UNIVERSITAT POLITÈCNICA DE VALÈNCIA


Bachelor's Degree in Aerospace Engineering

---

# Use of the genetic algorithm for the multi-objective optimisation of direct impulsive trajectories between the Earth and Mars

## Final Bachelor's Degree Project

---

**AUTHOR:** Bastida Pertegaz, Esther
**TUTORS:** Moll López, Santiago
Tatay Sangüesa, José

Universitat Politècnica de València
Escuela Técnica Superior de Ingeniería del Diseño

July, 2021

# Contents

# Abstract

This project employs the multi-objective genetic algorithm to optimize direct impulsive trajectories between the Earth and Mars.

In order to do so, a two-body problem modelled with impulsive manoeuvres will be assumed. Also, two and three-impulse trajectories will be assessed for each mission duration, selecting the one that minimises $\Delta v$. As for solving the transfer problem, Lambert arcs will be employed. The Lambert problem aims to determine an orbit from two position vectors and the Time Of Flight (TOF). The main objective of this choice of method is to reduce the number of parameters as much as possible, and therefore the computational cost. Also, an in-depth study will be carried out to select the most suitable genetic algorithm settings.

From a series of input parameters that include the launch date and the time of flight, the algorithm will optimise the manoeuvre in terms of minimum $\Delta v$ budget. After that, the program will render all necessary information about the transfer orbit and the trajectory will be plotted, so that the solution can be graphically seen. Moreover, for validating the goodness of the results, they will be compared to an ideal, two-impulse Hohmann transfer, and to real missions to Mars.

Finally, for this method to be able to target more than one objective at the same time, that is, $\Delta v$ and the time of flight, a Pareto frontier will be built. This plot offers a representation of each mission duration and its corresponding $\Delta v$, allowing the engineer to visually find a compromise between both variables, depending on the mission specifications.

**Keywords:** genetic algorithm, multi-objective optimisation, orbit transfer, $\Delta v$, time of flight, Lambert's problem.

# Resumen

Este proyecto emplea el algoritmo genético para realizar una optimización multiobjetivo de trayectorias impulsivas directas entre la Tierra y Marte.

Para lograr este propósito, se asumirá un problema de los dos cuerpos modelado con maniobras impulsivas. Además, para cada tiempo de vuelo, se calcularán las transferencias de dos y tres impulsos que minimicen $\Delta v$. Estas órbitas serán obtenidas empleando el problema de Lambert, el cual permite calcular una trayectoria partiendo de dos vectores posición. El objetivo de aplicar este método es reducir al mínimo el número de parámetros empleados, disminuyendo así también el coste computacional. Adicionalmente, se realizará un estudio exhaustivo con el fin de definir el valor más adecuado para los parámetros del algoritmo genético.

Partiendo de un conjunto de variables iniciales, entre las que se encuentran la fecha de salida y el tiempo de vuelo, el algoritmo optimizará la maniobra en términos de $\Delta v$ mínimo. Así, el programa proporcionará la información necesaria acerca de la órbita de transferencia, y se representará esta trayectoria con el fin de ofrecer una visión gráfica de la solución obtenida. Además, con el fin de validar los resultados, se compararán con la maniobra ideal bi-impulsiva de Hohmann, así como con misiones reales a Marte.

Finalmente, para que este método permita optimizar más de un parámetro al mismo tiempo, es decir, $\Delta v$ y el tiempo de vuelo, se elaborará un frente de Pareto. Este gráfico permite representar cada tiempo de vuelo con su correspondiente $\Delta v$, lo cual hace posible al ingeniero encontrar visualmente un compromiso entre ambas variables, dependiendo de las especificaciones de la misión.

**Palabras clave:** algoritmo genético, optimización multiobjetivo, transferencia orbital, $\Delta v$, tiempo de vuelo, problema de Lambert.

# Resum

Este projecte utilitza l'algoritme genètic per a realitzar una optimització multiobjectiu de trajectòries impulsives directes entre la Terra i Mart.

Amb aquest propòsit, s'assumirà un problema dels dos cossos amb maniobres impulsives. A més, per a cada temps de vol, es calcularan les transferències de dos i tres impulsos que minimitzen $\Delta v$. Estes òrbites seran obtingudes emprant el problema de Lambert, el qual permet calcular trajectòries partint de dos vectors posició. L'objectiu d'aplicar este mètode és disminuir al mínim el nombre de paràmetres empleats, reduint així també el cost computacional. Addicionalment, es realitzarà un estudi exhaustiu a fi de definir el valor més adequat per als paràmetres de l'algoritme genètic.

Partint d'una sèrie de variables inicials, entre les que es troben la data d'eixida i el temps de vol, l'algoritme optimitzarà la maniobra en termes de $\Delta v$ mínim. Així, el programa proporcionarà la informació necessària sobre l'òrbita de transferència, i es representarà la trajectòria a fi d'oferir una visió gràfica de la solució obtinguda. A més, a fi de validar els resultats, es compararan amb la maniobra ideal bi-impulsiva d'Hohmann, i amb missions reals a Mart.

Finalment, perquè este mètode permeta optimitzar més d'un paràmetre al mateix temps, és a dir, $\Delta v$ i el temps de vol, s'elaborarà un front de Pareto. Este gràfic permet representar cada temps de vol amb el seu corresponent $\Delta v$, la qual cosa fa possible a l'enginyer trobar visualment un compromís entre ambdós variables, depenent de les especificacions de la missió.

**Paraules clau:** algoritme genètic, optimització multiobjectiu, transferència orbital, $\Delta v$, temps de vol, problema de Lambert.

# Acknowledgements

I would like to give a special thanks to my project supervisors, Santiago Emmanuel Moll López and Pepe Tatay Sangüesa, for furthering my love for space and guiding me throughout this whole project.

Also, thanks to the Universitat Politècnica de València, for being the institution that has made my studies and this thesis possible.

My sincere gratitude to my family, for always supporting my goals and helping me become who I am. Also, to my friends Joan, Jorge, María, Javi and Antonio, who have accompanied me along this four-year journey.

Valencia, July 2021

Esther Bastida Pertegaz

# List of Figures

# List of Tables

# List of Symbols

$\Delta v$ Velocity change, measure of the total propellant

$\gamma$ Flight path angle

$\mathbf{T}$ Orbital period

$\mu$ Orbital constant

$\Omega$ Right Ascension of the Ascending Node (RAAN)

$\omega$ Argument of periapsis

$\overrightarrow{F_D}$ Drag force

$\overrightarrow{F_G}$ Gravitational force

$\overrightarrow{R}$ Position vector from the origin to a point

$\overrightarrow{r}$ Position vector

$\overrightarrow{v}$ Velocity vector

$\phi$ Phase angle

$\theta$ True anomaly

$\varepsilon$ Total energy of the orbit

$a$ Semi-major axis

$b$ Elevation in spherical coordinates

$b_i$ Number of bits allocated to a variable

$C_P$ Power cost

$e$ Eccentricity

$G$ Gravitational constant = $6.67408 \cdot 10^{-11} \mathrm{m}^3 \mathrm{kg}^{-1} \mathrm{s}^{-1}$

$g_0$ Gravitational acceleration at sea level

$G_i$ Constraint normalized to the unit

$g_i$ Constraint

$h$ Specific angular momentum

| | |
|---|---|
| $i$ | Inclination |
| $I_{sp}$ | Specific impulse |
| $l$ | Azimuth in spherical coordinates |
| $m$ | Mass |
| $m_{Sun}$ | Sun's mass |
| $m_s$ | Spacecraft's mass |
| $n$ | Angular velocity |
| $N_{bits}$ | Total number of bits |
| $p$ | Semi-latus rectum |
| $P_c$ | Power consumption |
| $P_m$ | Mutation probability |
| $P_s$ | Population size |
| $r$ | Distance |
| $r_a$ | Radius of the apoapsis |
| $R_i$ | Resolution of a variable |
| $r_p$ | Radius of the periapsis |
| $R_{x,y,z}$ | Rotation matrices |
| $t$ | Time instant |
| $T_{syn}$ | Synodic period |
| $v$ | Velocity |
| $w$ | Effective velocity |
| $x_i^L$ | Lower bound |
| $x_i^U$ | Upper bound |

# List of Abbreviations

**BSA** Bit String Affinity.

**PC** Personal Computer.

**RAAN** Right Ascension of the Ascending Node.

**SE** Solar Ecliptic Coordinate System.

**TOF** Time Of Flight.

# 1 Introduction

## 1.1 Motivation

Space exploration has become an increasingly relevant field during the last few years. From telecommunications to purely investigative reasons, the scope covered by the objective of space missions is astonishingly wide. Particularly, the interest in sending probes to Mars grows everyday, especially as its colonization could be possible in the future [1].

However, sending a spacecraft past the Earth's atmosphere costs companies and space agencies a great amount of money each year. Therefore, it makes sense that optimising all possible parameters is of the utmost importance to the space engineering field.

As the optimisation discipline can be applied to a wide range of operations, it is interesting to follow a spacecraft's life along its mission in order to identify possible improvement areas. After the launch phase, which takes the satellite from the Earth's surface to a desired orbit, the following processes occur:

1. **Orbit transfer**
   It is the process by which the spacecraft leaves its orbit and manoeuvres until reaching a target orbit.
2. **Orbit maintenance**
   Perturbations can cause a satellite to progressively abandon its orbit. Orbit maintenance is the process of making corrections so that the spacecraft stays in its desired path.
3. **Attitude control**
   Consists in orienting the spacecraft a certain way according to its mission objectives. Nowadays, this operation is carried out by means of additional actuators and generally does not require the use of propellant.
4. **De-orbiting**
   Once a satellite's life ends, it cannot continue orbiting, as it would become space debris. This is avoided by the spacecraft exiting its current orbit or being destroyed.

From these operations, orbit transfers consume up to a 70% of the total $\Delta v$ of a mission [2]. Therefore, optimising these manoeuvres would undoubtedly benefit space travel. This would allow, for example, to make the satellite smaller, thus reducing the total cost of the mission. Or the remaining $\Delta v$ could be employed in orbit maintenance, extending the spacecraft's life.

However, propellant use is not the only parameter that is worth taking into account. Multi-objective algorithms are able to target several variables at the same time, which would allow simultaneously optimising another important parameter: the time of flight.

Because there are an infinite number of possible trajectories between two states, the use of algorithms is generally the solution. This is why, in this work, a multi-objective genetic algorithm will be employed, so that an optimal transfer between the Earth and Mars can be successfully found.

# 2 Theoretical background

Throughout this section, two different points will be addressed. The first one will be devoted to the theoretical background related to orbital mechanics that has been needed for developing the project. In the second one, the working principle of the genetic algorithm and its specifications regarding this work will be explained.

## 2.1 Orbital mechanics

It is known that all celestial bodies move through orbits across space, affected by the spheres of influence of other celestial objects. The aim of this section is to introduce the formula that describes this orbital motion.

Once the motion of the Sun, planets and spacecraft are known, it is important to define the orbits that they follow, which represent their trajectories. Knowing these trajectories will allow finding a solution for our problem and calculating all necessary parameters, such as the fuel consumption or the time of flight.

### 2.1.1 Heliocentric ecliptic coordinates

Designating a reference system will be the first step for successfully describing the orbital motion of the spacecraft. The **Solar Ecliptic Coordinate System (SE)** has been chosen as the best option, as it allows expressing orbital parameters in a simple and approachable way. This reference system is broadly used for interplanetary missions in which the spacecraft escapes the spheres of influence of the planets, which is the case of this project.

The origin of this coordinate system is located at the centre of the Sun, and its fundamental plane will be the ecliptic, which is the plane of the Earth's orbit around the Sun. Its primary direction points towards the vernal equinox, which determines the OX axis in the Cartesian reference frame. The OZ axis, which is perpendicular to the ecliptic, and the OY axis are taken following the right-hand convention. Figure 1 shows the outline of this reference system.

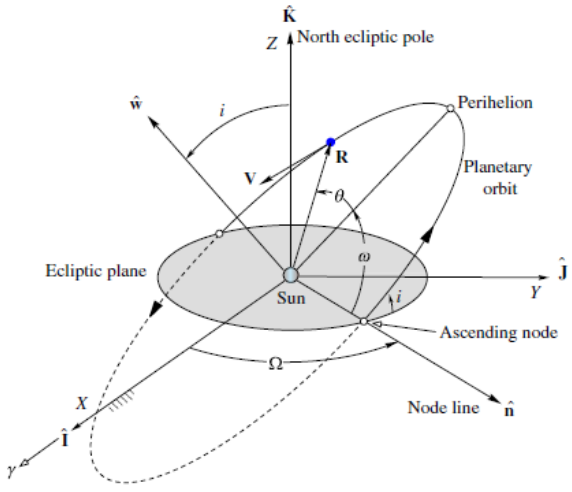



Figure 1: Solar Ecliptic Coordinate System [3]

It is interesting to mention that this coordinate system allows expressing the position of the planets and spacecraft with respect to the Sun both in rectangular and spherical form. In this work, the spherical form has been very useful for reasons that will be further explained in subsequent sections.

### 2.1.2 Simplifying hypotheses

Before defining the equations that will govern the orbital motion of the spacecraft, it is important to make certain simplifying assumptions. This way, the solution can be reached in a simpler and faster manner, as the computational cost is significantly reduced. It is important to mention that these assumptions have been selected in such a way that they do not compromise the reliability of the solutions to a high extent. They are the following:

1. The SE, which has been described in the previous section, will be considered inertial.
2. The two-body problem will be considered at all times. That is, only the Sun and the spacecraft will be taken into account for calculations. However, the Earth and Mars will play an essential role for defining the spacecraft launching and landing positions.
3. The spacecraft mass can be neglected if compared to that of the Sun. This relationship between both masses is indicated by expression (1).

$$m_s << m_{Sun} \tag{1}$$

4. The spacecraft mass does not change, which can be expressed by means of equation (2).

$$\Delta m_s = 0 \tag{2}$$

5. The Sun can be considered a perfectly spherical body with uniform density. This allows labeling it as a point mass, and therefore the Newton's law of gravitation (3) applies.

$$\overrightarrow{F_G} = G\frac{m_1 m_2}{r^3}\overrightarrow{r} \tag{3}$$

6. Manoeuvres will be considered impulsive, that is, they are carried out by means of an instantaneous change in the spacecraft velocity while its position remains the same. This allows neglecting the thrust force term when solving the equations of motion. As it will be commented in further sections, this assumption is able to accurately model reality provided that some conditions apply.
7. The spacecraft will be located at all times at a height over the atmosphere of the planets so that the drag force can be considered null, as indicated by equation (4).

$$F_D \approx 0 \tag{4}$$

8. The only force acting on the spacecraft will be the gravity force $\overrightarrow{F_G}$. Therefore, other forces caused by, for example, solar radiation or electromagnetic fields are not taken into account.

All these assumptions allow for a simplification of the equation of motion, which will be developed in the following section.

### 2.1.3 The orbital equation

In order to obtain the equation that governs the spacecraft motion, the two-body problem needs to be solved. These two bodies will be the Sun and the spacecraft travelling from the Earth to Mars. For doing this, a few steps need to be followed [4]. First, from equation (3) and using the second Newton law, the equation of motion of each of the bodies can be obtained, as it is shown by expressions (5) and (6).

$$\ddot{\overrightarrow{R_1}} = G\frac{m_2}{r^3}\overrightarrow{r} \tag{5}$$

$$\ddot{\overrightarrow{R_2}} = -G\frac{m_1}{r^3}\overrightarrow{r} \tag{6}$$

Figure 2 shows bodies 1 and 2 in an inertial reference system. For the present problem, these two bodies would be the Sun and the spacecraft, and the inertial frame of reference would be the SE, as it has been previously explained. It is important to mention that, in this work, the Sun will be located exactly at the origin of the inertial coordinate system.

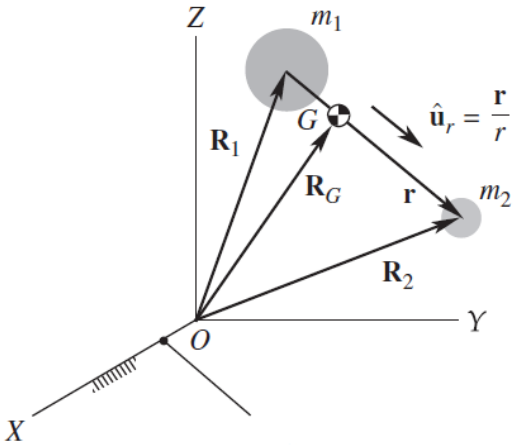



Figure 2: Motion in an inertial coordinate system [3]

The next step will be to find the equation that governs the relative motion between both bodies. The starting point is expression (7), which renders the position vector of body $m_2$ (the spacecraft) with respect to $m_1$ (the Sun).

$$\overrightarrow{r} = \overrightarrow{R_2} - \overrightarrow{R_1} \tag{7}$$

Deriving this equation twice with respect to time and substituting expressions (5) and (6), equation (8) is obtained.

$$\ddot{\overrightarrow{r}} = -\frac{\mu}{r^3}\overrightarrow{r} \tag{8}$$

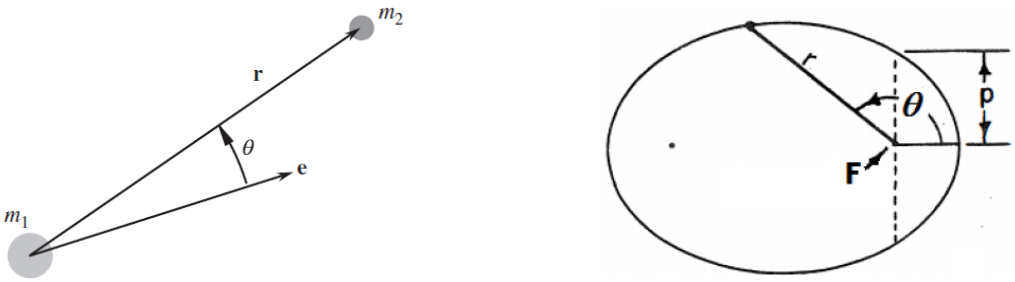
with:

$$\mu = G(m_1 + m_2) \tag{9}$$

The solution of equation (8) can be found in several references such as [5], and it is known as the **equation of orbital motion**:

$$r = \frac{p}{1 + e\cos\theta} \tag{10}$$

The parameters involved in equation (10) can be seen in a graphical way in Figures 3a and 3b.




(a) (b)

Figure 3: Orbital equation parameters [3], [4]

The variable $\theta$ denotes the true anomaly, and it represents the counterclockwise angle between the apse line in the direction of the periapsis and the spacecraft position vector $\overrightarrow{r}$.

Equation (10) is also known as the conic equation, as it describes one of the four possible conic sections, depending on the value of the eccentricity $e$. These conic sections are the circle ($e = 0$), the ellipse ($0 < e < 1$), the parabola ($e = 1$) and the hyperbola ($e > 1$), and they can be seen in Figure 4.

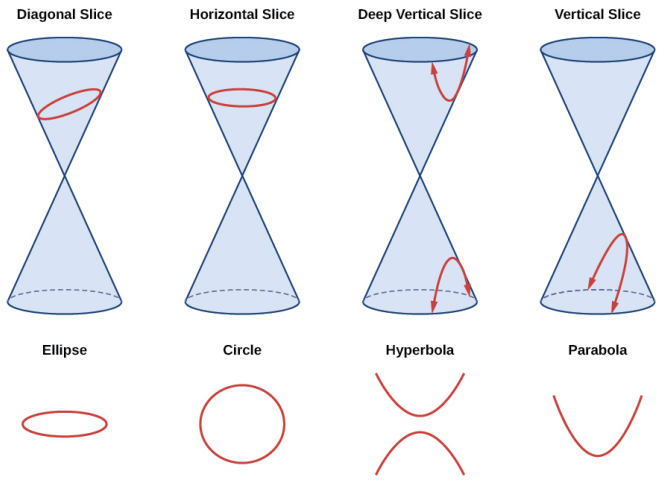



Figure 4: Conic sections [6]

### 2.1.4 Three-dimensional orbits

Once the motion of the spacecraft with respect to the Sun has been expressed by means of equation (10), the next step will be to characterize the trajectory it will follow. This path is

called an orbit, and it can be described by using different sets of parameters.

For example, a spacecraft's position in space (and therefore its orbit) can be completely defined by stating its three position and velocity vector components. This can be done, for instance, in the SE $(r_x, r_y, r_z, v_x, v_y, v_z)$, which has been previously explained and will be the main reference system employed in this project. For performing certain calculations, it could also be interesting to express the satellite's trajectory in the perifocal frame $(r_p, r_q, r_w, v_p, v_q, v_w)$, which is the orbital reference frame. Its origin is located at the primary focus of the orbit, its fundamental plane is the orbit's one and its principal direction is pointing to the periapsis. A graphical description of the perifocal frame can be observed in Figures 5a and 5b.

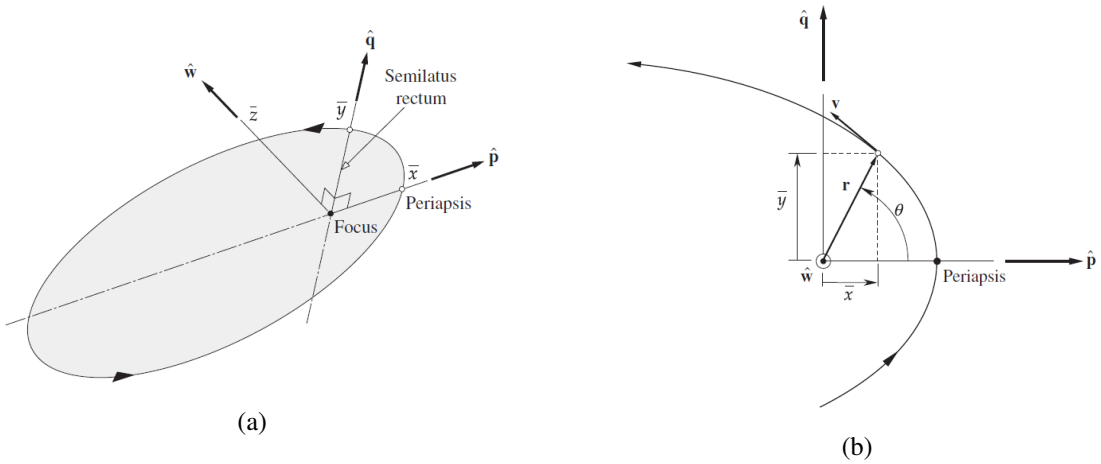



Figure 5: Perifocal frame of reference [3]

Both the heliocentric ecliptic and the perifocal frames are useful in terms of simplicity of the calculations. However, they fail to offer a clear view of the orbit's shape, size and orientation in space. To overcome this issue, it is helpful to define the orbit by means of a set of parameters known as **Keplerian orbital elements**. Before introducing these parameters, it is necessary to define the *node line*, which is the intersection between the orbital plane and the fundamental one (in our case, the ecliptic). The point on this node line where the spacecraft would pass above the ecliptic from below is called the *ascending node*. The node line vector $\overrightarrow{n}$ points outward from the centre of the Sun through the ascending node. The opposite end of the node line, where the satellite would start travelling below the ecliptic, is called the *descending node*. These parameters can be observed in Figure 6. Once these terms have been clarified, the six Keplerian orbital elements are defined as follows:

- **a: semi-major axis**
  It represents the length of half the major axis of the conic.
- **e: eccentricity**
  This parameter describes the shape of the orbit. It indicates the type of conic being represented, as it has already been explained. It is defined as the quotient between the focal distance and the semi-major axis. It can also be defined as the norm of the eccentricity vector $\overrightarrow{e}$, which goes from the focus to the periapsis of the orbit and is inside the orbital plane. It is one of the constants of the Keplerian orbit, since it does not change without external perturbations.

- **i: inclination**
  It is the angle between the ecliptic and the orbital plane, measured counterclockwise from the node line. Its value can go from $0°$ to $180°$.

- **$\Omega$: Right Ascension of the Ascending Node**
  It is the angle formed between the principal direction, which points to the vernal equinox, and the node line vector $\overrightarrow{n}$. This angle can take values from $0°$ to $360°$.

- **$\omega$: argument of periapsis**
  It is the angle formed by vectors $\overrightarrow{n}$ and $\overrightarrow{e}$, measured over the orbital plane in the direction followed by the spacecraft. Its value can vary from $0°$ to $360°$.

- **$\theta$: true anomaly**
  As it has been previously explained, it is the angle comprised between vectors $\overrightarrow{e}$ and $\overrightarrow{r}$ in the direction of the satellite's motion. It takes values from $0°$ to $360°$.

It is important to mention that angles $i$, $\Omega$ and $\omega$ are called the **orbital Euler angles**, and they are essential for locating a three-dimensional orbit in space. The semi-major axis and eccentricity allow defining the shape of the orbit, and $\theta$ indicates the position of the spacecraft along the trajectory. All these parameters can be seen in Figure 6.

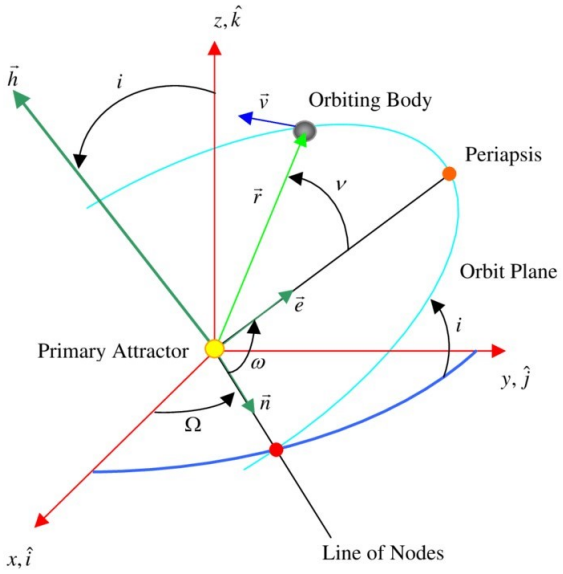



Figure 6: Keplerian orbital elements [7]

Apart from the Keplerian elements, there are other important parameters that can characterize an orbit. They are indicated below, as well as the expressions that allow calculating them.

- **$r_p$: periapsis radius**
  It is the distance from the principal focus to the orbit's closest point to the centre of attraction, which, in this case, will be the Sun.

$$r_p = a(1 - e) \tag{11}$$

- **$r_a$: apoapsis radius**
  It is the distance from the principal focus to the orbit's farthest point to the centre of

attraction.

$$r_a = a(1+e) \tag{12}$$

- **p: semi-latus rectum**
  It is the distance from the focus to the spacecraft when $\theta = 90°$. It can be observed in Figure 3b.

$$p = a(1-e^2) \tag{13}$$

- **$\varepsilon$: mechanical energy**
  Total energy of the orbit, that is, sum of the potential and kinetic energies. This is one of the keplerian orbit constants, since it is conserved along an orbit.

$$\varepsilon = -\frac{\mu}{2a} \tag{14}$$

- **h: specific angular momentum modulus**
  It is the modulus of the vector perpendicular to both the spacecraft's position and velocity vectors, which can be seen in Figure 7. It is constant throughout the orbit, which after some math [3], [8], validates Kepler's second law: "The radius vector drawn from the sun to the planet sweeps out equal areas in equal intervals of time".

$$h = |\overrightarrow{r} \times \overrightarrow{v}| \tag{15}$$

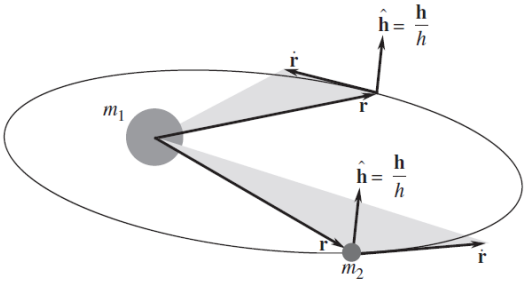



Figure 7: Angular momentum vector [3]

- **T: orbital period**
  It is the time it takes the spacecraft to travel along a full orbit.

$$T = 2\pi\sqrt{\frac{a^3}{\mu}} \tag{16}$$

Finally, the expressions for the spacecraft's velocity (17) and flight path angle (18) are also relevant. This last magnitude is the angle formed between the velocity vector and its tangential component, and can be seen in Figure 8.

$$v = \sqrt{\mu\left(\frac{2}{r} - \frac{1}{a}\right)} \tag{17}$$

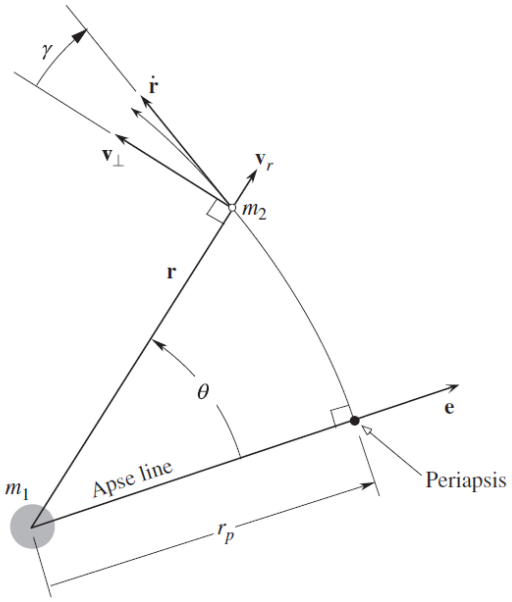



Figure 8: Flight path angle [3]

$$\gamma = \arccos \frac{h}{rv} \tag{18}$$

Once all these parameters have been introduced, it is important to know how to change between the heliocentric ecliptic coordinates, perifocal coordinates and Keplerian elements. For example, knowing the true anomaly $\theta$ allows calculating the perifocal coordinates as shown by expressions (19) and (20).

$$\overrightarrow{r}_{perifocal} = \begin{Bmatrix} r_p \\ r_q \\ r_w \end{Bmatrix} = \begin{Bmatrix} r\cos\theta \\ r\sin\theta \\ 0 \end{Bmatrix} \tag{19}$$

$$\overrightarrow{v}_{perifocal} = \begin{Bmatrix} \dot{r}_p \\ \dot{r}_q \\ \dot{r}_w \end{Bmatrix} = \begin{Bmatrix} -\frac{\mu}{h}\sin\theta \\ \frac{\mu}{h}(e+\cos\theta) \\ 0 \end{Bmatrix} \tag{20}$$

Now, changing from perifocal coordinates to heliocentric ecliptic ones requires the 3-1-3 rotation indicated by equation (21).

$$\begin{Bmatrix} r_x \\ r_y \\ r_z \end{Bmatrix} = R_z(-\Omega)\cdot R_x(-i)\cdot R_z(-\omega)\cdot \begin{Bmatrix} r_p \\ r_q \\ r_w \end{Bmatrix} \tag{21}$$

where $R_x$, $R_y$ and $R_z$ denote the rotation matrices with respect to the OX, OY and OZ axes, respectively.

### 2.1.5 Orbital manoeuvres

For a spacecraft to arrive to a desired target orbit, it is necessary that it undergoes an orbital manoeuvring process. Said process can be performed either by applying impulsive or continuous thrust. As it has been stated in section 2.1.2, **impulsive manoeuvres** will be assumed in this work. When applying this idealized model, results are satisfactory if the position of the spacecraft changes only slightly during the impulse. This is satisfied for high-thrust rockets which burn times are short compared with the orbital period of the spacecraft [3]. Therefore, if this is achieved, considering impulsive thrust offers reliable results while simplifying the calculations to a high extent, and reducing significantly the computational cost.

As it has been mentioned before, impulsive manoeuvres mean an instantaneous change in the velocity magnitude of the spacecraft without altering its position, as expression (22) portrays.

$$\overrightarrow{r}(t_0^+) = \overrightarrow{r}(t_0^-), \quad \overrightarrow{v}(t_0^+) = \overrightarrow{v}(t_0^-) + \Delta\overrightarrow{v} \tag{22}$$

Regarding the number of impulses that the satellite needs to perform, it has to be taken into account that any orbit transfer between two orbits that do not intersect requires at least two impulses. Also, it has been proved that no more than four burns are required for any optimal impulsive manoeuvre, as stated by reference [5]. As a consequence, two and three-impulse manoeuvres will be employed in this work in order to find orbital transfers between the Earth and Mars. Four impulse burns will not be assessed, as references such as [9] and [10] disregard them as a better option than two or three-impulse manoeuvres for similar case studies.

Regarding the manoeuvre itself in terms of computations, it is interesting to mention that the initial and final orbits have been expressed in Keplerian elements, while the transfer orbit or orbits are represented in rectangular coordinates for convenience. This evidences once again the importance of changing from one reference frame to another, as it has been stated before.

### 2.1.6 Orbit transfer optimisation

The orbit transfer optimisation problem aims to calculate the best transfer trajectory between an initial and a target orbit in terms of minimising some quantities. These quantities are generally the $\Delta v$ budget, which is an indicator of the propellant used, and the time of flight. The expression that relates $\Delta v$ with the propellant consumption is called the Tsiolkowsky rocket equation (23).

$$\Delta v = w \ln \frac{m_{initial}}{m_{final}} \tag{23}$$

$$w = I_{sp} \cdot g_0 \tag{24}$$

being:

$w$: effective velocity at which the rocket propellants are expelled
$m$: mass of the spacecraft
$I_{sp}$: specific impulse
$g_0$: gravitational acceleration at sea level

This optimisation problem only has an analytical solution for very specific cases, which means that it usually needs to be solved by employing algorithms such as the one used in the present work. One of the most well-known cases for which an analytical solution exists is the **Hohmann transfer** between co-planar, circular orbits. It consists in a **two-impulse** manoeuvre that results in an elliptical transfer orbit which is tangent to the initial and target orbits on their apse line, as shown in Figure 9.

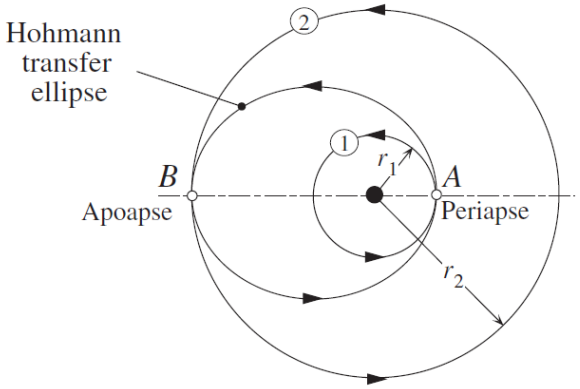



Figure 9: Hohmann orbit transfer [3]

The $\Delta v$ budget for this mission can be calculated by means of expression (27).

$$\Delta v_1 = v_2 - v_1 = \frac{h_H}{r_1} - \frac{h_1}{r_1} \tag{25}$$

$$\Delta v_2 = v_3 - v_2 = \frac{h_2}{r_2} - \frac{h_H}{r_2} \tag{26}$$

$$\Delta v_T = |\Delta v_1| + |\Delta v_2| = \sqrt{\frac{\mu}{r_1}}\left(\sqrt{\frac{1}{r_2/r_1}} - \sqrt{\frac{2}{(r_2/r_1)[1+(r_2/r_1)]}} + \sqrt{\frac{2(r_2/r_1)}{1+(r_2/r_1)}} - 1\right) \tag{27}$$

where the subscript $1$ refers to the initial orbit, $2$ to the final orbit, and $H$ to the Hohmann transfer orbit.

As for the transfer time, it could be calculated as half the orbital period of the transfer orbit (28).

$$TOF = \frac{T_H}{2} = \pi\sqrt{\frac{a_H^3}{\mu}} \tag{28}$$

Another case with an optimal analytical solution would be the **bi-elliptic Hohmann transfer**, which performs a **three-impulse** manoeuvre between co-planar, circular orbits. A scheme of this transfer can be seen in Figure 10.

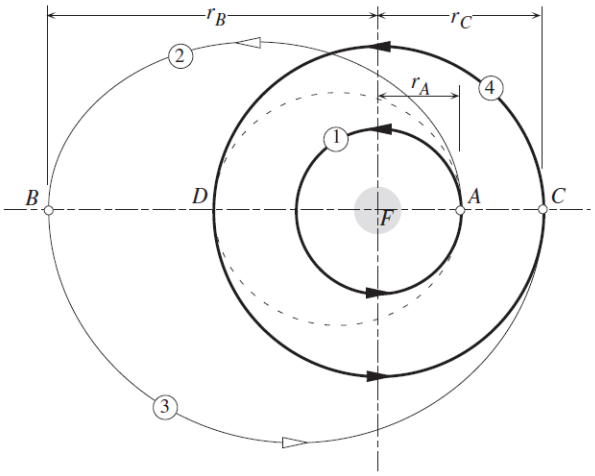



Figure 10: Bi-elliptic Hohmann orbit transfer [3]

As Figure 10 illustrates, a bi-elliptic Hohmann transfer from an inner orbit to an outer one would start at point A. After a semi-ellipse has been travelled by the satellite, a second impulse takes place at point B. The spacecraft moves then along a second elliptical orbit, and finally a third impulse occurs at point C in order to remain in the target orbit. It is important to mention that points A, B and C at which these three impulses occur are all located in the orbits' apse line.

Calculating the $\Delta v$ budget for this case can be done in a simple way by applying equations (29) to (32).

$$\Delta v_A = \sqrt{\frac{\mu}{r_A}}\left(\sqrt{\frac{2r_B}{r_A + r_B}} - 1\right) \tag{29}$$

$$\Delta v_B = \sqrt{\frac{\mu}{r_B}}\left(\sqrt{\frac{2r_C}{r_B + r_C}} - \sqrt{\frac{2r_A}{r_A + r_B}}\right) \tag{30}$$

$$\Delta v_C = \sqrt{\frac{\mu}{r_C}}\left(\sqrt{\frac{2r_B}{r_B + r_C}} - 1\right) \tag{31}$$

$$\Delta v_T = |\Delta v_A| + |\Delta v_B| + |\Delta v_C| \tag{32}$$

It is important to mention that $\Delta v_B$ will be smaller as $r_B$ increases. In fact, when $r_B$ tends to infinity, $\Delta v_B$ tends to 0 [3]. However, it needs to be taken into account that the time of flight is a limiting factor for the value of $r_B$.

Moreover, for the case of co-planar, circular orbits, the Hohmann transfer is more efficient than the bi-elliptic transfer when $r_C/r_A$ is less than 11.94 [3]. If this ratio is higher than 15.58, then the bi-elliptic transfer is more efficient. In between these two values, the optimal transfer depends on the ratio $r_B/r_A$ as seen in Figure 11.

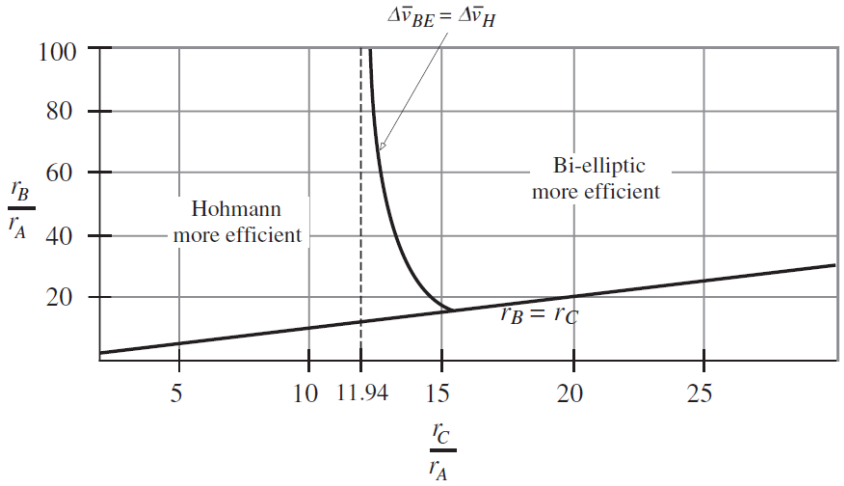



Figure 11: Optimal transfer between co-planar, circular orbits [3]

### 2.1.7 Synodic period

In order to correctly solve the optimisation problem for trajectories between the Earth and Mars, it is important to be familiar with the concept of synodic period. This value is the time it takes two objects to repeat a certain relative angular position between them.

This can be seen by looking at Figure 12, where $\phi$ is called the phase angle and can be calculated by means of expression (33), where $n$ represents the angular velocity of the planets. Stating that $\phi_0$ is the phase angle at a time $t = 0$, the synodic period is the time it will take the phase angle to become $\phi_0$ again. From equation (33), a short mathematical development [3] will lead to the final expression for calculating the synodic period (34).

$$\phi = \theta_2 - \theta_1 = \phi_0 + (n_2 - n_1)t \tag{33}$$

$$T_{syn} = \frac{2\pi}{|n_1 - n_2|} = \frac{T_1 T_2}{|T_1 - T_2|} \tag{34}$$

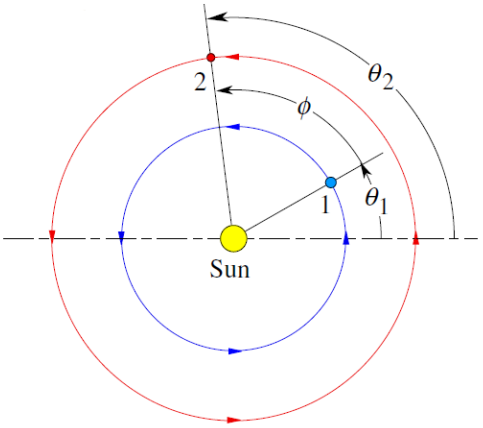



Figure 12: Planets in circular orbits around the Sun [3]

Equation (34) allows calculating the synodic period between the Earth and Mars, which will be employed for solving the orbital transfer optimisation problem.

$$\begin{aligned} T_{Earth} &= 365.26 \text{ days } (1 \text{ year}) \\ T_{Mars} &= 1 \text{ year } 321.73 \text{ days} = 687.99 \text{ days} \\ T_{syn} &= 777.9 \text{ days} \end{aligned}$$

## 2.2 The genetic algorithm

As the real optimisation problem for orbital trajectories between the Earth and Mars does not have an analytical solution, an algorithm needs to be used for its approach. In this work, the genetic algorithm will be employed, which is a powerful global optimisation method. This algorithm is a model of biological evolution based on Charles Darwin's theory of natural selection. It searches for the best possible variable combination by finding the "fittest" individuals within a certain population, and generating better solutions by reproduction amongst them. Genetic algorithms offer numerous advantages over traditional algorithms, such as their ability for solving complex problems and the parallelism they offer for manipulating different parameters at the same time [11].

However, care needs to be taken with genetic algorithms when selecting important parameters such as the initial population size, the mutation probability or the selection criteria for the new population. The choice of these parameters will strongly influence the level of accuracy of the results, and therefore it is essential to be thorough during the process.

As it has been previously mentioned, the genetic algorithm is a global search optimisation method, which is also called a zero-order method. This is due to the fact that no additional information about the objective function is needed.

Now, the working process of the algorithm will be explained. Generally, the following steps are carried out:

1. Encoding the solutions
   The algorithm will search for the optimal parameter combination inside a predefined closed space. This space is obtained by encoding the variables using binary strings. The accuracy of the search will depend on the number of bits selected, so a compromise needs to be found between the variable resolution and the computational cost. A string formed by $n$ bits that encodes a variable is called a *chromosome*, which is composed of *genes*. All chromosomes form a *population*, which changes every new *generation* in order to produce fitter solutions.

2. Defining a fitness function or selection criterion
   This consists in setting a function which value allows recognizing the fittest individuals.

3. Creating an initial population of individuals
   This population is **randomly** generated, and its size is selected by the programmer. Increasing the population size will improve the chances of finding an optimal solution, but it also means a higher computational cost.

4. Performing the evolution cycle
   This is done by evaluating the fitness of all individuals inside the population, and then carrying out the three following processes:

   - Selection
     All individuals are paired and a tournament process takes place. On the one hand, the best individuals according to the fitness function previously described survive in order to be parents for the next generation. On the other hand, the worst individual from each pair is discarded. Other selection procedures include ranking or the roulette wheel [12].

   - Crossover
     It represents the reproduction of individuals so that each pair generates two children, therefore the total population size remains the same. In this work, uniform crossover will be employed due to its effectiveness and efficiency according to sources such as [13]. However, different methods like the single-point crossover exist. In uniform crossover, the first child receives each of its bits from one of the parents with equal probability. The second child's bits will be the ones not selected for the first one. Figure 13 illustrates this method.

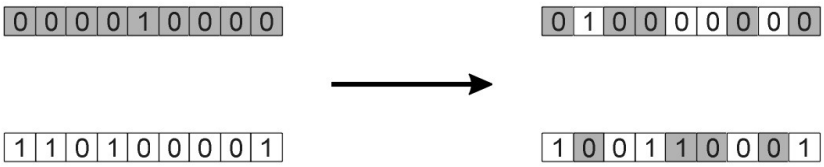



Figure 13: Uniform crossover example [14]

   - Mutation
     This operation is carried out by flipping randomly selected bits as shown in Figure 14. The assigned probability to this procedure is very low, and it can occur simultaneously at multiple chromosome locations. The objective of this step is to search in the neighbourhood of the current individual.

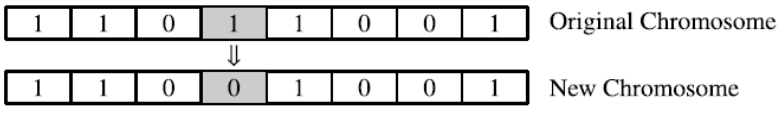



Figure 14: Mutation operation [11]

Once these three operations have been carried out, a new population arises and the process starts again until the stopping criteria is satisfied. For selecting these criteria, several approaches could be followed. Even though the algorithm allows setting a maximum number of generations, the programmer could choose to end the process when the fittest individual has not changed for several iterations. Another option is to stop iterating when the population is concentrated in a sufficiently small portion of the solution space, that is, chromosomes are very similar among them. For this project, this last method has been

chosen, as recommended by literature [15]. For doing so, a Bit String Affinity (BSA) value needs to be provided to the algorithm, as it will be explained in further sections.

5. Decoding the solution in order to obtain the optimal orbit transfer
   The last step will consist in decoding the obtained result, which is expressed in binary notation, to decimal notation.

### 2.2.1 Multi-objective optimisation

A multi-objective optimisation algorithm is one that simultaneously optimizes more than one parameter. Usually, there is not a single design solution that is optimal for every objective function, but rather this solution is formed by a set of designs. These designs are called non-dominated points, and improving one objective will always mean degrading another. In order to simplify the process of finding a trade-off between all objective functions, a Pareto frontier can be built. It consists in a graph formed by all non-dominated design points, which allows graphically finding a compromise between different objective function values. Figure 15 shows a generic example of a Pareto frontier plot.

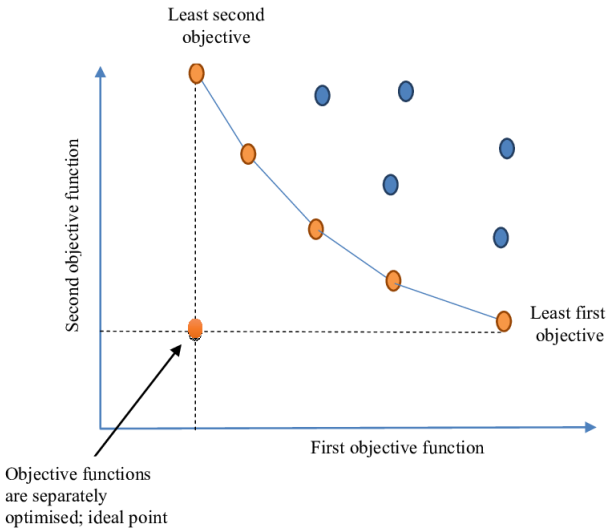



Figure 15: Pareto frontier example [16]

There are different ways of computationally solving a multi-objective optimisation problem. Two of the most widely used techniques are the $\varepsilon$-constraint and the weighting methods. In this work, the **$\varepsilon$-constraint approach** will be used, as it offers many advantages with respect to the weighting one [17], and has allowed obtaining effective results in numerous works such as [17, 18].

This method optimises one of the objectives (in our case, the $\Delta v$ budget), whilst the others (in the present work, the time of flight) are seen as constraints. This is achieved by adding inequality constraints to the model and establishing certain limits ($\varepsilon_i$) for these objectives. By changing said limits, new solutions are obtained and the Pareto frontier can be built.

# 3 Methodology

## 3.1 Orbital transfer problem

In this section, the approach followed for finding the optimal orbit transfer between the Earth and Mars will be explained. The employed software will be MATLAB, and it will be used for finding the orbital trajectories that minimise both $\Delta v$ and the time of flight. First of all, the initial orbit will be defined by its Keplerian elements ($a_0$, $e_0$, $i_0$, $\Omega_0$, $\omega_0$), and the same will be done for the final orbit ($a_f$, $e_f$, $i_f$, $\Omega_f$, $\omega_f$). The $\varepsilon$-constraint approach will be used and, as it has been explained in the previous section, $\Delta v$ is the main objective, whilst the time of flight will be the constrained one. Additionally, a constraint has been added to the program so that the satellite is always at least at 1 AU from the Sun.

The next step is to compute how the program will look for possible optimal solutions. The main objective of this work is to reduce the computational cost as much as possible, and therefore the number of variables needs to be minimum. For this reason, **Lambert's problem** will be employed for solving the different possible trajectories, as the efficiency of this method has been widely proved [19–21]. This approach allows to find a transfer arc when the initial and final positions, as well as the time of flight, are known. This is done by means of an iterative method, and it guarantees that the initial and final positions of the spacecraft will be correct, as they are an input. In order to implement this procedure, a robust and reliable algorithm found in literature will be used [22].

The next step will be to apply Lambert's problem to two and three-impulse manoeuvres. As it has already been stated, several references have not found four-impulse solutions to improve two and three-impulse ones [9,10], so four-impulse trajectories will be disregarded in this work.

For two-impulse trajectories, the problem has been solved by introducing the launch date and the time of flight as input variables. This way, the number of parameters that the genetic algorithm needs to solve has been reduced to the minimum. From these two variables, it is possible to obtain the initial and final true anomalies ($\theta_0$ and $\theta_f$, respectively), and use them to solve the single Lambert arc. These two angles are calculated by knowing the Earth and Mars' true anomalies at a reference chosen instant, J2000. For defining this concept, first the term *julian date* needs to be clarified. The julian date of a certain instant is the number of days that have passed since 1 January 4713 BC 12:00. J2000 is the instant corresponding to the julian date 2451545.0, and it is a widely used reference date. The true anomalies of both planets at this instant can be easily obtained from the *Horizons NASA web interface* [23]. Now, suppose that $\theta_{J2000}$ is the true anomaly of a planet at the instant J2000. Then, the true anomaly of that same planet some time later will be:

$$\theta = \theta_{J2000} + t_{J2000} \cdot n \tag{35}$$

being:

- $t_{J2000}$: time from J2000 to the date at which the true anomaly wants to be calculated
- $n$: angular velocity of the planet

As for three-impulse trajectories, two transfer orbits will be needed. Therefore, Lambert's problem needs to be solved twice. In this case, the input parameters (that is, the ones that need

to be searched by the genetic algorithm), will be the launch date, the times of flight for both arcs, and the spherical coordinates of the second impulse in the SE. These spherical coordinates are the modulus of the distance from the Sun ($r$) and two angles ($l$ and $b$). The use of spherical coordinates instead of rectangular ones is due to the fact that less bits need to be used for encoding the former, and therefore the algorithm will be faster. Figure 16 shows a scheme of these spherical coordinates, where $\phi$ would be $l$, and $\pi/2 - \theta$ would be $b$.

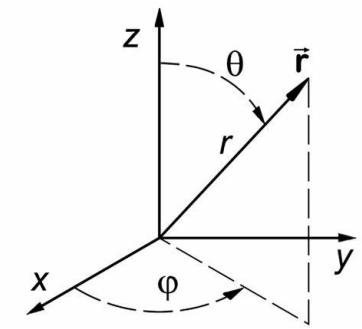



Figure 16: Spherical coordinates [24]

Once the launch date and the time of flight are known, it is easy to translate them into initial and final true anomalies, as it was done for the two-impulse case. Using these data, if the spherical coordinates of the intermediate impulse are translated into rectangular ones, it is possible to solve Lambert's problem twice and calculate both transfer arcs.

The following flow diagram shows the step-by-step solving of the orbit transfer problem for two and three-impulse trajectories. The red part of the chart only corresponds to three-impulse manoeuvres.

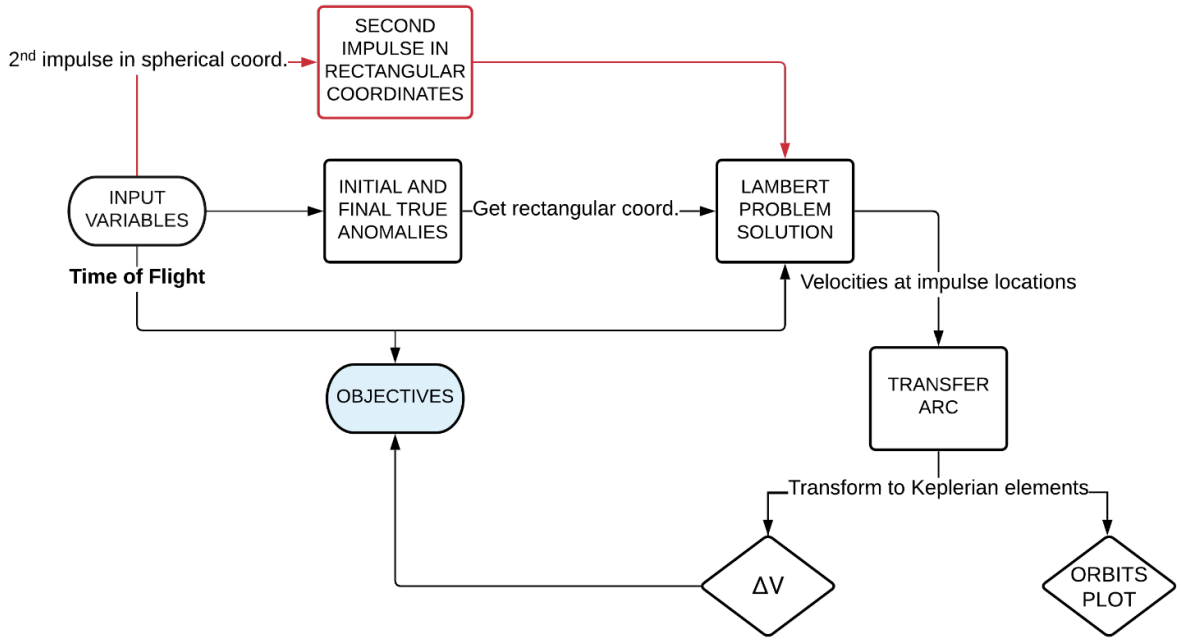



Figure 17: Orbit transfer calculation

## 3.2 Genetic algorithm settings

As it has been previously commented, correctly defining the genetic algorithm parameters will be key in obtaining accurate and fast results. In this work, the raw code for the genetic algorithm was obtained from literature [25], and an exhaustive analysis for properly defining its parameters was then carried out.

It should be taken into account that the genetic algorithm is not a calculus-based method but a searcher, as it iteratively calculates different solutions until the optimal one is found. This decision depends on the **stopping criteria** which, as it has already been mentioned, is based on the BSA value. This value has been selected so that the program stops the process when individual chromosomes present over a 90% of coincidence. Moreover, a maximum number of generations has been introduced as insurance, and its value is 200.

As for **encoding the solution space**, it has been stated that using more bits would increase the accuracy but also the computational cost. Therefore, a compromise between the two needs to be achieved. Since variables are coded in binary, the number of bits allocated to each variable can be calculated as:

$$b_i = \left\lceil \log_2 \left( \frac{x_i^U - x_i^L}{R_i} + 1 \right) \right\rceil \tag{36}$$

where:

$x_i^U$: upper bound
$x_i^L$: lower bound
$R_i$: resolution of variable $x_i$
$\lceil - \rceil$: *ceil* operator, which rounds the result to the nearest integer higher or equal than the element

Rewriting expression (36), the resolution can be isolated as follows:

$$R_i = \frac{x_i^U - x_i^L}{2^{b_i} - 1} \tag{37}$$

As for the bounds associated to each variable, they can be seen in Table 1, as well as the number of bits and resolution. The results shown in this table refer to an $\varepsilon$-constraint value of 250 days, so as to offer an example of possible values.

It is interesting to comment the upper bound for parameter $r$, which is the modulus of the distance to the second impulse in the case of three-impulse transfers. In order to establish this value, it has been considered that the farthest the spacecraft could travel would be along a very eccentric orbit (close to a straight line), which period was the TOF $\varepsilon$-constraint value. Therefore, the maximum distance from the Earth would be twice the semi-major axis of this orbit:

$$TOF_{max} = \varepsilon = T = 2\pi \sqrt{\frac{a^3}{\mu_S}} \rightarrow r = 2(1 + 10\%) \cdot a = 2.2 \cdot (4\pi^2 \varepsilon^2 \mu_S)^{1/3} \tag{38}$$

Additionally, a 10% margin has been added to this value for ensuring a sufficiently large solution space for $r$. Also, in order to calculate the maximum distance from the Sun instead of the Earth, 1 AU should be added to the result of equation (38).

| Variable | Bounds | Bits | Resolution |
|---|---|---|---|
| Launch date | 2458850 days $\leq \mathrm{d}_0 \leq$ 2458850+778 days | 15 | 1 h |
| Time of Flight | 0 h $\leq$ TOF $\leq \varepsilon$-constraint h | 13 | 1 h |
| r | 1 AU $\leq \mathrm{r} \leq 3.1078 \cdot 10^9$ km | 25 | 100 km |
| l | $0° \leq \mathrm{l} \leq 180°$ | 8 | $0.7059°$ |
| b | $0° \leq \mathrm{b} \leq 360°$ | 9 | $0.7045°$ |

Table 1: Genetic Algorithm variable encoding

As it can be seen in Table 1, the launch date bounds are expressed in **Julian days**, as they are easier to operate with. The lower bound corresponds to the 1 January 2020, so that a comparison with the latest launches to Mars could be made. Also, the interval defined by these bounds is 778 days. As it has been previously calculated, the synodic time between the Earth and Mars is 777.9 days. Therefore, in order to search for the optimal trajectory for all relative positions between both planets, the range of the launch date was set to be this synodic time.

Regarding the **population size**, it was set to 4 times the total number of bits, as recommended by literature [25]. It is important to remind that the initial population is randomly generated by the algorithm. As for the **mutation probability**, it can be calculated by means of the following expression:

$$P_m = \frac{N_{bits} + 1}{2 \cdot P_s \cdot N_{bits}} = \frac{N_{bits} + 1}{8 N_{bits}^2} \tag{39}$$

where $P_s$ is the population size.

Now, on the topic of how does the program handle constraints on any variable, they need to be introduced by means of **penalty functions**. This way, if a constraint is not met, a quantity proportional to the constraint violation is added by the program to $\Delta v$. Therefore, the result can no longer be optimal and the algorithm discards these solutions. In order to successfully implement this method, the penalty needs to be of the same order of magnitude than the originally calculated $\Delta v$, so that the result when violating a constraint is sufficiently increased.

The following steps need to be carried out in order to implement this approach. To begin with, all constraints have to be normalised to the unit as shown by equation (40). Then, the penalty has to be calculated by achieving that the constraint has the same order of magnitude as the objective function result. This can be done by multiplying the normalised constraint by a certain function, as shown by expression (41). In order to accomplish its purpose, this function needs to include a $log_{10}$ expression, as well as the *ceil* operator $\lceil - \rceil$. As the penalty function will only activate when the constraint is positive, which means there is a violation, and the $log_{10}$ operator includes an absolute value, the penalty will always be a positive quantity.

$$g_i(x) \leq c_i \rightarrow G_i = \frac{g_i(x)}{c_i} - 1 \leq 0 \tag{40}$$

$$\mathrm{Penalty_i} = 10^{\lceil \log_{10} |f| \rceil + 1} \cdot G_i \tag{41}$$

Finally, as it has been said, the initial population is randomly generated, and the final results will depend on this random selection. In order to ensure that a real optimal solution has been found, several iterations have been run for each case. This way, 25 iterations were selected for the two-impulse case, and 15 for the three-impulse one, so as not to increase the computational cost excessively. Also, as an additional verification, the whole code was run at least 5 times for each case.

It is important to highlight that, even though the final algorithm settings are the ones shown in this section, several trials were run before achieving satisfactory results. Therefore, an in-depth study has been carried out regarding the genetic algorithm specifically applied to the present case study. This is precisely what confers a research character to this project.

# 4 Presentation of results

In this section, the results obtained by following the previous methodology will be laid out. To begin with, a comparison with the two-impulse Hohmann transfer will be carried out in order to verify the goodness of the solutions. After that, the values for the optimal $\Delta v$ will be assessed for different mission durations. Finally, in order to accomplish a multi-objective optimization, a Pareto frontier including all results will be built. This way, engineering judgement can be applied depending on the mission objectives so as to decide the optimal $\Delta v$ and TOF altogether.

First, the Keplerian parameters of the initial and final orbits will be shown. They characterize the Earth and Mars' orbits, respectively, and are summed up in Table 2.

| **Initial Orbit** | | **Final Orbit** | |
|---|---|---|---|
| **Parameter** | **Value** | **Parameter** | **Value** |
| $a_0\ [km]$ | 149 562 903 | $a_f\ [km]$ | 227 938 631 |
| $e_0\ [-]$ | 0.0165 | $e_f\ [-]$ | 0.0935 |
| $i_0\ [^\circ]$ | 0.0180 | $i_f\ [^\circ]$ | 1.8494 |
| $\Omega_0\ [^\circ]$ | 264.8051 | $\Omega_f\ [^\circ]$ | 49.5409 |
| $\omega_0\ [^\circ]$ | 199.4599 | $\omega_f\ [^\circ]$ | 286.5163 |

Table 2: Initial and final orbit parameters

## 4.1 Comparison with Hohmann transfer

The first step for comparing the obtained solutions with the Hohmann transfer will be to apply expressions (25) to (28) in order to calculate the Hohmann orbital manoeuvre. Results are presented in Table 3.

| Parameter | Value |
|---|---|
| $a_t\ [km]$ | 188 792 513 |
| $e_t\ [-]$ | 0.2076 |
| $i_t\ [^\circ]$ | 0 |
| $\Omega_t\ [^\circ]$ | – |
| $\omega_t\ [^\circ]$ | – |
| $\theta_f - \theta_0\ [^\circ]$ | 180 |
| $\Delta v\ [km/s]$ | 5.5960 |
| $TOF\ [days]$ | 258.9152 |

Table 3: Hohmann transfer orbit parameters

It is important to point out that a Hohmann manoeuvre takes place between **co-planar, circular orbits** by definition. Therefore, the inclination $i_t$ of the transfer orbit will be zero. As a consequence, the ascending node does not exist and $\Omega_t$ and $\omega_t$ cannot be defined. Also, the calculated values are almost identical to those obtained from different references [26, 27], thus they can be considered valid.

As the obtained transfer time for the Hohmann manoeuvre has been 258.9 days, in order to compare these results with the ones provided by the optimiser, an $\varepsilon$-constraint value of 259 days is introduced to the algorithm. Table 4 shows the optimal transfer arc in terms of $\Delta v$ for two-impulse manoeuvres.

| Parameter | Value | Error |
|---|---|---|
| $a_t\ [km]$ | 194 651 242 | 3.1032% |
| $e_t\ [-]$ | 0.2192 | 5.5855% |
| $i_t\ [^\circ]$ | 1.4140 | $[-]$ |
| $\theta_f - \theta_0\ [^\circ]$ | 147.1025 | 18.2764% |
| $\Delta v\ [km/s]$ | 6.0356 | 7.8560% |
| $TOF\ [days]$ | 202.2104 | 21.9024% |

Table 4: Two-impulse genetic algorithm results for $\varepsilon = 259$ days

A graphical representation of this trajectory is shown in Figure 18. From the table above, it can be deduced that a mission duration of approximately 202 days has been selected by the algorithm as the one that renders the minimum $\Delta v$. This means that, for the real case of elliptic orbits in different planes, the Hohmann time of flight of 259 days is not the optimal one. The errors on the parameters of the transfer orbit are due to this reduction in the optimal time of

flight, as well as to the inclination and eccentricity of the initial and final orbits, which are not 0 as for the Hohmann transfer case.

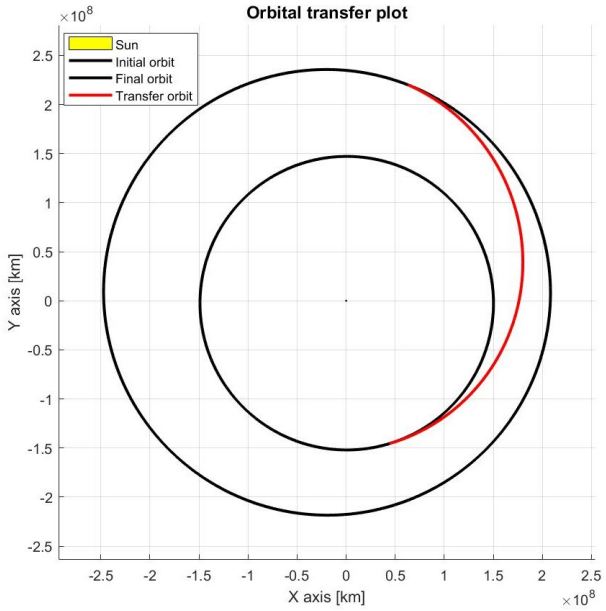



Figure 18: Two-impulse trajectory for $\varepsilon = 259$ days

As for the three-impulse trajectory, the obtained data for the transfer arcs has been collected in Table 5.

| | **Transfer arc 1** | **Transfer arc 2** | | |
|---|---|---|---|---|
| | **Value** | **Value** | | |
| $a_t\ [km]$ | 186 201 208 | 203 841 632 | | |
| $e_t\ [-]$ | 0.1836 | 0.1936 | **Total** | |
| $i_t\ [\,^\circ\,]$ | 0.3633 | 1.5612 | **Value** | **Error** |
| $\theta_f - \theta_0\ [\,^\circ\,]$ | 103.7626 | 77.0493 | 180.8119 | 0.4510% |
| $\Delta v\ [km/s]$ | $[-]$ | $[-]$ | 5.7710 | 3.1266% |
| $TOF\ [days]$ | 117.3421 | 141.2469 | 258.5889 | 0.1279% |

Table 5: Three-impulse genetic algorithm results for $\varepsilon = 259$ days

For the three-impulse orbital transfer, it can be seen how the genetic algorithm has selected a mission duration of 258.59 days as the one that minimizes the $\Delta v$ consumption. This value is very close to the Hohmann transfer TOF, being the relative error of only 0.1279%. Moreover, the errors for the phase angle $(\theta_f - \theta_0)$ and for $\Delta v$ are also fairly low.

It is helpful to take a look at the orbital transfer plot, which is represented by Figure 19.

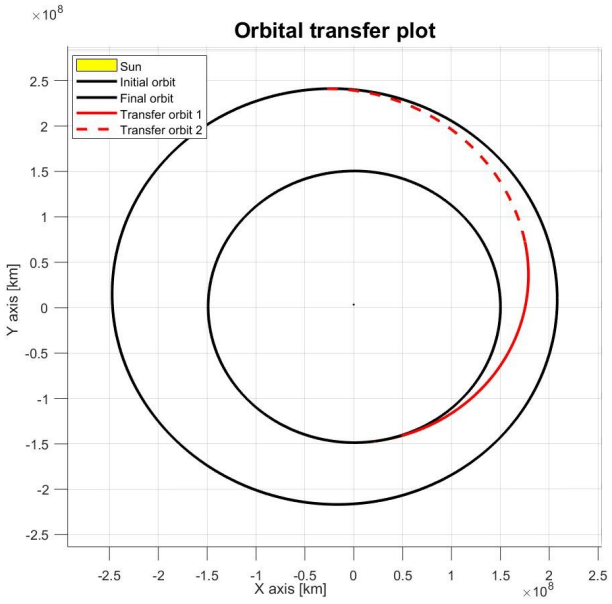



Figure 19: Three-impulse trajectory for $\varepsilon = 259$ days

However, the Hohmann transfer is a two-impulse one, while we are analyzing a three-impulse manoeuvre. From this statement, an important deduction can be made. The two-impulse Hohmann transfer is calculated between circular, co-planar orbits, as it has already been stated. Nevertheless, the real Earth and Mars orbits are neither circular nor co-planar. Therefore, in order to correct the small eccentricities and inclinations of both orbits, the real optimal trajectory for a TOF of approximately 259 days would not be a two-impulse, but a three-impulse one. In this trajectory, both transfer arcs present slight inclinations and eccentricities very similar to one another, so that they highly resemble a single transfer arc, as it can be seen in Figure 19. The relatively small errors that appear between the ideal, two-impulse Hohmann transfer and the real, three-impulse one, are precisely due to the eccentricity and different planes of the orbits. It makes sense that the variable presenting a higher error, while still being a low value, is $\Delta v$, as plane changes are a very influencing factor for its calculation.

In conclusion, as the obtained results in comparison with an ideal Hohmann transfer can be considered satisfactory, the validity of the genetic algorithm parameter settings has been verified.

## 4.2 Optimal transfer orbits

In this section, the results for a wide range of $\varepsilon$-constraint values will be presented in order to show the capabilities of the optimiser. TOF boundaries have been selected so that a broad scope of mission durations can be assessed, from short periods to large ones. After carrying out the process described in section 3, the trajectories that minimise $\Delta v$ are the ones shown in Table 6.

| $\mathbf{TOF_{limit}}$ **[days]** | **Impulses** | **Launch date** | **TOF [days]** | $\boldsymbol{\Delta v}$ **[km/s]** |
|---|---|---|---|---|
| 100 | 2 | 12 August 2020 23:17 | 100 | 12.4172 |
| 150 | 2 | 24 July 2020 4:36 | 150 | 7.5979 |
| 205 | 2 | 11 July 2020 11:10 | 202.2104 | 6.0356 |
| 215 | 2 | 11 July 2020 11:10 | 202.2104 | 6.0356 |
| 220 | 3 | 30 June 2020 0:30 | 219.8926 | 5.9574 |
| 225 | 3 | 30 June 2020 5:38 | 225 | 5.9103 |
| 250 | 3 | 29 June 2020 21:05 | 249.5422 | 5.7905 |
| 300 | 3 | 6 July 2020 18:54 | 292.4551 | 5.7450 |
| 350 | 3 | 9 July 2020 13:00 | 310.3491 | 5.7350 |
| 400 | 3 | 17 July 2020 14:45 | 383.8613 | 5.7261 |

Table 6: Minimum $\Delta v$ manoeuvres for different $\varepsilon$-constraint values

For each $\varepsilon$-constraint value, which is the limit TOF, the genetic algorithm calculates the two and three-impulse optimal manoeuvres, and the assembled code selects the one that minimizes $\Delta v$ among the two of them. Also, it was decided that a TOF of 400 days would be the highest value studied, as $\Delta v$ decreases at a very low rate from 250 days on, which does not compensate the increase in the mission duration.

Now, in order to offer a clearer view of these results, a Pareto frontier will be built from Table 6. This will also allow performing a **multi-objective optimisation**, as it helps the engineer find a compromise between $\Delta v$ and the TOF, depending on the mission.

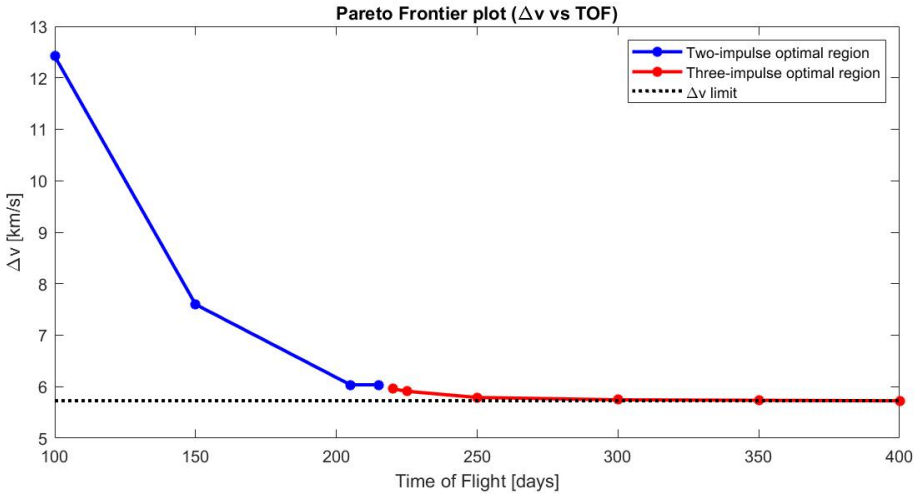



Figure 20: Pareto frontier plot

In this figure, three regions can be distinguished. The first one of them goes from a TOF of 100 days to 215 days, and it is the one at which two-impulse manoeuvres are optimal. The second region starts at 220 days until 400 days, and it is the three-impulse optimal region. It is interesting to point out that a non-optimal area appears between 215 and 220 days, which can be better seen in Figure 21. This occurs due to the fact that the maximum TOF for which two-impulse trajectories are optimal is around 202 days, which can also be checked by looking at Table 6. After this point, the two-impulse manoeuvre starts offering worse $\Delta v$ results, but three-impulse transfer orbits are still not optimal until a TOF of approximately 220 days. Finally, the black dotted line represents the smallest $\Delta v$ value encountered during the analysis.

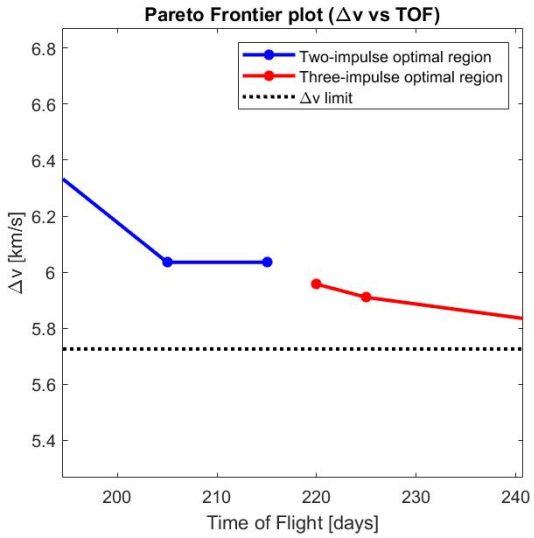



Figure 21: Zoom-in on the non-optimal region of the Pareto frontier plot

The next step will be to apply engineering judgement in order to select the optimal trajectory in terms of both $\Delta v$ budget and TOF. This analysis is what makes the optimization multi-objective, and is therefore an essential part of this work.

Depending on the mission requirements and specifications, the design job will point to a certain direction. This is why three different situations will be analysed: one where the TOF is prioritised, a second one in which both $\Delta v$ and the TOF want to be minimised, and a third one where achieving a minimum $\Delta v$ is the primary concern.

### 4.2.1 Time of Flight = 100 days

The first case that will be studied is the one for which the $\varepsilon$-constraint value for the TOF, that is, the limit inputted to the genetic algorithm, is 100 days. As it can be seen in both Table 6 and Figure 20, the $\Delta v$ value is 12.42 km/s, while the TOF is of 100 days. Therefore, this would be a mission where the $\Delta v$ value would not be important, as it is substantially high, whilst the TOF needs to be as small as possible.

The optimal trajectory for this mission duration is the **two-impulse** one shown in Figure 22. As expected, a relatively short transfer arc appears. By looking at the second impulse, given at the outer martian orbit, it seems logical that $\Delta v$ presents such a high value.

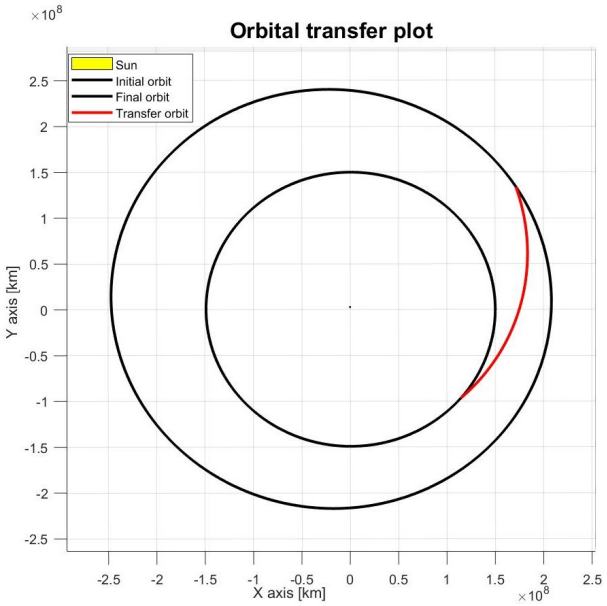



Figure 22: Two-impulse trajectory for $\varepsilon = 100$ days

The information about this manoeuvre is shown in Tables 7 and 8, which respectively sum up the Keplerian elements of the transfer orbit and the impulse specifications.

| **Parameter** | **Value** |
|---|---|
| $a_t\ [km]$ | 216 212 350 |
| $e_t\ [-]$ | 0.3222 |
| $i_t\ [\,^\circ\,]$ | 0.3928 |
| $\Omega_t\ [\,^\circ\,]$ | 139.1682 |
| $\omega_t\ [\,^\circ\,]$ | 149.5187 |
| $\theta_{t1}\ [\,^\circ\,]$ | 30.2685 |
| $\theta_{t2}\ [\,^\circ\,]$ | 108.8526 |

Table 7: Optimal transfer orbit parameters for $\varepsilon = 100$ days

| **Impulse** | **Coordinates** | | | | | |
|---|---|---|---|---|---|---|
| | **Rectangular** | | | **Spherical** | | |
| | x [km] | y [km] | z [km] | r [km] | b [ ° ] | l [ ° ] |
| 1 | 114 321 277 | -99 534 616 | 3860 | 151 579 993 | 0.0015 | -41.0444 |
| 2 | 171 495 157 | 131 779 410 | -1 452 121 | 216 283 404 | -0.3847 | 37.5392 |

Table 8: Impulse locations for $\varepsilon = 100$ days

### 4.2.2 Time of Flight = 250 days

This mission duration limit has been selected as the optimal one for the case where the time of flight is not a priority, but as much $\Delta v$ as possible needs to be saved. A constraint value of 250 days has been chosen instead of higher values because, as it can be seen in Table 6, the required propellant does not diminish significantly past this value. Therefore, the total $\Delta v$ would be of 5.79 km/s, and the TOF has a value of 249.54 days.

For this new case, the optimal trajectory is the **three-impulsive** one shown in Figure 23. Again, the transfer details have been summed up in Tables 9 and 10.

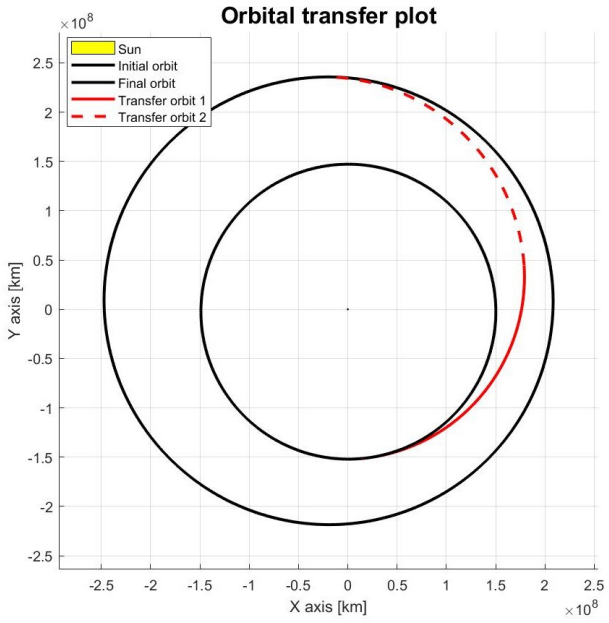



Figure 23: Three-impulse trajectory for $\varepsilon = 250$ days

| **Parameter** | **Value** | |
|---|---|---|
| | **Transfer arc 1** | **Transfer arc 2** |
| $a_t\ [km]$ | 185 874 388 | 202 667 112 |
| $e_t\ [-]$ | 0.1822 | 0.1964 |
| $i_t\ [\ ^\circ\ ]$ | 0.3569 | 1.4416 |
| $\Omega_t\ [\ ^\circ\ ]$ | 95.5814 | 28.2589 |
| $\omega_t\ [\ ^\circ\ ]$ | -179.4163 | -87.2804 |
| $\theta_{t1}\ [\ ^\circ\ ]$ | -0.6376 | 73.1072 |
| $\theta_{t2}\ [\ ^\circ\ ]$ | 97.9247 | 154.5096 |

Table 9: Optimal transfer orbit parameters for $\varepsilon = 250$ days

| **Impulse** | **Coordinates** | | | | | |
|---|---|---|---|---|---|---|
| | **Rectangular** | | | **Spherical** | | |
| | x [km] | y [km] | z [km] | r [km] | b [ ° ] | l [ ° ] |
| 1 | 14 641 629 | -151 295 917 | 889 | 152 002 736 | 0.0003 | -84.4724 |
| 2 | 178 783 764 | 44 874 272 | -1 135 481 | 184 332 916 | -0.3529 | 14.0900 |
| 3 | -22 617 500 | 235 686 056 | 5 493 797 | 236 832 536 | 1.3292 | 95.4816 |

Table 10: Impulse locations for $\varepsilon = 250$ days

### 4.2.3 Time of Flight = 205 days

To finalise, the most challenging case will be addressed. This is the situation where a true compromise wants to be found between propellant use and mission duration. By thoroughly analysing all data collected in Table 6 and in the Pareto frontier plot (Figure 20), it has been decided that introducing a limit TOF of 205 days renders the optimal results. Before this point, even though the TOF values are fairly low, $\Delta v$ has still got a considerable magnitude. Past this point, the increase in the TOF does not compensate the decrease in $\Delta v$, which is objectively low. However, the solution for a limit TOF of 205 days maintains the mission duration at a reasonable value (202.21 days), whilst offering a satisfactory result for $\Delta v$ (6.04 km/s). Also, as this transfer is a two-impulse one, it reduces the number of necessary parameters, thus increasing the mission reliability.

It is interesting to point out that the optimal solution the genetic algorithm offers for this case is the same as the two-impulse trajectory that was studied when comparing with the Hohmann case. This is due to the fact that, from a limit TOF of approximately 202 days, the optimal two-impulse trajectory does never change. From this value, the resultant $\Delta v$ keeps increasing, and three-impulse trajectories start being more optimal.

As it has been said, for the present case, the **two-impulse** manoeuvre is the optimal one. Even though this graph has already been shown in section 4.1, it will be presented again for clarity as Figure 24.

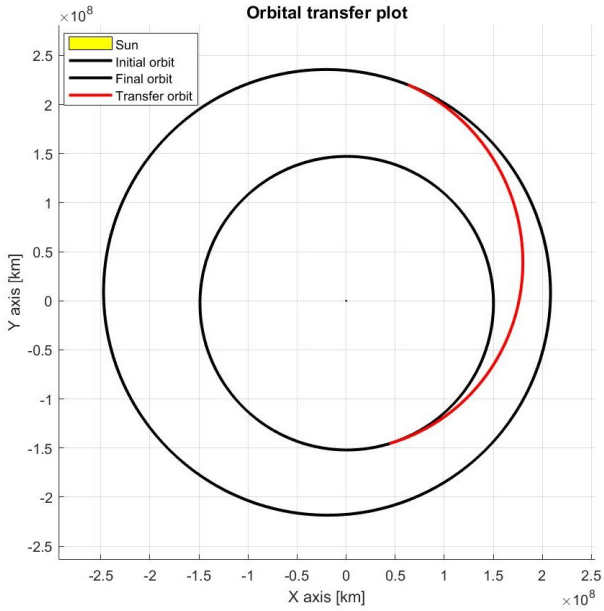



Figure 24: Two-impulse trajectory for $\varepsilon = 205$ days

As for the transfer orbit, even though some of its Keplerian parameters were already shown in Table 4, all of them are now summed up in Table 9. The coordinates of both impulses are included in Table 12.

| **Parameter** | **Value** |
|---|---|
| $a_t\ [km]$ | 194 651 242 |
| $e_t\ [-]$ | 0.2192 |
| $i_t\ [\,^\circ\,]$ | 1.4140 |
| $\Omega_t\ [\,^\circ\,]$ | -73.0876 |
| $\omega_t\ [\,^\circ\,]$ | -3.2615 |
| $\theta_{t1}\ [\,^\circ\,]$ | 3.2889 |
| $\theta_{t2}\ [\,^\circ\,]$ | 150.3914 |

Table 11: Optimal transfer orbit parameters for $\varepsilon = 205$ days

| **Impulse** | **Coordinates** | | | | | |
|---|---|---|---|---|---|---|
| | **Rectangular** | | | **Spherical** | | |
| | x [km] | y [km] | z [km] | r [km] | b [ ° ] | l [ ° ] |
| 1 | 44 296 681 | -145 433 126 | 1800 | 152 029 570 | 0.0007 | -73.0601 |
| 2 | 62 901 480 | 220 093 514 | 3 066 119 | 228 926 084 | 0.7674 | 74.0504 |

Table 12: Impulse locations for $\varepsilon = 205$ days

### 4.2.4 Results validation

In order to verify the obtained results, the case for a limit TOF of 205 days will be employed. It is reminded that this was selected as the option that optimized both the $\Delta v$ budget and the mission duration at the same time.

For validating this selection, it is enough to search for real Mars launches in the time period from the 1 January 2020 to 778 days later, which is the span covered by the genetic algorithm in this work. The search offers three relevant results [28], which are summed up in Table 13. The last row on said table corresponds to the optimal solution reached in this project.

| **Mission** | **Launch date** | **TOF [days]** |
|---|---|---|
| Emirates Mars | 19 July 2020 | 205 |
| Tianwen-1 | 23 July 2020 | 202 |
| Mars 2020 | 30 July 2020 | 203 |
| Genetic algorithm optimal | 11 July 2020 | 202.2104 |

Table 13: Missions to Mars from 2020 to 2022

It is easy to see that the result obtained by the genetic algorithm is highly similar to the three real Mars launches that took place in the year 2020. So much so, that the option selected as optimal in terms of both $\Delta v$ and TOF can be considered successfully validated. Moreover, another interesting conclusion can be extracted from Table 13. It can be deduced that, in the interval between 2020 and 2022, from mid to late July, there existed an optimal launch window to Mars. This is confirmed by both the results obtained in this project and summed up in Table 6, and the three real launches indicated in the table above.

# 5 Budget

This section aims to break down the total expense of the project, which can be divided into labour, energy, hardware and software costs.

## 5.1 Labour costs

These expenses include the ones associated to the hours that the student and both tutors have been working on the project, which are summed up in Tables 14 and 15. The second one of these tables already includes the joint information for both supervisors.

| **Concept** | **Hours** | **cost/h [€]** | **Subtotal [€]** |
|---|---|---|---|
| Documentation and research | 50 | 15 | 750 |
| Programming | 60 | 15 | 900 |
| Acquisition of results | 115 | 15 | 1725 |
| Memory writing | 80 | 15 | 1200 |
| **Total** | 305 | [−] | 4575 |

Table 14: Labour costs associated to the engineering student

| **Concept** | **Hours** | **cost/h [€]** | **Subtotal [€]** |
|---|---|---|---|
| Meetings | 30 | 40 | 1200 |
| Project revising | 40 | 40 | 1600 |
| **Total** | 70 | [−] | 2800 |

Table 15: Labour costs associated to both tutors

## 5.2 Energy consumption costs

During this work, a Personal Computer (PC) has been the main tool used by the student. A substantial amount of hours has been spent assembling code, as well as running it and writing the project. Therefore, the total costs should include the energetic expense of this hardware. Said PC is an Acer Swift SF314-52, and manufacturer data [29] states that its power consumption has a value of 45 W. As the total number of hours this computer has been used equals the total time the student has spent working on this project, this period would be 305 hours as stated in Table 14. Therefore, the total power consumption will be:

$$P_c = 45\ \mathrm{W} \cdot 305\ \mathrm{h} = 13725\ \mathrm{W} \cdot \mathrm{h} = 13.725\ \mathrm{kW} \cdot \mathrm{h} \tag{42}$$

Now, even though the cost of power in Spain changes depending on the day and hour [30], an average of 0.1694 $€/(\mathrm{kW} \cdot \mathrm{h})$ has been estimated. This way, the total cost of the power

consumed by the computer can be calculated as follows:

$$C_P = 13.725 \cdot 0.1694 = \mathbf{2.33}\ € \tag{43}$$

## 5.3 Hardware costs

Once the expense associated to the power consumed has been estimated, the cost related to the use of said computer needs to be calculated. As the student employed her personal computer for a limited amount of time, only the depreciation cost will be taken into account. For this purpose, the depreciation coefficient will be obtained from the Spanish Tax Agency [31].

| **Hardware** | **Cost [€]** | **Period [years]** | **Depreciation coeff. [%]** | **Total [€]** |
|---|---|---|---|---|
| Acer Swift SF314-52 | 313.77 | 0.5 | 0.25 | 39.22 |

Table 16: Hardware costs

It is pointed out that the computer was bought in 2017 for approximately 850 €. The cost represented in Table 16 is its price after applying the depreciation coefficient of 0.25 % for three and a half years of use, which corresponds to the time instant when the student started working in this project.

## 5.4 Software costs

Finally, for the completion of this academic work, several software licenses were used. The total cost of these programs is shown in Table 17.

| **Software** | **Hours** | **License cost [€/year]** | **cost/h [€]** | **Subtotal [€]** |
|---|---|---|---|---|
| MATLAB R2020b | 175 | 800 | 0.44 | 77 |
| Microsoft Office | 40 | 126 | 0.07 | 2.8 |
| Overleaf | 80 | 0 | 0 | 0 |
| | | | **Total** | 79.8 |

Table 17: Software costs

It is important to mention that, in order to calculate the hourly cost of a license, it was taken into account that one academic year comprises 60 ECTS, which translate to 1800 h.

## 5.5 Total costs

The following table sums up the total costs by taking into account the results obtained in previous sections.

| Concept | Amount [€] |
|---|---|
| Labour costs | 7375 |
| Electrical consumption costs | 2.33 |
| Hardware costs | 39.22 |
| Software costs | 79.8 |
| **Raw total** | 7496.35 |
| Industrial benefit (5%) | 374.82 |
| **Total with industrial benefit** | 7871.17 |
| VAT (21%) | 1652.95 |
| **Tender budget** | 9524.12 |

Table 18: Total costs

Therefore, the total raw cost is of:

SEVEN THOUSAND, FOUR HUNDRED AND NINETY-SIX EUROS WITH THIRTY-FIVE CENTS

And the total cost with industrial benefit is:

SEVEN THOUSAND, EIGHT HUNDRED AND SEVENTY-ONE EUROS WITH SEVENTEEN CENTS

Finally, the tender budget of this project is of:

NINE THOUSAND, FIVE HUNDRED AND TWENTY-FOUR EUROS WITH TWELVE CENTS

# 6 Solicitations

In this section, the most important regulations regarding the completion of a Bachelor's Degree Thesis will be presented.

As the academic institution at which the student has developed her Bachelor's Degree is the Universitat Politècnica de València, this organisation's regulations have been revised. Below, the most relevant ones are shown. It is pointed out that the original language in which these norms are written is in Spanish, and therefore they will be presented this way. The following text has been directly extracted from said regulations document [32].

**Artículo 1. Objeto**
La presente normativa tiene por objeto establecer el marco general regulatorio de las condiciones por la que se regirá en la Universidad Politècnica de València (en adelante, UPV) la matriculación, asignación, evaluación y otros aspectos de la tramitación académica y administrativa de los Trabajos Fin de Grado (TFG) y Trabajos Fin de Máster (TFM).

**Artículo 2. Ámbito de aplicación**

1. La presente normativa será de aplicación a las enseñanzas impartidas por la UPV conducentes a la obtención de los títulos de Grado y Máster Universitario de carácter oficial y validez en todo el territorio nacional (en adelante títulos oficiales).

2. Los TFG y TFM de los títulos oficiales que habiliten para el ejercicio de las profesiones reguladas se regirán por lo dispuesto en la correspondiente Orden Ministerial que establece los requisitos para la verificación del título, sin perjuicio de la aplicación, con carácter complementario, de lo que se indique en la presente Normativa Marco.

**Artículo 3. Naturaleza de los TFG y TFM**

1. Los TFG y TFM deberán estar orientados a la aplicación y evaluación de competencias asociadas al título.

2. En el caso de los TFG y en el de los TFM de títulos que habiliten para el ejercicio de profesiones reguladas deberán tener una orientación profesional. En el resto de casos, el TFM podrá tener orientación profesional o investigadora.

3. Los TFG y TFM consistirán en la realización de un trabajo o proyecto original en el que queden de manifiesto conocimientos, habilidades y competencias adquiridas por el estudiante a lo largo de sus estudios y, expresamente, las competencias asociadas a la materia TFG o TFM, tal y como se indique en la memoria de verificación.

4. La originalidad del trabajo a que se hace referencia en el punto anterior, debe entenderse sin menoscabo de que pueda ser parte independiente e individual de un trabajo integral desarrollado de manera conjunta entre estudiantes de una misma titulación o de diferentes titulaciones y ERT. En cualquier caso, la defensa del TFG y del TFM debe ser individual.

5. La materia TFG y TFM podrá organizarse mediante actividades de docencia reglada en forma de seminario, taller o similar; mediante trabajo autónomo y tutelado del estudiante; o mediante una mezcla de ambas.

6. El alcance, contenido y nivel de exigencia de los TFG y TFM deberá adecuarse a la asignación de ECTS que dicha materia haya recibido en la memoria de verificación [...].

7. Como cualquier otra materia de un plan de estudios, los TFG y TFM deberán disponer de una Guía Docente [...].

**Artículo 4. Tutores**

1. Cuando parte o la totalidad de los ECTS asignados a la materia se organicen mediante trabajo autónomo y tutelado, para la realización de su TFG o TFM, los estudiantes contarán con la dirección de un tutor académico que supervisará el trabajo académico y les dará apoyo en la gestión administrativa.[...].

**Artículo 8. Presentación**

1. Salvo que la naturaleza del trabajo lo impida, previamente a la defensa y calificación del TFG o TFM, el estudiante deberá presentar en la secretaría de la ERT el trabajo realizado, en formato electrónico, y redactado en castellano, valenciano o inglés. La presentación

del trabajo se realizará siguiendo el procedimiento establecido por el Área de Biblioteca y Documentación Científica, a los efectos de su posterior inclusión en los repositorios institucionales de la universidad.

Moreover, apart from the presented regulations, some conditions have to be met in order for the worker productivity to be optimal whilst minimising any possible health risks. Therefore, it must be ensured that:

- A proper body posture is maintained, and a comfortable and ergonomic chair is being used in order to prevent any muscular injuries.
- Natural light is always present during the daylight hours, and the workspace is sufficiently lit at all times. This will help reduce fatigue and avoid eye damage.
- Any connections to the electric grid must be safely made so as to prevent any harm to the equipment or to the engineer.
- The workload must be carefully regulated, as well as the amount and duration of the breaks taken.

As for the workplace, it is the student's choice to develop the work in any location that meets the abovementioned requirements, provided it allows maximum concentration and limits all distractions.

## 6.1 Technical aspects

Now, it is also a part of this section to present the hardware and software tools employed during the project completion.

As for the hardware used, it is an Acer Swift SF314-52 laptop with an Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz 2.90 GHz processor, an 8GB RAM and an Intel(R) HD Graphics 620 graphic card. The operating system it carries is the 64-bit Windows 10.

Regarding the software employed, it has already been listed in section 5. Now, the use of each program will also be mentioned.

- **MATLAB R2020b**
  This software was used for programming the necessary code that allowed the implementation of the genetic algorithm and the presentation of results and graphs.
- **Microsoft Teams**
  This was an essential tool that enabled online meetings between the student and both tutors.
- **Overleaf**
  This LaTeX online editor was used for writing the project memory.
- **Microsoft PowerPoint**
  Employed for assembling the project slide presentation.

# 7 Conclusions

After comparing the obtained results with the bi-impulsive Hohmann transfer and with three different missions to Mars, it can be said that the objectives of this project have been achieved. A successful implementation of the multi-objective genetic algorithm has been accomplished in order to optimise in a fast and reliable manner trajectories between the Earth and Mars. This is useful in many ways, as it allows reducing the mission costs and the human resources devoted to orbit calculations. Moreover, the allocated budget to the project is a reasonable quantity, as all resources have been efficiently used.

This has been possible due to several factors. In the first place, the parameter settings of the genetic algorithm were carefully thought, and numerous trials were run, until they were correctly defined. The variable resolution chosen has also been a determining factor for obtaining valid results, and it was the outcome of a compromise between the necessary accuracy and computational time. Regarding this computational time, calculating the two and three-impulse transfer orbits for each $\varepsilon$-constraint value takes less than 15 minutes. This can be considered reasonably fast, as the resolution for both the launch date and time of flight is 1 h, which is a fairly precise value.

Reaching successful results has also been possible by deeply understanding the orbital problem that was being solved and the corresponding Lambert solution. This allowed identifying the methodology that implied less variables, reducing this way the computational cost.

Also, the multi-objective character of the presented algorithm is worth mentioning. By building a Pareto frontier, the engineer can easily identify the mission that best fulfills the requirements.

In conclusion, for all of the above mentioned reasons, it is believed that this project has accomplished all of its objectives in a satisfactory manner. A powerful computational tool has been successfully developed and employed, and relevant and valid results have been presented.

# 8 Future work

As there is always room for further improvement and development, this project could be made even more comprehensive, and there are different pathways to do so.

For example, incorporating orbital perturbations caused by third-body accelerations would increase the accuracy of the model. Also, it could be interesting to take into account other areas of aerospace engineering such as solar energy. In order to do so, solar panels could be added to the spaceship travelling to Mars. This would mean calculating the orientation of the satellite so that such panels were facing the Sun at all times throughout the trajectory. Also, the occurrence of eclipses would need to be assessed, in order to identify the time spans when the panels would stop receiving sunlight.

An additional calculation that could be made would be to study the time window for which communication with the ground station would be possible. Finally, it has been considered that the propulsion model could be made more accurate. At the present moment, as it has been explained, impulsive manoeuvres are being taken into account in order to simplify the

calculations. However, this could be changed so as to represent a propulsive model based on continuous thrust. This would lead to incorporating the mass as one of the states and obtaining a more realistic idea of the cost saved.

On a different line of work, if the mission required so, a fly-by in Venus could be added to the existing code. This would definitely mean devoting several hours to deciding the most efficient, reliable way to implement the manoeuvre. However, the result would be an integral code that would be able to analyse an even wider range of mission possibilities.

Finally, as for the genetic algorithm parameter settings, it has been said that they are carefully thought and selected. However, there is always a margin for research and improvement, and an interesting way to explore the capabilities of the code would be by further analysing even more settings possibilities. Moreover, different optimisers such as the Evolutionary Algorithm, Particle Swarm Optimization, or Ant Colony Optimizer could be employed, and the results compared, in order to find the one that offers the best performance. Also, in this work, a single-objective algorithm was used, and then the user carried out a study so that the method could be a multi-objective one. Therefore, it would be interesting to employ a naturally multi-objective algorithm, and compare the obtained solutions to the present ones.

# 9 APPENDIX

## 9.1 MATLAB code

Listing 1: Lambert's problem [22]

```
% LAMBERT            Lambert-targeter for ballistic flights
%                    (Izzo, and Lancaster, Blanchard & Gooding)
%
% Usage:
%    [V1, V2, extremal_distances, exitflag] = lambert(r1, r2,
   tf, m, GM_central)
%
% Dimensions:
%             r1, r2 ->  [1x3]
%             V1, V2 ->  [1x3]
% extremal_distances ->  [1x2]
%              tf, m ->  [1x1]
%         GM_central ->  [1x1]
%
% This function solves any Lambert problem *robustly*. It uses
   two separate
% solvers; the first one tried is a new and unpublished
   algorithm developed
% by Dr. D. Izzo from the European Space Agency [1]. This
   version is extremely
% fast, but especially for larger [m] it still fails quite
   frequently. In such
% cases, a MUCH more robust algorithm is started (the one by
   Lancaster &
% Blancard [2], with modifcations, initial values and other
   improvements by
% R.Gooding [3]), which is a lot slower partly because of its
   robustness.
%
% INPUT ARGUMENTS:
%
   ================================================================

%    name         units    description
%
   ================================================================

%   r1, r1        [km]     position vectors of the two terminal
   points.
%     tf         [days]    time of flight to solve for
%      m           [-]     specifies the number of complete
```

```
   orbits to complete
%                          (should be an integer)
% GM_central   [km3/s2]   std. grav. parameter ( G M  = mu) of
   the central body
%
% OUTPUT ARGUMENTS:
%
   ================================================================

%   name              units   description
%
   ================================================================

%  V1, V2              [km/s]  terminal velocities at the end-
   points
%  extremal_distances  [km]    minimum(1) and maximum(2)
   distance of the
%                              spacecraft to the central body.
%  exitflag             [-]    Integer containing information on
    why the
%                              routine terminated. A value of +1
    indicates
%                              success; a normal exit. A value
   of -1
%                              indicates that the given problem
   has no
%                              solution and cannot be solved. A
   value of -2
%                              indicates that both algorithms
   failed to find
%                              a solution. This should never
   occur since
%                              these problems are well-defined,
   and at the
%                              very least it can be determined
   that the
%                              problem has no solution.
   Nevertheless, it
%                              still occurs sometimes for
   accidental
%                              erroneous input, so it provides a
    basic
%                              mechanism to check any
   application using this
%                              algorithm.
```

```
%
% This routine can be compiled to increase its speed by a
   factor of about
% 10-15, which is certainly advisable when the complete
   application requires
% a great number of Lambert problems to be solved. The entire
   routine is
% written in embedded MATLAB, so it can be compiled with the
   emlmex()
% function (older MATLAB) or codegen() function (MATLAB 2011a
   and later).
%
% To do this using emlmex(), make sure MATLAB's current
   directory is equal
% to where this file is located. Then, copy-paste and execute
   the following
% commands to the command window:
%
%    example_input = {...
%         [0.0, 0.0, 0.0], ...% r1vec
%         [0.0, 0.0, 0.0], ...% r2vec
%          0.0, ...           % tf
%          0.0, ...           % m
%          0.0};              % muC
%    emlmex -eg example_input lambert.m
%
% This is of course assuming your compiler is configured
   correctly. See the
% documentation of emlmex() on how to do that.
%
% Using codegen(), the syntax is as follows:
%
%    example_input = {...
%         [0.0, 0.0, 0.0], ...% r1vec
%         [0.0, 0.0, 0.0], ...% r2vec
%          0.0, ...           % tf
%          0.0, ...           % m
%          0.0};              % muC
%    codegen lambert.m -args example_input
%
% Note that in newer MATLAB versions, the code analyzer will
   complain about
% the pragma "%#eml" after the main function's name, and
   possibly, issue
% subsequent warnings related to this issue. To get rid of this
```

```
    problem, simply
% replace the "%#eml" directive with "%#codegen".
%
%
%
% References:
%
% [1] Izzo, D. ESA Advanced Concepts team. Code used available
   in MGA.M, on
%     http://www.esa.int/gsp/ACT/inf/op/globopt.htm. Last
   retreived Nov, 2009.
% [2] Lancaster, E.R. and Blanchard, R.C. "A unified form of
   Lambert's theorem."
%     NASA technical note TN D-5368,1969.
% [3] Gooding, R.H. "A procedure for the solution of Lambert's
   orbital boundary-value
%     problem. Celestial Mechanics and Dynamical Astronomy,
   48:145  165  ,1990.
%
% See also lambert_low_ExpoSins.


% Please report bugs and inquiries to:
%
% Name       : Rody P.S. Oldenhuis
% E-mail     : oldenhuis@gmail.com
% Licence    : 2-clause BSD (see License.txt)


% If you find this work useful, please consider a donation:
% https://www.paypal.me/RodyO/3.5

% If you want to cite this work in an academic paper, please
   use
% the following template:
%
% Rody Oldenhuis, orcid.org/0000-0002-3162-3660. "Lambert" <
   version>,
% <date you last used it>. MATLAB Robust solver for Lambert's
% orbital-boundary value problem.
% https://nl.mathworks.com/matlabcentral/fileexchange/26348



%
```

```
   ------------------------------------------------------------------

% Izzo's version:
% Very fast, but not very robust for more complicated cases
%
   ------------------------------------------------------------------

function [V1,...
          V2, ...
          extremal_distances,...
          exitflag,a] = lambert(r1vec,...
                                r2vec,...
                                tf,...
                                m,...
                                muC) %#coder

% original documentation:
%{
 This routine implements a new algorithm that solves Lambert's
    problem. The
 algorithm has two major characteristics that makes it
    favorable to other
 existing ones.

 1) It describes the generic orbit solution of the boundary
    condition
 problem through the variable X=log(1+cos(alpha/2)). By doing
    so the
 graph of the time of flight become defined in the entire real
    axis and
 resembles a straight line. Convergence is granted within few
    iterations
 for all the possible geometries (except, of course, when the
    transfer
 angle is zero). When multiple revolutions are considered the
    variable is
 X=tan(cos(alpha/2)*pi/2).

 2) Once the orbit has been determined in the plane, this
    routine
 evaluates the velocity vectors at the two points in a way that
     is not
 singular for the transfer angle approaching to pi (Lagrange
    coefficient
 based methods are numerically not well suited for this purpose
```

```
    ).

 As a result Lambert's problem is solved (with multiple
    revolutions
 being accounted for) with the same computational effort for
    all
 possible geometries. The case of near 180 transfers is also
    solved
 efficiently.

  We note here that even when the transfer angle is exactly
     equal to pi
 the algorithm does solve the problem in the plane (it finds X)
    , but it
 is not able to evaluate the plane in which the orbit lies. A
    solution
 to this would be to provide the direction of the plane
    containing the
 transfer orbit from outside. This has not been implemented in
    this
 routine since such a direction would depend on which
    application the
 transfer is going to be used in.

 please report bugs to dario.izzo@esa.int
%}

% adjusted documentation:
%{
 By default, the short-way solution is computed. The long way
    solution
 may be requested by giving a negative value to the
    corresponding
 time-of-flight [tf].

 For problems with |m| > 0, there are generally two solutions.
    By
 default, the right branch solution will be returned. The left
    branch
 may be requested by giving a negative value to the
    corresponding
 number of complete revolutions [m].
%}

% Authors
```

```
% .-`-.-`-.-`-.-`-.-`-.-`-.-`-.-`-.-`-.-`-.-`-.-`-.-`-.-`-.-`-.-`-.-`-.-`-.
% Name       : Dr. Dario Izzo
% E-mail     : dario.izzo@esa.int
% Affiliation: ESA / Advanced Concepts Team (ACT)

% Made more readible and optimized for speed by Rody P.S.
   Oldenhuis
% Code available in MGA.M on   http://www.esa.int/gsp/ACT/inf/
   op/globopt.htm

% last edited 12/Dec/2009

% ADJUSTED FOR EML-COMPILATION 24/Dec/2009

    % initial values
    tol = 1e-14;    bad = false;     days = 86400;

    % work with non-dimensional units
    r1 = sqrt(r1vec*r1vec.');  r1vec = r1vec/r1;
    V = sqrt(muC/r1);          r2vec = r2vec/r1;
    T = r1/V;                  tf    = tf*days/T; % also
       transform to seconds

    % relevant geometry parameters (non dimensional)
    mr2vec = sqrt(r2vec*r2vec.');
    % make 100% sure it's in (-1 <= dth <= +1)
    dth = acos( max(-1, min(1, (r1vec*r2vec.')/mr2vec)) );

    % decide whether to use the left or right branch (for multi
       -revolution
    % problems), and the long- or short way
    leftbranch = sign(m);   longway = sign(tf);
    m = abs(m);             tf = abs(tf);
    if (longway < 0), dth = 2*pi - dth; end

    % derived quantities
    c      = sqrt(1 + mr2vec^2 - 2*mr2vec*cos(dth)); % non-
       dimensional chord
    s      = (1 + mr2vec + c)/2;                     % non-
       dimensional semi-perimeter
    a_min  = s/2;                                    % minimum
       energy ellipse semi major axis
    Lambda = sqrt(mr2vec)*cos(dth/2)/s;              % lambda
       parameter (from BATTIN's book)
    crossprd = [r1vec(2)*r2vec(3) - r1vec(3)*r2vec(2),...
```

```
            r1vec(3)*r2vec(1) - r1vec(1)*r2vec(3),...% non-
               dimensional normal vectors
            r1vec(1)*r2vec(2) - r1vec(2)*r2vec(1)];
mcr         = sqrt(crossprd*crossprd.');           %
   magnitues thereof
nrmunit     = crossprd/mcr;                        % unit
   vector thereof

% Initial values
% ------------------------------------------------------------

% ELMEX requires this variable to be declared OUTSIDE the
   IF-statement
logt = log(tf); % avoid re-computing the same value

% single revolution (1 solution)
if (m == 0)

    % initial values
    inn1 = -0.5233;      % first initial guess
    inn2 = +0.5233;      % second initial guess
    x1   = log(1 + inn1);% transformed first initial guess
    x2   = log(1 + inn2);% transformed first second guess

    % multiple revolutions (0, 1 or 2 solutions)
    % the returned soltuion depends on the sign of [m]
else
    % select initial values
    if (leftbranch < 0)
        inn1 = -0.5234; % first initial guess, left branch
        inn2 = -0.2234; % second initial guess, left branch
    else
        inn1 = +0.7234; % first initial guess, right branch
        inn2 = +0.5234; % second initial guess, right
           branch
    end
    x1 = tan(inn1*pi/2);% transformed first initial guess
    x2 = tan(inn2*pi/2);% transformed first second guess
end

% since (inn1, inn2) < 0, initial estimate is always
   ellipse
xx    = [inn1, inn2];  aa = a_min./(1 - xx.^2);
bbeta = longway * 2*asin(sqrt((s-c)/2./aa));
% make 100.4% sure it's in (-1 <= xx <= +1)
```

```
aalfa = 2*acos(  max(-1, min(1, xx)) );

% evaluate the time of flight via Lagrange expression
y12  = aa.*sqrt(aa).*((aalfa - sin(aalfa)) - (bbeta-sin(
   bbeta)) + 2*pi*m);

% initial estimates for y
if m == 0
    y1 = log(y12(1)) - logt;
    y2 = log(y12(2)) - logt;
else
    y1 = y12(1) - tf;
    y2 = y12(2) - tf;
end

% Solve for x
% -----------------------------------------------------------

% Newton-Raphson iterations
% NOTE - the number of iterations will go to infinity in
   case
% m > 0  and there is no solution. Start the other routine
   in
% that case
err = inf;  iterations = 0; xnew = 0;
while (err > tol)
    % increment number of iterations
    iterations = iterations + 1;
    % new x
    xnew = (x1*y2 - y1*x2) / (y2-y1);
    % copy-pasted code (for performance)
    if m == 0, x = exp(xnew) - 1; else x = atan(xnew)*2/pi;
        end
    a = a_min/(1 - x^2);
    if (x < 1) % ellipse
        beta = longway * 2*asin(sqrt((s-c)/2/a));
        % make 100.4% sure it's in (-1 <= xx <= +1)
        alfa = 2*acos( max(-1, min(1, x)) );
    else % hyperbola
        alfa = 2*acosh(x);
        beta = longway * 2*asinh(sqrt((s-c)/(-2*a)));
    end
    % evaluate the time of flight via Lagrange expression
    if (a > 0)
        tof = a*sqrt(a)*((alfa - sin(alfa)) - (beta-sin(
```

```
            beta)) + 2*pi*m);
    else
        tof = -a*sqrt(-a)*((sinh(alfa) - alfa) - (sinh(beta
            ) - beta));
    end
    % new value of y
    if m ==0, ynew = log(tof) - logt; else ynew = tof - tf;
        end
    % save previous and current values for the next
       iterarion
    % (prevents getting stuck between two values)
    x1 = x2;  x2 = xnew;
    y1 = y2;  y2 = ynew;
    % update error
    err = abs(x1 - xnew);
    % escape clause
    if (iterations > 15), bad = true; break; end
end

% If the Newton-Raphson scheme failed, try to solve the
   problem
% with the other Lambert targeter.
if bad
    % NOTE: use the original, UN-normalized quantities
    [V1, V2, extremal_distances, exitflag] = ...
        lambert_LancasterBlanchard(r1vec*r1, r2vec*r1,
           longway*tf*T, leftbranch*m, muC);
    return
end

% convert converged value of x
if m==0, x = exp(xnew) - 1; else x = atan(xnew)*2/pi; end

%{
  The solution has been evaluated in terms of log(x+1) or
     tan(x*pi/2), we
  now need the conic. As for transfer angles near to pi the
      Lagrange-
  coefficients technique goes singular (dg approaches a
     zero/zero that is
  numerically bad) we here use a different technique for
     those cases. When
  the transfer angle is exactly equal to pi, then the ih
     unit vector is not
  determined. The remaining equations, though, are still
```

```
     valid.
%}

% Solution for the semi-major axis
a = a_min/(1-x^2);

% Calculate psi
if (x < 1) % ellipse
    beta = longway * 2*asin(sqrt((s-c)/2/a));
    % make 100.4% sure it's in (-1 <= xx <= +1)
    alfa = 2*acos( max(-1, min(1, x)) );
    psi  = (alfa-beta)/2;
    eta2 = 2*a*sin(psi)^2/s;
    eta  = sqrt(eta2);
else       % hyperbola
    beta = longway * 2*asinh(sqrt((c-s)/2/a));
    alfa = 2*acosh(x);
    psi  = (alfa-beta)/2;
    eta2 = -2*a*sinh(psi)^2/s;
    eta  = sqrt(eta2);
end

% unit of the normalized normal vector
ih = longway * nrmunit;

% unit vector for normalized [r2vec]
r2n = r2vec/mr2vec;

% cross-products
% don't use cross() (emlmex() would try to compile it, and
   this way it
% also does not create any additional overhead)
crsprd1 = [ih(2)*r1vec(3)-ih(3)*r1vec(2),...
           ih(3)*r1vec(1)-ih(1)*r1vec(3),...
           ih(1)*r1vec(2)-ih(2)*r1vec(1)];
crsprd2 = [ih(2)*r2n(3)-ih(3)*r2n(2),...
           ih(3)*r2n(1)-ih(1)*r2n(3),...
           ih(1)*r2n(2)-ih(2)*r2n(1)];

% radial and tangential directions for departure velocity
Vr1 = 1/eta/sqrt(a_min) * (2*Lambda*a_min - Lambda - x*eta)
   ;
Vt1 = sqrt(mr2vec/a_min/eta2 * sin(dth/2)^2);

% radial and tangential directions for arrival velocity
```

```
    Vt2 = Vt1/mr2vec;
    Vr2 = (Vt1 - Vt2)/tan(dth/2) - Vr1;

    % terminal velocities
    V1 = (Vr1*r1vec + Vt1*crsprd1)*V;
    V2 = (Vr2*r2n + Vt2*crsprd2)*V;

    % exitflag
    exitflag = 1; % (success)

    % also compute minimum distance to central body
    % NOTE: use un-transformed vectors again!
    extremal_distances = ...
        minmax_distances(r1vec*r1, r1, r2vec*r1, mr2vec*r1, dth
           , a*r1, V1, V2, m, muC);

end

%
   ------------------------------------------------------------------

% Lancaster & Blanchard version, with improvements by Gooding
% Very reliable, moderately fast for both simple and
   complicated cases
%
   ------------------------------------------------------------------

function [V1,...
          V2,...
          extremal_distances,...
          exitflag] = lambert_LancasterBlanchard(r1vec,...
                                                 r2vec,...
                                                 tf,...
                                                 m,...
                                                 muC) %#coder
%{
LAMBERT_LANCASTERBLANCHARD       High-Thrust Lambert-targeter

lambert_LancasterBlanchard() uses the method developed by
Lancaster & Blancard, as described in their 1969 paper. Initial
    values,
and several details of the procedure, are provided by R.H.
   Gooding,
as described in his 1990 paper.
%}
```

```
% Please report bugs and inquiries to:
%
% Name        : Rody P.S. Oldenhuis
% E-mail      : oldenhuis@gmail.com
% Licence     : 2-clause BSD (see License.txt)


% If you find this work useful, please consider a donation:
% https://www.paypal.me/RodyO/3.5

    % ADJUSTED FOR EML-COMPILATION 29/Sep/2009

    % manipulate input
    tol     = 1e-12;                                % optimum for
       numerical noise v.s. actual precision
    r1      = sqrt(r1vec*r1vec.');                  % magnitude of
       r1vec
    r2      = sqrt(r2vec*r2vec.');                  % magnitude of
       r2vec
    r1unit  = r1vec/r1;                             % unit vector
       of r1vec
    r2unit  = r2vec/r2;                             % unit vector
       of r2vec
    crsprod = cross(r1vec, r2vec, 2);               % cross product
        of r1vec and r2vec
    mcrsprd = sqrt(crsprod*crsprod.');              % magnitude of
       that cross product
    th1unit = cross(crsprod/mcrsprd, r1unit);       % unit vectors
       in the tangential-directions
    th2unit = cross(crsprod/mcrsprd, r2unit);
    % make 100.4% sure it's in (-1 <= x <= +1)
    dth = acos( max(-1, min(1, (r1vec*r2vec.')/r1/r2)) ); %
       turn angle

    % if the long way was selected, the turn-angle must be
       negative
    % to take care of the direction of final velocity
    longway = sign(tf); tf = abs(tf);
    if (longway < 0), dth = dth-2*pi; end

    % left-branch
    leftbranch = sign(m); m = abs(m);

    % define constants
```

```
c  = sqrt(r1^2 + r2^2 - 2*r1*r2*cos(dth));
s  = (r1 + r2 + c) / 2;
T  = sqrt(8*muC/s^3) * tf;
q  = sqrt(r1*r2)/s * cos(dth/2);

% general formulae for the initial values (Gooding)
% -------------------------------------------------

% some initial values
T0  = LancasterBlanchard(0, q, m);
Td  = T0 - T;
phr = mod(2*atan2(1 - q^2, 2*q), 2*pi);

% initial output is pessimistic
V1 = NaN(1,3);    V2 = V1;    extremal_distances = [NaN,
   NaN];

% single-revolution case
if (m == 0)
    x01 = T0*Td/4/T;
    if (Td > 0)
        x0 = x01;
    else
        x01 = Td/(4 - Td);
        x02 = -sqrt( -Td/(T+T0/2) );
        W   = x01 + 1.7*sqrt(2 - phr/pi);
        if (W >= 0)
            x03 = x01;
        else
            x03 = x01 + (-W).^(1/16).*(x02 - x01);
        end
        lambda = 1 + x03*(1 + x01)/2 - 0.03*x03^2*sqrt(1 +
           x01);
        x0 = lambda*x03;
    end

    % this estimate might not give a solution
    if (x0 < -1), exitflag = -1; return; end

% multi-revolution case
else

    % determine minimum Tp(x)
    xMpi = 4/(3*pi*(2*m + 1));
    if (phr < pi)
```

```
    xM0 = xMpi*(phr/pi)^(1/8);
elseif (phr > pi)
    xM0 = xMpi*(2 - (2 - phr/pi)^(1/8));
% EMLMEX requires this one
else
    xM0 = 0;
end

% use Halley's method
xM = xM0;  Tp = inf;  iterations = 0;
while abs(Tp) > tol
    % iterations
    iterations = iterations + 1;
    % compute first three derivatives
    [dummy, Tp, Tpp, Tppp] = LancasterBlanchard(xM, q,
       m);%#ok
    % new value of xM
    xMp = xM;
    xM  = xM - 2*Tp.*Tpp ./ (2*Tpp.^2 - Tp.*Tppp);
    % escape clause
    if mod(iterations, 7), xM = (xMp+xM)/2; end
    % the method might fail. Exit in that case
    if (iterations > 25), exitflag = -2; return; end
end

% xM should be elliptic (-1 < x < 1)
% (this should be impossible to go wrong)
if (xM < -1) || (xM > 1), exitflag = -1; return; end

% corresponding time
TM = LancasterBlanchard(xM, q, m);

% T should lie above the minimum T
if (TM > T), exitflag = -1; return; end

% find two initial values for second solution (again
   with lambda-type patch)
%
   -------------------------------------------------------------


% some initial values
TmTM  = T - TM;    T0mTM = T0 - TM;
[dummy, Tp, Tpp] = LancasterBlanchard(xM, q, m);%#ok
```

```
    % first estimate (only if m > 0)
    if leftbranch > 0
        x   = sqrt( TmTM / (Tpp/2 + TmTM/(1-xM)^2) );
        W   = xM + x;
        W   = 4*W/(4 + TmTM) + (1 - W)^2;
        x0  = x*(1 - (1 + m + (dth - 1/2)) / ...
            (1 + 0.15*m)*x*(W/2 + 0.03*x*sqrt(W))) + xM;

        % first estimate might not be able to yield
          possible solution
        if (x0 > 1), exitflag = -1; return; end

    % second estimate (only if m > 0)
    else
        if (Td > 0)
            x0 = xM - sqrt(TM/(Tpp/2 - TmTM*(Tpp/2/T0mTM -
               1/xM^2)));
        else
            x00 = Td / (4 - Td);
            W = x00 + 1.7*sqrt(2*(1 - phr));
            if (W >= 0)
                x03 = x00;
            else
                x03 = x00 - sqrt((-W)^(1/8))*(x00 + sqrt(-
                   Td/(1.5*T0 - Td)));
            end
            W      = 4/(4 - Td);
            lambda = (1 + (1 + m + 0.24*(dth - 1/2)) / ...
                (1 + 0.15*m)*x03*(W/2 - 0.03*x03*sqrt(W)));
            x0     = x03*lambda;
        end

        % estimate might not give solutions
        if (x0 < -1), exitflag = -1; return; end

    end
end

% find root of Lancaster & Blancard's function
% --------------------------------------------

% (Halley's method)
x = x0; Tx = inf; iterations = 0;
while abs(Tx) > tol
    % iterations
```

```
        iterations = iterations + 1;
        % compute function value, and first two derivatives
        [Tx, Tp, Tpp] = LancasterBlanchard(x, q, m);
        % find the root of the *difference* between the
        % function value [T_x] and the required time [T]
        Tx = Tx - T;
        % new value of x
        xp = x;
        x  = x - 2*Tx*Tp ./ (2*Tp^2 - Tx*Tpp);
        % escape clause
        if mod(iterations, 7), x = (xp+x)/2; end
        % Halley's method might fail
        if iterations > 25, exitflag = -2; return; end
    end

    % calculate terminal velocities
    % -----------------------------

    % constants required for this calculation
    gamma = sqrt(muC*s/2);
    if (c == 0)
        sigma = 1;
        rho   = 0;
        z     = abs(x);
    else
        sigma = 2*sqrt(r1*r2/(c^2)) * sin(dth/2);
        rho   = (r1 - r2)/c;
        z     = sqrt(1 + q^2*(x^2 - 1));
    end

    % radial component
    Vr1    = +gamma*((q*z - x) - rho*(q*z + x)) / r1;
    Vr1vec = Vr1*r1unit;
    Vr2    = -gamma*((q*z - x) + rho*(q*z + x)) / r2;
    Vr2vec = Vr2*r2unit;

    % tangential component
    Vtan1    = sigma * gamma * (z + q*x) / r1;
    Vtan1vec = Vtan1 * th1unit;
    Vtan2    = sigma * gamma * (z + q*x) / r2;
    Vtan2vec = Vtan2 * th2unit;

    % Cartesian velocity
    V1 = Vtan1vec + Vr1vec;
    V2 = Vtan2vec + Vr2vec;
```

```
    % exitflag
    exitflag = 1; % (success)

    % also determine minimum/maximum distance
    a = s/2/(1 - x^2); % semi-major axis
    extremal_distances = minmax_distances(r1vec, r1, r1vec, r2,
        dth, a, V1, V2, m, muC);

end

% Lancaster & Blanchard's function, and three derivatives
   thereof
function [T, Tp, Tpp, Tppp] = LancasterBlanchard(x, q, m)

    % protection against idiotic input
    if (x < -1) % impossible; negative eccentricity
        x = abs(x) - 2;
    elseif (x == -1) % impossible; offset x slightly
        x = x + eps;
    end

    % compute parameter E
    E  = x*x - 1;

    % T(x), T'(x), T''(x)
    if x == 1 % exactly parabolic; solutions known exactly
        % T(x)
        T = 4/3*(1-q^3);
        % T'(x)
        Tp = 4/5*(q^5 - 1);
        % T''(x)
        Tpp = Tp + 120/70*(1 - q^7);
        % T'''(x)
        Tppp = 3*(Tpp - Tp) + 2400/1080*(q^9 - 1);

    elseif abs(x-1) < 1e-2 % near-parabolic; compute with
       series
        % evaluate sigma
        [sig1, dsigdx1, d2sigdx21, d3sigdx31] = sigmax(-E);
        [sig2, dsigdx2, d2sigdx22, d3sigdx32] = sigmax(-E*q*q);
        % T(x)
        T = sig1 - q^3*sig2;
        % T'(x)
        Tp = 2*x*(q^5*dsigdx2 - dsigdx1);
```

```
        % T''(x)
        Tpp = Tp/x + 4*x^2*(d2sigdx21 - q^7*d2sigdx22);
        % T'''(x)
        Tppp = 3*(Tpp-Tp/x)/x + 8*x*x*(q^9*d3sigdx32 -
           d3sigdx31);

    else % all other cases
        % compute all substitution functions
        y  = sqrt(abs(E));
        z  = sqrt(1 + q^2*E);
        f  = y*(z - q*x);
        g  = x*z - q*E;

        % BUGFIX: (Simon Tardivel) this line is incorrect for E
           ==0 and f+g==0
        % d  = (E < 0)*(atan2(f, g) + pi*m) + (E > 0)*log( max
           (0, f + g) );
        % it should be written out like so:
        if (E<0)
            d = atan2(f, g) + pi*m;
        elseif (E==0)
            d = 0;
        else
            d = log(max(0, f+g));
        end

        % T(x)
        T = 2*(x - q*z - d/y)/E;
        %  T'(x)
        Tp = (4 - 4*q^3*x/z - 3*x*T)/E;
        % T''(x)
        Tpp = (-4*q^3/z * (1 - q^2*x^2/z^2) - 3*T - 3*x*Tp)/E;
        % T'''(x)
        Tppp = (4*q^3/z^2*((1 - q^2*x^2/z^2) + 2*q^2*x/z^2*(z -
            x)) - 8*Tp - 7*x*Tpp)/E;

    end
end

% series approximation to T(x) and its derivatives
% (used for near-parabolic cases)
function [sig, dsigdx, d2sigdx2, d3sigdx3] = sigmax(y)

    % preload the factors [an]
    % (25 factors is more than enough for 16-digit accuracy)
```

```
    persistent an;
    if isempty(an)
        an = [
            4.000000000000000e-001;     2.142857142857143e-001;
                    4.629629629629630e-002
            6.628787878787879e-003;     7.211538461538461e-004;
                    6.365740740740740e-005
            4.741479925303455e-006;     3.059406328320802e-007;
                    1.742836409255060e-008
            8.892477331109578e-010;     4.110111531986532e-011;
                    1.736709384841458e-012
            6.759767240041426e-014;     2.439123386614026e-015;
                    8.203411614538007e-017
            2.583771576869575e-018;     7.652331327976716e-020;
                    2.138860629743989e-021
            5.659959451165552e-023;     1.422104833817366e-024;
                    3.401398483272306e-026
            7.762544304774155e-028;     1.693916882090479e-029;
                    3.541295006766860e-031
            7.105336187804402e-033];
    end

    % powers of y
    powers = y.^(1:25);

    % sigma itself
    sig = 4/3 + powers*an;

    % dsigma / dx (derivative)
    dsigdx = ( (1:25).*[1, powers(1:24)] ) * an;

    % d2sigma / dx2 (second derivative)
    d2sigdx2 = ( (1:25).*(0:24).*[1/y, 1, powers(1:23)] ) * an;

    % d3sigma / dx3 (third derivative)
    d3sigdx3 = ( (1:25).*(0:24).*(-1:23).*[1/y/y, 1/y, 1,
       powers(1:22)] ) * an;

end


%
   ------------------------------------------------------------------

% Helper functions
```

```
%
    ------------------------------------------------------------------


% compute minimum and maximum distances to the central body
function extremal_distances = minmax_distances(r1vec, r1,...
                                               r2vec, r2,...
                                               dth,...
                                               a,...
                                               V1, V2,...
                                               m,...
                                               muC)

    % default - minimum/maximum of r1,r2
    minimum_distance = min(r1,r2);
    maximum_distance = max(r1,r2);

    % was the longway used or not?
    longway = abs(dth) > pi;

    % eccentricity vector (use triple product identity)
    evec = ((V1*V1.')*r1vec - (V1*r1vec.')*V1)/muC - r1vec/r1;

    % eccentricity
    e = sqrt(evec*evec.');
    % apses
    pericenter = a*(1-e);
    apocenter  = inf;                    % parabolic/hyperbolic
        case
    if (e < 1), apocenter = a*(1+e); end % elliptic case

    % since we have the eccentricity vector, we know exactly
       where the
    % pericenter lies. Use this fact, and the given value of [
       dth], to
    % cross-check if the trajectory goes past it
    if (m > 0) % obvious case (always elliptical and both apses
        are traversed)
        minimum_distance = pericenter;
        maximum_distance = apocenter;
    else % less obvious case
        % compute theta1&2 ( use (AxB)-(CxD) = (C B)(D A) - (
           C A)(B D) ))
        pm1 = sign( r1*r1*(evec*V1.') - (r1vec*evec.')*(r1vec*
           V1.') );
```

```
        pm2 = sign( r2*r2*(evec*V2.') - (r2vec*evec.')*(r2vec*
            V2.') );
        % make 100.4% sure it's in (-1 <= theta12 <= +1)
        theta1 = pm1*acos( max(-1, min(1, (r1vec/r1)*(evec/e)
            .')) );
        theta2 = pm2*acos( max(-1, min(1, (r2vec/r2)*(evec/e)
            .')) );
        % points 1&2 are on opposite sides of the symmetry axis
            -- minimum
        % and maximum distance depends both on the value of [
            dth], and both
        % [theta1] and [theta2]
        if (theta1*theta2 < 0)
            % if |th1| + |th2| = turnangle, we know that the
                pericenter was
            % passed
            if abs(abs(theta1) + abs(theta2) - dth) < 5*eps(dth
                )
                minimum_distance = pericenter;
            % this condition can only be false for elliptic
                cases, and
            % when it is indeed false, we know that the orbit
                passed
            % apocenter
            else
                maximum_distance = apocenter;
            end
        % points 1&2 are on the same side of the symmetry axis.
            Only if the
        % long-way was used are the min. and max. distances
            different from
        % the min. and max. values of the radii (namely, equal
            to the apses)
        elseif longway
            minimum_distance = pericenter;
            if (e < 1), maximum_distance = apocenter; end
        end
    end

    % output argument
    extremal_distances = [minimum_distance, maximum_distance];

end
```

# References


[1] Evgeny Kuzmin, Dmitry Goritsyn, and Irina Khasanova. Colonization of Mars. `https://mars-colony.life/`. Accessed: 10/07/2021.

[2] Delft University Aerospace Engineering. Spacecraft engineering: Delta-v (velocity increment) budget. `http://lr.tudelft.nl`. Accessed: 05/07/2021.

[3] Howard D. Curtis. *Orbital Mechanics for Engineering Students*. Elsevier, 2010.

[4] José Antonio Moraño Fernández. Orbits, satellites and relativity notes. 2020.

[5] Guido Colasurdo. *Astrodynamics*. Politecnico di Torino, 2006.

[6] OpenStax College. *Algebra and Trigonometry*. OpenStax CNX.

[7] Brent Barbee. Mission planning for the mitigation of hazardous near Earth objects. 2021.

[8] Alessandro Zavoli. Orbital mechanics notes. 2020.

[9] Denilson Paulo Souza dos Santos, Antônio Fernando Bertachini de Almeida Prado, and Guido Colasurdo. Four-impulsive rendezvous maneuvers for spacecraft in circular orbits using genetic algorithms. *Mathematical Problems in Engineering*, 2012.

[10] José Tatay Sangüesa. Multi-objective optimisation of impulsive orbital trajectories. *Final Bachelor's Degree Project*, 2019.

[11] Xin-She Yang. Genetic algorithms. In *Nature-Inspired Optimization Algorithms*, chapter 6, pages 91–100. Academic Press, second edition, 2021.

[12] Ryan Champlin. Selection methods of genetic algorithms. *Student Scholarship - Computer Science*, 2018.

[13] Xiao-Bing Hu and Ezequiel Di Paolo. An efficient genetic algorithm with uniform crossover for air traffic control. *Computers operations research*, pages 245–259, 2009.

[14] A.E. Eiben and Jim E. Smith. *Introduction To Evolutionary Computing*, volume 45. 2003.

[15] William Crossley, Kamlesh Nankani, and Daniel Raymer. Comparison of bit-string affinity and consecutive generation stopping criteria for genetic algorithms. $42^{nd}$ *AIAA Aerospace Sciences Meeting and Exhibit*, 2004.

[16] Alireza Alinezhad, Abolfazl Kazemi, and Mojgan Khorasani. Presenting a model for decoupling points in supply chain networks. *International Journal of Logistics Systems and Management*, 33:383–403, 2019.

[17] George Mavrotas. Effective implementation of the $\varepsilon$-constraint method in multi-objective mathematical programming problems. *Applied mathematics and computation*, pages 455–465, 2009.

[18] George Mavrotas and Kostas Florios. An improved version of the augmented $\varepsilon$-constraint method (AUGMECON2) for finding the exact pareto set in multi-objective integer programming problems. *Applied mathematics and computation*, pages 9652–9669, 2013.

[19] Dario Izzo. Revisiting Lambert's problem. *Celestial Mechanics and Dynamical Astronomy*, 2014.

[20] David Ottesen and Ryan P. Russell. Unconstrained spacecraft trajectory optimization using embedded boundary value problems. *AAS/AIAA Astrodynamics Specialist Conference*, 2019.

[21] Ryan P. Russell. On the solution to every Lambert problem. *Celestial Mechanics and Dynamical Astronomy*, 2019.

[22] Rody Oldenhuis. Robust solver for Lambert's orbital-boundary value problem. `https://nl.mathworks.com/matlabcentral/fileexchange/26348`. Accessed: 25/01/2021.

[23] California Institute of Technology Jet Propulsion Laboratory. HORIZONS interface. `https://ssd.jpl.nasa.gov/horizons.cgi`. Accessed: 20/01/2021.

[24] P.wormer. Spherical polar coordinates. `https://upload.wikimedia.org/wikipedia/commons/7/75/Spherical_polar_coordinates.png`. Accessed: 06/06/2021.

[25] William Crossley. Genetic algorithm introduction. *AAE 55000: Multidisciplinary Design Optimization*, 2018.

[26] Jürgen Giesen. Hohmann transfer orbit applet. `http://www.jgiesen.de/hohmann/`. Accessed: 09/06/2021.

[27] David P. Stern. Flight to Mars: calculations. `http://www.phy6.org/stargaze/Smars2.htm`. Accessed: 09/06/2021.

[28] NASA's Science Mission Directorate. Mars exploration program historical log. `https://mars.nasa.gov/mars-exploration/missions/historical-log/`. Accessed: 20/06/2021.

[29] Acer. Swift 3 technical specifications. `https://www.acer.com/ac/es/MX/content/model/NX.GNUAL.014`. Accessed: 10/07/2021.

[30] Selectra. ¿cuál es el precio del kWh en mercado regulado? `https://selectra.es/energia/info/que-es/precio-kwh`. Accessed: 10/07/2021.

[31] Agencia Tributaria. Tabla de coeficientes de amortización lineal. `https://www.agenciatributaria.es/AEAT.internet/Inicio/_Segmentos_/Empresas_y_profesionales/Empresas/Impuesto_sobre_Sociedades/Periodos_impositivos_a_partir_de_1_1_2020/Base_imponible/Amortizacion/Tabla_de_coeficientes_de_amortizacion_lineal_.shtml`. Accessed: 10/07/2021.

[32] Universitat Politècnica de València. Normativa marco de trabajos fin de grado y fin de máster. 2013.