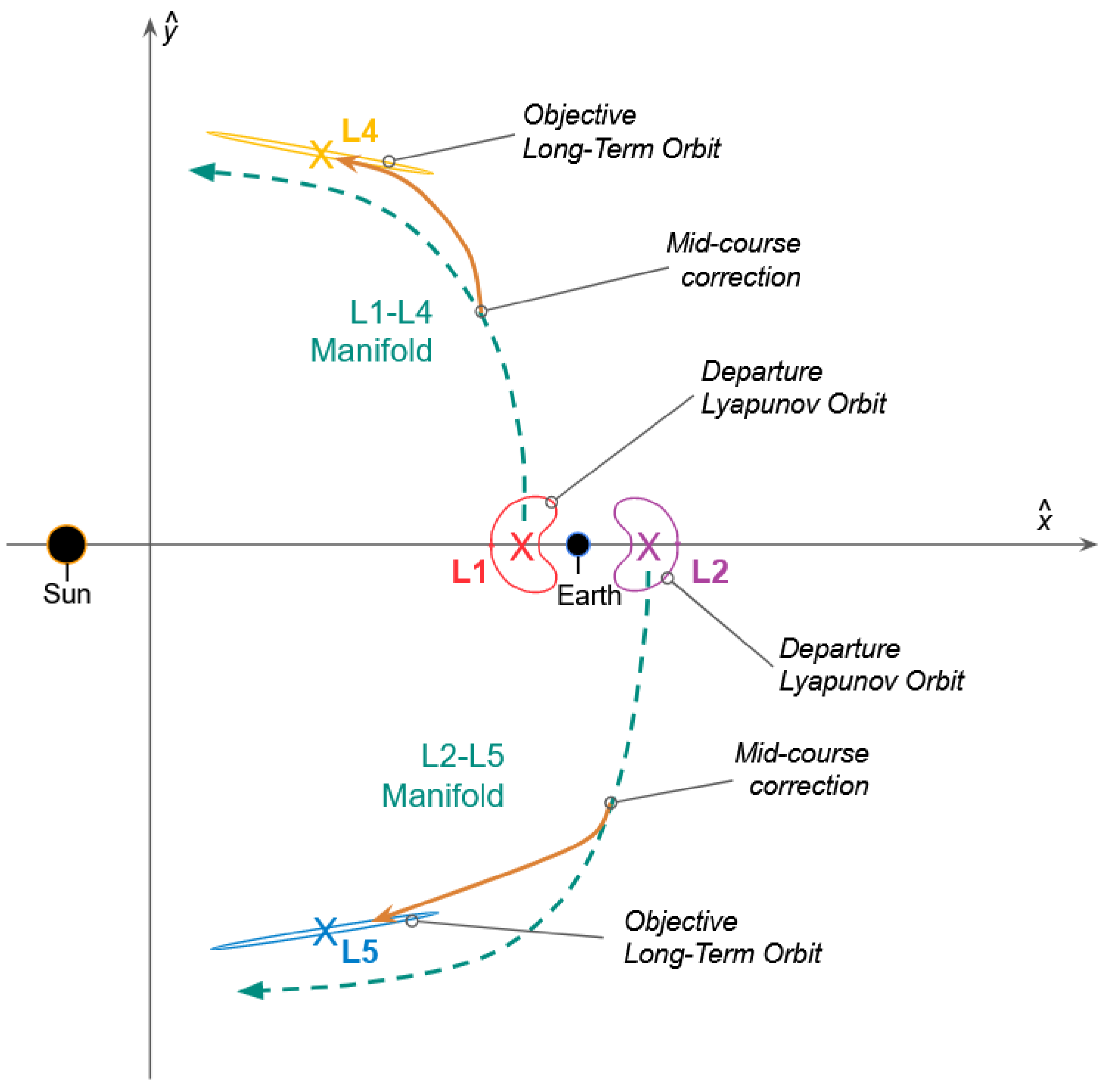


Figure 1.1: Schematic visualization of the paths chosen to Lagrange Point 4 - on the orbit of the smaller primary, ahead of it - (L4) and Lagrange Point 5 - on the orbit of the smaller primary, behind it - (L5) taking advantage of the unstable manifolds originating at Lagrange Point 1 - between the two main primaries - (L1) and Lagrange Point 2 - on the far side of the smaller primary - (L2).

1.2 Brief Introduction and Justification

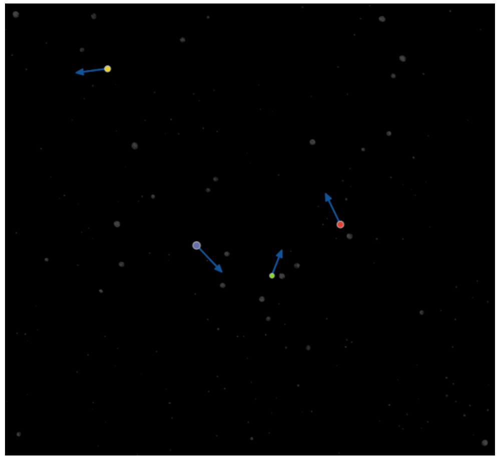
Newton's formulation of the Law of Universal Gravitation [6] (1.1) gave scientists a tool of unprecedented exactitude to describe the movement of every celestial body. Nevertheless, this tool came with intrinsic difficulties, namely its chaotic nature. When three or more bodies with mass are the object of study, even without taking into account other forces at play (not to mention relativistic behaviours), the system quickly and undoubtedly shows its chaotic behaviour. This is known as the n-body problem [7], and besides its chaotic nature, it has been proven to have no analytical solution for 3 or more bodies; although numerical methods may solve it with remarkable exactitude [8].

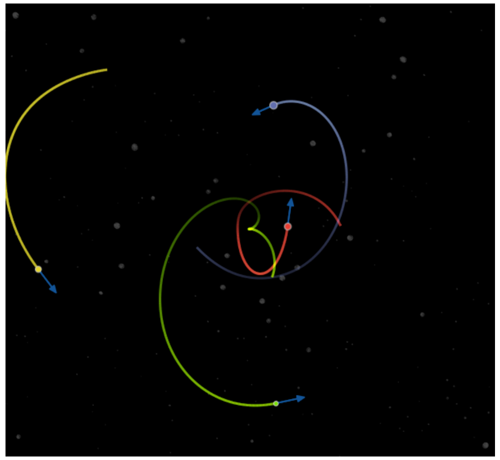
$$F = -G\frac{m_1 m_2}{r^2} \tag{1.1}$$

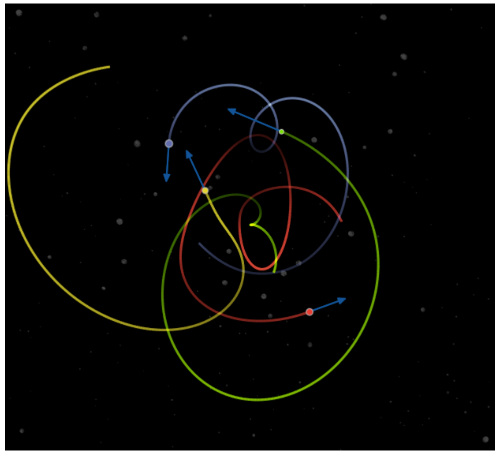
As an example, Figure 1.2a illustrates an isolated system with four bodies, all with different masses and starting velocities. Note that the diameters of the dots are directly proportional to their masses, while the blue arrows represent instantaneous velocities. Using numerical methods, the trajectory of the masses can be propagated to, for instance, $5\ \mathrm{s}$ and $10\ \mathrm{s}$ into the future (see Figure 1.2b and Figure 1.2c). Trajectories twist around and intertwine in complex paths, as in every instance of time, the gravitational pull that each body exerts on the other three changes directions. Note that the continuous coloured lines represent their past trajectories, starting at $\mathrm{t} = 0\mathrm{s}$.

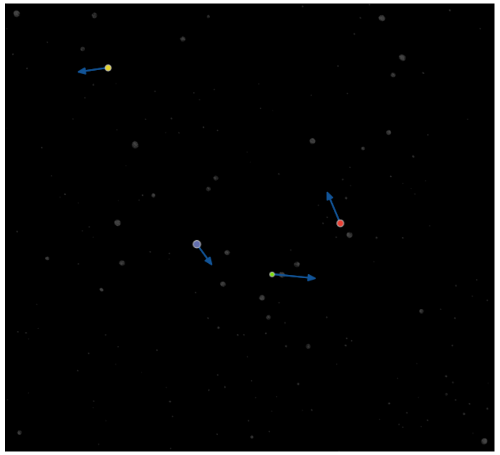
Interestingly, though, if the starting conditions are modified even minimally, the outcome changes significantly. Following the previous example, if the bodies are maintained in the same starting positions and with the same masses, but with a slight variation of the starting velocities (see Figure 1.2d). Note how the starting state has been only slightly altered, with the only variation being of less than 20% in the initial velocity modulus and no changes in the direction but in one of the bodies, at around $30\ \mathrm{deg}$. Yet the outcome at the same time stamps, $\mathrm{t} = 5\ \mathrm{s}$ and $\mathrm{t} = 10\ \mathrm{s}$ drastically changes (see Figure 1.2e and specially Figure 1.2f, where even the yellow body has gone too far for it to be shown in the picture). So, as Lorenz put it, *"Chaos [is] when the present determines the future, but the approximate present does not approximately determine the future."* [9]

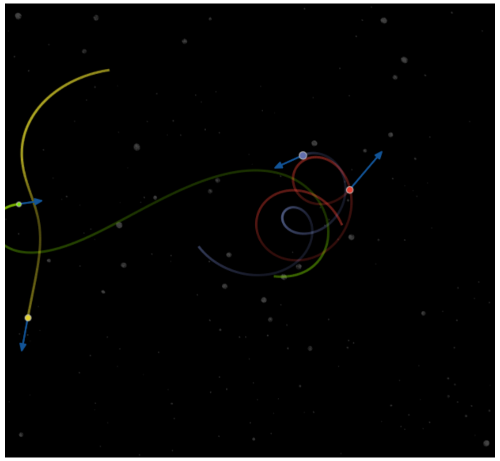
In the midst of this chaos, however, marginally stable situations do exist. While they do not necessarily exist in every possible arrangement of bodies with mass, they can be found. In the case of this bachelor final thesis, focus is made on the Lagrange Points [1, 2]. A first simplified approach (having static bodies, and with no centrifugal forces at play) to these would be an equilibrium point like the one depicted in Figure 1.3, where the gravitational pull of the two bodies cancel out like $\overrightarrow{\mathrm{F}_{\mathrm{m1}}} = -\overrightarrow{\mathrm{F}_{\mathrm{m2}}}$.


(a) The example four-body system at $\mathsf{t} = 0\mathsf{s}$.


(b) The example four-body system at $\mathsf{t} = 5\mathsf{s}$.


(c) The example four-body system at $\mathsf{t} = 10\mathsf{s}$.


(d) The slightly modified four-body system, at $\mathsf{t} = 0\mathsf{s}$.


(e) The slightly modified four-body system, at $\mathsf{t} = 5\mathsf{s}$.

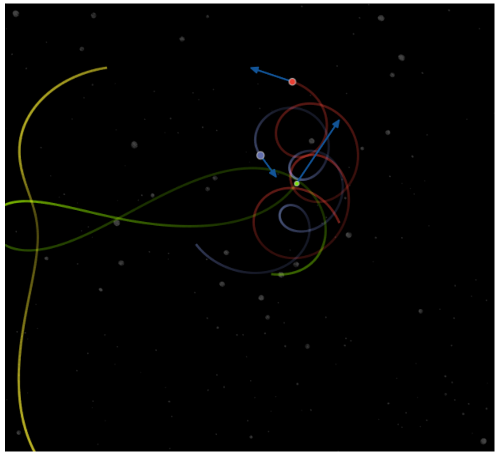

(f) The slightly modified four-body system, at $\mathsf{t} = 10\mathsf{s}$.

Figure 1.2: An example of the chaotic nature of the n-body problem simulated using *MinuteLabs.io*'s algorithm and User Interface [10].

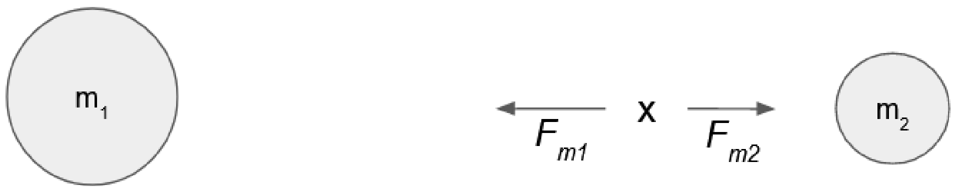


Figure 1.3: Two stationary masses, $\mathsf{m_1}$ and $\mathsf{m_2}$, with no other forces at play, create an equilibrium point (**x** in the figure) where the gravitational pull of the two bodies cancel out like $\overrightarrow{\mathsf{F_{m1}}} = -\overrightarrow{\mathsf{F_{m2}}}$.

This mental image, albeit useful, does miss the complexity that makes Lagrange Points such interesting places. Thus, instead of describing them from two stationary masses, let us use the mathematical formulation of the Circular-Restricted Three-Body Problem (CR3BP). The reasons why this simplification makes a lot of sense for the Earth-Sun system will be discussed later (see Part 2), but for now the CR3BP will be a model to be able to solve multi-body dynamics and its chaos with more ease; where the only masses that will be considered will be the Sun ($\mathsf{m_1}$), the Earth ($\mathsf{m_2}$), and the spacecraft ($\mathsf{m_3}$).

In this simplified model, a total of five points exist where the third mass $\mathsf{m_3}$ can be placed in force equilibrium, where all involved forces, real and fictitious, cancel out [11]. These are the Lagrange Points and are named accordingly, depending on their location; L1, L2, Lagrange Point 3 - on the orbit of the smaller primary, opposite to it - (L3), L4, and L5 (see Figure 1.4). Note that Lagrange Points 1 to 3 can also be referred to as Eulerian points. Nonetheless, for the sake of simplicity and for it being quite consistent in the literature, the arguably arbitrary[i] naming convention of Lagrange Points 1 to 5 will be kept.

The main point of using CR3BP will be to describe two new trajectories to L4 and L5. These two points are of special interest because, unlike L1, L2, and L3; L4 and L5 are not only statically stable (Figure 1.3), but also dynamically stable [11]. This can be graphically understood by mapping the effective pseudo-potential energy that the third body has at each point in this CR3BP space (Figure 1.5). There, L4 and L5 are clearly seen as hilltops, while L1, L2, and L3 are saddle points. The stability of these two points makes them not only privileged positions for Sun observation, as few missions have noticed and used before [3]; but it has become more relevant in recent years, as it has been confirmed as a parking spot for "Trojan" asteroids [4, 5].

[i]It can be argued that they are ordered this way because of the descending values the Jacobi constant has at each point when numbered this way, and because L4 is ahead of the smaller primary while L5 is behind it. Nevertheless, choosing the values of the Jacobi constant in a descending way is, again, an arbitrary yet widely used convention.

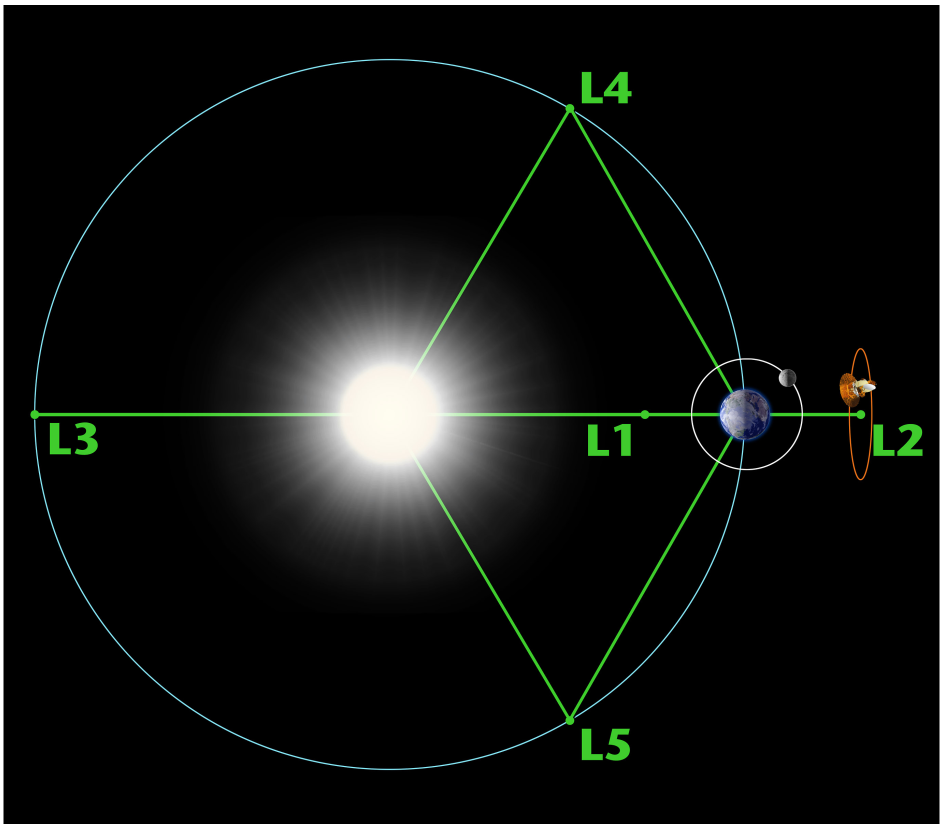


Figure 1.4: The five Lagrange Points of the Earth-Sun system (green); Earth's orbit (blue), the Moon's orbit (grey), and the James Webb Space Telescope Halo orbit around L2 (orange). Figure from Barnett et al. [2].

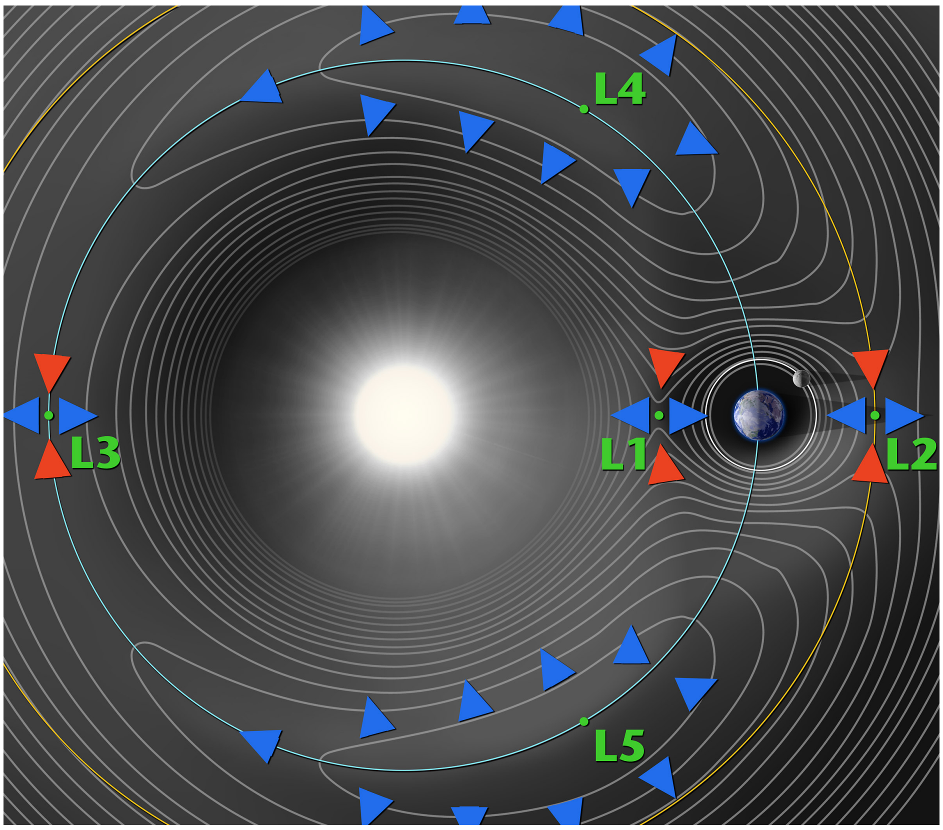


Figure 1.5: Contour map of the pseudo-potential energy in the CR3BP Earth-Sun system. Arrows show whether the gradient is "uphill" or "downhill" from each Lagrange Point. Figure from Barnett et al. [2].

The confirmed existence of Earth Trojan asteroids makes going to L4 and L5 specially relevant for asteroid science, since transfers there can be easier than transfers to the asteroid belt. Plus, the Trojan asteroids are, thanks to their very nature as objects located around the Lagrange Points, located in more predictable orbits. Even towing asteroids there may prove a better idea than to any other Lagrange Point or Low Earth Orbit (LEO); for orbits around L4 and L5 are more stable and located at a safer distance from Earth.

Nevertheless, and while not many missions have exploited these Lagrange Points to their advantage, L4 and L5 are not new discoveries. The present work, instead, aims to describe a new way to reach them with less energy requirements from LEO. The main idea is to make use of the unstable nature of L1 and L2 as staging grounds for the transfer to L4 and L5. This instability will lead to a trajectory leading to the vicinity of L4 or L5, in a way that only trajectory-correction burns will be needed. This has been described by Farrés et al. using solar sails [12], and Sood et al. [13], but its usability and advantages using traditional propulsion remains to be tested.

Thus, by proposing new, more energy-efficient, transfers to L4 and L5 of the Earth-Sun system, this study aims to contribute to what humankind can visit, explore, and learn from our cosmic neighbourhood that is the Solar System.

1.3 Scope

This project is to provide, in its final form:

- A comprehensive overview of the formulation needed to create the CR3BP model.
- Detailed method of application of the CR3BP model to the Sun-Earth system.
- The obtaining of periodic orbits around L1, L2, L4, and L5 using CR3BP approximation.
- The obtaining of transfer orbits between L1/L2 and L4/L5 using CR3BP approximation.

Conversely, the following topics fall out of the scope of this thesis:

- Detail the return transfers from L4 and L5 to LEO or any other Lagrange Point.
- Propose, suggest, nor detail characteristics of potential spacecraft would need to perform the described transfer to L4 and L5.
- Propose a mission that would use the described transfer to L4 and L5.
- Take into account the burn time needed to perform the velocity changes needed to enter the proposed transfer orbits. All burns will be assumed instantaneous, as the transfers are meant for traditionally-propelled spacecraft.

It will include the following high-level deliverables:

- Project Charter.
- Project Budget (available in appendix B).
- Compressed file with the code and algorithms produced (available in appendix A).
- Final thesis.
- Presentation.

1.4 Requeriments

In this section, the basic requirements, specifications, and restrictions needed to consider this thesis finished will be described.

- **Numerical orbit propagator**: The first requirement is to develop a computational algorithm to numerically solve the CR3BP given any initial conditions, so the orbit of the third body can be propagated forward through time by just calling a function.
- **Identification of the location of L1, L2, L4, and L5**: The fifth degree equations that describe the location of the Lagrange points for a given system need to be solved for the Earth-Sun case, and the mapping of their location with the necessary precision (10^{-10} non-dimensional units)[ii].
- **Identification of key periodic orbits around L1, L2, L4, and L5**: Both the departure orbits around L1 and L2 and the destination orbits around L4 and L5 shall have to be mapped to accurately describe the transfer paths between them.
- **Optimization of Transfer Orbits**: Initial conditions needed for the orbits around L1 and L2 to enter the found trajectories to L4 and L5, and the needed corrections to enter the final target orbits around these two last Lagrange points. These corrections will be optimized, regarding at least one parameter, to minimize the changes of velocity required to perform the transfer.

Overall, the main goal for every requirement is that the resulting origin and destination orbits, and transfers are proven possible with an error tolerance of less than 10^{-5} non-dimensional units[iii].

[ii]Non-dimensional units refer to how magnitudes are traditionally non-dimensionalized in the CR3BP. Exact method and characteristic magnitudes for this non-dimensionalization will be further detailed in the next part.

[iii]Idem

Part #02

To L4 and L5 in the CR3BP

2.1 Theoretical Framework

CR3BP is the model that is to be used for the resolution of the complex and chaotic behaviour of orbital mechanics in the Solar System. The brief introduction provided in Part 1 is expanded as it follows.

The CR3BP Sun-Earth model is based upon with the following assumptions [11, 14]:

a) Only point masses exist in the studied universe. All the mass of any of the bodies is thus concentrated at its centre, ignoring in this way the effects that the actual mass distribution around the body has on the others and on itself.

b) Only three point masses exist in the studied universe; with one of them being significantly smaller in mass when compared to the other two, in a way in which its mass can be neglected ($\mathsf{m}_1 \geq \mathsf{m}_2 >> \mathsf{m}_3$). Thus, a Three-Body Problem.

c) The two main masses will orbit the barycentre of the system in fixed circular orbits, with only the path of the third body free and to-be-determined. Thus, Circular Restricted.

d) No relativistic effects nor acceleration from Solar Radiation Pressure is to be considered.

e) Any velocity change the spacecraft does will be considered instantaneous.

This set of assumptions is justified as being a fairly factual approximation to a real life scenario where m_1 is the Sun, m_2 is the masses of the Earth and the Moon combined, and m_3 is the hypothetical spacecraft; for:

a) The studied orbits will be far enough from the two main point masses for any perturbations to the third body orbit, to be relevant enough. Figure 2.1 depicts an example of the perturbations that may have to be taken into account for orbits around Earth, and their impacts). There will be no operations in situations where these perturbations are significant enough to be considered (for instance, LEO, the Earth atmosphere, the Sun atmosphere, etc).

b) As stated before, the area where the spacecraft, m_3, will not be found in the vicinity of either the main point masses, so any imperfections in their gravitational attraction will be considered negligible. To this simplification another is added, for the mass of the Moon is added to the point-mass of the Earth with the goal of making the model more realistic. The mass of the Moon is a quarter of that of the Earth [15] and, despite the Sun being several orders of magnitude more massive than any combination of those two bodies, adding such a massive body like the Moon has been considered adequate to achieve more precision.

   Besides that, there is the fact that the rest of the bodies of the solar system will be ignored. As stated before, this is a valid hypothesis because the Sun represents $99,8\%$ of the mass of the Solar System [15]; and because the area of operations, where all the Lagrange Points are located, is in the vicinity of the orbit of the Earth around the Sun and, thus, the perturbations that other masses of the Solar System could cause would be significantly less than in zones further away from Earth's orbit around the Sun.

c) The orbit the Earth follows around the Sun is almost perfectly circular. The eccentricity of its orbit is

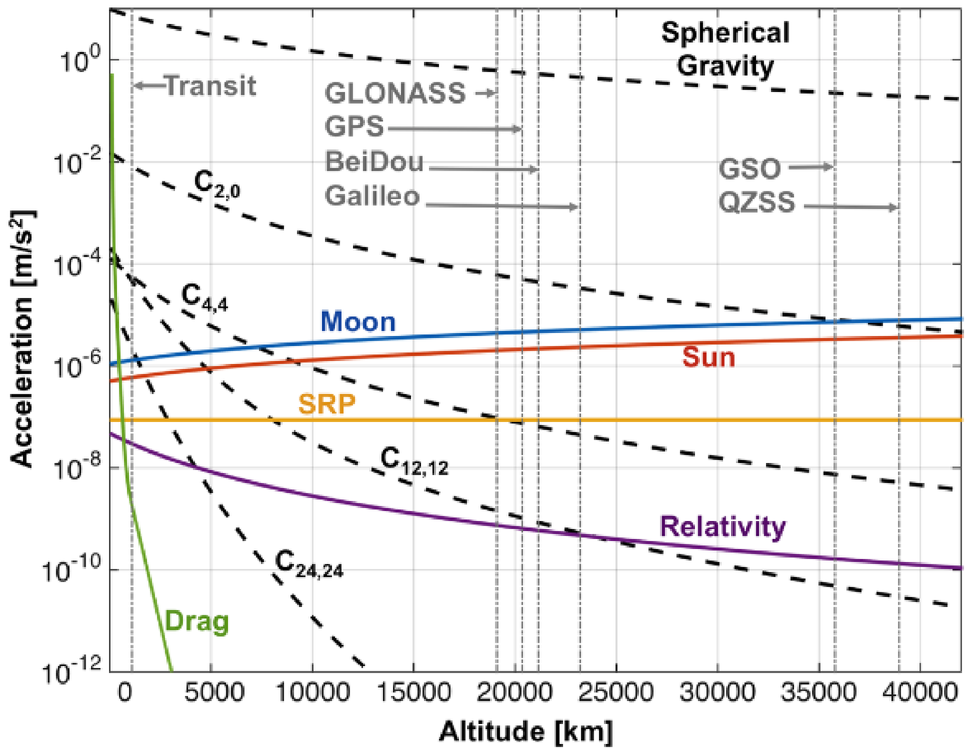


Figure 2.1: Orbital perturbations around Earth, as a function of altitude. Figure from Reid [16].

between 0,0034 and 0,058, with this variation defined by the 100.000 year-long Milankovitch cycles. Not only it would still be safe to assume a perfectly circular orbit at the peak of these cycles, but added to that this cycle currently (at the time of writing) is near its most circular, slowly increasing its eccentricity [17]; thus making it an even better hypothesis.

d) Relativistic effects can be safely considered negligible because the velocities that the third body will be reaching will be very far from that of the speed of light; and because relativistic effects that would arise from being too close to any of the gravitational wells of the two main bodies are even less than the perturbations provoked by their lack of perfectly spherical gravity – which in turn has been already considered negligible (again, see Figure 2.1 with orbits near the Earth as an example of the effects these perturbations have on orbiting spacecraft). Solar Radiation Pressure is also safe to be ignored, for it has minimal impact when compared to the other assumptions, based on the primary assumption that the spacecraft is not using Solar Sail technology.

e) The spacecraft will be assumed to use traditional, chemical, propulsion systems. Thus, considering the changes in speed instantaneous is an acceptable assumption.

With the assumptions out of the way, it is time for a deeper dive into the CR3BP.

2.1.1 Reference Frames

The usual Reference Frame (RF), the most known for our Solar System, is a fixed one with the Sun at its centre (an inertial or Galilean RF). In this RF, the Sun is, as stated, at the centre of the coordinate system, with the x-axis pointing in the direction of the Earth. The z-axis is defined by the angular velocity vector (ω) of the Earth orbit, and the y-axis is perpendicular to those two, following the right-hand rule (see Figure 2.2).

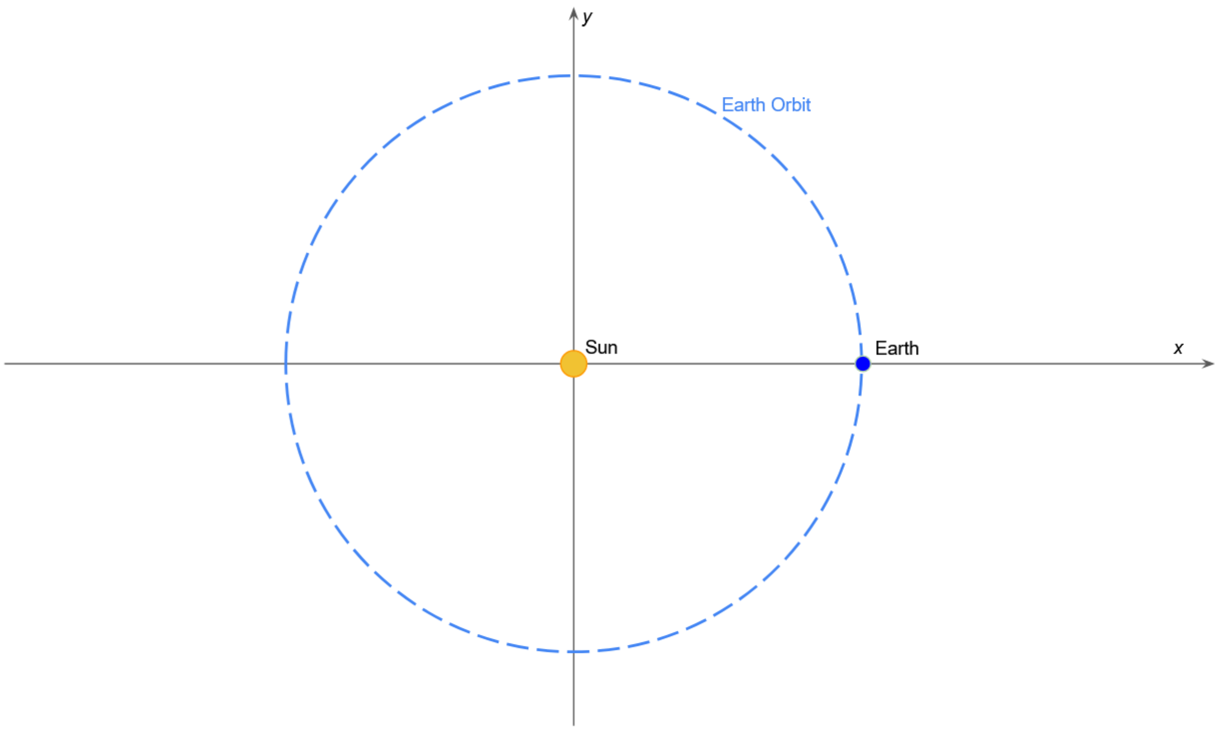
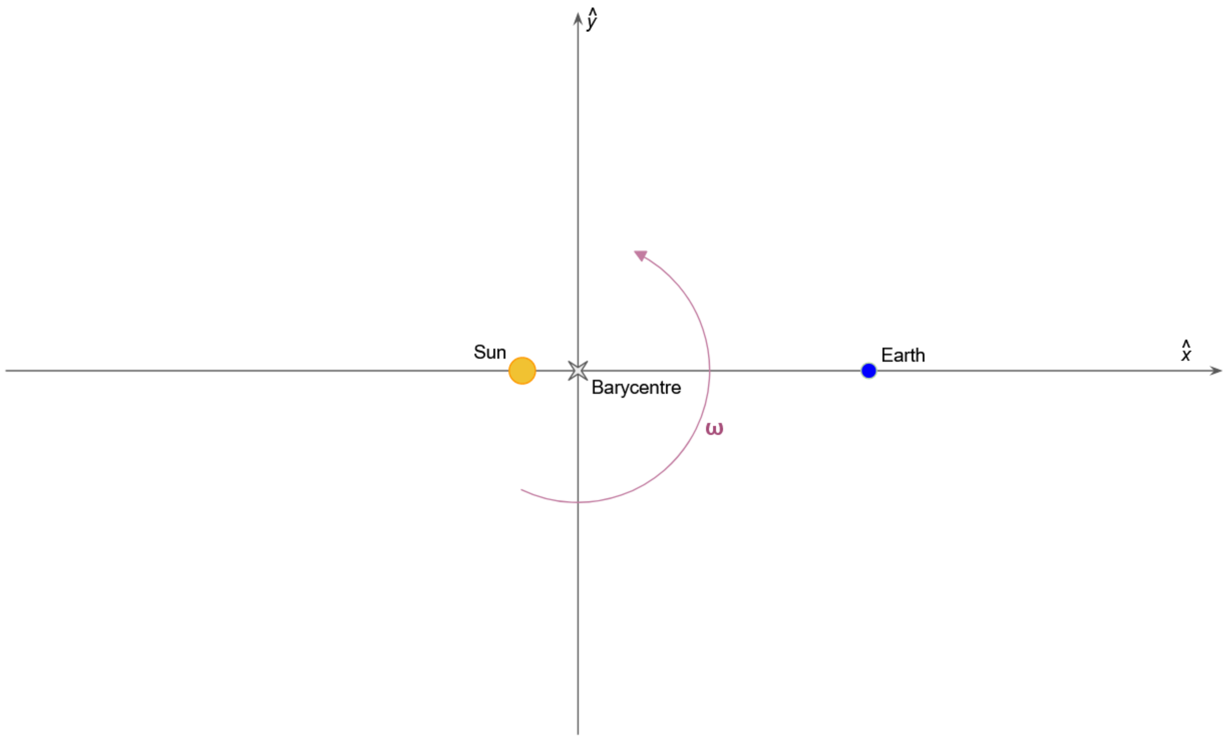


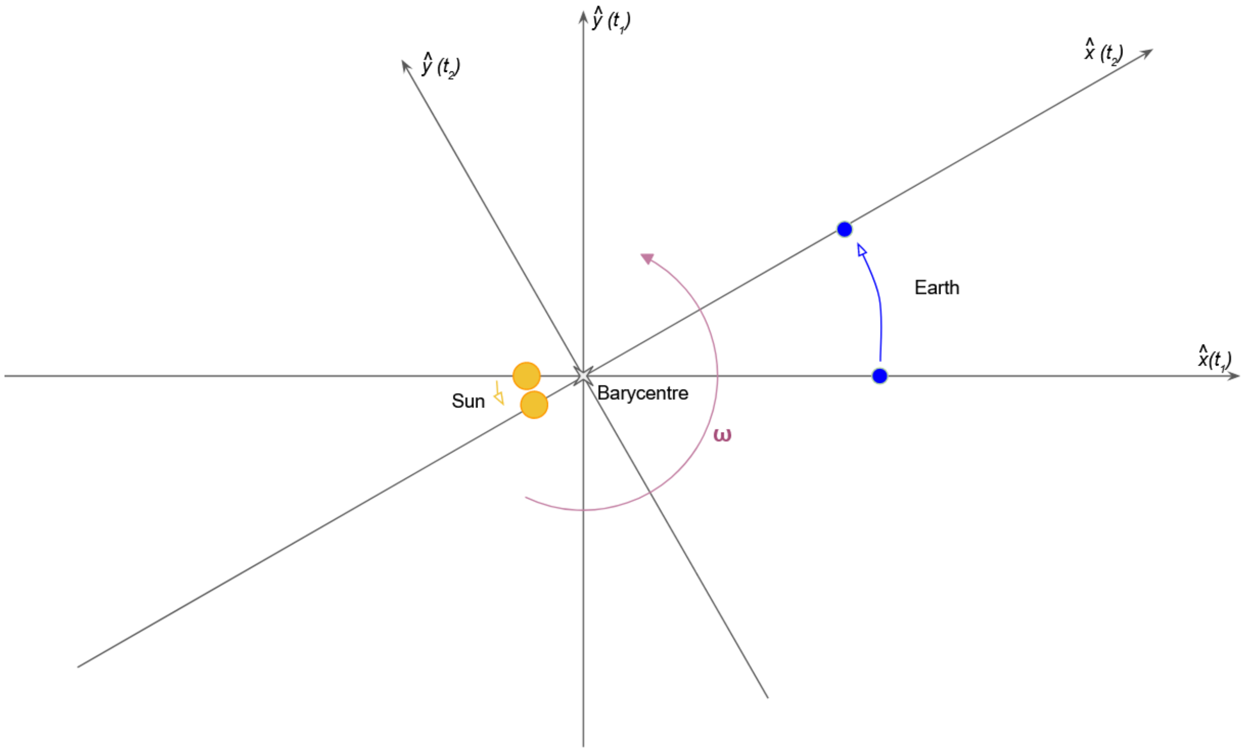
Figure 2.2: Fixed RF for the Sun-Earth system.

This approach, however, will rarely be used in the studied case, for having one of the gravitational wells moving around in this way will make things unnecessarily complex. Instead, a rotating RF is to be defined. This will be, by definition, non-inertial, and will have its centre and origin at the barycentre (or centre of masses) of the two main bodies. This rotating RF will rotate with the angular velocity (ω) of the Earth orbit, and again its direction will define the z-axis, with the x-axis being again defined by the connecting line between the Sun and the Earth. The y-axis will be perpendicular to these two and, thus, contained within the orbital plane (see Figure 2.3a, please note the change in notation in the axis, from x and y in the fixed RF to $\hat{x}$ and $\hat{y}$ in the rotating RF).

Using a rotating RF, means that fictitious forces will have to be taken into account when computing the forces that act on the third body. It is important to keep in mind that any graphic representation in the rotating RF is not static in time. Except for the barycentre, any fixed point in the rotating RF is actually moving around the barycentre with an angular velocity of ω (see Figure 2.3b).



(a) Rotating RF for the Sun-Earth system.



(b) The rotating RF at two different instants of time t_1 and t_2 from a fixed RF point of view

Figure 2.3: Two figures representing the nature of the chosen rotating RF for the Sun-Earth system.

2.1.2 Characteristic magnitudes and Non-dimensionalization

Working with full-length units of mass, time, and space is not the most agile thing to do with the vast magnitudes seen in planetary-sized problems. Not only that, but by working a way around units, CR3BP can become a general problem to be adapted to any two combination of three bodies, not limited to the case of study (the Sun-Earth system). Non-dimenzionalization is, thus, the best way to work with the problem at hand and also a way to generalize the system for any similar mass relation between the two primaries [11, 14, 18].

To do that, characteristic magnitudes are going to be used. This way, when any magnitude is divided by a characteristic magnitude of the same unit type, the result is a non-dimensional magnitude, with Non-Dimensional Units (NDU) (2.1).

$$\frac{magnitude\ [units]}{characteristic\ magnitude\ [units]} = non - dimensional\ magnitude\ [NDU] \tag{2.1}$$

$$m^* = m_1 + m_2 \tag{2.2}$$

$$l^* = l_1 + l_2 \tag{2.3}$$

$$t^* = \sqrt{\frac{l^{*3}}{Gm^*}} = \frac{2\pi}{\omega} \tag{2.4}$$

For the CR3BP, three main characteristic magnitudes will be at play: characteristic mass m^* (2.2), characteristic length l^* (2.3), and characteristic time t^* (2.4). Note that from now on, G will be the Universal Gravitational constant [6], m_1 and m_2 will be the masses of the two main primaries, and l_1 and l_2 will be the distances from the barycentre to the two main primaries. ω will remain as the angular velocity at which the two main primaries orbit the barycentre [11, 14, 18].

Nevertheless, it will be of interest to introduce a fourth characteristic magnitude: the mass parameter μ (2.5), that not only will be used to non-dimensionalize the primary masses and will allow for a relation between them and the spatial dimension.

$$\mu = \frac{m_2}{m^*} = -\frac{l_1}{l^*} \implies 1 - \mu = \frac{l_2}{l^*} \tag{2.5}$$

The fact that the mass parameter conveniently corresponds with the location on the x-axis of the primary body (once the sign is reversed) in NDU is quite easy to see from the definition of the barycentre [18]. If one is to play around a bit, with it, the previous definition of μ appears naturally (2.6, 2.7, and 2.8). From these, (2.5) is obtained. Note that the handy transformation $1 - \mu = m_1/m^*$ has been used. Note also how, when using it as a coordinate, l_1 needs a negative sign since the m_1 is located on the negative side of the x-axis (Figure 2.3a).

$$x_{barycentre} = 0 = \frac{-l_1 m_1 + l_2 m_2}{m^*} \tag{2.6}$$

$$0 = (\mu m^* - m^*)l_1 + \mu m^*(l^* - l_1) \tag{2.7}$$

$$0 = -m^* l_1 + \mu m^* l^* \tag{2.8}$$

Having defined that, it is time to dive to the concrete problem to be analysed: the Sun-Earth system. In this case, the main body, m_1, will undoubtably be the Sun. For the secondary mass m_2, the combined masses of the Earth and the Moon will be used. This criterium, while not consistent in the literature, is deemed adequate, for it does not dramatically alter the final mass parameter of the problem, yet does take into account the mass of the Moon, which at around a sixth part that of the Earth, is relevant enough to not be ignored. That being said, the exact magnitudes used can be consulted in Table 2.1, with data from Park et al. from National Aeronautics and Space Administration (NASA) - Jet Propulsion Laboratory (JPL) [15].

Parameter [units]	**Magnitude**
$G\ [m^3kg^{-1}s^{-2}]$	$6,67430 \cdot 10^{-11}$
$Gm_{Sun}\ [m^3s^{-2}]$	$1,32712440018 \cdot 10^{20}$
$Gm_{Earth}\ [m^3s^{-2}]$	$3,98600435507 \cdot 10^{14}$
$Gm_{Moon}\ [m^3s^{-2}]$	$4,902800118 \cdot 10^{12}$
$l_1 + l_2\ [m]$	$1,49597870700 \cdot 10^{11}$

Table 2.1: Planetary data for the Sun-Earth system. Extracted from Park et al. (JPL) [15].

2.1.3 Governing Equations

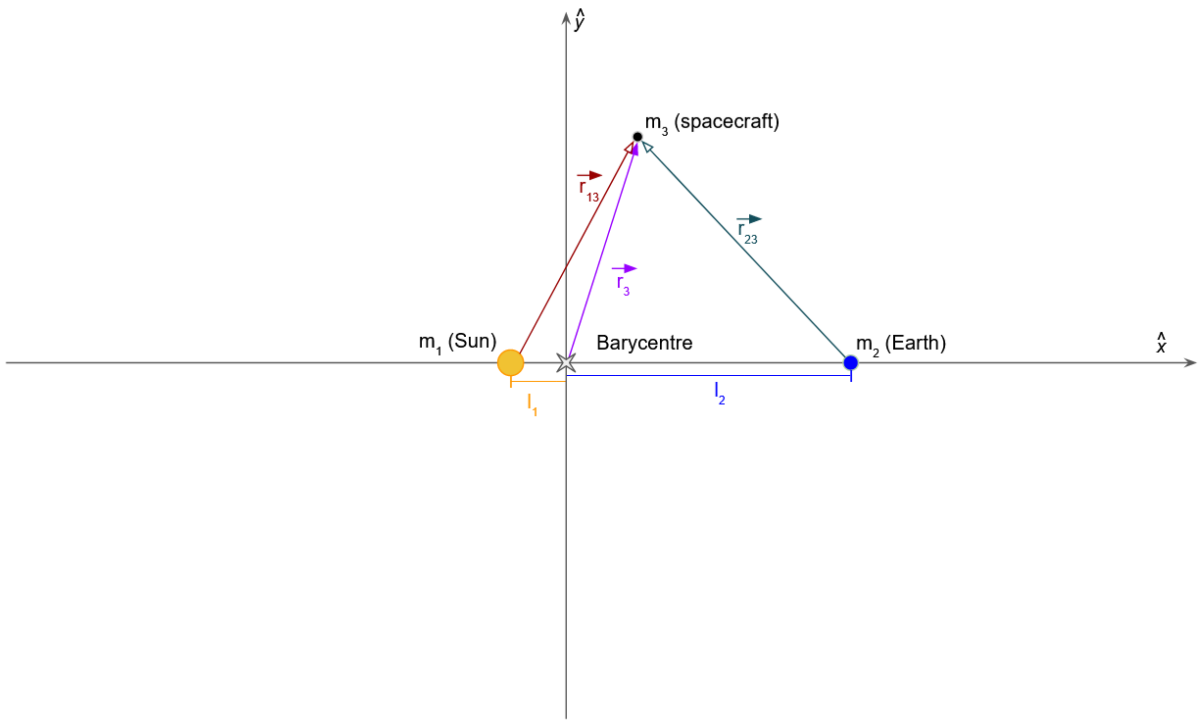


Figure 2.4: CR3BP representation in the rotating RF.

With that out of the way, it is time to talk about the problem properly. In CR3BP, the paths that the two main bodies, m_1 and m_2, are to follow are set as circular orbits around the barycentre, leaving the third, m_3, to move around attracted by the pull of the other two. To that effect, l_1 and l_2 have been previously described as the distances between two main primaries and the barycentre; to which will be added $\overrightarrow{r_3}$, as the vector from the barycentre to the third body (in the studied case, a massless spacecraft), as well as $\overrightarrow{r_{13}}$ and $\overrightarrow{r_{23}}$ as the vectors from m_1 and m_2 to m_3, respectively (see Figure 2.4). Note that this is an $\mathbb{R}^3$ space, meaning that all vectors have three components [11].

The problem to be solved, then, is the one regarding where m_3 is located and how its location evolves over a period of time. Or, what is the same, the evolution of $\overrightarrow{r}$ with time. To that effect, Newton's Second Law of Motion applied to the third body will come handy as a starting point (2.9), where $\overrightarrow{a}$ is the third body's acceleration in absolute terms. This equation will be modified to adapt it to the fact that it is operating in a rotating RF, in a way that the left side is left with the sum of the two gravitational pulls. On the other side, only the relative, centrifugal, and Coriolis accelerations are left (2.10). Note that $\overrightarrow{a_r}$ and $\overrightarrow{v_r}$ are the relative acceleration and velocity of m_3 in relation to the rotating RF respectively.

$$\sum_{i=1}^{n} \overrightarrow{F_i} = m_3 \cdot \overrightarrow{a} \tag{2.9}$$

$$\frac{\sum_{i=1}^{2} \overrightarrow{F_i}}{m_3} = \overrightarrow{a_r} + 2\overrightarrow{\omega} x \overrightarrow{v_r} + \overrightarrow{\omega} x (\overrightarrow{\omega} x \overrightarrow{r_3}) \tag{2.10}$$

Completing the left-hand side with the already discussed Law of Universal Gravitation (1.1), it can be seen that the value of m_3 disappears (2.11). It is important to note that the fact that m_3 is no longer be relevant is not due to a mathematical conclusion, but because the assumptions of the CR3BP lead to this equations making sense. Note that for simplicity, the vector form of the Law of Universal Gravitation has been applied with the transformation $\overrightarrow{r_n}^{-2} = |\overrightarrow{r_n}|^{-3} \cdot \overrightarrow{r_n}$.

$$-\frac{Gm_1}{|\overrightarrow{r_{13}}|^3}\overrightarrow{r_{13}} - \frac{Gm_2}{|\overrightarrow{r_{23}}|^3}\overrightarrow{r_{23}} = \overrightarrow{a_r} + 2\overrightarrow{\omega} x \overrightarrow{v_r} + \overrightarrow{\omega} x (\overrightarrow{\omega} x \overrightarrow{r_3}) \tag{2.11}$$

Since the objective is to determine the motion of m_3 in the rotating RF, $\overrightarrow{a_r}$ the second derivative with respect to time (t) of $\overrightarrow{r}$, will be the value to be found (2.12).

$$\overrightarrow{a_r} = -2\overrightarrow{\omega} x \overrightarrow{v_r} - \overrightarrow{\omega} x (\overrightarrow{\omega} x \overrightarrow{r_3}) - \frac{Gm_1}{|\overrightarrow{r_{13}}|^3}\overrightarrow{r_{13}} - \frac{Gm_2}{|\overrightarrow{r_{23}}|^3}\overrightarrow{r_{23}} \tag{2.12}$$

Now it will be more useful to use full vector notation for the acceleration terms, and solving the vector products knowing that $\overrightarrow{\omega} = (0\ 0\ \omega)^T$ (2.13). Note that X, Y, and Z are still referring to the position of m_3 with regard to the rotating RF, dimensionalized.

$$\begin{pmatrix} a_X \\ a_Y \\ a_Z \end{pmatrix} = -2 \begin{pmatrix} -\omega v_Y \\ \omega v_X \\ 0 \end{pmatrix} + \omega^2 \begin{pmatrix} X \\ Y \\ 0 \end{pmatrix} - \frac{Gm_1}{|\overrightarrow{r_{13}}|^3}\overrightarrow{r_{13}} - \frac{Gm_2}{|\overrightarrow{r_{23}}|^3}\overrightarrow{r_{23}} \tag{2.13}$$

From here on, non-dimensionalization will be of use. By using the characteristic magnitudes seen in the previous section (2.2, 2.4, and 2.5), and reordering the equations in a way that the unknowns and their derivatives are all at the left-hand side, the set of governing equations in NDU is obtained (2.14). It is quite remarkable to see how detached the equations regarding the ẑ direction are from the other two, allowing the problem to be solved independently for a $\mathbb{R}^2$ case formed by the x̂ and ŷ dimensions. This will be quite useful later on, for it is interesting to study trajectories only taking into account the movement in the x̂ and ŷ directions independently of that in the ẑ one. This makes a $\mathbb{R}^2$ simplified CR3BP analysis an almost equally good approximation to a real-life scenario as long as the motion is restricted to the orbital plane.

$$\begin{cases} \ddot{x} - 2\dot{y} - x = -\frac{1-\mu}{d^3}(x+\mu) - \frac{\mu}{r^3}[x-(1-\mu)] \\ \ddot{y} + 2\dot{x} - y = -\frac{1-\mu}{d^3}y - \frac{\mu}{r^3}y \\ \ddot{z} = -\frac{1-\mu}{d^3}z - \frac{\mu}{r^3}z \end{cases} \tag{2.14}$$

Note how now that NDU are in use, the notation to refer to the position of the third body has changed to (lowercase) x, y, z, and τ (for non-dimensional time t). Similarly, in NDU, $|\overrightarrow{r_{13}}|$ becomes d, while $|\overrightarrow{r_{23}}|$ becomes r (not to be confused with $\overrightarrow{r_3}$).

$$\frac{|\overrightarrow{r_{13}}|}{l^*} = d = |\sqrt{(x+\mu)^2 + y^2 + z^2}| \tag{2.15}$$

$$\frac{|\overrightarrow{r_{23}}|}{l^*} = r = |\sqrt{(x-1+\mu)^2 + y^2 + z^2}| \tag{2.16}$$

Finally, for the acceleration and velocity, a change for simplicity has also been made using overhead dot notation (2.15, 2.16, 2.17, 2.18, and 2.19).

$$\frac{\overrightarrow{r_3}}{l^*} = \begin{pmatrix} x & y & z \end{pmatrix}^T \tag{2.17}$$

$$\begin{pmatrix} v_X \\ v_Y \\ v_Z \end{pmatrix} \frac{t^*}{l^*} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix} = \frac{\delta}{\delta\tau} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} \tag{2.18}$$

$$\begin{pmatrix} a_X \\ a_Y \\ a_Z \end{pmatrix} \frac{t^{*2}}{l^*} = \begin{pmatrix} a_x \\ a_y \\ a_z \end{pmatrix} = \frac{\delta^2}{\delta\tau^2} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{pmatrix} \tag{2.19}$$

2.1.4 Orbit Propagation

Solving the governing equations of CR3BP (2.14) for a given time interval is known in the literature as propagating an orbit [11]. Being able to propagate any orbit for any given starting conditions and for any given time frame is of utmost importance to use CR3BP as a model.

$$\int \frac{\delta s}{\delta \tau} \delta \tau = s \tag{2.20}$$

To that end, it will prove key to define the governing equations in a state space. The state space is perfect because the governing equations are a set of ordinary differential equations, so defining a state space (s) with as well as its first derivative with respect to time ($\delta \mathsf{s}/\delta\tau$) allows for the easy numerical integration of the problem given any initial state s (2.20, 2.21, and 2.22). Note that in the later two equations $\delta\mathsf{s}$ is used instead of the full notation of $\delta\mathsf{s}/\delta\tau$ for simplicity.

$$\begin{cases} \mathsf{s} = (\mathsf{x}\ \mathsf{y}\ \mathsf{z}\ \dot{\mathsf{x}}\ \dot{\mathsf{y}}\ \dot{\mathsf{z}})^\mathsf{T} \\ \delta\mathsf{s} = (\dot{\mathsf{x}}\ \dot{\mathsf{y}}\ \dot{\mathsf{z}}\ \ddot{\mathsf{x}}\ \ddot{\mathsf{y}}\ \ddot{\mathsf{z}})^\mathsf{T} \end{cases} \tag{2.21}$$

$$\begin{cases} \dot{\mathsf{x}} = \delta\mathsf{s}(1) = \mathsf{s}(4) \\ \dot{\mathsf{y}} = \delta\mathsf{s}(2) = \mathsf{s}(5) \\ \dot{\mathsf{z}} = \delta\mathsf{s}(3) = \mathsf{s}(6) \\ \ddot{\mathsf{x}} = \delta\mathsf{s}(4) = 2\mathsf{s}(5) + \mathsf{s}(1) - \frac{1-\mu}{\mathsf{d}^3}(\mathsf{s}(1)+\mu) - \frac{\mu}{\mathsf{r}^3}[\mathsf{s}(1) - (1-\mu)] \\ \ddot{\mathsf{y}} = \delta\mathsf{s}(5) = -2\mathsf{s}(4) + \mathsf{s}(2) - \frac{1-\mu}{\mathsf{d}^3}\mathsf{s}(2) - \frac{\mu}{\mathsf{r}^3}\mathsf{s}(2) \\ \ddot{\mathsf{z}} = \delta\mathsf{s}(6) = -\frac{1-\mu}{\mathsf{d}^3}\mathsf{s}(3) - \frac{\mu}{\mathsf{r}^3}\mathsf{s}(3) \end{cases} \tag{2.22}$$

Here can be seen again how detached this problem is between the $\hat{\mathsf{x}}$ and $\hat{\mathsf{y}}$ dimensions, and the $\hat{\mathsf{z}}$ dimension. As long as $\mathsf{z} = \dot{\mathsf{z}} = \ddot{\mathsf{z}} = 0$, a 2-D analysis can be made without issue. Separately, detached analysis can be done even in the cases with movement in the $\hat{\mathsf{z}}$ axis. This problem then can be integrated numerically using a myriad of methods. In this work, Python [19] and the library SciPy have been chosen because of its broad popularity, ease-of-use, and overall major maintenance and reliability when compared when other Python libraries [20]. While the full code can be consulted in Appendix A; in short, the way it solves the ordinary differential equation is by applying an explicit 8th order Runge-Kutta method, as *"[it] should be used for non-stiff problems [. . . , and it is] recommended for solving with high precision"* [21]. Interestingly, this solver needs not the step size, but rather uses a tolerance cap which can be set at will. For this work, a relative tolerance of 10^{-10} NDU and an absolute tolerance of 10^{-13} NDU are used. Note that due to the decoupling that exists regarding the $\hat{\mathsf{z}}$ direction, the stability that exists when operating within the $\mathsf{x}-\mathsf{y}$ plane, and how only operating in 2-D greatly reduces the computational load when integrating these equations, this study has been done assuming planar motion. This is also possible because the manifolds (which will be descrived in-depth in Section 2.4.2.), that connect L1 and L2 to L4 and L5 can be traced also within the $\mathsf{x}-\mathsf{y}$ plane, thus eliminating the need of also computing unneeded displacements within the $\hat{\mathsf{z}}$ direction.

2.2 Equilibrium: the Lagrange Points

Lagrange Points are the main focus of study for this work. Understanding how they come to be and finding their location is, thus, of utmost importance. Note that, due to this work being centred in L1, L2, L4, and L5; while L3 will be discussed but not described as in-depth as the others.

2.2.1 Pseudo-Potential Function and the Jacobi Constant

Since gravity is a conservative force, it can be interesting to define a potential field Ω, from which the divergence of such is just the desired force (2.23). For the classic two-body problem this is not a complex operation, but for CR3BP the following steps are needed.

$$\nabla_{(\vec{r})}\Omega(\vec{r}) = \vec{F_r} \tag{2.23}$$

Since the operating RF is rotating, it is more appropriate to define it as a pseudo-potential field Ω^* (2.24). This field will be of remarkable interest because of the conclusions that will be extracted when manipulating it, plus it will be used to define undoubtably useful mathematical tools such as the Jacobi constant (Jc) [11].

$$\Omega^*(\vec{r_3}) = \frac{1-\mu}{d} + \frac{\mu}{r} + \frac{1}{2}(x^2+y^2) \tag{2.24}$$

This allows for the rewriting of the equations of motion in terms of this pseudo-potential function (2.25) [14].

$$\begin{cases} \ddot{x} - 2\dot{y} = \frac{\delta\Omega^*}{\delta x} \\ \ddot{y} + 2\dot{x} = \frac{\delta\Omega^*}{\delta y} \\ \ddot{z} = \frac{\delta\Omega^*}{\delta z} \end{cases} \tag{2.25}$$

From the pseudo-potential function, the definition of the Jc can also be extracted using a series of energy integrals [11, 14]. The Jc (2.26) is a single number representation of a given state of m_3. It is useful to think it “analogous” to the mechanical energy of the third body. However, note when doing this association that when the mechanical energy increases, the Jc decreases, and vice versa.

$$J_C = 2\Omega^* - |v|^2 \tag{2.26}$$

One of the main uses of the Jc is the description it gives of the forbidden areas that exist for a given state of the third body. By plotting the intersection of a zero-velocity surface ($|v| = 0$) with the pseudo-potential field as defined with the Jc (2.27), for a given Jc, one will be able to see the space of possible locations for m_3 for that given Jc. This is to mean that to get to certain points in the CR3BP space, a lower Jc might be necessary (for instance, both equilateral Lagrange Points L4 and L5 need a remarkably low Jc for them to be accessible [11]).

$$2\Omega^* - J_C = 0 \tag{2.27}$$

2.2.2 Existence

The fact that the Lagrange Points do exist can be proven with a few equations, that will hint their actual location. First, though, it is necessary to understand what is a Lagrange Point. A Lagrange Point is defined as that point in CR3BP space where the divergence of the pseudo-potential is equal to zero ($\nabla_{\vec{r_3}}\Omega^* = 0$) [11]. This means that in those points, the sum of forces in the rotating RF will be zero.

With that, a good place to start is by more thoroughly developing the divergence of the pseudo-potential field (2.28).

$$\nabla\Omega^* = \begin{pmatrix} \frac{\delta\Omega^*}{\delta x} \\ \\ \frac{\delta\Omega^*}{\delta y} \\ \\ \frac{\delta\Omega^*}{\delta z} \end{pmatrix} = \begin{pmatrix} x - \frac{(1-\mu)(x+\mu)}{d^3} - \frac{\mu(x-1+\mu)}{r^3} \\ \\ y - \frac{(1-\mu)y}{d^3} - \frac{\mu y}{r^3} \\ \\ -\frac{(1-\mu)z}{d^3} - \frac{\mu z}{r^3} \end{pmatrix} \tag{2.28}$$

From that expansion, it is clear how $\nabla\Omega^* = 0$ will result in three equations to be equated to zero. It is interesting to start by looking at the equation corresponding to the z-axis, for it will simplify the rest later. By factoring $(-z)$ out, it is evident how the only possible solution is $z = 0$, for $\left(\frac{1-\mu}{d^3} + \frac{\mu}{r^3}\right)$ will never be zero for any value of μ nor any state of the third body (2.29).

$$-z\left(\frac{1-\mu}{d^3} + \frac{\mu}{r^3}\right) = 0 \implies z = 0 \tag{2.29}$$

Now that the possible solutions of $\nabla\Omega^* = 0$ are set to be within the plane of the ecliptic, it is time to shift to the equation regarding the y-axis. In this case, similarly to the previous one, y will be factored out, resulting in two options. This time, however, neither of them can be discarded, instead resulting in two distinct paths to get to a possible solution (2.30).

$$y\left(1 - \frac{(1-\mu)}{d^3} - \frac{\mu}{r^3}\right) = 0 \implies \begin{cases} y = 0 \\ \text{or} \\ 1 - \frac{(1-\mu)}{d^3} - \frac{\mu}{r^3} = 0 \end{cases} \tag{2.30}$$

The first set of possible solutions, corresponding to $y = z = 0$, is what is known in the literature as Collinear or Eulerian points. Knowing $y = z = 0$, the equation referring to the x-axis can be developed by replacing d and r with their actual definition (2.15 and 2.16); which finally results in equation (2.31). It is of utmost importance to note that one cannot "simply" simplify the square root and the squared parentheses in any of the denominators, for this would mean losing possible solutions.

$$0 = x - \frac{(1-\mu)(x+\mu)}{(\sqrt{(x+\mu)^2})^3} - \frac{\mu(x-1+\mu)}{(\sqrt{(x-1+\mu)^2})^3} \tag{2.31}$$

Instead, three possible solutions can be identified that result in this final equation being zero and obviously avoiding any by-zero division. The three points which correspond to these three solutions will be named, accordingly, L1 (2.32), L2 (2.33), and L3 (2.34).

$$L1 : \begin{cases} x > -\mu \\ x < 1-\mu \end{cases} \tag{2.32}$$

$$L2 : \begin{cases} x > -\mu \\ x > 1-\mu \end{cases} \tag{2.33}$$

$$L3 : \begin{cases} x < -\mu \\ x < 1-\mu \end{cases} \tag{2.34}$$

Now, the path that is left to explore is the one remaining from equation (2.30), where $y \neq 0$. In that case, the points obtained will be within the plane of the ecliptic, but not contained within the x-axis. To find these final points, a system of two equations will be needed, for both the equation regarding the x-axis and the one regarding the y-axis will need to be zero (2.35).

$$\begin{cases} y\left(1 - \frac{(1-\mu)}{d^3} - \frac{\mu}{r^3}\right) = 0 \qquad \text{(where } y \neq 0\text{)} \\ x - \frac{(1-\mu)(x+\mu)}{d^3} - \frac{\mu(x-1+\mu)}{r^3} = 0 \end{cases} \tag{2.35}$$

These have two solutions, as will be seen in the following section, meaning two more Lagrange Points. These are known in the literature as Equilateral points because of their geometric relationship with the two primaries; and are named accordingly, L4 and L5.

2.2.3 Location

To pinpoint the exact location of the now identified Lagrange Points, a deeper dive into the previously discussed equations will be needed. Starting with L1, the solution that is to be found is one that satisfies equations 2.31 and 2.32. To more easily define the problem, it is useful to make a slight change of variable from x to γ_1 following $x_{L1} = 1 - \mu - \gamma_1$. Cleverly, this change will guarantee equation (2.32) for any real positive value of γ_1; and will transform equation (2.31) into a quintic equation that can be easily solved numerically (2.36). In this work, Python's NumPy [22] library has been used, which solves these equations using Newton's method. All the developed code is available at Appendix A.

$$\gamma_1^5 - (3-\mu)\gamma_1^4 + (3-2\mu)\gamma_1^3 - \mu\gamma_1^2 + 2\mu\gamma_1 - \mu = 0 \tag{2.36}$$

In a similar fashion, for L2 (2.31, and 2.33) will be subjected to the change $x_{L2} = 1 - \mu + \gamma_2$, thus resulting in another quintic equation to be solved numerically (2.37).

$$\gamma_2^5 + (3-\mu)\gamma_2^4 + (3-2\mu)\gamma_2^3 - \mu\gamma_2^2 - 2\mu\gamma_2 - \mu = 0 \tag{2.37}$$

The actual physical meaning of these variable changes means that instead of the actual value of x for L1 or L2 in the x-axis, the unknowns (γ_1 and γ_2) are the distances of L1 or L2 from m_2, located at $x = 1-\mu$, in NDU (see Figure 2.5). Note that, while it is easy to imagine a similar solution for the case of L3, but since this work does not concern with that Lagrange point, finding it is considered ultimately out of the scope for this research, and thus will not be further discussed.

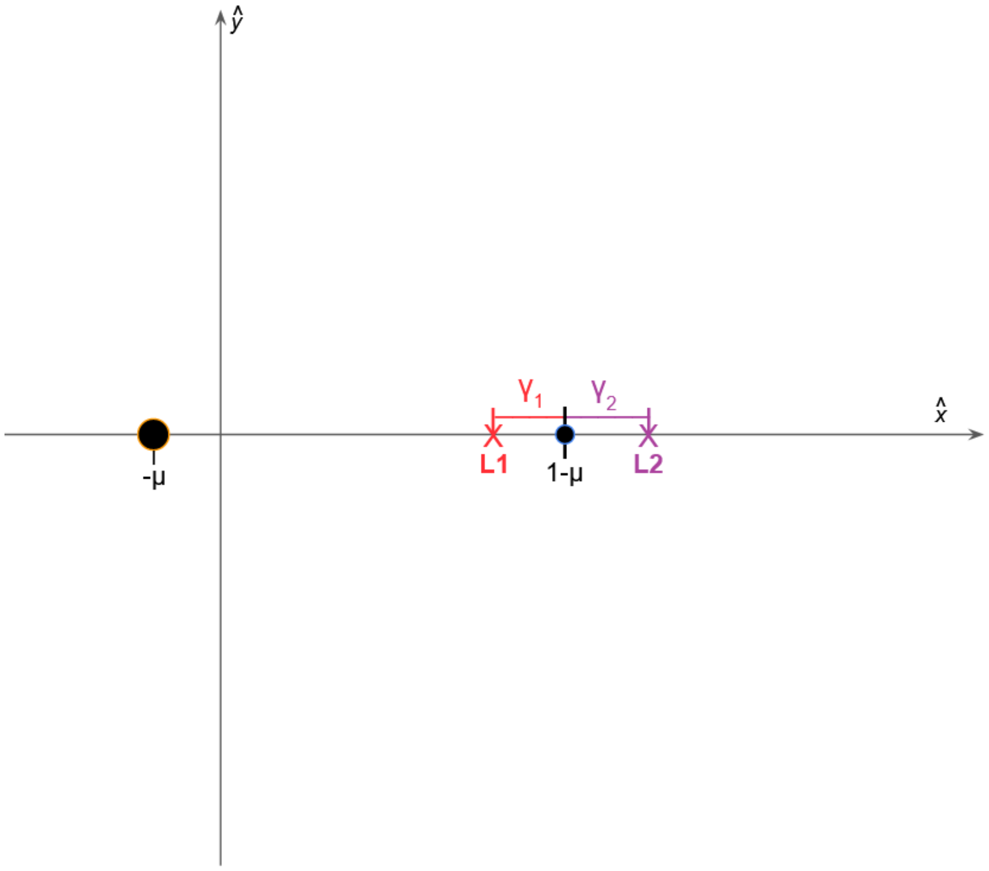


Figure 2.5: Representation of the physical meaning of the variable changes from x to γ_1 or γ_2 (all in NDU).

Moving forward to L4 and L5, things get a little more circuitous. The system of equations that arise from imposing the condition of $y \neq 0$ (2.35) is not direct. Instead, variables α and β will be introduced (2.38).

$$\begin{cases} \alpha = \frac{1}{d^3} \\ \beta = \frac{1}{r^3} \end{cases} \tag{2.38}$$

Solving for these variables instead means that the results will provide the inverse of the distance between the equilateral Lagrange points and the two main bodies (2.39).

$$\begin{cases} x-(1-\mu)(x+\mu)\alpha-\mu(x-1+\mu)\beta=0 \\ 1-(1-\mu)\alpha-\mu\beta=0 \end{cases} \tag{2.39}$$

This can be easily solved matricially. To that end, the system will be rewritten (2.40).

$$[A]\,\underline{x}+\underline{b}=\underline{0}$$

$$\text{where}$$

$$\begin{cases} [A]=\begin{bmatrix} -(1-\mu)(x+\mu) & -\mu(x-1+\mu) \\ \mu-1 & -\mu \end{bmatrix} \\ \underline{a}=\begin{bmatrix}\alpha\\ \beta\end{bmatrix} \;;\; \underline{b}=\begin{bmatrix}x\\ 1\end{bmatrix} \;;\; \underline{0}=\begin{bmatrix}0\\ 0\end{bmatrix} \end{cases} \tag{2.40}$$

The results of which are, regardless of μ, in $\alpha=\beta=1$. This is a newsworthy conclusion, for it means that the two main primaries and a Lagrange point not located on the x-axis necessarily form an equilateral triangle of side equal to 1 NDU (2.41). This has been illustrated in Figure 2.6. From which, unsurprisingly, L4 and L5 get their alternative name: Equilateral Lagrange points (note that only two points satisfy this condition when the needed condition of $z=0$ is also taken into account).

$$\begin{cases} x_{L4}=x_{L5}=\frac{1}{2}-\mu \\ y_{L4}=\frac{\sqrt{3}}{2} \;;\; y_{L5}=-\frac{\sqrt{3}}{2} \end{cases} \tag{2.41}$$

In the studied case, the Sun-Earth system, the location of the four Lagrange Points of interest (every one but L3), has been solved numerically working with the data from Table 2.1. Again, all relevant code can be found in appendix A, but the exact results are displayed in Table 2.2. In it, it is interesting to see how the ascending notation to refer to the different Lagrange Points is aligned with the decreasing values of the Jc[i] at each one of these points, with the only exception being L4 and L5 that, by their very nature, end up having the same value (the tie is finally broken by the criterium that L4 is ahead of the Earth while L5 is trailing behind). These values, when plotted in a graph, to scale, results in Figure 2.7.

	L1	L2	L4	L5
x	$0,989985982341$	$1,01007520002$	$4,99996959577$	$4,999969595779$
y	0	0	$\sqrt{3}/2$	$-\sqrt{3}/2$
z	0	0	0	0
J_C	$3,000897942$	$3,0008938876$	$2,99999695958$	$2,99999695958$

Table 2.2: L1, L2, L4, and L5 location for the Sun-Earth system, and the Jc for each one. All in NDU.

[i]Note that this, in turn, means that L4 and L5 require more mechanical energy to be accessed than L1 and L2.

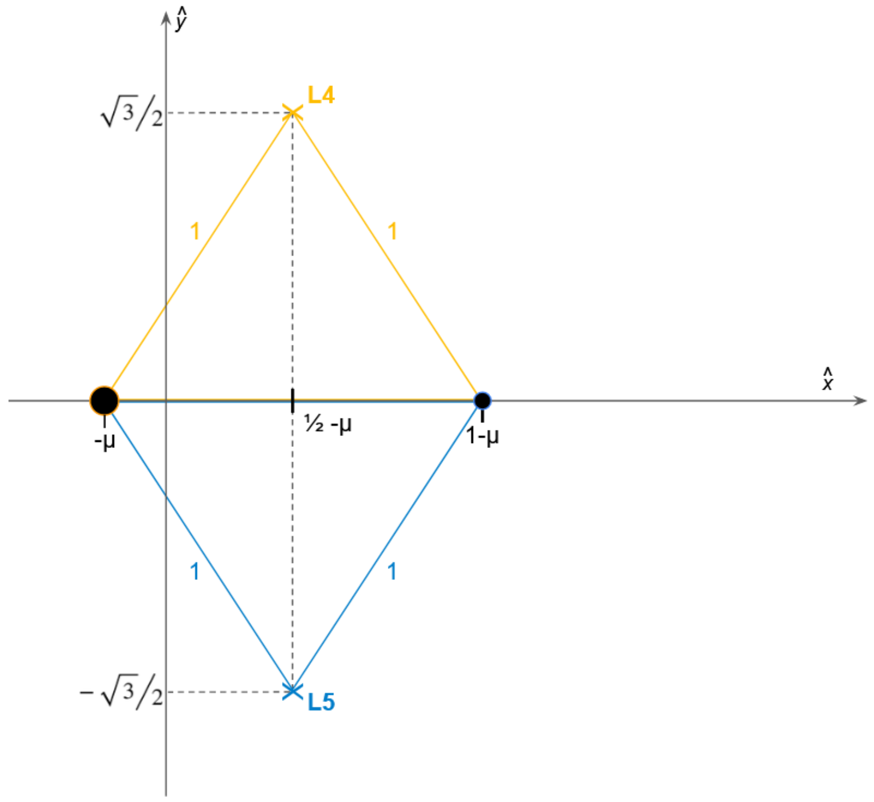


Figure 2.6: Location of L4 and L5 in the rotating RF, both being the third vertex of an equilateral triangle with side length of 1 with the main primaries at the other two vertices (all in NDU).

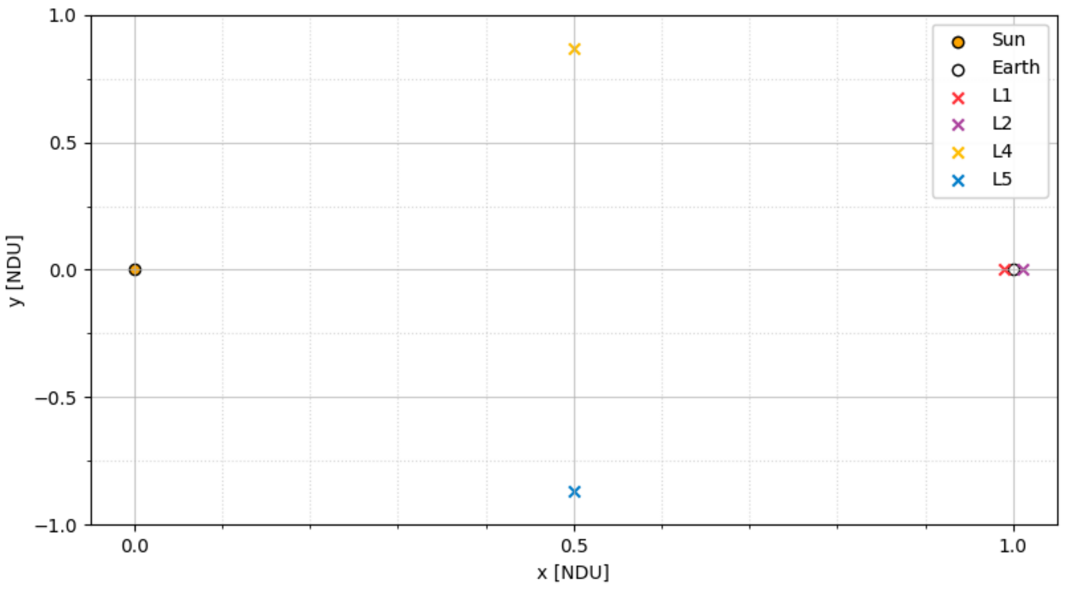


Figure 2.7: Relevant Lagrange Points' exact position, with the Sun and Earth for reference.

2.3 Periodic Motion

While being exactly in a Lagrange Point is not impossible, being in a trajectory around them gives one more opportunities to explore possible movement around the system than being in a fixed point. It is then, useful, to define the term **periodic orbit**. This is a trajectory for m_3 in CR3BP space where, after a given period of time (which will be defined as an orbital period), the initial state ($s_0 = [x_0,\ y_0,\ \dot{x}_0,\ \dot{y}_0]^T$) and the final state ($s_f = [x_f,\ y_f,\ \dot{x}_f,\ \dot{y}_f]^T$) is the same. This means that, when represented in a figure, a periodic orbit is a closed trajectory that loops back into itself, periodically repeating the same movement [11, 23]. Similar enough states to those that generate periodic orbits can result in near-periodic motion, that can be useful too.

It is crucial to remember that, while in a two-body model, this results in conic trajectories, this will not be the case in CR3BP. They will not necessarily even be symmetric with respect to some axis or plane (see Figure 2.8, where a selection of periodic orbits are selected, and the obvious lack of symmetry of the blue, yellow, and green orbits can be clearly noticed).

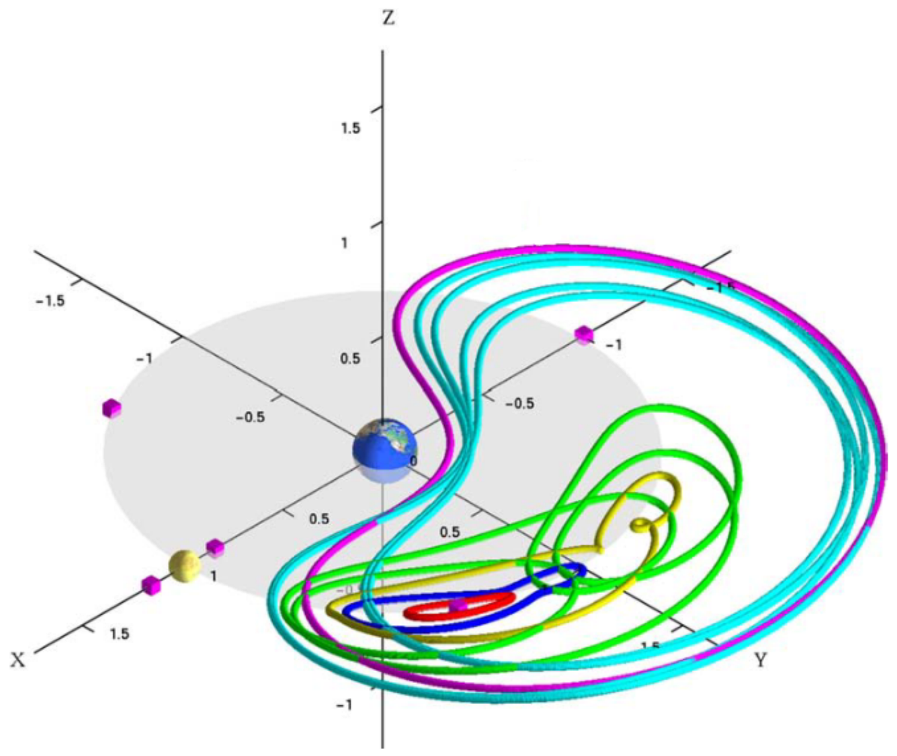


Figure 2.8: In the Earth-Moon system, a selection of periodic orbits are represented - illustrating the bifurcation point between the S3 and long-term planar families of periodic orbits. Red, blue, yellow, green are used for the planar long-period family around L4 (Lagrange Points are represented with pink boxes). Meanwhile, cyan and magenta represent the S3 family. Figure from J. Doedel et al. [23], slightly modified.

These periodic orbits around the Lagrange Points will serve as the starting and ending point for the desired transfers. Thus, mapping the whole families of orbits is out of the scope of this work. Instead, with a few periodic orbits around the desired Lagrange Points the basis to compute possible transfers will be obtained.

2.3.1 Poincaré Section

In a periodic orbit, the Poincaré section is crossed in the same direction with the same state vector ($\mathsf{s} = [\mathsf{x},\ \mathsf{y},\ \dot{\mathsf{x}},\ \dot{\mathsf{y}}]^\mathsf{T}$). Then, defining a Poincaré section and checking with which state the trajectory crosses it is a useful and common way to determine whether it is periodic, quasi-periodic, or chaotic [11]. It is a planar section that intersects the trajectory at least once. Usefully, this eliminates time as a variable, for the trajectory only needs to be propagated all the way to the crossing in the given direction with the Poincaré section.

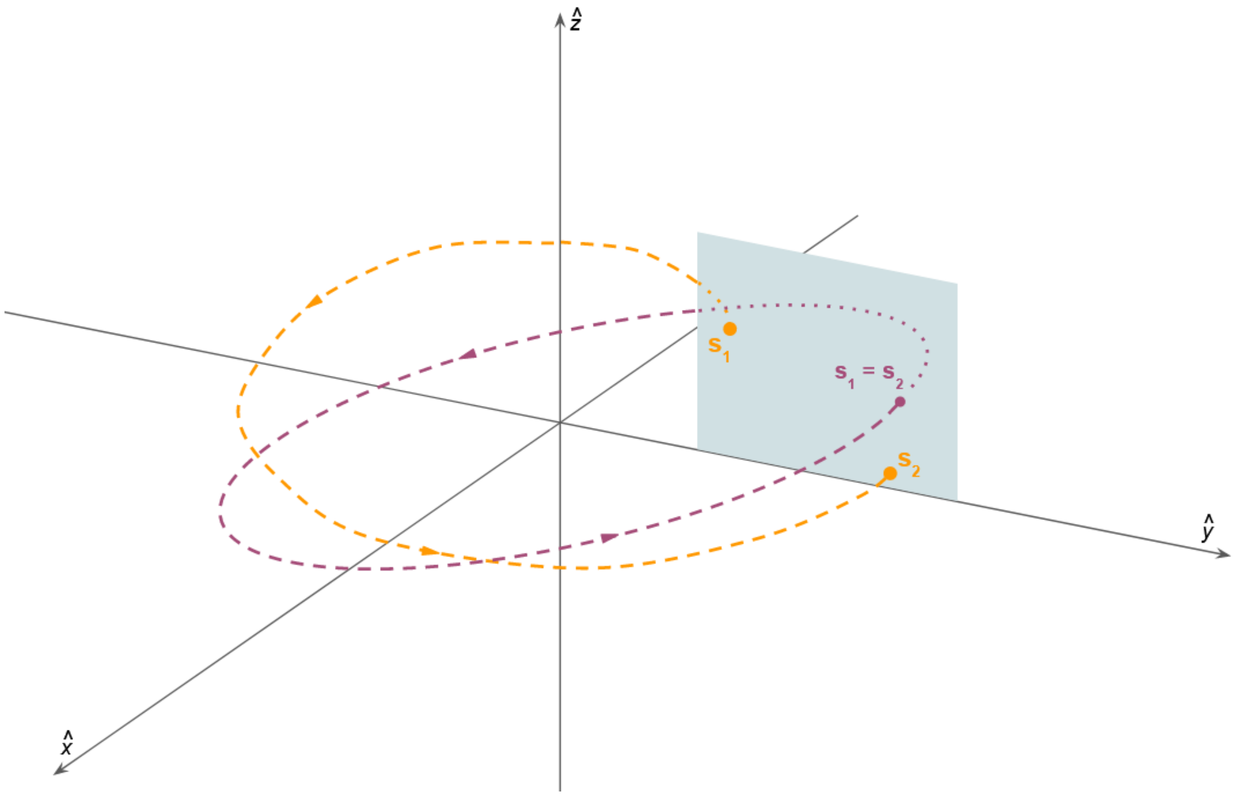


Figure 2.9: Schematic example of a Poincaré section at $\hat{\mathsf{z}} = 0$ (very-light blue), and the position of the initial (s_1) and final (s_2) states of a periodic orbit (magenta), and a non-periodic orbit (orange).

This is a very useful technic, but must be used carefully. One cannot assume that any Poincaré section will work for any trajectory. Moreover, some periodic orbits might cross the Poincaré section in the same direction in many places. For instance, the same Poincaré section placed at $\hat{\mathsf{z}} = 0$ intersects with the orbit in green in Figure 2.8 a total of 6 times, 3 in the same direction (three times when $0.5 < \mathsf{x} < 0.75$ NDU, and three more when $0.75 < \mathsf{x} < 1$ NDU). This is all to mean that, despite being an easy-to-understand concept, the application is not trivial.

2.3.2 Linearization of the CR3BP

To obtain the desired periodic motion around the Lagrange Points of study, the idea is to find the state vector the spacecraft needs to have in order to find itself in such orbit. Obtaining this state vector, however, is not trivial. If one has an initial guess, it is possible to imagine a refining algorithm that propagates the initial guess to a given Poincaré section, and then refines that data by minimizing or making zero the difference between the initial state and the point obtained at the crossing of the Poincaré section. The details regarding this refining process will be detailed in the following sections but, crucially, an initial guess is needed for any such algorithm to work.

Nevertheless, CR3BP is extremely chaotic by nature. This means that the initial guess needs to be quite good for any root-finding algorithm to find the exact periodic motion that is desired. This is caused by the fact that, for any given point in space, infinite periodic orbits may be found with very different characteristics.

A common way to solve this problem consist in linearizing the CR3BP equations of motion around the equilibrium point of interest [14, 18]. From that linearization, sufficiently good initial conditions for a periodic orbit in the vicinity of the equilibrium points will be found, thus being able to refine them to find the real periodic motion. To do so, the original equation of motion in CR3BP using the pseudo-potential function will be recovered (2.25). To them, a small perturbation will be applied (2.42). In it, ξ and η are the perturbations in the $\hat{x}$ and $\hat{y}$ directions, respectively. Note that, since the linearization is being done around the equilibrium points, which are all in turn located at $z = 0$, all this process can be done in 2D for simplicity's sake.

$$\begin{cases} x = x_{Li} + \xi \\ y = y_{Li} + \eta \end{cases} \quad \text{for } i \in [1,2,3,4,5] \quad \text{(the Lagrange Points)} \tag{2.42}$$

Now the equations will be rewritten, but in the terms of that variation (2.43). To obtain that, one is to simply expand them about the Lagrange Points position with a Taylor series and remove the terms of order higher than two, the linearized equations are obtained [14, 18, 24].

$$\begin{cases} \ddot{\xi} - 2\dot{\eta} = \Omega^*_{xx}\xi + \Omega^*_{xy}\eta \\ \ddot{\eta} + 2\dot{\xi} = \Omega^*_{yx}\xi + \Omega^*_{yy}\eta \end{cases} \tag{2.43}$$

Note that in order to do that from the pseudo-potential function, the partial derivatives of Ω^* evaluated at the Lagrange points are needed (2.44). Note how $\Omega^*_{xy} = \Omega^*_{yx}$).

$$\begin{aligned} U &= \begin{pmatrix} \frac{\delta^2\Omega^*}{\delta x \delta x} & \frac{\delta^2\Omega^*}{\delta x \delta y} \\ \frac{\delta^2\Omega^*}{\delta y \delta x} & \frac{\delta^2\Omega^*}{\delta y \delta y} \end{pmatrix} = \begin{pmatrix} \Omega^*_{xx} & \Omega^*_{xy} \\ \Omega^*_{yx} & \Omega^*_{yy} \end{pmatrix} \\ &= \begin{pmatrix} 1 - \frac{(1-\mu)}{d^3} - \frac{\mu}{r^3} + \frac{3(1-\mu)(x+\mu)^2}{d^5} + \frac{3\mu(x-1+\mu)^2}{r^5} & \frac{3(1-\mu)(x+\mu)y}{d^5} + \frac{3\mu(x-1+\mu)y}{r^5} \\ \frac{3(1-\mu)(x+\mu)y}{d^5} + \frac{3\mu(x-1+\mu)y}{r^5} & 1 - \frac{(1-\mu)}{d^3} - \frac{\mu}{r^3} + \frac{3(1-\mu)y^2}{d^5} + \frac{3\mu y^2}{r^5} \end{pmatrix} \end{aligned} \tag{2.44}$$

Thus, the linearized equations have the predictable form of equation (2.45), where $\dot{\bar{s}} = (\xi \ \eta \ \dot{\xi} \ \dot{\eta})^{\mathsf{T}}$.

$$\dot{\bar{s}} = A\bar{s} \tag{2.45}$$

Where A is the Jacobian of the equations of motion, which actually takes the form of a 4x4 matrix when working in 2-D (2.46) [24].

$$A = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \Omega^*_{xx} & \Omega^*_{xy} & 0 & 2 \\ \Omega^*_{yx} & \Omega^*_{yy} & -2 & 0 \end{pmatrix} \tag{2.46}$$

This problem then can be solved by computing the eigenvalues and eigenvectors of the A matrix. For each Lagrange Point, four eigenvalues will be obtained: two real and two complex for the three collinear points, and four complex for the two equilateral ones. Not only that, they will be paired up, being with the same value yet with different sign. On one hand, they will be referred to as $\pm\lambda$ and $\pm i\omega_1$ for the collinear points. On the other hand, $\pm i\omega_1$ and $\pm i\omega_2$ will be used for the equilateral ones (differentiating with λ and ω the real and complex ones, respectively). Then the eigenvectors corresponding to each eigenvalue can be placed in columns in a matrix C (2.47) [24]. In this C matrix the complex part of the eigenvectors is stored as a real numbers in the form of the U_n vectors, while the V_n vectors correspond to the real parts of the eigenvectors.

$$\begin{aligned} C &= \begin{bmatrix} V_\lambda & U_\lambda & V_{\omega_1} & U_{\omega_1} \end{bmatrix} \quad \text{(for L1, L2, and L3)} \\ C &= \begin{bmatrix} V_{\omega_1} & U_{\omega_1} & V_{\omega_2} & U_{\omega_2} \end{bmatrix} \quad \text{(for L1, L2, and L3)} \end{aligned} \tag{2.47}$$

In doing so, the C 4x4 matrix can be used to perform the variable change $\epsilon = C\zeta$, where $\zeta = (\xi \ \eta \ \dot{\xi} \ \dot{\eta})^{\mathsf{T}}$ and $\epsilon = (\epsilon_1 \ \epsilon_2 \ \epsilon_3 \ \epsilon_4)^{\mathsf{T}}$ (note that these ϵ are just the new variables fruit of this variable change). The system then becomes one with the form of $\dot{\epsilon} = A\epsilon$, and the integration is direct (2.48) [24].

$$\begin{aligned} &\begin{cases} \epsilon_1(t) = A_1 e^{\lambda t} \\ \epsilon_2(t) = A_2 e^{-\lambda t} \\ \epsilon_3(t) = A_3 \cos\omega_1 t \\ \epsilon_4(t) = A_4 \sin\omega_1 t \end{cases} \quad \text{(for L1, L2, and L3)} \\ &\begin{cases} \epsilon_1(t) = A_1 \cos\omega_1 t \\ \epsilon_2(t) = A_2 \sin\omega_1 t \\ \epsilon_3(t) = A_3 \cos\omega_2 t \\ \epsilon_4(t) = A_4 \sin\omega_2 t \end{cases} \quad \text{(for L1, L2, and L3)} \end{aligned} \tag{2.48}$$

Then, by undoing the change, the final linearized equations around the equilibrium points are obtained: equation (2.49) for the collinear points and equation (2.51) for the equilateral ones [24].

$$\begin{cases} \xi(t) = A_1 e^{\lambda t} + A_2 e^{-\lambda t} + A_3 \cos \omega_1 t + A_4 \sin \omega_1 t \\ \eta(t) = cA_1 e^{\lambda t} + cA_2 e^{-\lambda t} + \kappa A_3 \cos \omega_1 t + \kappa A_4 \sin \omega_1 t \\ \dot{\xi}(t) = \frac{A_1}{\lambda} e^{\lambda t} - \frac{A_2}{\lambda} e^{-\lambda t} - A_3 \omega_1 \sin \omega_1 t + A_4 \omega_1 \cos \omega_1 t \\ \dot{\eta}(t) = \frac{cA_1}{\lambda} e^{\lambda t} - \frac{cA_2}{\lambda} e^{-\lambda t} - \kappa A_3 \omega_1 \sin \omega_1 t + \kappa A_4 \omega_1 \cos \omega_1 t \end{cases} \tag{2.49}$$

$$\begin{aligned} \kappa &= -\frac{\omega_1^2 + 1}{2\omega_1} \\ c &= \frac{\lambda^2 - 1}{2\lambda} \end{aligned} \tag{2.50}$$

$$\begin{pmatrix} \xi(t) \\ \eta(t) \\ \dot{\xi}(t) \\ \dot{\eta}(t) \end{pmatrix} = A_1 \cos \omega_1(t) \overrightarrow{V_{\omega_1}} + A_2 \sin \omega_1(t) \overrightarrow{U_{\omega_1}} + A_3 \cos \omega_2(t) \overrightarrow{V_{\omega_2}} + A_4 \sin \omega_2(t) \overrightarrow{U_{\omega_2}} \tag{2.51}$$

It is important to notice that for the collinear points, the constants κ an c appear when describing the η parameter and its derivative; which take the form defined in (2.50). Notice how in the case of the equilateral points, the components of the eigenvectors appear instead, which in turn are the building blocks of the C matrix (2.47).Note also that in both cases, the integration constants A_n still appear, these can take different, arbitrarily small values (of the order of 10^{-5} to 10^{-4}). They must be small because these define the amplitudes of the to-be-obtained orbits; and when linearizing around a Lagrange Point, the further away the state is to the point where linearization has been made the less good of an approximation the linearization becomes. All in all, this values will determine the family of periodic orbits that is being approximated (see Table 2.3, with examples of orbits within those families available at Figure. 2.10) [14, 18, 24, 13].

In summary, a good enough initial guess for the state vector (s) is needed to make any refining algorithm work in its task of finding the desired periodic orbits. This is obtained by choosing a starting point, an objective type of orbit (Table 2.3), and two arbitrarily small values of A_n and A_{n+1}. The linearization of the equation of motions near the equilibrium points described in this section will then return the desired initial guess according to the point chosen and the desired family of orbits. This is because this way either (2.49) or (2.51) will be fully defined by this inputs, and $\dot{\xi}$ and $\dot{\eta}$ can be solved for. These two solutions will be the required initial guess for the velocities at the chosen starting point.

It is, nonetheless, important to remember that this is only one method to obtain a good-enough initial guess. This guess, regardless of the obtaining method, need to be refined by ensuring that the trajectory crosses the chosen Poincaré Section with the same state vector after being propagated one orbital period.

	$A_1 = A_2 = 0$	$A_3 = A_4 = 0$
L1, L2, L3	Lyapunov orbits	*non-periodic motion*
L4, L5	Long-term Planar orbits	Short-term Planar orbits

Table 2.3: Expected orbits obtained from zeroing certain A_n values and arbitrarily defining the remaining pair.

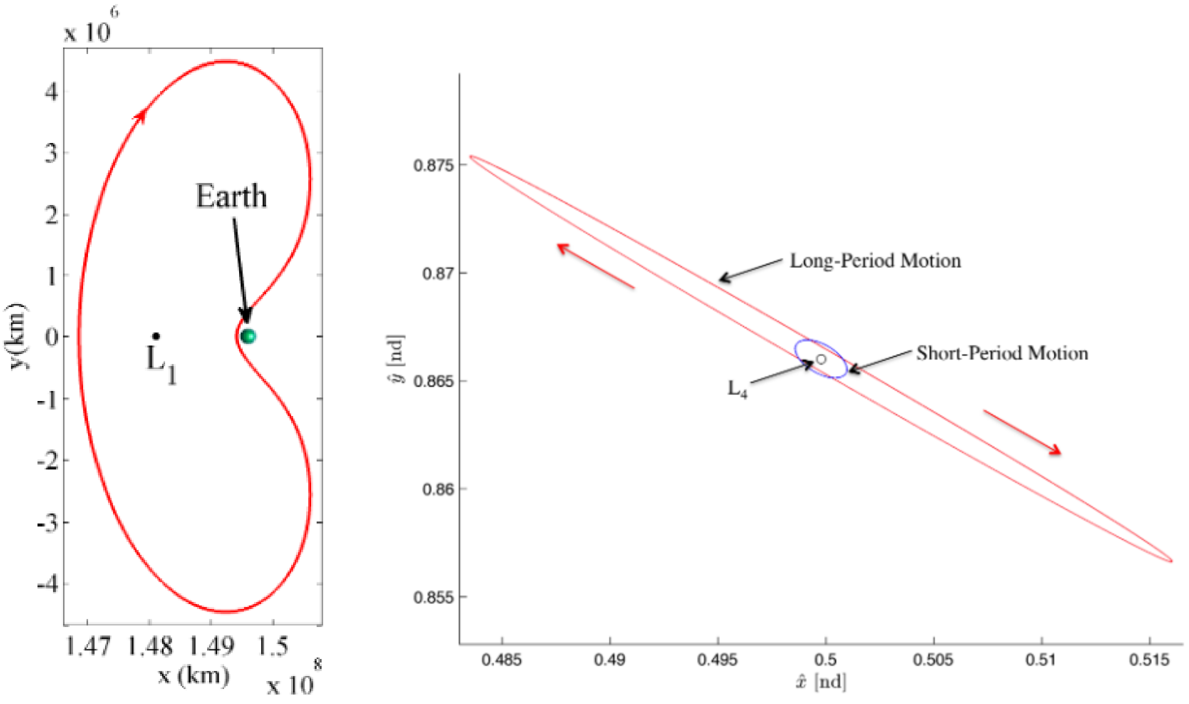


Figure 2.10: Example periodic orbits around Lagrange Points of interest. [Left] Lyapunov orbit around L1 of the Sun - Earth system. Figure by Doedel et al. [13]. [Right] A long-term and a short-term planar orbit (referred to as long-period and short-period motion in the figure) around the Saturn-Titan system. Figure by Van Aderlecht [14].

2.3.3 Periodic Orbits: Collinear Points

In the case of the Collinear points of interest, L1 and L2, there is a closer initial guess for s that can be used other than the one obtained by linearizing the system. This will not be available for the two equilateral points, where the linearization method will be needed. NASA—JPL has published their results on mapping the myriad of families of stable orbits around the Collinear points of different possible combinations of bodies in the solar system (or, what is the same, for different mass parameters μ) [25]. These results will, however, be only used as departure points, for the mass parameter used for their Sun-Earth computation does not take into account the mass of the Moon, while this work does.

Thus, as a starting point, one of the closest possible Lyapunov planar orbit to the equilibrium point will be used. This initial conditions can be refined to find the matching orbit in the studied mass parameter. Then, a continuation scheme can be applied, where the departure point will be slightly modified, and the previous, refined, velocities at $\mathsf{t} = 0$ can be used as initial conditions (see Figure 2.11) [11]. This will be followed until a limited set of Lyapunov orbits (for both L1 and L2) of varying amplitudes near the equilibrium points is obtained. Nonetheless, it is interesting to note that, despite only the orbits in the vicinity of the collinear points have been mapped, the scheme could theoretically be continued all the way to the vicinity of Earth, up to the point where the innermost point of the orbit gets too close to Earth that the assumptions that allow the CR3BP model to work do not make sense any more (for the orbit would be entering the atmosphere, amongst other reasons).

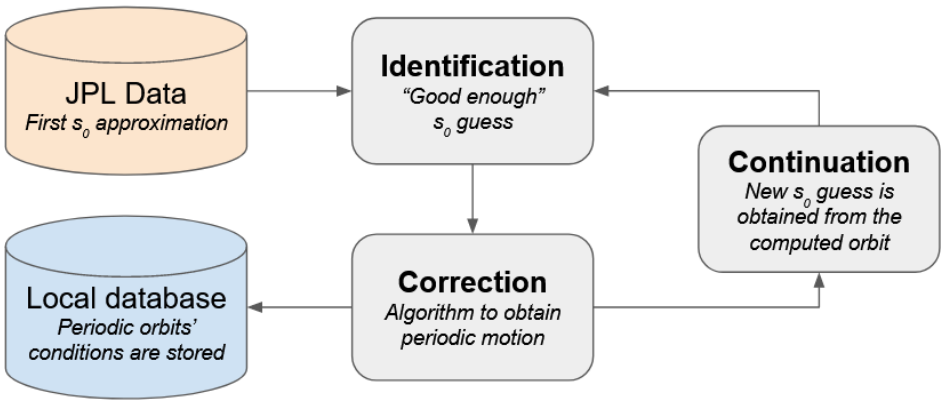

JPL Data
First s0 approximation
Identification
"Good enough" s0 guess
Continuation
New s0 guess is obtained from the computed orbit
Local database
Periodic orbits' conditions are stored
Correction
Algorithm to obtain periodic motion

Figure 2.11: Identification—Correction—Continuation scheme applied to the collinear points.

The "correction" part of the scheme, however, may deserve more explanation. The objective is to obtain periodic motion given a fixed point in space ($\mathsf{x_0}$, $\mathsf{y_0}$, the departure point), and a sufficiently good first approximation of the first temporal derivatives ($\dot{\mathsf{x}}_{0_{\text{guess}}}$, $\dot{\mathsf{y}}_{0_{\text{guess}}}$). By only modifying the derivatives, the algorithm must find the solution that results in periodic motion. By using the previously described Poincaré section, a $\mathbb{R}^2$ to $\mathbb{R}^4$ function can be designed to be then zeroed out. In this function, the inputs are the temporal derivatives at the departure point (the aforementioned two velocities: $\dot{\mathsf{x}}_0$, $\dot{\mathsf{y}}_0$); and the outputs are the differences between $\mathsf{s_0}$ and $\mathsf{s_f}$ ($\mathsf{s_f}$ being the final state vector, 4x1, once the Poincaré section is reached exactly one orbital period later).

By making the output of this function $s_f - s_0$, periodic motion will be obtained when this output is equal to zero.

However, optimization efforts in computational engineering are primarily dedicated to $\mathbb{R}^n$ to $\mathbb{R}^n$ functions; and thus, since further optimization is out of the scope of this work, a slight modification has been done to this initially intuitive correction scheme to transform it into a zero search of an $\mathbb{R}^2$ to $\mathbb{R}^2$ function. To do so, it is interesting to first define the Poincaré section that will be used. Knowing that Lyapunov orbits are symmetric with respect to the $x - z$ plane, defining the section at $y = 0$ makes a lot of sense. This actually discards one of the variables that needs to be zeroed out, for once the Poincaré section is reached exactly one orbital period later, y will necessarily be equal to zero (for if that was not the case, the Poincaré section would not have been reached). Then, for the collinear points case, a simple sum that combines the remaining three variables into two will be designed (2.52), where the variables with the f subindex are referring to the state vector once the Poincaré section is reached, while the 0 subindex refers to the departure conditions. Indispensably, note how this solution does not guarantee $s_0 = s_f$. This can result in incorrect solutions arising when zeroing this function, so a check mechanism will be needed. Nevertheless, the fact that the position in the x axis appears twice prioritizes that the position is the same at t_0 and t_f, which is the main concern when trying to find periodic or quasi-periodic motion.

$$f_{L1,L2}(\dot{x}_0, \dot{y}_0) = \begin{bmatrix} (x_f + \dot{x}_f) - (x_0 + \dot{x}_0) \\ (x_f + \dot{y}_f) - (x_0 + \dot{y}_0) \end{bmatrix} \tag{2.52}$$

This $\mathbb{R}^2$ to $\mathbb{R}^2$ function can be efficiently zeroed out using a variety of equation-solving algorithms. For this study, a modification of Powell's hybrid method has been used (included in Python's SciPy library), which interestingly does not need a differentiable function to work [19, 20, 26]. Again, the developed code can be consulted in Appendix A.

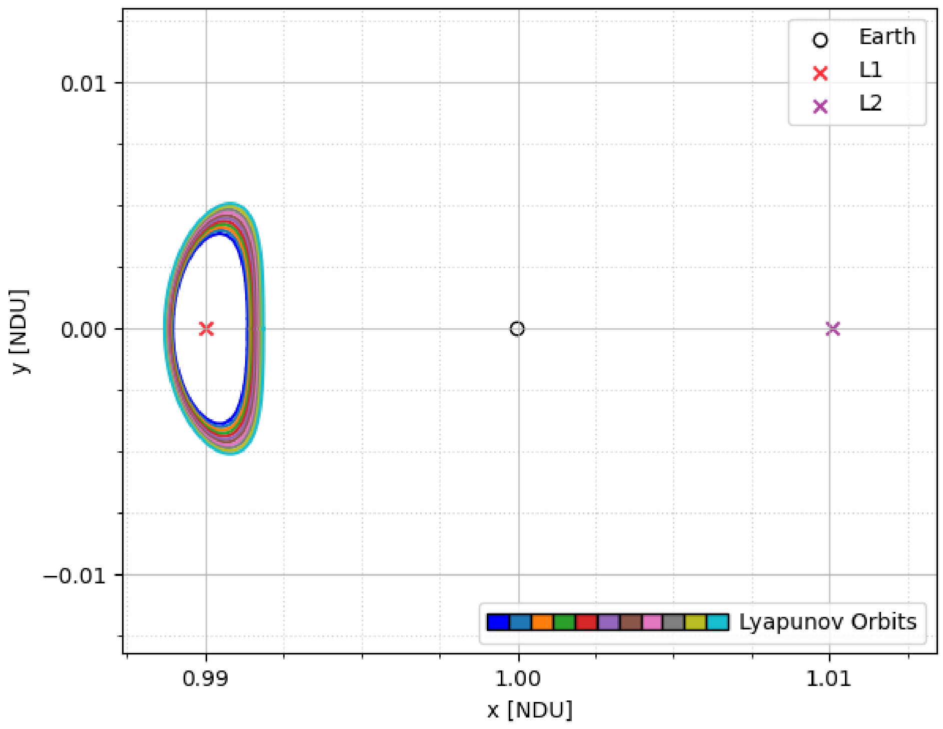
Satisfactory results of this correction algorithm are then used as the initial guess for the next orbit to be found (the "continuation" part of the original scheme), which will start at a point $x_0 + h$, where h is the step size ($h \approx 10^{-6}$). By applying this scheme in sufficient small steps, a set of Lyapunov orbits in the vicinity of L1 and L2 are obtained, which are periodic with a tolerance of 10^{-7} NDU in the studied case (see Figure 2.12 and 2.13). Exact results for the obtained, corrected, initial conditions for the mapped orbits are available in Table 2.4 and 2.5.

x_0	y_0	$\dot{x}_0$	$\dot{y}_0$	**Orbital period**
$0,991360$	0	0	$-0,008414$	$3,048650$
$0,991409$	0	0	$-0,008694$	$3,051168$
$0,991459$	0	0	$-0,008975$	$3,053787$
$0,991509$	0	0	$-0,009256$	$3,056482$
$0,991559$	0	0	$-0,009535$	$3,059254$
$0,991609$	0	0	$-0,009814$	$3,062101$
$0,991659$	0	0	$-0,010092$	$3,065024$
$0,991709$	0	0	$-0,010368$	$3,068023$
$0,991759$	0	0	$-0,010644$	$3,071098$
$0,991809$	0	0	$-0,010919$	$3,074250$
$0,991859$	0	0	$-0,011193$	$3,077478$

Table 2.4: Initial conditions and orbital period of the mapped Lyapunov orbits around L1. All in NDU.

x_0	y_0	$\dot{x}_0$	$\dot{y}_0$	**Orbital period**
$1,011030$	0	0	$-0,006897$	$3,081845$
$1,011080$	0	0	$-0,007303$	$3,085405$
$1,011130$	0	0	$-0,007714$	$3,089278$
$1,011180$	0	0	$-0,008132$	$3,093486$
$1,011230$	0	0	$-0,008555$	$3,098055$
$1,011280$	0	0	$-0,008984$	$3,103012$
$1,011330$	0	0	$-0,009420$	$3,108386$
$1,011380$	0	0	$-0,009862$	$3,114211$
$1,011430$	0	0	$-0,010310$	$3,120521$
$1,011480$	0	0	$-0,010765$	$3,127355$
$1,011530$	0	0	$-0,011227$	$3,134751$

Table 2.5: Initial conditions and orbital period of the mapped Lyapunov orbits around L2. All in NDU.



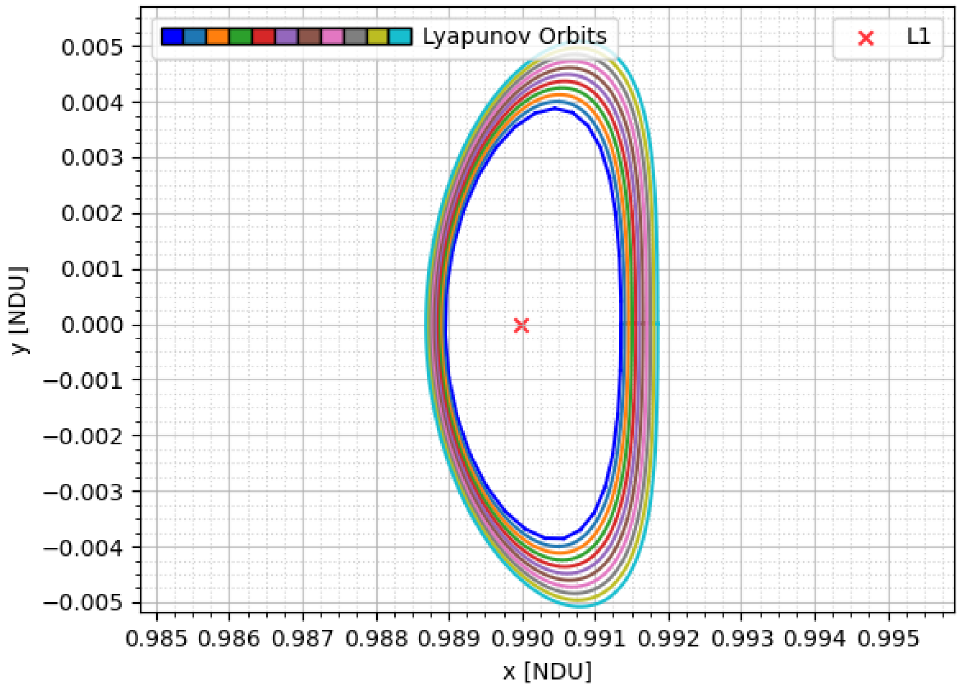
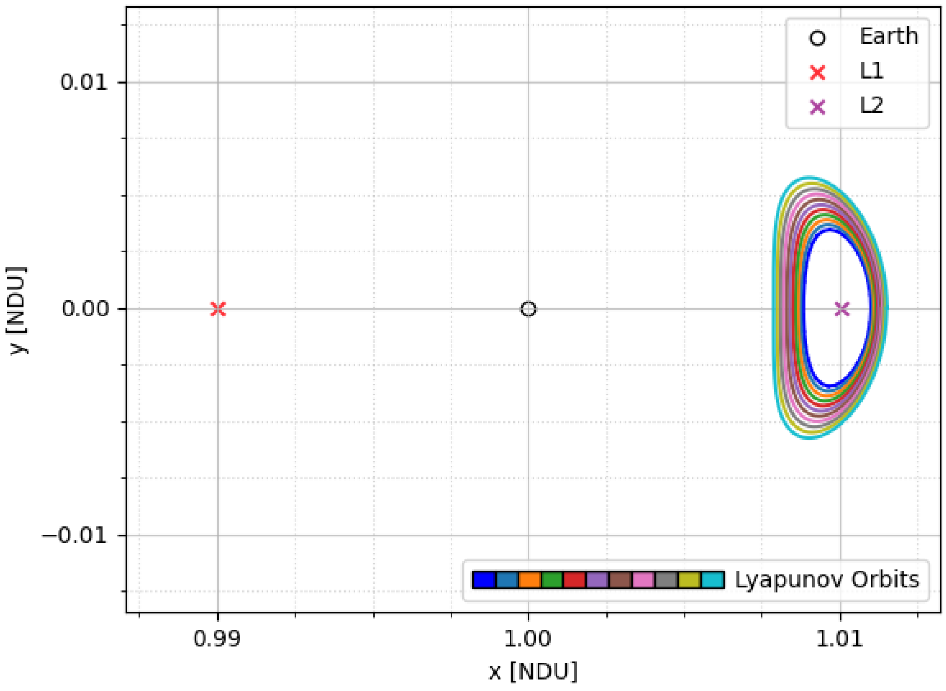


Figure 2.12: Set of Lyapunov orbits in the vicinity of L1. Different colours are used for different amplitudes, with a difference of 10^{-6} NDU between each two. [Top] Zoomed-out view of the obtained orbits. [Bottom] Zoomed-in view of the obtained orbits.



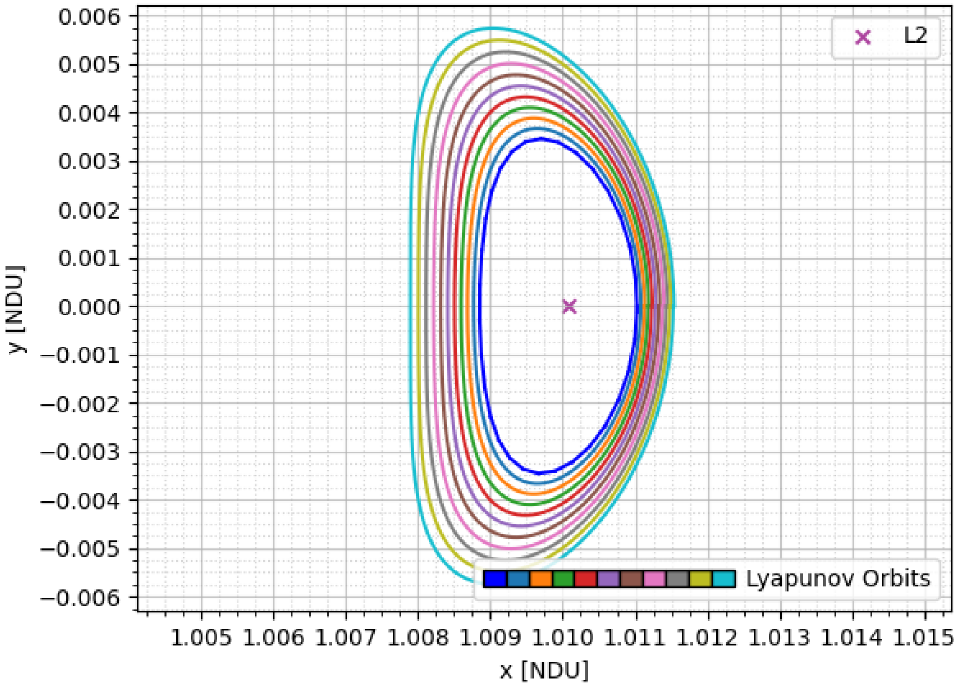


Figure 2.13: Set of Lyapunov orbits in the vicinity of L2. Different colours are used for different amplitudes, with a difference of 10^{-6} NDU between each two. [Top] Zoomed-out view of the obtained orbits. [Bottom] Zoomed-in view of the obtained orbits.

2.3.4 Periodic Orbits: Equilateral Points

For the equilateral points case, a similar Identification-Correction-Continuation scheme has been applied, albeit with a major change. NASA—JPL has not published a similar dataset with conditions for periodic motion for the case of L4 and L5, meaning that for this equilibrium points the results of linearizing the equations of motion in the vicinity of those equilibrium points will be needed as the set of initial conditions for mapping the desired periodic orbits. This results in a slightly different scheme (Figure 2.14).

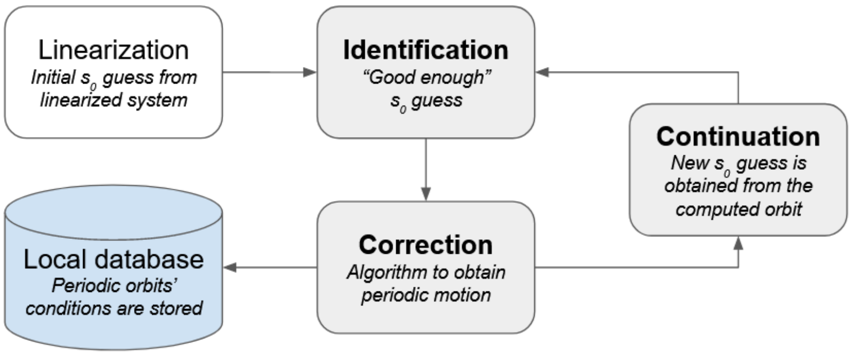

Linearization
Initial s_0 guess from linearized system
Identification
"Good enough" s_0 guess
Continuation
New s_0 guess is obtained from the computed orbit
Local database
Periodic orbits' conditions are stored
Correction
Algorithm to obtain periodic motion

Figure 2.14: Flowchart: Identification-Correction-Continuation scheme applied to the equilateral points.

When talking about the initial step, the linearization of the problem, it is indispensable to chose the family of periodic orbits that is to be mapped. As described in Table 2.3, depending on the values of the integration constants chosen when linearizing, a different family of periodic orbits will be mapped. Therefore, either the long-term or the short-term planar orbits (see Figure 2.10) around the equilateral points should be chosen as objective for the proposed transfers. For this study, the long-term planar orbits have been chosen, mainly because the convergence of the designed algorithm was reasonably better than that of the short-term ones. Despite that, short-term orbits should result in similar conclusions, and no in-depth analysis has been performed as to why this convergence speed differs, for it is out of the scope of this work.

Similarly to the collinear points, the whole orbit family will not be mapped. Instead, a set of periodic orbits in the vicinity of L4 and L5, though the family does continue until some orbits reach as far as L3 and beyond [23]. Also as in the previous case, the correction step of the scheme deserves a bit more of a nuanced explanation. In this case, due to the nature of the orbits being mapped, a Poincaré section defined by a constant x value makes a lot of sense. Since the orbits will be around the equilateral points, this x value is set at $\mathsf{x} = \mathsf{x}_{\mathsf{L4}} = \mathsf{x}_{\mathsf{L5}}$, a value that should only be crossed twice as long as the orbits are purely long-term (and do not combine long and short term motion). To correct for the fact that an $\mathbb{R}^{\mathsf{n}}$ to $\mathbb{R}^{\mathsf{n}}$ function is preferred, a two-solutions to be zeroed-out is again targetted. Since this time it is the x coordinate the one that is going to be the same after every orbital period because of the election of the Poincaré section, the function that the roots will be found for is a slight variation of the previous one (2.52), adapting to the Poincaré section choice (2.53). This, again, ensures that the algorithm has to work finding the roots of an $\mathbb{R}^2$ to $\mathbb{R}^2$ function at the cost of not necessarily ensuring that every single variable is zero; and again satisfactory results of this correction algorithm are then used as the initial guess for the next orbit to be found (the "continuation" part of the original scheme), which will start at a point $\mathsf{y}_0+\mathsf{h}$, where h is the step size ($\mathsf{h} \approx 10^{-5}$).

$$f_{L4,L5}(\dot{x}_0, \delta\dot{y}_0) = \begin{bmatrix} (\mathsf{y}_\mathsf{f} + \dot{\mathsf{x}}_\mathsf{f}) - (\mathsf{y}_0 + \dot{\mathsf{x}}_0) \\ \\ (\mathsf{y}_\mathsf{f} + \dot{\mathsf{y}}_\mathsf{f}) - (\mathsf{y}_0 + \dot{\mathsf{y}}_0) \end{bmatrix} \tag{2.53}$$

Overall, this results in a set of periodic orbits around the equilateral points that are a part of the long-term planar orbits family (see Figure 2.15 and 2.16). These will be the orbits to be targetted by the transfers, and reaching them will be the ultimate objective of this study. It is interesting to note that, despite looking like almost straight lines when viewed from afar, when actually zoomed-into (Figure 2.15 and 2.16) they are actually describing elliptical-like motion – albeit with very little amplitude. It is recalled that in CR3BP there no motion is described by conics, unlike in the two-body problem, and thus they are actually not ellipses. When zooming in, it is also interesting to note how, despite targeting trajectories with purely long-term motion, some short-term variations appear in some cases[ii]. This is mainly to the fact that the correction algorithm works with the modifications related to obtaining the roots of a $\mathbb{R}^2$ to $\mathbb{R}^2$ function instead of actually finding the true roots of a $\mathbb{R}^2$ to $\mathbb{R}^4$ or $\mathbb{R}^2$ to $\mathbb{R}^3$ function[iii]. Again, all the related code is available in Appendix A. Exact results for the obtained, corrected, initial conditions for the mapped orbits are available in Table 2.6 and 2.7.

[ii]This refers to the fact that motion that would on its own result in a short-term orbit gets added to the predominant long-term motion of the orbit. This has been studied in-depth by Van Aderlecht, for instance [14].

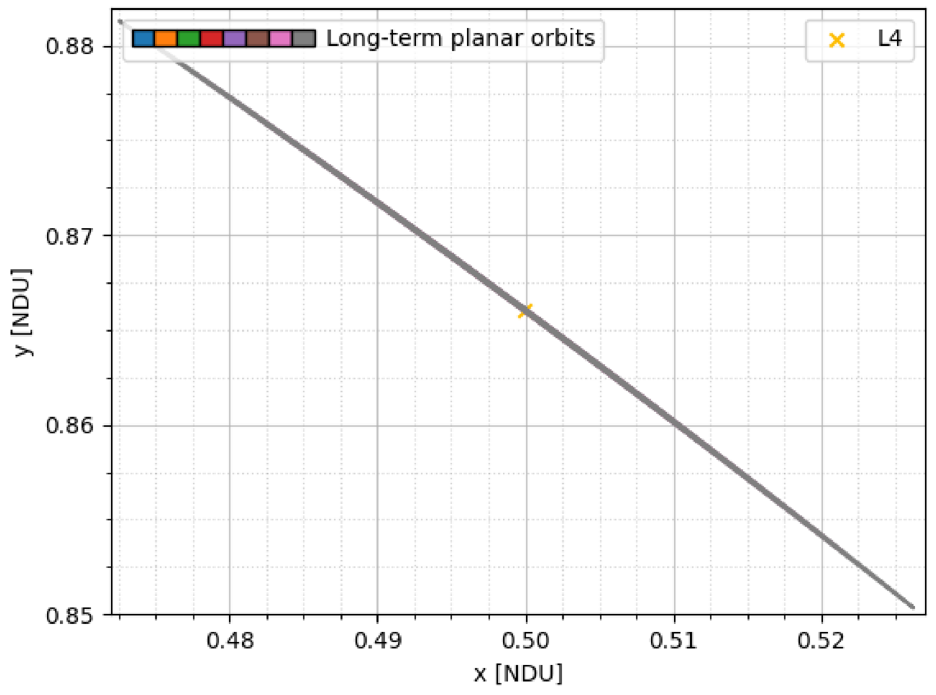
[iii]Note that a $\mathbb{R}^2$ to $\mathbb{R}^3$ function is the same in these cases as a $\mathbb{R}^2$ to $\mathbb{R}^4$ function because when propagating to a Poincaré section defined by a constant value of x or y, either $\mathsf{x}_\mathsf{f} = \mathsf{x}_0$ or $\mathsf{y}_0 = \mathsf{y}_\mathsf{f}$ is guaranteed.

x_0	y_0	$\dot{x}_0$	$\dot{y}_0$	**Orbital period**
$0,499994$	$0,866075$	$5,4 \cdot 10^{-5}$	$-3,1 \cdot 10^{-5}$	$1386,987619$
$0,499994$	$0,866085$	$6,5 \cdot 10^{-5}$	$-3,8 \cdot 10^{-5}$	$1387,010295$
$0,499994$	$0,866105$	$8,8 \cdot 10^{-5}$	$-5,1 \cdot 10^{-5}$	$1387,068495$
$0,499994$	$0,866115$	$9,9 \cdot 10^{-5}$	$-5,7 \cdot 10^{-5}$	$1387,104024$
$0,499994$	$0,866125$	$1,10 \cdot 10^{-4}$	$-6,4 \cdot 10^{-5}$	$1387,143843$
$0,499994$	$0,866135$	$1,22 \cdot 10^{-4}$	$-7,1 \cdot 10^{-5}$	$1387,187955$

Table 2.6: Initial conditions and orbital period of the mapped long-term planar orbits around L4. All in NDU.

x_0	y_0	$\dot{x}_0$	$\dot{y}_0$	**Orbital period**
$0,499994$	$-0,866045$	$-2,0 \cdot 10^{-5}$	$-1,2 \cdot 10^{-5}$	$1386,945361$
$0,499994$	$-0,866055$	$-3,2 \cdot 10^{-5}$	$-1,8 \cdot 10^{-5}$	$1386,955252$
$0,499994$	$-0,866065$	$-4,3 \cdot 10^{-5}$	$-2,5 \cdot 10^{-5}$	$1386,969421$
$0,499994$	$-0,866075$	$-5,4 \cdot 10^{-5}$	$-3,2 \cdot 10^{-5}$	$1386,987868$
$0,499994$	$-0,866105$	$-8,8 \cdot 10^{-5}$	$-5,1 \cdot 10^{-5}$	$1387,068899$
$0,499994$	$-0,866115$	$-9,9 \cdot 10^{-5}$	$-5,7 \cdot 10^{-5}$	$1387,104479$
$0,499994$	$-0,866125$	$-1,10 \cdot 10^{-4}$	$-6,4 \cdot 10^{-5}$	$1387,144349$
$0,499994$	$-0,866135$	$-1,22 \cdot 10^{-4}$	$-7,0 \cdot 10^{-5}$	$1387,188516$

Table 2.7: Initial conditions and orbital period of the mapped long-term planar orbits around L5. All in NDU.



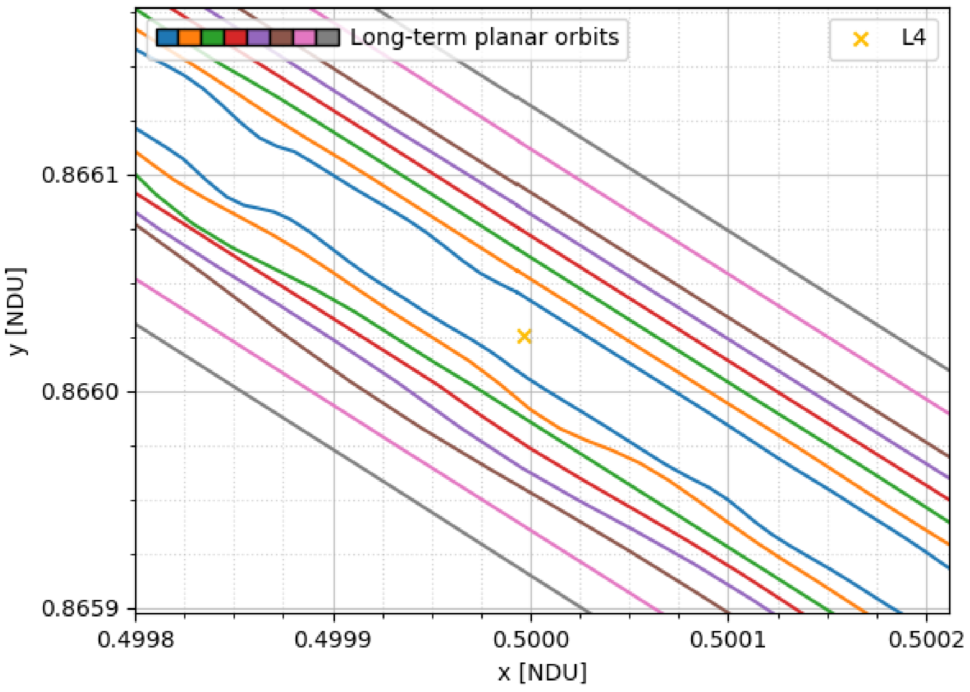
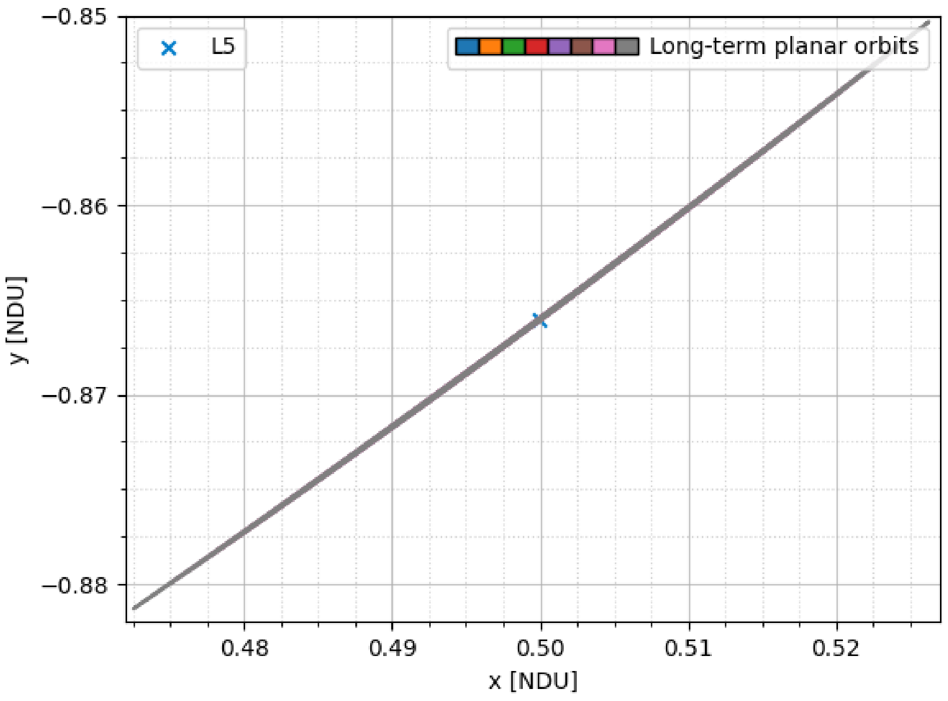


Figure 2.15: Set of long-term planar orbits in the vicinity of L4. Different colours are used for different amplitudes, with a difference of 10^{-6} NDU between each two. [Top] Zoomed-out view of the obtained orbits. [Bottom] Zoomed-in view of the obtained orbits.



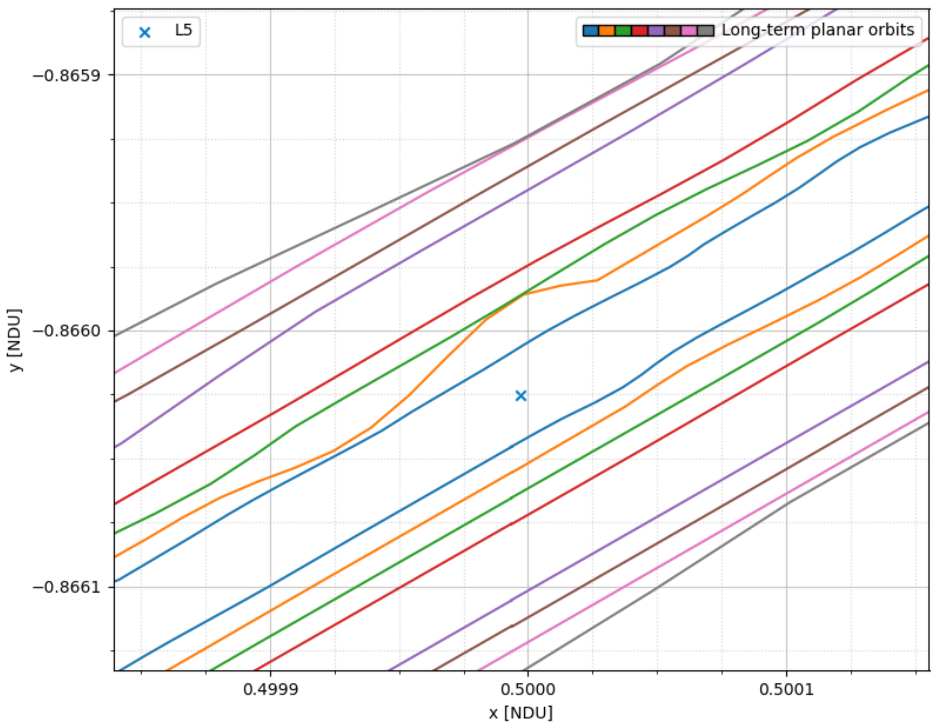


Figure 2.16: Set of long-term planar orbits in the vicinity of L5. Different colours are used for different amplitudes, with a difference of 10^{-6} NDU between each two. [Top] Zoomed-out view of the obtained orbits. [Bottom] Zoomed-in view of the obtained orbits.

2.4 Transfers

Now that a set of possible Lyapunov orbits around L1 and L2 have been computed along with long-term planar orbits around L4 and L5, it is possible to proceed to the final stage of this thesis. Using as a departure one of the found Lyapunov orbits, and as objective one of the found long-term orbits; the first step to map possible transfers to those objective orbits is to explore the natural instability of the collinear points. Then, it is necessary to find how it can be exploited in a way that they lead the spacecraft as swiftly as possible from one of the departure Lyapunov orbits to one of the objective long-term orbits.

To that end, a mathematical tool known as the State Transition Matrix (STM) [11] will be used to characterize the stability and instability of a previously-mapped orbit around one of the collinear points. Since the collinear points are saddle-point stable, they will necessarily have an unstable direction, where a perturbation will lead to most change from the original trajectory. By mapping this unstable direction, the whole unstable manifold out of that orbit can be plotted, thus revealing a path a spacecraft can take virtually for free, for only a small perturbation in the right direction will lead to it following that manifold. The chosen unstable manifold will be one that leads into the vicinity of one of the collinear points. By performing a mid-course correction change in velocity, an intersection with the objective long-term orbits around either L4 or L5 is to be achieved. This way, once that point is reached, another minimal speed correction can be done to achieve orbital insertion (see Figure 1.1). These paths are of interest because they promise to be quite energy efficient [13], for they use the intrinsic instability that exist in the collinear points to obtain trajectories leading to the vicinity of the desired destinations.

2.4.1 State Transition Matrix

The STM is a mathematical tool that defines the variation between two states in CR3BP after a given increment of time (2.54). In it, Φ is the STM that transforms a state vector at time zero $\mathsf{s}(\mathsf{t}_0)$ to a state vector at some future point in time $\mathsf{s}(\mathsf{t})$ [11].

$$\delta s(t) = \Phi(t, t_0)\delta s(t_0) \tag{2.54}$$

To obtain the STM for a given trajectory, it will need to be propagated along with the equations of movement corresponding to that trajectory (2.55) [11, 24, 13]. Note that A is the Jacobian of the equations of motion, previously defined in (2.46). Rather obviously, the STM from t_0 to t_0 is the identity matrix ($\mathbb{I}$), so that can be used as the starting value of Φ when propagating it along the equations of movement.

$$\begin{cases} \dot{\Phi}(\mathsf{t}, \mathsf{t}_0) = \mathsf{A}\Phi(\mathsf{t}, \mathsf{t}_0) \\ \Phi(\mathsf{t}_0, \mathsf{t}_0) = \mathbb{I} \end{cases} \tag{2.55}$$

When the STM describes the movement of a closed trajectory (one that returns with the same state vector to the departing Poincaré section after exactly one orbital period), it is known in the literature as Monodromy matrix [11]. The Monodromy matrix is exactly what will be computed in this case for Lyapunov orbits around L1 and L2; and will become the basic tool needed to make use of the intrinsic instability existing in the collinear points to take the spacecraft to either L4 and L5; and has thus been computed accordingly, with the code available in Appendix A.

2.4.2 Manifolds

The STM/Monodromy matrix is needed for the ultimate purpose of mapping the manifolds that arise from the saddle-point nature of L1 and L2. This stability can be easily demonstrated by obtaining the eigenvalues (λ)[iv] of the STM. In L4 and L5 the stability is peak-like, meaning it is in all directions marginally stable, which results in the module of the eigenvectors being equal to one. Meanwhile, in L1 and L2 the stability is saddle-point-like, which results in two eigenvalues being equal to one, while the two remaining will be more and less than one respectively (Table 2.8) [11, 13].

	$\|\lambda_1\|$	$\|\lambda_2\|$	$\|\lambda_3\|$	$\|\lambda_4\|$
Collinear Points	>1	1	1	<1
Equilateral Points	1	1	1	1

Table 2.8: Expected eigenvalues (λ) from a Monodromy matrix of an orbit around the Lagrange Points.

The eigenvector that results from the eigenvalue that is larger than one of a Monodromy matrix from an orbit around L1 or L2 points towards the unstable direction, and thus will be of interest. By applying a set of small enough perturbations h in the direction of that instability (from $\mathsf{h} \approx 10^{-7}$ to $\mathsf{h} \approx 10^{-3}$ NDU), the whole manifold that exits the collinear point and leads away from Earth can be mapped. The resulting trajectories that follow that manifold can be propagated all the way to one of the equilateral points (L4 for the manifold exiting from L1 and L5 for the manifold exiting from L2). One of the optimizations that this study performs is related to finding the exact perturbation h in the initial Lyapunov orbit that leads to the closest approach possible to either L4 or L5.

It is, however, crucial that the direction the unstable eigenvector points towards is getting away from Earth. In Figure 2.17 it can be seen how the stable eigenvector (that related to $\lambda < 1$) will push a sufficiently small perturbation in that direction back into the periodic Lyapunov orbit around L1. In contrast, a gentle push in the direction of the unstable vector (that related to $\lambda > 1$) will otherwise lead the spacecraft into a manifold, either towards Earth or towards L4. To ensure that the trajectory taken is within the manifold that leads to the equilateral points, a check of the direction of the unstable eigenvector will be performed when computing its value. This will be done by ensuring that the eigenvector is negative in the $\hat{\mathsf{x}}$ direction when in orbit around L1; or alternatively positive when in orbit around L2. If needed, then, a reversal in its sign will be performed to ensure the manifold is mapped in the direction of interest for this study. This will be done by simply multiplying by -1; since these calculations are just to get the direction the perturbation needs to be applied towards (no perturbation has been applied at this stage).

[iv] Note that these eigenvalues (λ) are not the same as the ones described during the linearization of the equations of motion near the equilibrium points.

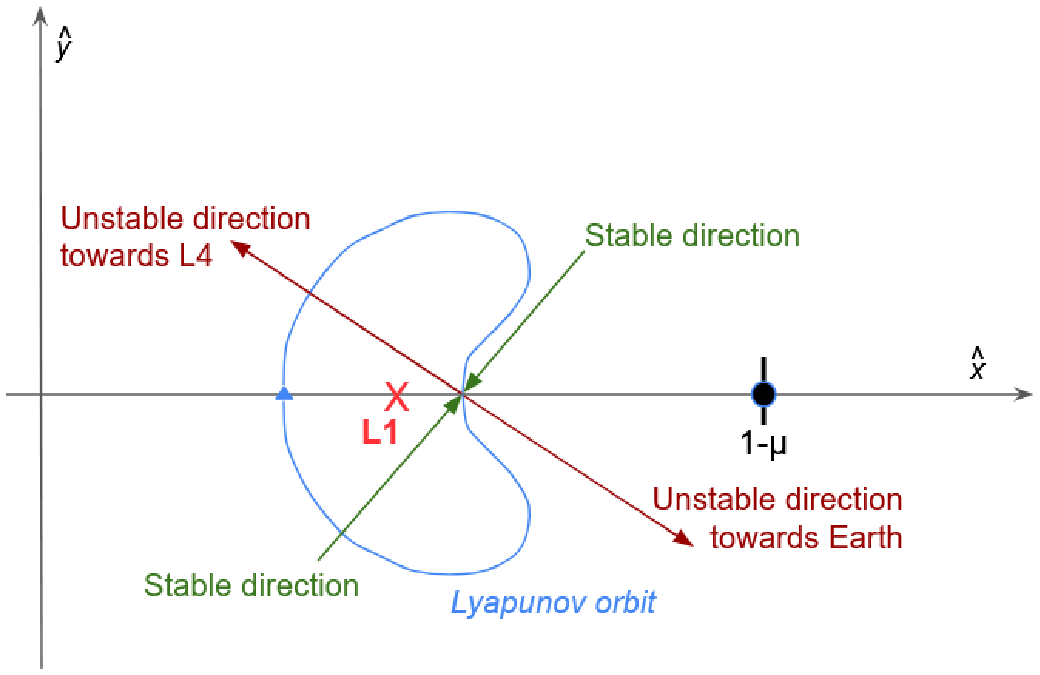


Figure 2.17: Schematic representation of the stable and unstable eigenvectors from a Monodromy matrix related to a Lyapunov orbit around L1.

2.4.3 Mid-course correction and orbital injection

Ensuring the spacecraft is a trajectory contained within the desired manifold is the first step into getting the desired transfer trajectories; and, as it has been seen in the previous section, can be achieved with a very small change in the departure Lyapunov orbit. Nevertheless, the energy requirements of a periodic orbit around L4 or L5 are quite higher than those of Lyapunov orbits around L1 or L2. This energetic requirement can be seen when the Jc has been computed for each of the Lagrange Points of interest, where L4 and L5 have a lower Jc value associated to them (Table 2.2), being in this way the least accessible of the Lagrange Points to in energy terms [11].

It is then, necessary, to perform a mid-course change of velocity to ensure the energy requirements are met and the manifold (that would otherwise continue to L3 and beyond) is left behind. This will be the first significant change of speed (ΔV) the spacecraft will need to perform to reach L4 or L5. Comparatively, the required change in the state vector to enter the manifold is negligible from the departure Lyapunov orbit around L1 or L2. The exact point of this mid-course correction is where the other optimization that this study will be performed. This is because depending on the trajectory taken within the manifold the ideal burn point can vary drastically. The rule of thumb used in the two-body problem that any correction in trajectory should be done as further away from the objective as possible, then, no longer applies in the CR3BP.

In order to get that mid-course correction, a shooting scheme has been applied. This works very similarly to the correction part of the periodic-orbit searching scheme; albeit a bit simpler, for now the function to be zeroed-out will be from $\mathbb{R}^2$ to $\mathbb{R}^2$. Departing from a point of a trajectory contained within the manifold

x_1, y_1; two variables will be available to be tweaked: the velocities $\dot{x_1}$ and $\dot{y_1}$. Changing these velocities will change which point of a Poincaré section set at either $x = x_{L4/L5}$ or $y = y_{L4/L5}$ (depending on the case and choosing either x or y as a constant to get better convergence when compared to the other option). Then, the equation to be zeroed out takes the form of (2.56), where x_2 and y_2 is the final position of the propagation of the orbit all the way to one of the equilateral points, and $x_{oL4/oL5}$, $y_{oL4/oL5}$ is the initial position of one of the previously mapped orbits around either L4 or L5. Zeroing-out (2.56), then, ensures that the post-correction trajectory passes through the initial point of the chosen orbit around one of the equilateral points. So, the first ΔV will be the difference in velocity needed at the mid-course correction point.

$$f_{Transfer}(\dot{x_1}, \dot{y_1}) = \begin{bmatrix} x_2 - x_{oL4/oL5} \\ \\ y_2 - y_{oL4/oL5} \end{bmatrix} \tag{2.56}$$

Notwithstanding, there is one more ΔV needed: the change in velocity at that point to ensure that the spacecrafts remains in that orbit. This is known as an orbital injection burn, and is calculated as a difference of speeds at the intersection point between the previously obtained trajectory and the chosen objective orbit (2.57); where $\dot{x}_{oL4/oL5}$ and $\dot{y}_{oL4/oL5}$ are the required speeds to be in the objective orbit.

$$\Delta V_2 = \begin{bmatrix} \dot{x_2} - \dot{x}_{oL4/oL5} \\ \\ \dot{y_2} - \dot{y}_{oL4/oL5} \end{bmatrix} \tag{2.57}$$

A summary of the required significant changes of speed, ΔV_1 and ΔV_2 and their approximate location can be consulted in Figure 2.18.

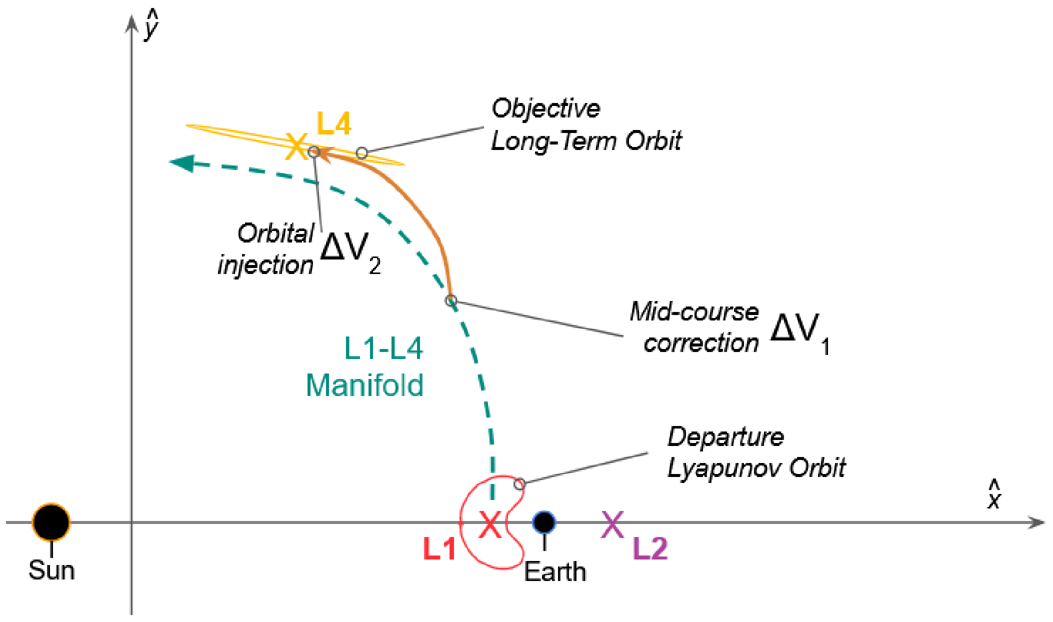


Figure 2.18: Schematic representation including the points where ΔV_1 and ΔV_2 will be needed for the case of the transfer from L1 to L4

2.4.4 From L1 to L4

The present study has selected as a case study the largest of the mapped Lyapunov orbits around L1 as the departure orbit (very light-blue orbit in Figure 2.12). This because the manifold exiting it had the closest approach to L4. Note, however, that mapping and then choosing much larger Lyapunov orbits around L1 defeats the purpose of transferring between L1 and L4, for they are arguably no longer in the vicinity of the Lagrange Point of departure. As a final destination orbit, the largest of the mapped long-term, planar orbits around L4 has been chosen. This is mainly because they are more visible when plotted, but their origin point - where the shooting scheme is aiming - is actually very similar to the rest of the orbits (as it has been seen in Figure 2.15). Thus, the results are not expected to differ drastically if one were to change that final objective orbit for another of the mapped ones.

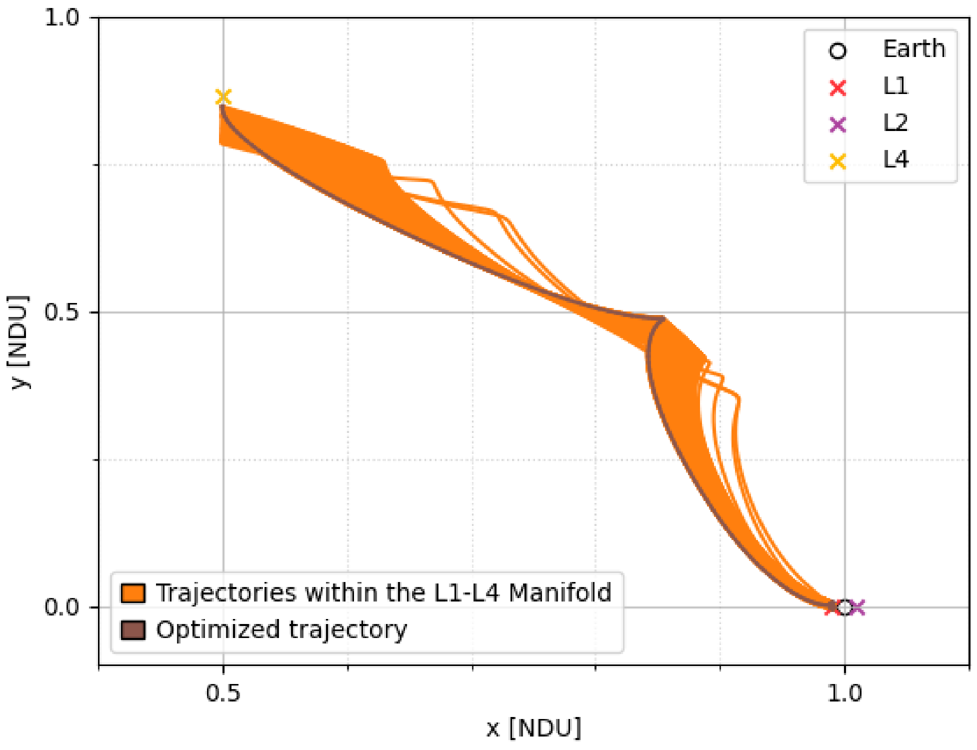
Departing from the chosen Lyapunov orbit around L1 it is interesting to map the myriad of possible trajectories that are a part of the manifold that leads to L4. To do so, a perturbation is applied in the unstable direction described by the eigenvector corresponding to the eigenvalue with a value higher than one. By applying progressively larger perturbations (from $\mathsf{h} = 10^{-7}$ to $\mathsf{h} = 5 * 10^{-4}$) in that direction and mapping the trajectory that results from each of these perturbations, the contours of the manifold appear. Notice how, despite the difference in the initial perturbation, all the trajectories result in very similar results (see Figure 2.19), all contained within the manifold.

This is quite remarkable, because from the infinity of possible trajectories within the L1-L4 manifold, one can choose the one that is more convenient. As previously stated, all of them have a basically negligible cost of entrance, for all of them result from very small perturbations to the departure orbit; and the only real cost will be that of insertion into the final, long-term periodic orbit around L4. Thus, in the study case, the trajectory within the manifold that has been chosen is, from all possible departure Lyapunov orbits and all mapped manifold-contained trajectories, that which has the closest approach to the Lagrange Point of interest. This has resulted in a close approach to L4 of around $0,02$ NDU, at $\hat{y} = 0,84685498$ NDU, for an initial perturbation of $\mathsf{h} = 7,10 \cdot 10^{-6}$ NDU.

Now that the trajectory contained within the manifold has been chosen, the next optimization is done. The main idea is to find the point between L1 and L4 where the mid-course correction burn should be performed, minimizing the total ΔV requirement; that is composed by the mid-course correction and the orbital injection. By combining both ΔV requirements, a comparatively simple $\mathbb{R}^1$ to $\mathbb{R}^1$ function has been used (2.58). Note how it has as an output corresponding to the total required ΔV budget, while having as an input the position in the $\hat{x}$ axis where the mid-course correction burn shall be done.

$$f_{\Delta V}(x) = |\Delta V_1| + |\Delta V_2| \tag{2.58}$$

This has been minimized using Python's [19] SciPy's [20] library, which includes a "minimize_scalar" function that allows for the minimization of any function with output of size 1. This function will locate a local



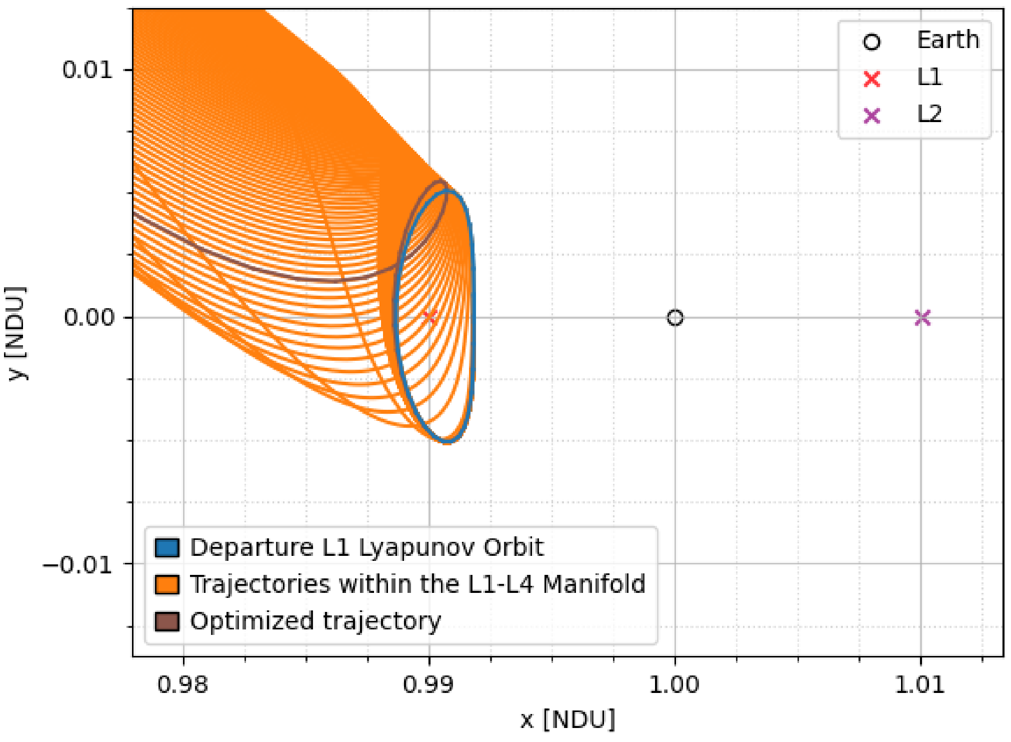


Figure 2.19: Set of possible manifold-contained trajectories from L1 to L4. [Top] Zoomed-out view of the manifold-contained trajectories. [Bottom] Zoomed-in view.

minimum close to the initial guess given, which has been set at around halfway, at $\hat{x} = x_{L4} + 0.3$. The output has been bounded, so it is located between $x = x_{L4} + 0.1$ and $x = x_{L4} + 0.4$, so that it happens between when the spacecraft is in its way to the equilateral point, and not after that. Overall, this strategy has resulted in the following results (all in NDU), while the full path can be consulted in Figure 2.20.

- Optimal burn point found at: $x = 0,7238716626920861$, at time $t = 11,219269293667233$.
- $\Delta V_1 = [-0,01651274 \quad 0,01400461]$[v]
- $\Delta V_2 = [0,0471737 \quad 0,06194482]$[vi]
- For a total ΔV budget of $0,09951396114237993$.
- And a total transfer time of 19.39274519857272.

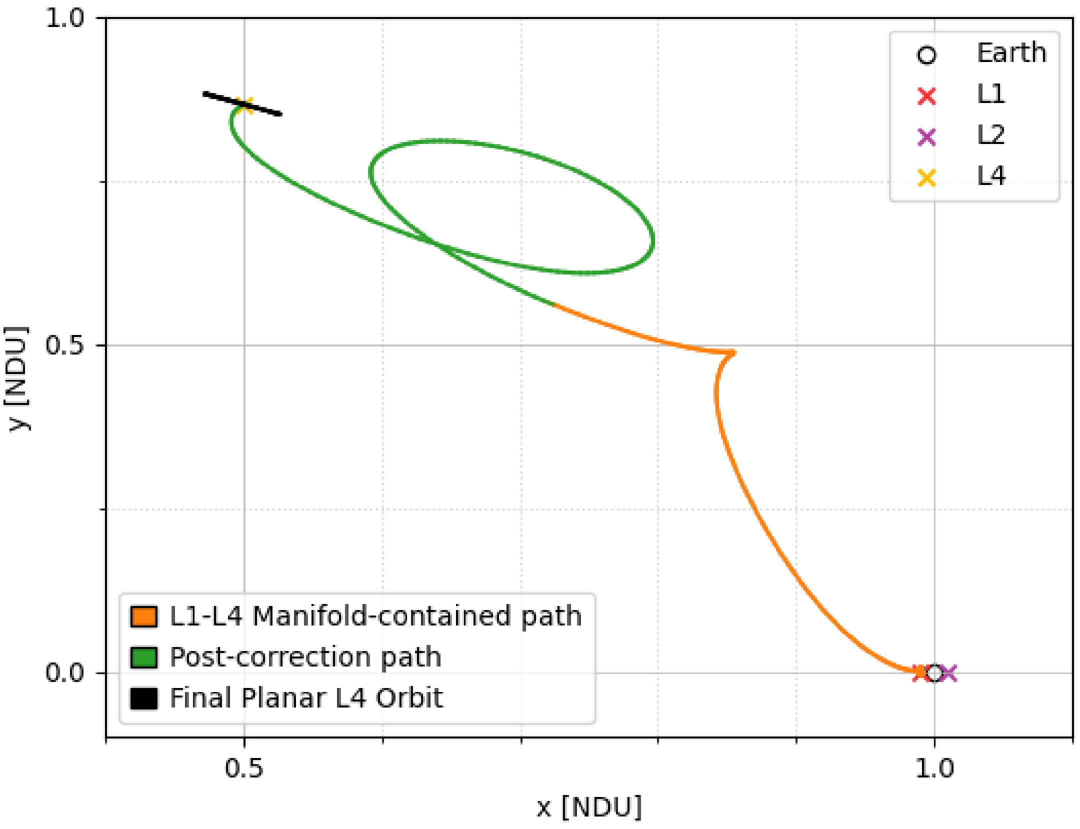


Figure 2.20: Optimized L1 to L4; departing from the largest mapped Lyapunov orbit and arriving at the largest mapped long-period orbit around L4.

[v]The vector is referring to the required changes in the $\hat{x}$ and $\hat{y}$ directions.
[vi]Idem

2.4.5 From L2 to L5

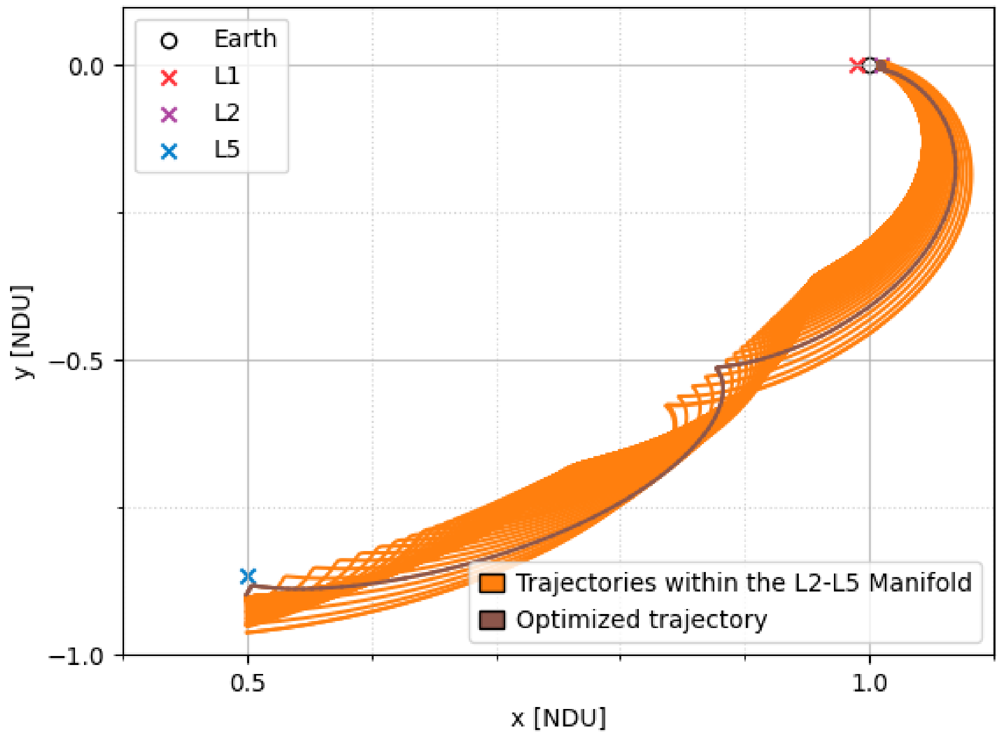
In the same way that it has been done for the L1-L4 manifold, the largest Lyapunov orbit around L2 has been chosen as the departure orbit (very light blue in Figure 2.13). Meanwhile, the objective long-term planar orbit around L5 is also the largest of the mapped set. By applying progressively larger perturbations (from $h = 10^{-7}$ to $h = 5 * 10^{-4}$) in the unstable direction of the departure point of the Lyapunov orbit around L2, a set of trajectories contained within the L2-L5 manifold have been plotted (Figure 2.21). The closest obtained approach to L5 from these trajectories has been obtained for an initial perturbation of $h = 3,1 \cdot 10^{-6}$ NDU, which results in an approach at $\hat{y} = -0.8958491972354155$ NDU. Similarly, this is located at $0,03$ NDU from the true location of L5.

Again, the path contained within the manifold that has the closest approach to L5 is the one that is optimized upon; by finding the minimum ΔV budget required for both the mid-course correction and the orbital injection. The same equation (2.58) is used as the minimizing objective, only that this time it is applied to the characteristics of L5, which has to take into account that a trajectory from L2 to L5 actually crosses a Poincaré section set at $y = y_{L5}$ twice (Figure 2.22). Nonetheless, by requiring a second crossing of that Poincaré section, better convergence when applying the shooting scheme has been obtained than using a Poincaré section set at $x = x_{L5}$. Again, all the developed code can be consulted in Appendix A.

All in all, the following results have been obtained for this second case (again, all in NDU), while the full path can be consulted in Figure 2.22.

- Optimal burn point found at: $x = 0,66048479670194171$, at time $t = 16,08661637377853$.
- $\Delta V_1 = [-0,00555063 \quad -0,01448192]$[vii]
- $\Delta V_2 = [-0,01396819 \quad 0,04815692]$[viii]
- For a total ΔV budget of $0,06565099794109891$.
- And a total transfer time of $17,451542946886626$.

[vii]The vector is referring to the required changes in the $\hat{x}$ and $\hat{y}$ directions.
[viii]Idem



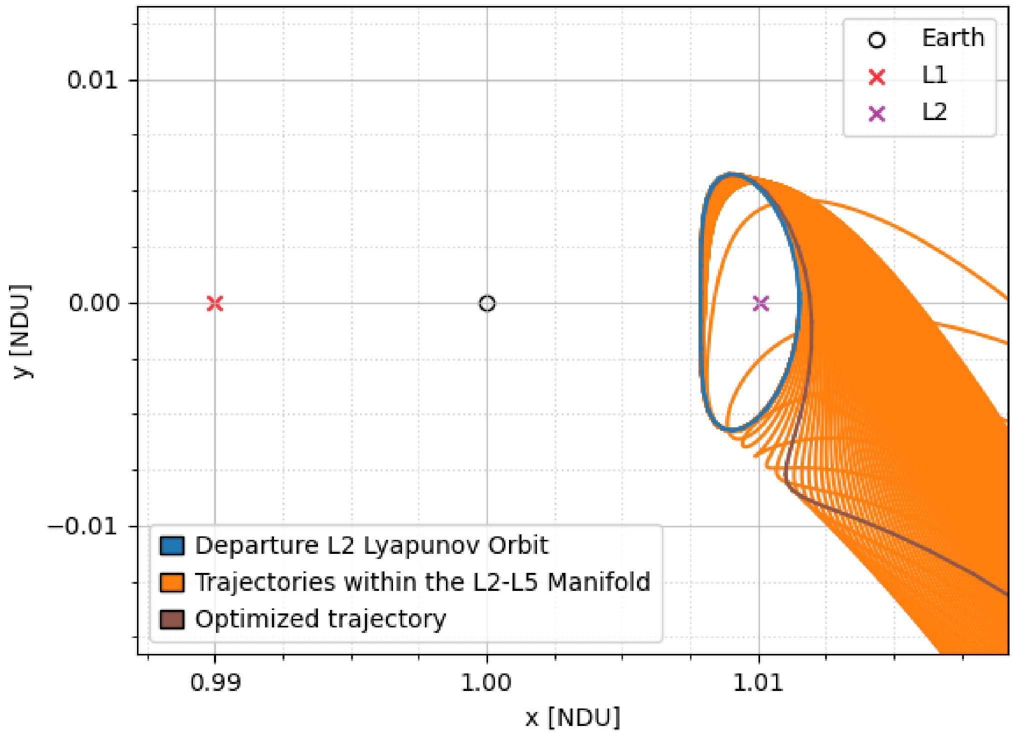


Figure 2.21: Set of possible manifold-contained trajectories from L2 to L5. [Top] Zoomed-out view of the manifold-contained trajectories. [Bottom] Zoomed-in view.

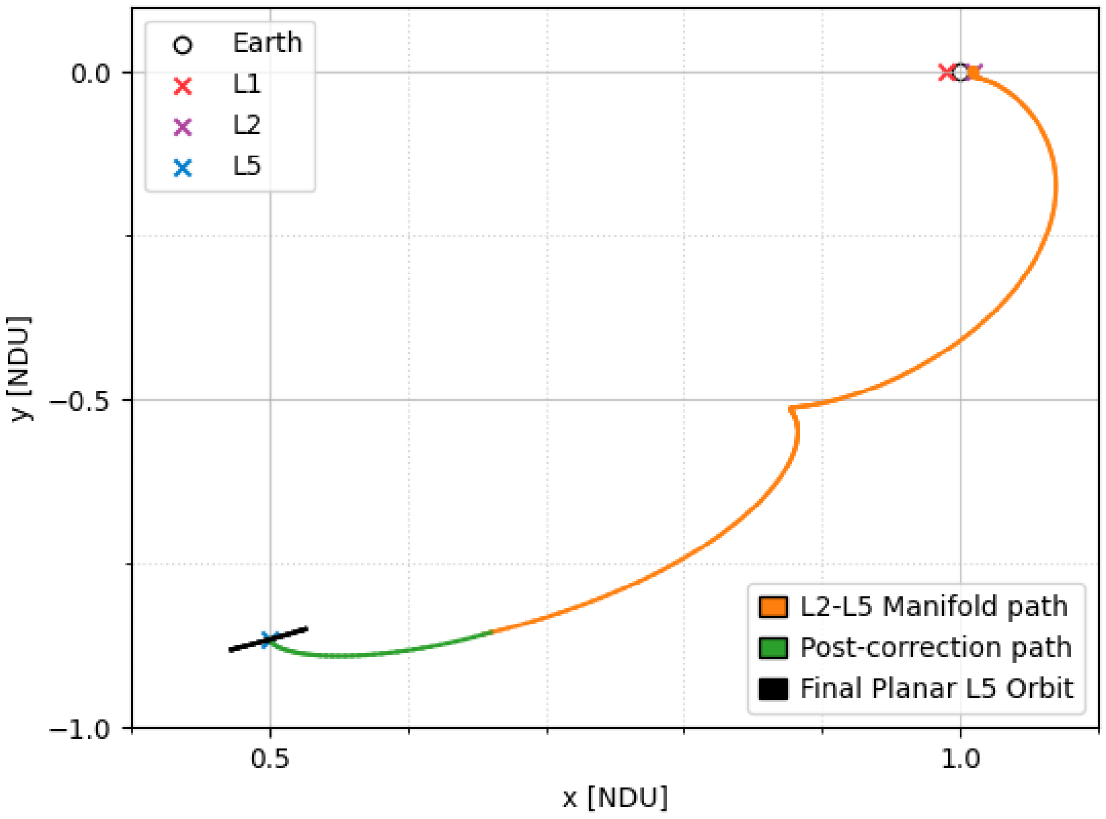


Figure 2.22: Optimized L2-L5 transfer; departing from the largest mapped Lyapunov orbit and arriving at the largest mapped long-period orbit around L5.

Part #03

Results and Conclusions

This part is dedicated to the dimensionalization and contextualization of the results, accompanied by a thorough discussion of their implications. Overall conclusions are then drawn from the work presented, and further research paths stemming from the methods used and results obtained are outlined.

3.1 Results analysis

The first step into contextualizing the results is to re-dimensionalize them to provide the necessary context as to what is being discussed. To do so, (2.1) is recovered, allowing for the necessary conversion of the obtained results into dimensional ones. All in all, this results in the following ΔV and time requirements for the optimized transfers from L1 to L4 and L2 to L5, which are attached in Table 3.1. Note how the results show the total ΔV budget required to get to the final planar orbits. The requirements shown correspond to the modulus of the required velocity changes in the x and y directions, detailed in NDU in Part 2. It is also worth noting that ΔV_2 happens at the final moment of the transfer, for once it is performed the objective orbit is achieved and the transfer is considered complete.

Transfer	ΔV_1 (m/s)	t_1 (days)	ΔV_2 (m/s)	**Transfer time (days)**	**Total ΔV (m/s)**
L1 to L4	$644,893$	$652,20$	$2319,104$	$1127,35$	$2963,997$
L2 to L5	$461,938$	$935,15$	$1493,460$	$1014,50$	$1955,398$

Table 3.1: ΔV and time requirements to perform the optimized transfers along the L1-L4 and L2-L5 manifolds.

These results come to show that the proposed optimized transfers taking advantage of the L1-L4 and L2-L5 manifolds are not only possible but plausible ways to get to the equilateral Lagrange Points of the Sun-Earth system. With an overall ΔV budget between 2 and 3 km/s, these requirements are not too dissimilar to other interplanetary transfer requirements. For instance, a two-body, Hohmann transfer to Mars takes requires approximately 3 km/s, and that is without counting the ΔV required to scape the sphere of influence of Earth, nor the ΔV required for orbital injection [27]. This makes getting to an equilateral point from L1 or L2 actually cheaper in ΔV budget terms than one of the most popular of the interplanetary transfers from Earth, including the required orbital injection.

Notably, there is a remarkable difference in the total ΔV budget between the two transfers found, of around 1 km/s. This is despite the fact that the actual difference in the starting orbits is not considerable. This has thus been attributed to the optimizations efforts applied having better results for the L2-L5 manifold than in the L1-L4 one.

Interestingly too, with a transfer time at around 3 years, these manifold-based trajectories are also plausible and useful from a temporal perspective. It is, nevertheless, a significant amount of time for the distance required - but not an unfathomable amount. Manifold-based transfers like the ones detailed in this study clearly trade a longer trip in favour of lower ΔV requirements. While using traditional two-body trajectories to get to, for instance, Mars, takes significantly less (NASA's Curiosity rover took 7 months) [28]; a three-year journey through the Solar System not rare. Though it still would, for example, be significantly shorter than the 5-year long trip NASA's Juno prove to Jupiter took between 2011 and 2016 [29]; even when accounting for the needed transfer time to either L1 or L2. NASA's James Webb Space Telescope, for instance, took approximately a month to get from LEO to a Halo orbit around L2 of the Sun-Earth system [30].

All in all, the results obtained through the optimization scheme implemented are promising in the sense that they are feasible for getting future asteroid-science or Sun-observation related missions to L4 or L5 without employing nearly as much ΔV as an interplanetary mission would need.

3.2 Conclusions

The present bachelor's final thesis has proven that the underlying instability existing around the collinear Lagrange Points can be used to access places hardly explored by humankind in an inexpensive enough manner. While further optimizations are still possible and are discussed in the following section, this work addresses the feasibility and usefulness from a ΔV budget and timespan perspective.

Departing from a Lyapunov orbit in the vicinity of either L1 or L2, this study proves that it is possible to get into a trajectory contained within a manifold (the L1-L4 manifold or the L2-L5) by making a variation in the state vector negligible from a ΔV budget perspective. This trajectory leads the spacecraft to the vicinity of the objective orbits around L4 or L5. Thus, there are two main contributors to the ΔV budget: a mid-course correction to ensure that the spacecraft has enough energy to be able to access the Lagrange Point of interest; and more relevantly, the orbital injection to ensure that once either L4 or L5 are reached the spacecraft stays there in a periodic orbit (Figure 1.1).

The total ΔV budget needed to perform the proposed transfers has been minimized regarding two aspects. Firstly, regarding the initial perturbation needed to escape the departing Lyapunov orbit. Secondly, the point in which the mid-course correction needs to be performed to reach the desired orbit around the final destination (either L4 or L5). Overall, these optimizations have resulted in a total ΔV budget required of $2964,00$ m/s for the L1-L4 transfer, and $1955,37$ m/s for the L2-L5 transfer; which places this transfers at a lower ΔV cost than other interplanetary transfers.

The results presented in this work comply with the error and tolerance margins required in Part 1. This last part was of utmost importance, for when operating with NDU it is easy to forget the magnitudes that are being described. An error above 10^{-5} results in a significant enough perturbation that could drastically change the obtained trajectory (as it has been seen in Part 2 when discussing the obtention of manifolds) [11]. Therefore, the largest tolerance used to ensure convergence of the periodic-orbit finding algorithm was of 10^{-6} NDU, an order of magnitude less than the required. Note that this error concerns the confirmation that the orbit is, in fact, periodic, and does not concern the actual error regarding the state vector of the spacecraft. When propagating or finding the actual transfers, the tolerance levels have been set much lower, at 10^{-10} NDU.

In conclusion, this bachelor's final thesis has achieved all the requirements set for it in Part 1 satisfactorily. Orbit-propagation code has been developed, periodic motion has been mapped around the Lagrange Points of interest, transfer paths between them have been optimized, and the tolerance requirements have been surpassed. Moreover, the obtained optimized transfers have proven to be accomplishable with ΔV budgets of less than that of other interplanetary missions, and are actually realizable in a timespan within what is usual of interplanetary missions. Undoubtably, then, this successful project opens the door for further research in the field of transfers taking advantage of the nature of the CR3BP.

3.3 Further research

The continuation of this work has two very clear and distinct paths to follow: result validation, and optimization improvement. The first concerns the testing of the results using higher fidelity models, subject them to more strict conditions to see if the trajectories would still be feasible in more real-life-like conditions or if they would otherwise need further corrections. The second is regarding optimization, for in this work only limited optimization has been performed, and there are a myriad of paths to explore in the search for more optimal paths that may be able to perform similar trajectories at lower ΔV costs.

Regarding results validation, software like NASA's General Mission Analysis Tool (GMAT) could be used [31]. GMAT is a mission-certified software used to validate trajectory proposal for real-life missions. It includes all the relevant astral objects in the Solar System in order to validate trajectories taking into account perturbations made by other planets and moons. It also has built-in different models to accurately consider the non-sphericity of many astral objects, as well as atmospheric and solar radiation pressure models to be able to alter the trajectories as they would be in a real-life-like situation. Additionally, it would be quite relevant to perform a comparison using GMAT's results with other possible paths, and include the ΔV requirements regarding launch to LEO, and the needed transfer from LEO to either L1 or L2.

When considering optimization improvements, multiple paths could be explored in the search for cheaper trajectories in terms of ΔV. These include, but are not limited to:

- **Optimization regarding departure periodic orbit around L1/L2 and objective periodic orbit around L4/L5**. In this work, optimization has only been applied choosing one departure orbit and one objective orbit. Different combinations of departure and objective orbits may potentially lead to ΔV savings.
- **Optimization regarding the departure point**. When calculating the direction of the small perturbation should be applied towards in order to enter a trajectory contained within one of the manifolds, the STM/Monodromy matrix was only computed for one point in the departure Lyapunov orbit

(Figure 2.17). Computing that direction in other points of the departure orbit and applying the perturbation there may also potentially lead to more efficient transfers.

- **Optimization regarding the objective point**. In Part 2, a shooting scheme has been applied in order to compute the required mid-course correction in a way that the correction lead the trajectory into an intersection with the departure point of the target periodic orbit around either L4 or L5. Allowing this intersection to be made with any point of the target periodic orbit could potentially lead to ΔV savings reducing the velocity change needed for the mid-course correction and/or for the orbital injection.

All things considered, this study opens the door for a myriad of future research into taking advantage of the underlying nature of the gravitational pulls of the Sun and the Earth. The future looks bright for future missions that may potentially make use of trajectories such as the ones described here.

References

1. CORNISH, Neil J. The Lagrange Points. *WMAP Education and Outreach*. 1998. Available also from: http://map.gsfc.nasa.gov/ContentMedia/lagrange.pdf.
2. BARNETT, Amanda et al. *What is a Lagrange Point?* [online]. NASA, 2020 [visited on 2022-09-16]. Available from: https://solarsystem.nasa.gov/resources/754/what-is-a-lagrange-point/.
3. RUST, David; DAVILA, Joseph, et al. The Sun and Heliosphere in Three Dimensions: Report of the NASA Science Definition Team for the STEREO Mission. 1997. Available also from: https://stereo.gsfc.nasa.gov/img/stdt.pdf.
4. MORTON, Erin; JONES, Nancy Neal. *OSIRIS-REx Searchers for Earth-Trojan Asteroids* [online]. NASA, 2017 [visited on 2022-09-19]. Available from: https://www.nasa.gov/feature/goddard/2017/osiris-rex-begins-earth-trojan-asteroid-search.
5. HUI, Man-To; WIEGERT, Paul A.; THOLEN, David J.; FÖHRING, Dora. The Second Earth Trojan 2020 XL5. *The Astrophysics Journal Letter*. 2020, vol. 922, no. 2, p. L25. Available from DOI: 10.3847/2041-8213/ac37bf.
6. NEWTON, Sir Isaac. *The mathematical principles of natural philosophy*. London (Printed for B. Motte), 1729. Available also from: https://archive.org/details/bub_gb_Tm0FAAAAQAAJ/mode/2up.
7. PEALE, Stanton J. et al. Celestial Mechanics - The n-body Problem. *Encyclopedia Britannica* [online]. 2022 [visited on 2022-09-19]. Available from: https://www.britannica.com/science/celestial-mechanics-physics/The-n-body-problem.
8. PEALE, Stanton J. et al. Celestial Mechanics - The three-body Problem. *Encyclopedia Britannica* [online]. 2022 [visited on 2022-09-19]. Available from: https://www.britannica.com/science/celestial-mechanics-physics/The-three-body-problem.
9. DANFORTH, Christopher M. *Chaos in an Atmosphere Hanging on a Wall* [online]. Mathematics of Planet Earth, 2013 [visited on 2022-09-19]. Available from: http://mpe.dimacs.rutgers.edu/2013/03/17/chaos-in-an-atmosphere-hanging-on-a-wall/.
10. PALFREE, Jasper; SALAZAR, Ever; REICH, Henry, et al. *Chaotic Planets* [online]. MinuteLabs.io, 2014 [visited on 2022-09-19]. Available from: http://labs.minutelabs.io/Chaotic-Planets/.
11. GUZZETTI, D.; MARTIN, K.; MISKIOĞLU, E. *Designing the Moonshot: An Introduction to Gravitational Multi-Body Dynamics*. Auburn University, 2021. Available also from: https://auburncatalog.instructure.com/courses/1913/pages/please-cite-this-material-as?module_item_id=32256.
12. FARRES, Ariadna; HEILIGERS, Jeannette; MIGUEL BAÑOS, Narcís. Road map to L4/L5 with a solar sail. *Aerospace Science and Technology*. 2019, vol. 95, p. 105458. Available from DOI: 10.1016/j.ast.2019.105458.
13. SOOD, Rohan; HOWELL, Kathleen. L4, L5 Solar Sail Transfers and Trajectory Design: Solar Observations and Potential Earth Trojan Exploration. In: [online]. 2016 [visited on 2023-01-02]. Available from: https://www.researchgate.net/publication/298528458_L4_L5_Solar_Sail_Transfers_and_Trajectory_Design_Solar_Observations_and_Potential_Earth_Trojan_Exploration.
14. VAN ANDERLECHT, Alexandre G. Tadpole Orbits in the L4/L5 region: Construction and Links to other families of periodic orbits. *Purdue University*. 2016. Available also from: https://engineering.purdue.edu/people/kathleen.howell.1/Publications/Masters/2016_VanAnderlecht.pdf.

15. PARK, Ryan S.; FOLKNER, William M.; WILLIAMS, James G.; BOGGS, Dale H. The JPL Planetary and Lunar Ephemerides DE440 and DE441. *The Astronomical Journal*. 2021, vol. 161, no. 3, p. 105. Available from DOI: 10.3847/1538-3881/abd414.
16. REID, Tyler. Orbital Diversity for Global Navigation Satellite Systems. 2017. Available also from: https://www.researchgate.net/publication/323245224_Orbital_Diversity_for_Global_Navigation_Satellite_Systems.
17. BUIS, Alan. *Milankovitch (Orbital) Cycles and Their Role in Earth's Climate - Climate Change: Vital Signs of the Planet* [online]. NASA and JPL, 2013 [visited on 2022-09-20]. Available from: https://climate.nasa.gov/news/2948/milankovitch-orbital-cycles-and-their-role-in-earths-climate/.
18. HAAPALA, Amanda F. Trajectory design using periapse maps and invariant manifolds. *Purdue University*. 2010. Available also from: https://engineering.purdue.edu/people/kathleen.howell.1/Publications/masters/2010_Haapala.pdf.
19. *Python.org* [online]. Python Software Foundation, 2003 [visited on 2023-01-01]. Available from: https://www.python.org/.
20. SCIPY COMMUNITY, The. *SciPy Documentation* [online]. 2022. [visited on 2022-11-01]. Available from: https://docs.scipy.org/doc/scipy/index.html.
21. SCIPY COMMUNITY, The. *scipy.integrate.solve_ivp* [online]. 2022. [visited on 2022-11-01]. Available from: https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html.
22. BERG, Sebastian; GOMMERS, Ralf; HARRIS, Charles; HOYER, Stephan; PAWSON, Inessa; PICUS, Matti; WALT, Stéfan van der; MENDONÇA, Melissa Weber; WIESER, Eric. *NumPy* [online]. NumPy, 2003 [visited on 2023-01-01]. Available from: https://numpy.org/.
23. J.DOEDEL, E.; A.ROMANOV, V.; PAFFENROTH, Randy; B.KELLER, H.; DICHMANN, Donald; GALÁN, J.; VANDERBAUWHEDE, Andre. Elemental periodic orbits associated with the libration points in the circular restricted 3-body problem. *International Journal of Bifurcation and Chaos*. 2011, vol. 17. Available from DOI: 10.1142/S0218127407018671.
24. CANALIAS VILA, Elisabet. Contributions to Libration Orbit Mission Design using Hyperbolic Invariant Manifolds [online]. 2007 [visited on 2022-12-01]. Available from: http://hdl.handle.net/10803/5927.
25. PARK, Ryan; CHAMBERLIN, Alan B., et al. *Solar System Dynamics: Three-Body Periodic Orbits* [online]. NASA and JPL, 2022 [visited on 2022-11-01]. Available from: https://ssd.jpl.nasa.gov/tools/periodic_orbits.html.
26. SCIPY COMMUNITY, The. *scipy.optimize.root* [online]. 2022. [visited on 2022-12-01]. Available from: https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.root.html.
27. STERN, David P. *Flight to Mars: Calculations* [online]. NASA, 2006 [visited on 2023-01-04]. Available from: https://pwg.gsfc.nasa.gov/stargaze/Smars2.htm.
28. *Trip to Mars - NASA Mars* [online]. NASA Science, 2020 [visited on 2023-01-04]. Available from: https://mars.nasa.gov/mars2020/timeline/cruise/.

29. GREICIUS, Tony; DUNBAR, Brian. *Juno Overview* [online]. NASA, 2022 [visited on 2023-01-04]. Available from: https://www.nasa.gov/mission_pages/juno/overview/index.html.
30. FISHER, Alan; DUNBAR, Brian. *Webb's Journey to L2 Is Nearly Complete* [online]. NASA, 2022 [visited on 2023-01-10]. Available from: https://blogs.nasa.gov/webb/2022/01/21/webbs-journey-to-l2-is-nearly-complete/.
31. MITCHELL, Darryl; ANIS, Adil, et al. *General Mission Analysis Tool (GMAT)* [online]. NASA, 2019 [visited on 2022-09-16]. Available from: https://opensource.gsfc.nasa.gov/projects/GMAT/index.php.
32. *Matplotlib - Visualization for Python* [online]. Matplotlib Project, 2022 [visited on 2023-01-01]. Available from: https://matplotlib.org/.
33. MONGE, Javier González. *RayleighLordAnimations* [online]. GitHub, 2022 [visited on 2022-12-01]. Available from: https://github.com/RayleighLord/RayleighLordAnimations/blob/master/publication%5C%20quality%5C%20figures/fig_config.py.
34. GESTIÓ ACADÈMICA, Servei de. *Preus dels estudis de grau* [online]. Universitat Politècnica de Catalunya, 2022 [visited on 2023-01-02]. Available from: https://www.upc.edu/sga/ca/matricula/preus/PreusGrau.
35. *PC de Sobremesa* [online]. PC Componentes, 2022 [visited on 2023-01-02]. Available from: https://www.pccomponentes.com/nitropc-pack-silver-intel-i5-11400f-16gb-1tb-256gb-ssd-gtx1650-monitor-22-fullhd.

Appendix A

CR3BP Code

For the code of this thesis, Python version 3.10.7 [19] has been used, along commonly used libraries to ease coding: NumPy [22], SciPy [20], and MatPlotLib [32]. In this Appendix, relevant information on the developed code, as well as the code files, are attached.

A.1 Guide into consulting and using this code

The developed code for this thesis is modular at its core. This is to mean that the code is structured in a way to be reusable for other need, by implementing very broad functions that may be used for a variety of needs. For instance, the mass parameter μ is just an input from NASA-JPL that is stored in the GMdata.txt data file, but it could theoretically be changed for any other mass parameter of choice. The Python file that contains the whole set of functions is the conveniently named aux_functions.py file, from where the rest of the code calls the functions needed to obtain the desired results.

On the other hand, data from relevant steps is stored in a folder (called "data") to avoid repeating computations each time it is needed. This means, however, that all code should be executed in order to generate the data required for every following step (the only standalone file is the general, aux_functions.py functions file that can be repurposed for any other needs). Thus, the totality of the code should be executed in the following order:

1. lagrange_points.py
2. Computation of periodic orbits
   - LyapunovL1.py
   - LyapunovL2.py
   - PlanarL4.py
   - PlanarL5.py
3. Computation of Transfers
   - TransfersL1.py
   - TransfersL2.py

For convenience, Figure A.1 depicts the code structure to obtain the objective, optimized transfer paths from L1 and L2 to L4 and L5. The resulting data from the code's execution is structured in the following manner:

- **GMdata.txt**: *Includes:*
  G *(Universal Gravitational Constant)*
  $\mathsf{GM}_{\mathsf{Sun}}$ *(mass parameter of the Sun)*
  $\mathsf{GM}_{\mathsf{Earth}}$ *(mass parameter of the Earth)*
  $\mathsf{GM}_{\mathsf{Moon}}$ *(mass parameter of the Moon)*
  All from NASA-JPL [15].

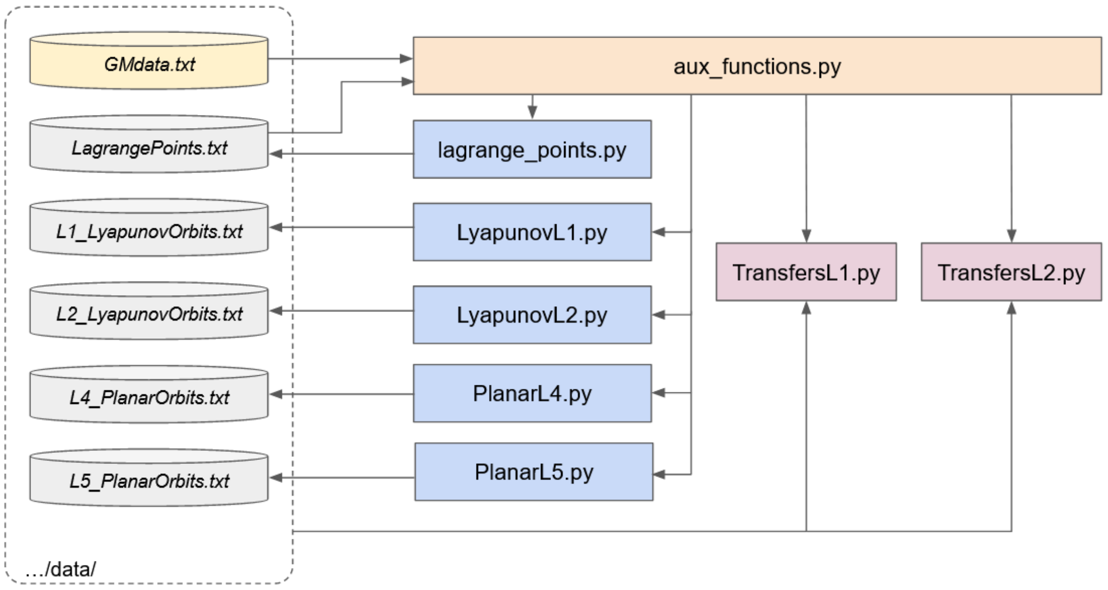


Figure A.1: Flowchart of the code, subsequent modules, and data files developed for this thesis.

- **LagrangePoints.txt**: *Includes:*

  x_{L1} *($\hat{x}$ coordinate of L1)*

  x_{L2} *($\hat{x}$ coordinate of L2)*

  x_{L4} *($\hat{x}$ coordinate of L4)*

  x_{L5} *($\hat{x}$ coordinate of L5)*

- **L1_LyapunovOrbits.py**: *Includes:*

  x_{0_1} y_{0_1} $\dot{x}_{01}$ $\dot{y}_{01}$ T_{orbit_1} *(Initial state vector of the First Lyapunov orbit around L1, plus its orbital period)*

  x_{0_2} y_{0_2} $\dot{x}_{02}$ $\dot{y}_{02}$ T_{orbit_2} *(Initial state vector of the Second Lyapunov orbit around L1, plus its orbital period)*

  ...

  x_{0_n} y_{0_n} $\dot{x}_{0n}$ $\dot{y}_{0n}$ T_{orbit_n} *(Initial state vector of the nth Lyapunov orbit around L1, plus its orbital period)*

- **L2_LyapunovOrbits.py**: *Same as L1_LyapunovOrbits.txt*

- **L4_PlanarOrbits.py**: *Same as L1_LyapunovOrbits.txt*

- **L5_PlanarOrbits.py**: *Same as L1_LyapunovOrbits.txt*

A.2 aux_functions.py

This Python file is the backbone of all the developed code. A description of the inputs, outputs, needs, and other relevant information regarding every developed function is included here before the full Python file for easier understanding of the code.

- **add_grid** (adds a grid to the current plot) [i]

[i]This function has been extracted from Monge [33].

  - Inputs: ax (axis object), lines (bool), locations (4x1 array-like)
  - Outputs: *None*

- **load_basic_data** (computes the mass parameter μ)
  - Inputs: *None*
  - Outputs: mu_earthsun (float)
  - Requires: .txt file at "data/GMdata.txt"
- **import_LP** (imports locations of the Lagrange Points)
  - Inputs: *None*
  - Outputs: L1 (float), L2 (float), L4 (float), L5 (float)
  - Requires: .txt file at "data/LagrangePoints.txt"
- **system_plot** (plots the basic CR3BP space with Sun, Earth, and the Key Lagrange Points)
  - Inputs: mu_earthsun (float), L1 (float), L2 (float), L4 (float), L5 (float), xlim (2x1 array-like), ylim (2x1 array-like), zoom (int), showSun (bool), showEarth (bool), showL1 (bool), showL2 (bool), showL4 (bool), showL5 (bool)
  - Outputs: ax (axis object)
- **Jacobi_c** (computes the Jc of a given state vector)
  - Inputs: s (4x1 array-like), mu (float)
  - Outputs: Jc (float)
- **CR3BP_ds** (returns the first temporal derivatives of a given state vector)
  - Inputs: s (4x1 array-like), mu (float)
  - Outputs: ds (4x1 array-like)
- **shoot** (propagates a state vector a given amount of time, and optionally outputs it as a figure)
  - Inputs: mu (float), s (4x1 array-like), tspan (float), style (string), t0 (float), res (float)
  - Outputs: sol (solution-type object)
- **shoot_to_poincare_x** (propagates a given state vector until it crosses a Poincaré section defined at a constant $\hat{x}$ coordinate a desired number of times)
  - Inputs: mu (float), s_ini (4x1 array-like), bstep (float), style (string), crossings (float), x_obj (float)
  - Outputs: s_end (4x1 array-like), t_poincare (float)

- **shoot_to_poincare_y** (propagates a given state vector until it crosses a Poincaré section defined at a constant ŷ coordinate a desired number of times)
  - Inputs: mu (float), s_ini (4x1 array-like), bstep (float), style (string), crossings (float), y_obj (float)
  - Outputs: s_end (4x1 array-like), t_poincare (float)
- **CR3BP_A** (returns the corresponding Jacobian of the equations of motion A for a given point)
  - Inputs: x (float), y (float), mu (float)
  - Outputs: A (4x4 array-like)
- **clear_small** (removes very small numbers that result from numerical processes and that should be considered zero)
  - Inputs: matrix (array-like), tol (float)
  - Outputs: matrix (array-like)
- **remove_i** (removes imaginary parts form a given set of eigenvectors)
  - Inputs: veps (array-like)
  - Outputs: C (array-like)
- **lin_eq_c1** (returns the linear approximation of the equations of motion around the collinear points, assuming $A3 = A4 = 0$)[ii]
  - Inputs: vaps (float), A1 (float), A2 (float), y0 (float)
  - Outputs: t0 (float), x (float), dx (float), dy (float)
- **lin_eq_c2** (returns the linear approximation of the equations of motion around the collinear points, assuming $A1 = A2 = 0$) [iii]
  - Inputs: vaps (float), A3 (float), A4 (float), y0 (float)
  - Outputs: t0 (float), x (float), dx (float), dy (float)
- **lin_eq_1** (returns the linear approximation of the equations of motion around the equilateral points, assuming $A3 = A4 = 0$)
  - Inputs: C, mvaps (float), A1 (float), A2 (float), y0 (float)
  - Outputs: t0 (float), x (float), dx (float), dy (float)
- **lin_eq_2** (returns the linear approximation of the equations of motion around the equilateral points, assuming $A1 = A2 = 0$)

[ii] Note that this function has not been utilized in this project for data from NASA-JPL has been used instead as initial approximation to find the periodic orbits around L1 and L2 [25].
[iii] Idem.

  - Inputs: C, mvaps (float), A3 (float), A4 (float), y0 (float)
  - Outputs: t0 (float), x (float), dx (float), dy (float)

- **variationals** (returns the first derivative of an STM for a given state-vector)
  - Inputs: x (4x1 array-like), mu (float)
  - Outputs: var (4x4 array-like)
- **variationals_prop** (returns the STM of a given trajectory after propagating them a given timespan)
  - Inputs: mu (float), s0 (4x1 array-like), tspan (float), t0 (float)
  - Outputs: phi (4x4 array-like)

```python
# Aux. functions shared by many of the scripts developed for L1, L2, L4, and L5 motion.
# Aitor Urruticoechea 2022
import numpy as np
from scipy import optimize
from scipy import integrate
from scipy import linalg
import matplotlib.pyplot as plt
from matplotlib.ticker import MultipleLocator

########################
# FUNCTION FROM https://github.com/RayleighLord/RayleighLordAnimations/blob/
↪  master/publication\%20quality\%20figures/fig_config.py
def add_grid(ax, lines=True, locations=None):
    """Add a grid to the current plot.
    Args:
        ax (Axis): axis object in which to draw the grid.
        lines (bool, optional): add lines to the grid. Defaults to True.
        locations (tuple, optional):
            (xminor, xmajor, yminor, ymajor). Defaults to None.
    """

    if lines:
        ax.grid(lines, alpha=0.5, which="minor", ls=":")
        ax.grid(lines, alpha=0.7, which="major")

    if locations is not None:

        assert (
            len(locations) == 4
        ), "Invalid entry for the locations of the markers"

        xmin, xmaj, ymin, ymaj = locations

        ax.xaxis.set_minor_locator(MultipleLocator(xmin))
        ax.xaxis.set_major_locator(MultipleLocator(xmaj))
        ax.yaxis.set_minor_locator(MultipleLocator(ymin))
        ax.yaxis.set_major_locator(MultipleLocator(ymaj))
###################



# Import basic data
```

```python
def load_basic_data():
    try:
        basic_data = np.loadtxt('data/GMdata.txt')
    except:
        raise Exception('This code needs data on the G constant and the planetary masses!')
    G = basic_data[0]
    GM_sun = basic_data[1]
    GM_earth = basic_data[2]
    GM_moon = basic_data[3]
    mu_earthsun = (GM_earth + GM_moon)/(GM_sun + GM_earth + GM_moon)
    return mu_earthsun

# Import Lagrange Points
def import_LP():
    try:
        LagrangePoints = np.loadtxt('data/LagrangePoints.txt')
    except:
        raise Exception('This code needs to be run after the Lagrange Points have been
        ↪  calculated. Run lagrange_points.py first!')
    L1 = LagrangePoints[0]
    L2 = LagrangePoints[1]
    L4 = LagrangePoints[2]
    L5 = LagrangePoints[3]
    return L1,L2,L4,L5


# Function to plot CR3BP key points
def system_plot(mu_earthsun,L1,L2,L4,L5,xlim=[-0.05, 1.05],ylim=[-1, 1],zoom=0, showSun=True,
↪  showEarth=True, showL1=True, showL2=True, showL4=True, showL5=True):
    ax = plt.axes()
    if zoom==0:
        add_grid(ax, locations=(0.1,0.5,0.25,0.5))
    elif zoom==1:
        add_grid(ax, locations=(0.0025,0.01,0.0025,0.01))
    elif zoom==2:
        add_grid(ax, locations=(0.00025,0.001,0.00025,0.001))
    elif zoom==3:
        add_grid(ax, locations=(0.000025,0.0001,0.000025,0.0001))
    if showSun: plt.scatter(-mu_earthsun, 0, c='orange',edgecolor='black', label='Sun')
    if showEarth: plt.scatter(1-mu_earthsun,0, c='white',edgecolor='black', label='Earth')
    if showL1: plt.scatter(L1,0,c='#ff2f36', marker="x", label='L1')
    if showL2: plt.scatter(L2,0,c='#ac429e', marker="x", label='L2')
    if showL4: plt.scatter(L4,np.sqrt(3)/2,c='#febd00', marker="x", label='L4')
    if showL5: plt.scatter(L5,-np.sqrt(3)/2,c='#007dca', marker="x", label='L5')
    first_legend = plt.legend()
    plt.xlabel('x [NDU]')
    plt.ylabel('y [NDU]')
    try:
        plt.xlim(xlim)
        plt.ylim(ylim)
    except:
        pass
    ax.add_artist(first_legend)
    return ax

def Jacobi_c(s,mu):
    x = s[0]
    y = s[1]
    vx = s[2]
    vy = s[3]
    r = np.linalg.norm([x+mu-1, y]) #dist to smaller primary
    d = np.linalg.norm([x+mu, y]) #dist to larger primary
    omega = ((1-mu)/(d)) + (mu/(r)) + (1/2)*(x**2 + y**2)
    Jc = 2*omega - np.linalg.norm([vx,vy])**2
    return Jc

# CR3BP Movement description (2D)
def CR3BP_ds(s, mu):
    x = s[0]
    y = s[1]
    vx = s[2]
    vy = s[3]
    r = np.linalg.norm([(s[0])+mu-1, (s[1])]) #dist to smaller primary
    d = np.linalg.norm([(s[0])+mu, (s[1])]) #dist to larger primary
    ds = np.zeros(4)
    ds[0] = vx
    ds[1] = vy
    ds[2] = 2*vy + x - ((1-mu)/(d**3))*(x+mu) - (mu/(r**3))*(x-(1-mu))
    ds[3] = -2*vx + y - ((1-mu)/(d**3))*y - mu*y/(r**3)
    return ds

# CR3BP Propagation Function
```

```
def shoot(mu, s, tspan, style='none', t0=0, res=100):
    FUN = lambda t,x: np.array(CR3BP_ds(x,mu))
    sol = integrate.solve_ivp(FUN, [t0, tspan], s, t_eval = np.linspace(t0,tspan,res),
    ↪ method='DOP853', rtol = 1.e-10, atol = 1e-13)
    if style!='none':
        try:
            plt.plot(sol.y[0,:],sol.y[1,:],style)
            plt.pause(0.05)
        except:
            pass
    return sol

# Pointcare section (with x=k)
def shoot_to_poincare_x(mu, s_ini, bstep=3, style='none', crossings=2, x_obj=None):
    x = s_ini[0]
    y = s_ini[1]
    if x_obj == None: x0 = x
    else: x0 = x_obj
    dx = s_ini[2]
    dy = s_ini[3]
    t = 0
    n_cross = 0
    while n_cross < crossings:
        x_prev = x
        y_prev = y
        dx_prev = dx
        dy_prev = dy
        sol_t = shoot(mu, [x,y,dx,dy], t+bstep, style=style, t0=t,res=2)
        x = float(sol_t.y[0,1])
        y = float(sol_t.y[1,1])
        dx = float(sol_t.y[2,1])
        dy = float(sol_t.y[3,1])
        if (x_prev<x0 and x>x0) or (x_prev>x0 and x<x0):
            n_cross = n_cross+1
        t = t+bstep
    t = t-bstep
    x = x_prev
    y = y_prev
    dx = dx_prev
    dy = dy_prev
    def poincare_crossing(time):
        sol_t = shoot(mu, [x,y,dx,dy], time, t0=t,res=2, style='none')
        x_poincare = float(sol_t.y[0,1])
        return x_poincare - x0
    t_poincare = optimize.newton(poincare_crossing, t+bstep/2,maxiter=1000)
    sol_t = shoot(mu, [x,y,dx,dy], t_poincare, style=style, t0=t,res=2)
    x = float(sol_t.y[0,1])
    y = float(sol_t.y[1,1])
    dx = float(sol_t.y[2,1])
    dy = float(sol_t.y[3,1])
    s_end = [x,y,dx,dy]
    return s_end, t_poincare

# Pointcare section (with y=k)
def shoot_to_poincare_y(mu, s_ini, bstep=0.1, style='none',crossings=2, y_obj=None ):
    x = s_ini[0]
    y = s_ini[1]
    if y_obj == None: y0 = y
    else: y0 = y_obj
    dx = s_ini[2]
    dy = s_ini[3]
    t = 0
    n_cross = 0
    while n_cross < crossings:
        x_prev = x
        y_prev = y
        dx_prev = dx
        dy_prev = dy
        sol_t = shoot(mu, [x,y,dx,dy], t+bstep, style=style, t0=t,res=2)
        x = float(sol_t.y[0,1])
        y = float(sol_t.y[1,1])
        dx = float(sol_t.y[2,1])
        dy = float(sol_t.y[3,1])
        if (y_prev<y0 and y>y0) or (y_prev>y0 and y<y0):
            n_cross = n_cross+1
        t = t+bstep
    t = t-bstep
    x = x_prev
    y = y_prev
    dx = dx_prev
    dy = dy_prev
    def poincare_crossing(time):
```

```python
            sol_t = shoot(mu, [x,y,dx,dy], time, t0=t,res=2, style='none')
            y_poincare = float(sol_t.y[1,1])
            return y_poincare - y0
        t_poincare = optimize.newton(poincare_crossing, t+bstep/2,maxiter=1000)
        sol_t = shoot(mu, [x,y,dx,dy], t_poincare, style=style, t0=t,res=2)
        x = float(sol_t.y[0,1])
        y = float(sol_t.y[1,1])
        dx = float(sol_t.y[2,1])
        dy = float(sol_t.y[3,1])
        s_end = [x,y,dx,dy]
        return s_end, t_poincare


# System definition (s' = A s) (2D)
def CR3BP_A(x,y,mu):
    r = +np.sqrt((x-1+mu)**2+y**2) #dist to smaller primary
    d = +np.sqrt((x+mu)**2+y**2) #dist to larger primary
    Uxx = 1 - (1-mu)/d**3 - mu/r**3 + (3*(1-mu)*(x+mu)**2)/d**5 +(3*mu*(x-1+mu)**2)/r**5
    Uxy = (3*(1-mu)*(x+mu)*y)/d**5 + (3*mu*(x-1+mu)*y)/r**5
    Uyx = Uxy
    Uyy = 1 - (1-mu)/d**3 - mu/r**3 + (3*(1-mu)*y**2)/d**5 + (3*mu*y**2)/r**5
    A = [[0, 0, 1, 0],[0, 0, 0, 1],[Uxx, Uxy, 0, 2],[Uyx, Uyy, -2, 0]]
    return A

# Numerical processes often times leave behind very small results that can be considered zero
def clear_small(matrix,tol):
    if np.ndim(matrix)==2: #CASE 2D
        for n in range(np.size(matrix,0)):
            for m in range(np.size(matrix,1)):
                real = np.real(matrix[n,m])
                imag = np.imag(matrix[n,m])
                if real<tol and real>-tol:
                    matrix[n,m] = np.imag(matrix[n,m])*1j
                if imag<tol and imag>-tol:
                    matrix[n,m] = np.real(matrix[n,m])
    else: #CASE 1D
        for n in range(np.size(matrix,0)):
            real = np.real(matrix[n])
            imag = np.imag(matrix[n])
            if real<tol and real>-tol:
                matrix[n] = imag*1j
            if imag<tol and imag>-tol:
                matrix[n] = real
    return matrix

# Remove imaginary part from eignvectors to create C matrix
def remove_i(veps):
    C = np.zeros([np.size(veps,0),np.size(veps,1)])
    for a in range(int(np.size(veps,0)/2)):
        C[:,2*a] = np.real(veps[:,2*a])
        C[:,2*a+1] = np.imag(veps[:,2*a])
    return C

# First set of lineal equations for collinear points
def lin_eq_c1(vaps, A1, A2, y0):
    lam = np.real(vaps[0])
    c = (lam**2-1)/(2*lam)
    def y_fun(t):
        return c*A1*np.e**(lam*t) + c*A2*np.e**(-lam*t) - y0
    t0 = optimize.fsolve(y_fun,0.1)
    x = A1*np.e**(lam*t0) + A2*np.e**(-lam*t0)
    dx = (A1/lam) *np.e**(lam*t0) - (A2/lam) *np.e**(-lam*t0)
    dy = (c*A1/lam) *np.e**(lam*t0) - (c*A2/lam) * np.e**(-lam*t0)
    return [t0, x, dx, dy]

# Second set of lineal equations for collinear points
def lin_eq_c2(vaps, A3, A4, y0):
    w1 = np.imag(vaps[2])
    kappa = -(w1**2+1)/(2*w1)
    def y_fun(t):
        return kappa*A3*np.cos(w1*t) + kappa*A4*np.sin(w1*t) - y0
    t0 = optimize.fsolve(y_fun, 0)
    x = A3*np.cos(w1*t0) + A4*np.sin(w1*t0)
    dx = -A3*w1*np.sin(w1*t0) +  A4*w1*np.cos(w1*t0)
    dy = -kappa*A3*w1*np.sin(w1*t0) +  kappa*A4*w1*np.cos(w1*t0)
    return [t0, x, dx, dy]

# First set of lineal equations for equilateral points
def lin_eq_1(C, mvaps, A1, A2, y0):
    # A3 = A4 = 0
    Vw1_0 = C[0,0]
    Vw1_1 = C[1,0]
    Vw1_2 = C[2,0]
    Vw1_3 = C[3,0]
    Uw1_0 = C[0,1]
    Uw1_1 = C[1,1]
    Uw1_2 = C[2,1]
    Uw1_3 = C[3,1]
    w1 = mvaps[0]
    #w2 = mvaps[2]
    def y_fun(t):
        return A1*np.cos(w1*t)*Vw1_1 + A2*np.sin(w1*t)*Uw1_1 - y0
    t0 = optimize.fsolve(y_fun,0.1)
    x = A1*np.cos(w1*t0)*Vw1_0 + A2*np.sin(w1*t0)*Uw1_0
    dx = A1*np.cos(w1*t0)*Vw1_2 + A2*np.sin(w1*t0)*Uw1_2
    dy = A1*np.cos(w1*t0)*Vw1_3 + A2*np.sin(w1*t0)*Uw1_3
    return [t0, x, dx, dy]

# Second set of Lineal equations for equilateral points
def lin_eq_2(C, mvaps, A3, A4, y0):
    # A1 = A2 = 0
    Vw2_0 = C[0,2]
    Vw2_1 = C[1,2]
    Vw2_2 = C[2,2]
    Vw2_3 = C[3,2]
    Uw2_0 = C[0,3]
    Uw2_1 = C[1,3]
    Uw2_2 = C[2,3]
    Uw2_3 = C[3,3]
    #w1 = mvaps[0]
    w2 = mvaps[2]
    def y_fun(t):
        return A3*np.cos(w2*t)*Vw2_1 + A4*np.sin(w2*t)*Uw2_1 - y0
    t0 = optimize.fsolve(y_fun,0.1)
    x = A3*np.cos(w2*t0)*Vw2_0 + A4*np.sin(w2*t0)*Uw2_0
    dx = A3*np.cos(w2*t0)*Vw2_2 + A4*np.sin(w2*t0)*Uw2_2
    dy = A3*np.cos(w2*t0)*Vw2_3 + A4*np.sin(w2*t0)*Uw2_3
    return [t0, x, dx, dy]

# Variationals - STM
def variationals(x, mu):
    s = [x[0], x[1], x[2], x[3]]
    ds = CR3BP_ds(s, mu)
```

```python
    r = +np.sqrt((x[0]-1+mu)**2+x[1]**2) #dist to smaller primary
    d = +np.sqrt((x[0]+mu)**2+x[1]**2) #dist to larger primary
    Uxx = 1 - (1-mu)/d**3 - mu/r**3 + (3*(1-mu)*(x[0]+mu)**2)/d**5 +(3*mu*(x[0]-1+mu)**2)/r**5
    Uxy = (3*(1-mu)*(x[0]+mu)*x[1])/d**5 + (3*mu*(x[0]-1+mu)*x[1])/r**5
    Uyx = Uxy
    Uyy = 1 - (1-mu)/d**3 - mu/r**3 + (3*(1-mu)*x[1]**2)/d**5 + (3*mu*x[1]**2)/r**5
    lam_d = np.array([
        [x[6]                     , x[10]                     , x[14]                      ,
        ↪  x[18]],
        [x[7]                     , x[11]                     , x[15]                      ,
        ↪  x[19]],
        [Uxx*x[4]+Uxy*x[5]+2*x[7] , Uxx*x[8]+Uxy*x[9]+2*x[11] , Uxx*x[12]+Uxy*x[13]+2*x[15] ,
        ↪  Uxx*x[16]+Uxy*x[17]+2*x[19]],
        [Uyx*x[4]+Uyy*x[5]-2*x[6] , Uyx*x[8]+Uyy*x[9]-2*x[10] , Uyx*x[12]+Uyy*x[13]-2*x[14] ,
        ↪  Uyx*x[16]+Uyy*x[17]-2*x[18]],
    ])
    var = np.append(ds, lam_d[:,0])
    var = np.append(var, lam_d[:,1])
    var = np.append(var, lam_d[:,2])
    var = np.append(var, lam_d[:,3])
    return var

def variationals_prop(mu, s0, tspan, t0=0):
    tspan = round(tspan, 5)
    FUN = lambda t, x: np.array(variationals(x, mu))
    init = np.append(np.array(s0), np.identity(4))
    sol = integrate.solve_ivp(FUN, [t0, tspan], init, t_eval =
    ↪  np.linspace(t0,tspan,int(tspan*100000)), method='DOP853',rtol = 1.e-10, atol = 1e-13)
    STM = sol.y[:,-1]
    phi = np.zeros([4,4])
    for n in range(4):
        phi[n,0] = STM[4+4*n]
        phi[n,1] = STM[4+4*n+1]
        phi[n,2] = STM[4+4*n+2]
        phi[n,3] = STM[4+4*n+3]
    return phi
```

A.3 Lagrange_Points.py

This Python code solves the quintic equations to find the exact location of the Lagrange Points of interest. For enhanced usability, the individual functions to find each Lagrange Point have been defined, so they can be imported by a third party.

```python
# Finding L1, L2, L4 & L5 (CR3BP) using scipy's Newton-Raphson method
# data form JPL-NASA - Park, R.S., et al., 2021
# Aitor Urruticoechea 2022

import numpy as np
from scipy import optimize
from aux_functions import *

try:
    basic_data = np.loadtxt('data/GMdata.txt')
except:
    raise Exception('This code needs data on the G constant and the Planetary masses!')
G = basic_data[0]
GM_sun = basic_data[1]
GM_earth = basic_data[2]
GM_moon = basic_data[3]

mu_earthsun = (GM_earth + GM_moon)/(GM_sun + GM_earth + GM_moon)


def findL1(guess,mu): #x coordinates (y,z = 0)
    def fun_gamma1(gamma1):
        return 1-mu-gamma1-(1-mu)/((1-gamma1)**2) + mu/(gamma1**2)
    fprime_gamma1 = lambda gamma1: -1- 2*(1-mu)/((1-gamma1)**3) - 2*mu/(gamma1**3)
    root = optimize.newton(fun_gamma1, guess, fprime_gamma1)
    return 1- mu -root

def findL2(guess,mu): #x coordinates (y,z = 0)
    def fun_gamma2(gamma2):
        return 1-mu+gamma2-(1-mu)/((1+gamma2)**2) - mu/(gamma2**2)
    fprime_gamma2 = lambda gamma2: 1- 2*(mu-1)/((1+gamma2)**3) + 2*mu/(gamma2**3)
    root = optimize.newton(fun_gamma2, guess, fprime_gamma2)
    return 1- mu +root

def findL4L5(mu): #x coordinates (y = sqrt(3)/2 NDU, z = 0)
```

```python
    return 1/2 - mu

L1 = findL1(0.1, mu_earthsun)
L2 = findL2(0.1, mu_earthsun)
L4 = findL4L5(mu_earthsun)
L5 = findL4L5(mu_earthsun)

np.savetxt('data/LagrangePoints.txt',[L1,L2,L4,L5])

system_plot(mu_earthsun,L1,L2,L4,L5)
plt.show()

```

A.4 L1_LyapunovOrbits.py

Using the Identification-Correction-Continuation method detailed in Part 2, a set of periodic orbits around L1 have been mapped. Here, it is interesting to note that the steps taken when the continuation is made to the next orbit need to be very small to ensure convergence due to the saddle-point stability of L1. Thus, only one every a hundred orbits calculated is actually stored and displayed on screen.

```python
# Plotting key Lyapunov orbits around L1
# data form JPL-NASA - Park, R.S., et al., 2021
# Aitor Urruticoechea 2022
import numpy as np
from scipy import optimize
from scipy import integrate
from scipy import linalg
import matplotlib.pyplot as plt
from matplotlib.patches import Patch
from aux_functions import *

# Basic operational data
mu_earthsun = load_basic_data()

# Data to be Imported
L1, L2, L4, L5 = import_LP()

# JPL gives the initial conditions
x = 0.99136
y = 0
dx = 0
```

```python
dy = -0.00841535092

# Plots
ax = system_plot(mu_earthsun,L1,L2,L4,L5,xlim=[0.97,1.02],ylim=[-0.025, 0.025],zoom=1,
↪ showSun=False, showL4=False, showL5=False)
#ax = system_plot(mu_earthsun,L1,L2,L4,L5,xlim=[0.985,0.995],ylim=[-0.005, 0.005],zoom=2,
↪ showSun=False, showL4=False, showL5=False, showEarth = False, showL2=False)

def objective(ds0,s0,mu, style='none', step=0.5):
    s_fin, t = shoot_to_poincare_y(mu, [s0[0],s0[1],ds0[0],ds0[1]], style=style, bstep=step)
    s_f = []
    s_f.append(s_fin[0]+s_fin[2])
    s_f.append(s_fin[0]+s_fin[3])
    return np.array(s_f)-np.array([s0[0]+ds0[0], s0[0]+ds0[1]])


print("")
print(">> L1 Lyapunov Orbits <<")
print("")

fid1 = open('data/L1_LyapunovOrbits.txt','w')
success_1 = 0
fail_1 = 0
success_2 = 0
fail_2 = 0
total = 0

tol = 10**(-6)
x0 = x

s0 = [x0, y]
ds0 = [dx, dy]
root1 = optimize.root(objective, ds0, args=(s0,mu_earthsun,'none',0.5))
s_fin, t_1 = shoot_to_poincare_y(mu_earthsun, [s0[0],s0[1],root1.x[0],root1.x[1]], style='-b')
print('Orbit at: [s0]; ds0; time')
print(s0)
print(str(root1.x[0]) + ' '+ str(root1.x[1]))
print(t_1)
print('')
fid1.write(str(s0[0]) + " " + str(s0[1]) + " " + str(root1.x[0]) + " " + str(root1.x[1]) + " "
↪ + str(t_1) + '\n')
```

```

count = 1
x0 = x0 + 5*10**(-7)
while count<1001:
    s0 = [x0, y]
    #ds0 = [root1.x[0], root1.x[1]]
    ds0 = [0, root1.x[1]]
    root1 = optimize.root(objective, ds0, args=(s0,mu_earthsun,'none',0.5), method='hybr', tol=10**(-10))
    s_fin, t_1 = shoot_to_poincare_y(mu_earthsun, [s0[0],s0[1],root1.x[0],root1.x[1]])
    if root1.success and (np.array([s0[0],s0[1],root1.x[0],root1.x[1]])-np.array(s_fin)<tol).all():
        count = count+1
        x0 = x0 + 5*10**(-7)
        if count%100 == 0: #Very small steps are taken, so only 1 in 1000 orbits are stored
            print('Orbit at: [s0]; ds0; time')
            print(s0)
            print(str(root1.x[0]) + ' '+ str(root1.x[1]))
            print(t_1)
            shoot(mu_earthsun,[s0[0],s0[1],root1.x[0],root1.x[1]], t_1, style='-',res=3000)
            fid1.write(str(s0[0]) + " " + str(s0[1]) + " " + str(root1.x[0]) + " " + str(root1.x[1]) + " " + str(t_1) + '\n')
            print('')
    elif not(root1.success and (np.array([s0[0],s0[1],root1.x[0],root1.x[1]])-np.array(s_fin)<tol).all()):
        print('Warning: Solution not good enough (count: ' + str(count)+').' )

fid1.close()
legend_object0 = Patch(facecolor='blue', edgecolor='black')
legend_object1 = Patch(facecolor='C0', edgecolor='black')
legend_object2 = Patch(facecolor='C1', edgecolor='black')
legend_object3 = Patch(facecolor='C2', edgecolor='black')
legend_object4 = Patch(facecolor='C3', edgecolor='black')
legend_object5 = Patch(facecolor='C4', edgecolor='black')
legend_object6 = Patch(facecolor='C5', edgecolor='black')
legend_object7 = Patch(facecolor='C6', edgecolor='black')
legend_object8 = Patch(facecolor='C7', edgecolor='black')
legend_object9 = Patch(facecolor='C8', edgecolor='black')
legend_object10 = Patch(facecolor='C9', edgecolor='black')
```

```
plt.legend(handles=[legend_object0, legend_object1, legend_object2, legend_object3,
↪ legend_object4, legend_object5, legend_object6,legend_object7, legend_object8,
↪ legend_object9, legend_object10],labels=['','','', '','','','','','','', 'Lyapunov
↪ Orbits'],ncols=11, handletextpad=0.5, handlelength=1.0, columnspacing=-0.5,loc='lower
↪ right')
plt.show()
```

A.5 L2_LyapunovOrbits.py

In the same way of L1, an Identification-Correction-Continuation scheme has been applied only saving a small fraction of the orbits generated (this time, one every two hundred).

```
# Plotting key Lyapunov orbits around L2
# data form JPL-NASA - Park, R.S., et al., 2021
# Aitor Urruticoechea 2022
import numpy as np
from scipy import optimize
from scipy import integrate
from scipy import linalg
import matplotlib.pyplot as plt
from matplotlib.patches import Patch
from aux_functions import *

# Basic operational data
mu_earthsun = load_basic_data()

# Data to be Imported
L1, L2, L4, L5 = import_LP()

# JPL gives the initial conditions
x = 1.01103
y = 0
dx = 0
dy = -0.006897268596

# Plots
ax = system_plot(mu_earthsun,L1,L2,L4,L5,xlim=[0.97,1.02],ylim=[-0.025, 0.025],zoom=1,
↪ showSun=False, showL4=False, showL5=False)
#ax = system_plot(mu_earthsun,L1,L2,L4,L5,xlim=[1.004, 1.016],ylim=[-0.006, 0.006],zoom=2,
↪ showSun=False, showL4=False, showL5=False, showEarth = False, showL1=False)

```

```python
def objective(ds0,s0,mu, style='none', step=0.1):
    s_fin, t = shoot_to_poincare_y(mu, [s0[0],s0[1],ds0[0],ds0[1]], style=style, bstep=step)
    s_f = []
    s_f.append(s_fin[0]+s_fin[2])
    s_f.append(s_fin[0]+s_fin[3])
    return np.array(s_f)-np.array([s0[0]+ds0[0], s0[0]+ds0[1]])


print("")
print(">> L2 Lyapunov Orbits <<")
print("")

fid1 = open('data/L2_LyapunovOrbits.txt','w')
success_1 = 0
fail_1 = 0
success_2 = 0
fail_2 = 0
total = 0

tol = 10**(-6)
x0 = x

s0 = [x0, y]
ds0 = [dx, dy]
root1 = optimize.root(objective, ds0, args=(s0,mu_earthsun,'none',0.5))
s_fin, t_1 = shoot_to_poincare_y(mu_earthsun, [s0[0],s0[1],root1.x[0],root1.x[1]], style='-b')
print('Orbit at: [s0]; ds0; time')
print(s0)
print(str(root1.x[0]) + ' '+ str(root1.x[1]))
print(t_1)
print('')
fid1.write(str(s0[0]) + " " + str(s0[1]) + " " + str(root1.x[0]) + " " + str(root1.x[1]) + " "
 ↪ + str(t_1) + '\n')

count = 1
x0 = x0 + 5*10**(-7)
while count<2001:
    s0 = [x0, y]
    ds0 = [0, root1.x[1]]
    root1 = optimize.root(objective, ds0, args=(s0,mu_earthsun,'none',0.5), method='hybr',
     ↪ tol=10**(-10))
    s_fin, t_1 = shoot_to_poincare_y(mu_earthsun, [s0[0],s0[1],root1.x[0],root1.x[1]])
    if root1.success and
    ↪  (np.array([s0[0],s0[1],root1.x[0],root1.x[1]])-np.array(s_fin)<tol).all():
        count = count+1
        x0 = x0 + 2.5*10**(-7)
        if count%200 == 0: #Very small steps are taken, so only 1 in 200 orbits are stored
            print('Orbit at: [s0]; ds0; time')
            print(s0)
            print(str(root1.x[0]) + ' '+ str(root1.x[1]))
            print(t_1)
            shoot(mu_earthsun,[s0[0],s0[1],root1.x[0],root1.x[1]], t_1, style='-',res=3000)
            fid1.write(str(s0[0]) + " " + str(s0[1]) + " " + str(root1.x[0]) + " " +
            ↪  str(root1.x[1]) + " " + str(t_1) + '\n')
            print('')
    elif not(root1.success and
    ↪  (np.array([s0[0],s0[1],root1.x[0],root1.x[1]])-np.array(s_fin)<tol).all()):
        print('Warning: Solution not good enough (count: ' + str(count)+').' )


fid1.close()
legend_object0 = Patch(facecolor='blue', edgecolor='black')
legend_object1 = Patch(facecolor='C0', edgecolor='black')
legend_object2 = Patch(facecolor='C1', edgecolor='black')
legend_object3 = Patch(facecolor='C2', edgecolor='black')
legend_object4 = Patch(facecolor='C3', edgecolor='black')
legend_object5 = Patch(facecolor='C4', edgecolor='black')
legend_object6 = Patch(facecolor='C5', edgecolor='black')
legend_object7 = Patch(facecolor='C6', edgecolor='black')
legend_object8 = Patch(facecolor='C7', edgecolor='black')
legend_object9 = Patch(facecolor='C8', edgecolor='black')
legend_object10 = Patch(facecolor='C9', edgecolor='black')
plt.legend(handles=[legend_object0, legend_object1, legend_object2, legend_object3,
↪  legend_object4, legend_object5, legend_object6,legend_object7, legend_object8,
↪  legend_object9, legend_object10],labels=['','','', '','','','','','','', 'Lyapunov
↪  Orbits'],ncols=11, handletextpad=0.5, handlelength=1.0, columnspacing=-0.5,loc='lower
↪  right')
plt.show()
```

A.6 L4_PlanarOrbits.py

In the same way of the collinear points, an Identification-Correction-Continuation scheme has been applied. Note, however, the equations of motion need to be linearized to obtain the starting orbit, so the first line vary slightly when compared to the previous ones. This time, larger steps when the continuation is made to the next orbit are possible, and thus every orbit is saved and displayed.

```python
# Plotting key Lyapunov orbits around L4
# Aitor Urruticoechea 2022
import numpy as np
from scipy import optimize
from scipy import integrate
from scipy import linalg
import matplotlib.pyplot as plt
from matplotlib.patches import Patch
from aux_functions import *

# Basic operational data
mu_earthsun = load_basic_data()

# Data to be Imported
L1, L2, L4, L5 = import_LP()

# Linealized system gives the initial conditions
x = L4
y = np.sqrt(3)/2 # L4 location
A = np.array(CR3BP_A(x,y,mu_earthsun)) # Function from aux_functions

vaps = linalg.eigvals(A)
veps = linalg.eig(A)[1]
vaps = clear_small(vaps,10**(-10))
veps = clear_small(veps,10**(-10))

omega_vaps = np.imag(vaps)
C = remove_i(veps)
invC = linalg.inv(C)

check = clear_small(invC@A@C,10**(-10))

# Plots
```

```python
system_plot(mu_earthsun,L1,L2,L4,L5,xlim=[0.4998,0.50015],ylim=[0.8659,
↪ 0.8661],zoom=3,showSun=False, showEarth=False,showL1=False,showL2=False,showL5=False)
#system_plot(mu_earthsun,L1,L2,L4,L5,xlim=[0.472,0.527],ylim=[0.85,
↪ 0.882],zoom=1,showSun=False, showEarth=False,showL1=False,showL2=False,showL5=False)


# Objective function
def objective(ds0,s0,mu, style='none', step=5):
    s_fin, t = shoot_to_poincare_x(mu, [s0[0],s0[1],ds0[0],ds0[1]], style=style, bstep=step)
    s_f = []
    s_f.append(s_fin[1]+s_fin[2])
    s_f.append(s_fin[1]+s_fin[3])
    return np.array(s_f)-np.array([s0[1]+ds0[0], s0[1]+ds0[1]])

print("")
print(">> L4 Planar Orbits <<")
print("")

fid1 = open('data/L4_PlanarOrbits.txt','w')
success_1 = 0
fail_1 = 0
success_2 = 0
fail_2 = 0
total = 0

tol = 10**(-8)
can_continue = False
y0 = y
while not can_continue:
    y0 = y0 + 10**(-5)
    [t0_1, x0_1, dx_1, dy_1] = lin_eq_1(C, omega_vaps, 10**(-5), 10**(-5), y0)
    s0 = [float(x+x0_1),y0]
    root1 = optimize.root(objective, [dx_1,dy_1], args=(s0,mu_earthsun,'none',5),
    ↪ method='hybr', tol=10**(-10))
    s_fin, t_1 = shoot_to_poincare_x(mu_earthsun, [s0[0],s0[1],root1.x[0],root1.x[1]])
    if root1.success and
    ↪ (np.array([s0[0],s0[1],root1.x[0],root1.x[1]])-np.array(s_fin)<tol).all():
        #The initial orbit does incorporate some short-term movement so it is not plotted
        #print('Orbit at: [s0]; ds0; time')
        #print(s0)
        #print(str(root1.x[0]) + ' '+ str(root1.x[1]))
        #print(t_1)
        #shoot(mu_earthsun,[s0[0],s0[1],root1.x[0],root1.x[1]], t_1, style='-',res=3000)
        #fid1.write(str(s0[0]) + " " + str(s0[1]) + " " + str(root1.x[0]) + " " +
        ↪  str(root1.x[1]) + " " + str(t_1) + '\n')
        print('')
        can_continue = True

while y0 <= y + 10*10**(-5):
    y0 = y0 + 10**(-5)
    s0[1] = y0
    ds0 = [root1.x[0], root1.x[1]]
    root1 = optimize.root(objective, ds0, args=(s0,mu_earthsun,'none',5), method='hybr',
    ↪  tol=10**(-10))
    s_fin, t_1 = shoot_to_poincare_x(mu_earthsun, [s0[0],s0[1],root1.x[0],root1.x[1]])
    if root1.success and
    ↪  (np.array([s0[0],s0[1],root1.x[0],root1.x[1]])-np.array(s_fin)<tol).all():
        print('Orbit at: [s0]; ds0; time')
        print(s0)
        print(str(root1.x[0]) + ' '+ str(root1.x[1]))
        print(t_1)
        shoot(mu_earthsun,[s0[0],s0[1],root1.x[0],root1.x[1]], t_1, style='-',res=3000)
        fid1.write(str(s0[0]) + " " + str(s0[1]) + " " + str(root1.x[0]) + " " +
        ↪  str(root1.x[1]) + " " + str(t_1) + '\n')
        print('')

fid1.close()
legend_object1 = Patch(facecolor='C0', edgecolor='black')
legend_object2 = Patch(facecolor='C1', edgecolor='black')
legend_object3 = Patch(facecolor='C2', edgecolor='black')
legend_object4 = Patch(facecolor='C3', edgecolor='black')
legend_object5 = Patch(facecolor='C4', edgecolor='black')
legend_object6 = Patch(facecolor='C5', edgecolor='black')
legend_object7 = Patch(facecolor='C6', edgecolor='black')
legend_object8 = Patch(facecolor='C7', edgecolor='black')
legend_object9 = Patch(facecolor='C8', edgecolor='black')
legend_object10 = Patch(facecolor='C9', edgecolor='black')

```

```python
plt.legend(handles=[legend_object1, legend_object2, legend_object3, legend_object4,
↪  legend_object5, legend_object6,legend_object7,
↪  legend_object8,legend_object9,legend_object10],labels=['','','','','','','','','',
↪  'Long-term planar orbits'],ncols=11, handletextpad=0.5, handlelength=1.0,
↪  columnspacing=-0.5,loc='upper left')
plt.show()
```

A.7 L5_PlanarOrbits.py

In the same way of L4, an Identification-Correction-Continuation scheme has been applied once the linearized equations of motion have resulted in a periodic orbit around L5.

```python
# Plotting key Lyapunov orbits around L5
# Aitor Urruticoechea 2022
import numpy as np
from scipy import optimize
from scipy import integrate
from scipy import linalg
import matplotlib.pyplot as plt
from matplotlib.patches import Patch
from aux_functions import *

# Basic operational data
mu_earthsun = load_basic_data()

# Data to be Imported
L1, L2, L4, L5 = import_LP()

# Linealized system gives the initial conditions
x = L5
y = -np.sqrt(3)/2 # L5 location
A = np.array(CR3BP_A(x,y,mu_earthsun)) # Function from aux_functions

vaps = linalg.eigvals(A)
veps = linalg.eig(A)[1]
vaps = clear_small(vaps,10**(-10))
veps = clear_small(veps,10**(-10))

omega_vaps = np.imag(vaps)
C = remove_i(veps)
invC = linalg.inv(C)
```

```python

check = clear_small(invC@A@C,10**(-10))

# Plots
# system_plot(mu_earthsun,L1,L2,L4,L5, xlim=[0.472,0.527], ylim=[-0.882,-0.85], zoom=1,
↪  showSun=False,  showEarth=False, showL1=False, showL2=False, showL4=False)
system_plot(mu_earthsun,L1,L2,L4,L5, xlim=[0.4998,0.50015], ylim=[-0.8661,-0.8659], zoom=3,
↪  showSun=False,  showEarth=False, showL1=False, showL2=False, showL4=False)

def objective(ds0,s0,mu, style='none', step=5):
    s_fin, t = shoot_to_poincare_x(mu, [s0[0],s0[1],ds0[0],ds0[1]], style=style, bstep=step)
    s_f = []
    s_f.append(s_fin[1]+s_fin[2])
    s_f.append(s_fin[1]+s_fin[3])
    return np.array(s_f)-np.array([s0[1]+ds0[0], s0[1]+ds0[1]])

print("")
print(">> L5 Planar Orbits <<")
print("")

fid1 = open('data/L5_PlanarOrbits.txt','w')
success_1 = 0
fail_1 = 0
success_2 = 0
fail_2 = 0
total = 0

tol = 10**(-8)
print('Long-period orbits')
can_continue = False
y0 = y
while not can_continue:
    y0 = y0 - 10**(-5)
    [t0_1, x0_1, dx_1, dy_1] = lin_eq_1(C, omega_vaps, 10**(-5), 10**(-5), y0)
    s0 = [float(x+x0_1),y0]
    root1 = optimize.root(objective, [dx_1,dy_1], args=(s0,mu_earthsun,'none',5),
    ↪  method='hybr', tol=10**(-10))
    s_fin, t_1 = shoot_to_poincare_x(mu_earthsun, [s0[0],s0[1],root1.x[0],root1.x[1]])
    if root1.success and
    ↪  (np.array([s0[0],s0[1],root1.x[0],root1.x[1]])-np.array(s_fin)<tol).all():
        #The initial orbit does incorporate some short-term movement so it is not plotted
        #print('Orbit at: [s0]; ds0; time')
        #print(s0)
        #print(str(root1.x[0]) + ' '+ str(root1.x[1]))
        #print(t_1)
        #shoot(mu_earthsun,[s0[0],s0[1],root1.x[0],root1.x[1]], t_1+10, style='-',res=3000)
        #fid1.write(str(s0[0]) + " " + str(s0[1]) + " " + str(root1.x[0]) + " " +
        ↪  str(root1.x[1]) + " " + str(t_1) + '\n')
        print('')
        can_continue = True

while y0 >= y-10*10**(-5):
    y0 = y0 - 10**(-5)
    s0[1] = y0
    ds0 = [root1.x[0], root1.x[1]]
    root1 = optimize.root(objective, ds0, args=(s0,mu_earthsun,'none',5), method='hybr',
    ↪  tol=10**(-10))
    s_fin, t_1 = shoot_to_poincare_x(mu_earthsun, [s0[0],s0[1],root1.x[0],root1.x[1]])
    if root1.success and
    ↪  (np.array([s0[0],s0[1],root1.x[0],root1.x[1]])-np.array(s_fin)<tol).all():
        print('Orbit at: [s0]; ds0; time')
        print(s0)
        print(str(root1.x[0]) + ' '+ str(root1.x[1]))
        print(t_1)
        shoot(mu_earthsun,[s0[0],s0[1],root1.x[0],root1.x[1]], t_1, style='-',res=3000)
        fid1.write(str(s0[0]) + " " + str(s0[1]) + " " + str(root1.x[0]) + " " +
        ↪  str(root1.x[1]) + " " + str(t_1) + '\n')
        print('')
fid1.close()

legend_object1 = Patch(facecolor='C0', edgecolor='black')
legend_object2 = Patch(facecolor='C1', edgecolor='black')
legend_object3 = Patch(facecolor='C2', edgecolor='black')
legend_object4 = Patch(facecolor='C3', edgecolor='black')
legend_object5 = Patch(facecolor='C4', edgecolor='black')
legend_object6 = Patch(facecolor='C5', edgecolor='black')
legend_object7 = Patch(facecolor='C6', edgecolor='black')
legend_object8 = Patch(facecolor='C7', edgecolor='black')

```

```
plt.legend(handles=[legend_object1, legend_object2, legend_object3, legend_object4,
↪  legend_object5, legend_object6,legend_object7,
↪  legend_object8],labels=['','','','','','','', 'Long-term planar orbits'],ncols=11,
↪  handletextpad=0.5, handlelength=1.0, columnspacing=-0.5,loc='upper right')


plt.show()
```

A.8 L1_Transfers.py

To find the desired transfers, first the STM/Monodromy matrix of a departure orbit around L1 is computed by propagating the necessary equations. From there, the relevant unstable direction is calculated to optimize the initial needed perturbation to get the closest approach to L4. That is saved to then optimize for the point in the trajectory where doing a mid-course change of velocity minimizes the total ΔV budget including the later needed orbital insertion into a periodic orbit around L4.

```python
# Transfers from L1
# Aitor Urruticoechea 2022
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Patch
from aux_functions import *
from scipy import linalg

# Basic operational data
mu_earthsun = load_basic_data()

# Data to be Imported
L1, L2, L4, L5 = import_LP()

# Plots
system_plot(mu_earthsun,L1,L2,L4,L5,xlim=[0.97,1.02],ylim=[-0.025, 0.025],zoom=1,
↪  showSun=False, showL4=False, showL5=False)
#system_plot(mu_earthsun,L1,L2,L4,L5,xlim=[0.4,1.1],ylim=[-0.1,1],zoom=0,showSun=False,showL5=False)

# Initial conditions: periodic orbit
L1_orbits = np.loadtxt('data/L1_LyapunovOrbits.txt')
L4_orbits = np.loadtxt('data/L4_PlanarOrbits.txt')
s0_L4 = np.delete(L4_orbits[-1,:],4,0) # largest
s0 = np.delete(L1_orbits[-1,:],4,0) # largest

# Propagation for an orbital period, gives the STM
s_end_orbit, t_orbit = shoot_to_poincare_y(mu_earthsun, s0, style='C0')
phi = variationals_prop(mu_earthsun, s0, t_orbit)

# Eigenvalues and eigenvectors of the STM gives the direction of instability
vaps = linalg.eigvals(phi)
#print(vaps) #(this is to check the values are the expected one - or close enough)
veps = linalg.eig(phi)[1]
found = False
n = -1
while not found:
    n = n+1
    if abs(vaps[n])>1.05: found = True
unstable_dir = veps[n,:] / np.linalg.norm(veps[n,:]) # Normalized unstable direction
if unstable_dir[0] > 0: unstable_dir = -unstable_dir
# Exploring that instability
```

```
maximum_y = [0,0,0,0]
s_transfer = [0,0,0,0]
h_transfer = 0
for i in range(1,500,5):
    h = i*10**(-7)
    s_new = s0 + h*unstable_dir
    s_end, t_end = shoot_to_poincare_x(mu_earthsun, s_new, style=None, crossings=1, x_obj=L4)
    shoot(mu_earthsun, s_new, t_end, style='C1',res=1000)
    if s_end[1] > maximum_y[1]:
        maximum_y = s_end
        s_transfer = s_new
        h_transfer = h

shoot_to_poincare_x(mu_earthsun, s_transfer, style='C5', bstep=0.1, crossings=1, x_obj=L4)
print("Manifold's closest approach to L4 at y: " + str(maximum_y[1]))
print("For an initial perturbation of h: " + str(h_transfer))
legend_object1 = Patch(facecolor='C0', edgecolor='black')
legend_object2 = Patch(facecolor='C1', edgecolor='black')
legend_object3 = Patch(facecolor='C5', edgecolor='black')
#plt.legend(handles=[legend_object2, legend_object3],
#           labels=['Trajectories within the L1-L4 Manifold','Optimized trajectory'],ncols=1,
↪   handletextpad=0.5, handlelength=1.0, columnspacing=-0.5,loc='lower left')

s_end_orbit, t_orbit = shoot_to_poincare_y(mu_earthsun, s0, style='C0')
plt.legend(handles=[legend_object1, legend_object2, legend_object3],
           labels=['Departure L1 Lyapunov Orbit', 'Trajectories within the L1-L4 Manifold',
           ↪   'Optimized trajectory'],ncols=1, handletextpad=0.5, handlelength=1.0,
           ↪   columnspacing=-0.5,loc='lower left')
plt.show()


#Plots again
system_plot(mu_earthsun,L1,L2,L4,L5,xlim=[0.4,1.1],ylim=[-0.1,1],zoom=0,showSun=False,showL5=False)
s_end_orbit, t_orbit = shoot_to_poincare_y(mu_earthsun, s0, style='C0')

#Objective
s_obj = [s0_L4[0], s0_L4[1]]
def objective(ds, s0, s_obj, mu, style='none', step=0.01):
    s_fin, t = shoot_to_poincare_x(mu, [s0[0], s0[1], ds[0], ds[1]], style=style, crossings=2,
    ↪   bstep=step, x_obj=s0_L4[0])
    s_f = [s_fin[0], s_fin[1]]
    return np.array(s_f) - np.array(s_obj)

def to_minimize(halfway_point, style_1='none',style_2='none',show=False, step=0.05):
    s_halfway, t_half = shoot_to_poincare_x(mu_earthsun, s_transfer, style=style_1,
    ↪  crossings=1, x_obj=(L4+halfway_point), bstep=step)
    s0_halfway = [s_halfway[0], s_halfway[1]]
    ds_halfway = [s_halfway[2], s_halfway[3]]
    if show: print('Burn 1 at t: ' + str(t_half))
    success = False
    ds_guess = ds_halfway
    count=0
    while not(success) and count<11: # Retry until solution converges
        count+=1
        root1 = optimize.root(objective, ds_guess, args=(s0_halfway, s_obj,
        ↪  mu_earthsun,'none',step),method='hybr', tol=10**(-10))
        success = root1.success
        ds_guess = root1.x
    s_fin, t_1 = shoot_to_poincare_y(mu_earthsun,
    ↪  [s0_halfway[0],s0_halfway[1],root1.x[0],root1.x[1]], style=style_2, crossings=1,
    ↪  bstep=step, y_obj=s_obj[1])
    #if show: print(s_fin)
    deltav1 = np.array(root1.x)-np.array(ds_halfway)
    if show: print('Frist delta-v (halfway point): ' + str(deltav1) + ' [dx,dy] (absolute: ' +
    ↪  str(np.linalg.norm(deltav1)) + ')')
    deltav2 = np.array([s_fin[2],s_fin[3]]) - np.array([s0_L4[2],s0_L4[3]])
    if show: print('Second delta-v (L4):           ' + str(deltav2) + ' [dx,dy] (absolute: ' +
    ↪  str(np.linalg.norm(deltav2)) + ')')
    if show: print('Total transfer time: ' + str(t_half+t_1))
    #if not(success): return np.linalg.norm(deltav1)+np.linalg.norm(deltav2) + 100 # If the
    ↪  solution has not converged, the function will return an exagerated response so the
    ↪  solver does not take it as a valid solution
    return np.linalg.norm(deltav1)+np.linalg.norm(deltav2)


L4plusX = optimize.minimize_scalar(to_minimize, 0.3,
↪  bounds=(0.1,0.4),method='bounded',args=('none','none',False,0.02),options={'maxiter':7})

print('Optimal burn point found at x: ' + str(L4+L4plusX.x))

totalDv = to_minimize(L4plusX.x,style_1='C1',style_2='C2',show=True)
shoot_to_poincare_x(mu_earthsun, s0_L4, bstep=30, style='black')
```

```python
print('For a total deltaV of: ' + str(totalDv))

legend_object11 = Patch(facecolor='black', edgecolor='black')
legend_object1 = Patch(facecolor='C0', edgecolor='black')
legend_object2 = Patch(facecolor='C1', edgecolor='black')
legend_object3 = Patch(facecolor='C2', edgecolor='black')
plt.legend(handles=[legend_object2, legend_object3, legend_object11],
           labels=['L1-L4 Manifold-contained path', 'Post-correction path', 'Final Planar L4
    ↪  Orbit'],ncols=1, handletextpad=0.5, handlelength=1.0,
    ↪  columnspacing=-0.5,loc='lower left')
plt.show()
```

A.9 L2_Transfers.py

In the same way of the transfers from L1 to L4, the closest approach to L5 is optimized for from the unstable direction departing from a Lyapunov orbit around L2 . Then, the required ΔV budget is also minimized regarding the position of the mid-course change of speed, accounting for the two changes of velocity needed (mid-course and orbital insertion).

```python
# Transfers from L2
# Aitor Urruticoechea 2022
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.patches import Patch
from aux_functions import *
from scipy import linalg

# Basic operational data
mu_earthsun = load_basic_data()

# Data to be Imported
L1, L2, L4, L5 = import_LP()

# Plots
system_plot(mu_earthsun,L1,L2,L4,L5,xlim=[0.97,1.02],ylim=[-0.025, 0.025],zoom=1,
↪  showSun=False, showL4=False, showL5=False)
#system_plot(mu_earthsun,L1,L2,L4,L5,xlim=[0.4,1.1],ylim=[-1,0.1],zoom=0,showSun=False,showL4=False)

# Initial conditions: periodic orbit
L2_orbits = np.loadtxt('data/L2_LyapunovOrbits.txt')
L5_orbits = np.loadtxt('data/L5_PlanarOrbits.txt')
s0_L5 = np.delete(L5_orbits[-1,:],4,0) # largest
s0 = np.delete(L2_orbits[-1,:],4,0) # largest

# Propagation for an orbital period, gives the STM
s_end_orbit, t_orbit = shoot_to_poincare_y(mu_earthsun, s0, style='C0')
phi = variationals_prop(mu_earthsun, s0, t_orbit)

# Eigenvalues and eigenvectors of the STM gives the direction of instability
vaps = linalg.eigvals(phi)
#print(vaps) #(this is to check the values are the expected one - or close enough)
veps = linalg.eig(phi)[1]
found = False
n = -1
while not found:
    n = n+1
    if abs(vaps[n])> 1.05: found = True
unstable_dir = veps[n,:] / np.linalg.norm(veps[n,:]) # Normalized unstable direction
if unstable_dir[0] < 0: unstable_dir = -unstable_dir
# Exploring that instability
```

```python
maximum_y = [-10,-10,-10,-10]
s_transfer = [0,0,0,0]
h_transfer = 0
for i in range(1,500,5):
    h = i*10**(-7)
    s_new = s0 + h*unstable_dir
    s_end, t_end = shoot_to_poincare_x(mu_earthsun, s_new, style=None, crossings=1, x_obj=L5)
    shoot(mu_earthsun, s_new, t_end, style='C1',res=1000)
    if s_end[1] > maximum_y[1]:
        maximum_y = s_end
        s_transfer = s_new
        h_transfer = h

shoot_to_poincare_x(mu_earthsun, s_transfer, style='C5', bstep=0.1, crossings=1, x_obj=L4)
print("Manifold's closest approach to L5 at y: " + str(maximum_y[1]))
print("For an initial perturbation of h: " + str(h_transfer))
legend_object1 = Patch(facecolor='C0', edgecolor='black')
legend_object2 = Patch(facecolor='C1', edgecolor='black')
legend_object3 = Patch(facecolor='C5', edgecolor='black')
#plt.legend(handles=[legend_object2, legend_object3],
#           labels=['Trajectories within the L2-L5 Manifold', 'Optimized trajectory'],ncols=1,
↪  handletextpad=0.5, handlelength=1.0, columnspacing=-0.5,loc='lower right')
s_end_orbit, t_orbit = shoot_to_poincare_y(mu_earthsun, s0, style='C0')
plt.legend(handles=[legend_object1, legend_object2, legend_object3],
           labels=['Departure L2 Lyapunov Orbit', 'Trajectories within the L2-L5 Manifold',
           ↪  'Optimized trajectory'],ncols=1, handletextpad=0.5, handlelength=1.0,
           ↪  columnspacing=-0.5,loc='lower left')

plt.show()



#Plots again
system_plot(mu_earthsun,L1,L2,L4,L5,xlim=[0.4,1.1],ylim=[-1,0.1],zoom=0,showSun=False,showL4=False)
s_end_orbit, t_orbit = shoot_to_poincare_y(mu_earthsun, s0, style='C0')

s_obj = [s0_L5[0], s0_L5[1]]
def objective(ds, s0, s_obj, mu, style='none', step=0.01):
    s_fin, t = shoot_to_poincare_y(mu, [s0[0], s0[1], ds[0], ds[1]], style=style, crossings=2,
    ↪  bstep=step, y_obj=s_obj[1])
    s_f = [s_fin[0], s_fin[1]]
    return np.array(s_f) - np.array(s_obj)
```

```

def to_minimize(halfway_point, style_1='none',style_2='none',show=False, step=0.05):
    s_halfway, t_half = shoot_to_poincare_x(mu_earthsun, s_transfer, style=style_1, crossings=1, x_obj=(L5+halfway_point), bstep=step)
    s0_halfway = [s_halfway[0], s_halfway[1]]
    ds_halfway = [s_halfway[2], s_halfway[3]]
    if show: print('Burn 1 at t: ' + str(t_half))
    success = False
    ds_guess = ds_halfway
    count=0
    while not(success) and count<11: # Retry until solution converges
        count+=1
        root1 = optimize.root(objective, ds_guess, args=(s0_halfway, s_obj, mu_earthsun,'none',step),method='hybr', tol=10**(-10))
        success = root1.success
        ds_guess = root1.x
    s_fin, t_1 = shoot_to_poincare_y(mu_earthsun, [s0_halfway[0],s0_halfway[1],root1.x[0],root1.x[1]], style=style_2, crossings=2, bstep=step, y_obj=s_obj[1])
    #if show: print(s_fin)
    deltav1 = np.array(root1.x)-np.array(ds_halfway)
    if show: print('Frist delta-v (halfway point): ' + str(deltav1) + ' [dx,dy] (absolute: ' + str(np.linalg.norm(deltav1)) + ')')
    deltav2 = np.array([s_fin[2],s_fin[3]]) - np.array([s0_L5[2],s0_L5[3]])
    if show: print('Second delta-v (L5):            ' + str(deltav2) + ' [dx,dy] (absolute: ' + str(np.linalg.norm(deltav2)) + ')')
    if show: print('Total transfer time: ' + str(t_half+t_1))
    if not(success): return np.linalg.norm(deltav1)+np.linalg.norm(deltav2) + 100 # If the solution has not converged, the function will return an exagerated response so the solver does not take it as a valid solution
    return np.linalg.norm(deltav1)+np.linalg.norm(deltav2)


L5plusX = optimize.minimize_scalar(to_minimize, 0.2, bounds=(0.1,0.4),method='bounded',args=('none','none',False,0.02),options={'maxiter':7})

print('Optimal burn point found at x: ' + str(L5+L5plusX.x))

totalDv = to_minimize(L5plusX.x,style_1='C1',style_2='C2',show=True)
shoot_to_poincare_x(mu_earthsun, s0_L5, bstep=30, style='black')
print('For a total deltaV of: ' + str(totalDv))
```

```

legend_object11 = Patch(facecolor='black', edgecolor='black')
legend_object1 = Patch(facecolor='C0', edgecolor='black')
legend_object2 = Patch(facecolor='C1', edgecolor='black')
legend_object3 = Patch(facecolor='C2', edgecolor='black')
plt.legend(handles=[legend_object2, legend_object3, legend_object11],
           labels=['L2-L5 Manifold path', 'Post-correction path', 'Final Planar L5
           ↪  Orbit'],ncols=1, handletextpad=0.5, handlelength=1.0,
           ↪  columnspacing=-0.5,loc='lower right')
plt.show()
```

Appendix B

Budget

In this appendix, a detailed approach to the required budget for the realization of this thesis is presented. This includes the required academic fees for the evaluation of this final degree thesis [34], the required work hours by the author and by both director and co-director; as well as other needs like suitable hardware for the execution of the developed code [35], and an approximation for the utilities-related expenses of working that amount of hours from home. Interestingly, other necessary expenses for the realization of this work result in zero values as a direct result of choices made during the past half-year. These include using an open-source and free-to-use programming language like Python and its related packages (such as NumPy or SciPy) [19, 20, 22]; as well as attending the free CR3BP course by Auburn University in collaboration with NASA-JPL "Designing the moonshot" [11]. All detailed expenses can be consulted in Table B.1 and Figure B.1.

Expense	Unit price	Amount	Subtotal
Academic fees [34]	18,46 €	12	221,52 €
Work hours (undergraduate)	12,00 €	500	6.000,00 €
Work hours (PhD)	30,00 €	25	750,00 €
"Designing the moonshot" course fees [11]	0,00 €	1	0,00 €
Hardware [35]	1.019,00 €	1	1.019,00 €
Software Licence (Python) [19]	0,00 €	1	0,00 €
Utilities*	100,00 €	7	700,00 €
**monthly price has been approximated*		**TOTAL**	**8.690,52 €**

Table B.1: Budget disclosure for this thesis.

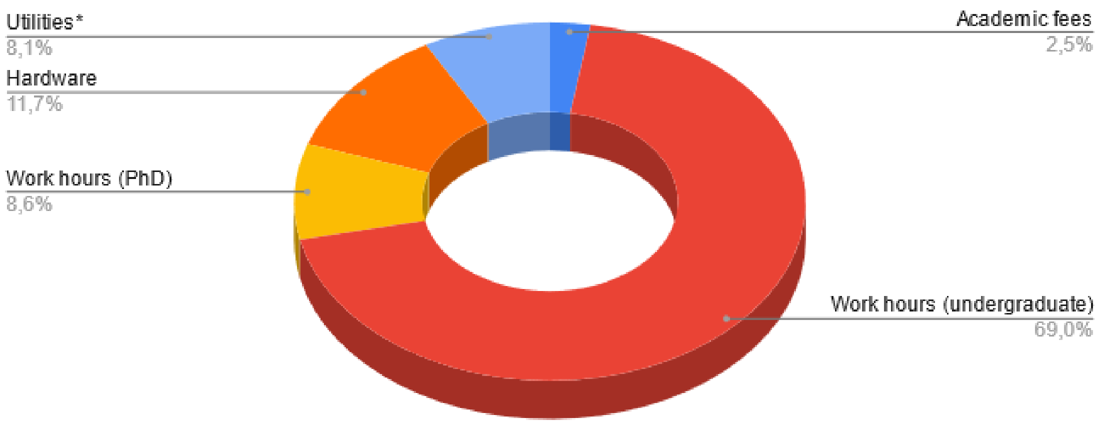


Figure B.1: Pie chart with the proportional budget expenses for this thesis.