UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeries
Industrial i Aeronàutica de Terrassa

Institut Supérieur de l'Aéronautique et de l'Espace
SUPAERO

Validation and visualization of trajectories computaiton

Escola Superior d'Enginyeria Industrial, Aeroespacial

i Audiovisual de Terrassa

-

Institut Supérieur de l'Aéronautique

et de l'Espace – SUPAERO

Màster Universitari en Enginyeria Aeronàutica

Report

Author: **Albert Guillén Buisan**

Director: **Stéphanie Lizy-Destrez**

June 2018

Abstract

This project describes and implements a method to visualize different types of spacecraft trajectories by using commercial open software from well-known public enterprises in the space industry. In order to do so, a preliminary state of the art is done to determine which are the visualization tools best fitting the needs depending on the constraints in each situation. Once selected the tools with which the visualizations will be performed, a brief chapter is dedicated to explain how spacecraft trajectories must be generated in order to be compatible with the software, as well as which are the essential files to run the simulations properly.

Afterwards, several trajectories are generated and visualized regarding the Keplerian motion to illustrate the influence of orbital parameters when designing an orbit and other scenarios like manoeuvres and heliosynchronism. Then, some simulations have been created regarding the circular restricted three body problem (CR3BP) to illustrate non-linear dynamics around Lagrangian points, concretely Halo, NRO, DRO and vertical Lyapunov orbits around L1 and L2 points in the Earth – Moon system.

List of contents

List of figures

List of tables

Abbreviations

CCSDS	Consultative Committee for Space Data Systems
CIC	Centre d'Ingénierie Concourante
LEO	Low Earth Orbit
MEO	Medium Earth Orbit
GEO	Geostationary Earth Orbit
SSO	Sun Synchronous Orbit
CNES	Centre National d'Études Spatiales
NASA	National Aeronautics and Space Administration
RAAN	Right Ascension of the Ascending Node
JD	Julian Date
CR3BP	Circular Restricted 3 Body Problem
MJD	Modified Julian Date
NRO	Nearly Rectilinear Orbit
DRO	Distant Retrograde Orbit
VLYAP	Vertical Lyapunov Orbit

1 Aim of the project and objectives

The main objective of this project is to validate and visualize spacecraft trajectories in two body problem and three body problem.

At the end of the project it must be possible to produce some videos and animations to illustrate space mechanics theory and other research activities being held in the department.

In order to do so, other preliminary objectives shall be fulfilled. One of these objectives is to find a polyvalent visualization tool that permits visualizing all types of trajectories, Keplerian or non-Keplerian, depending on the scenario's requirements as well.

Another objective once the tool is selected is to establish and implement a method to generate and export spacecraft trajectories so that they are compatible with the chosen software and the visualization is successful.

2 State of the art of visualization tools

First of all, a state of the art of the different existing visualization tools for orbits and other space applications has been performed. The strengths and weaknesses as well as easy access to each visualization tool have been analysed regarding our needs in order to find the one which fits best in each case. This also means that only open source software has been considered in the study.

It is interesting to note that all visualization tools evaluated in this section have been developed by major enterprises from the space industry like CNES and NASA, so reliability and validity of preloaded data like ephemerides, celestial bodies' characteristics is ensured.

Another important aspect that has been taken into account and it is quite important for all space related works is to check that all these tools comply with all the CCSDS standards. Therefore, tools not complying with this requirement have not been taken into account in this analysis.

An initial list of applications has been created after an extensive research. This list is composed of the following applications:

- Celestia
- GMAT
- Cosmographia
- VTS
- Celestlab
- Orekit

2.1 Celestia

Celestia is a three dimension software that simulates many different types of celestial objects, from planets and moons to star clusters and galaxies. You can visit every object in the expandable database and view it from any point in space and time. The position and movement of solar system objects is calculated accurately in real time at any rate desired.

What it is really interesting to fulfil the objectives of this project is that Celestia is designed in a way that allows the user to interact with the solar system objects and to add new ones. So, using Celestia it will be possible to add new spacecraft

with their respective trajectories calculated previously and to visualize them in the solar system in this case [1].

Hereafter, a summarised list of the most relevant characteristics from Celestia is presented:

- Celestia respects the CCSDS protocol established.
- Celestia supports different types of trajectory data. Sampled Orbits for example can be used to represent spacecraft designed trajectories, or you can use NASA's SPICE kernels for various solar system objects or defining an orbit by defining its orbital parameters.
- It uses .xyz or .xyzv files as input for trajectories' data. Note that difference between both file extensions is to include velocity fields or not in the orbit discretization. (describe document format, reference frames, time format, etc)
- Scenario's configuration must be defined in a .scc file.
- In the .scc file the reference frame can be modified depending on the needs. For example, a customized two-vector reference frame between Earth and Moon. Ideal for representing Halo, Lissajous or Lyapunov orbits around Lagrangian points and respective transfers from LEO orbits.
- It is in the .ssc file where it is possible to define a custom orbit referencing the corresponding .xyz(v) file or just define an elliptical, circular or parabolic orbit by defining its main orbital parameters like eccentricity, epoch, semi-major axis, inclination and period.

2.2 Cosmographia

This is an interactive tool used to produce 3D visualizations of planet ephemerides, sizes, shapes, spacecraft trajectories and orientations. Cosmographia has many user controls, allowing one to manage what is displayed, what vantage point is used, and how fast the animation progresses.

As Cosmographia is a tool developed by NASA as well as Celsestia is, they presents lots of similarities. In fact, they even share several functionalities when

importing trajectories and load different scenarios. Then, as they are quite similar tools it is nonsense to use both. It may be noted then that Celestia has been more used in recent years in applications similar to what has been done in this project and presents higher reliability and background than Cosmographia. This is the reason why finally Cosmographia has been discarded as the tool to be used to represent our trajectories.

2.3 VTS

VTS is a tool created and designed by CNES that enables to produce satellite animations in 2D or 3D environments. The architecture of VTS is designed as an extensible platform, able to work with an unlimited number of compatible applications. One of these compatible applications is Celestia, which is its main 3D visualization tool. Celestia, as explained before, is an open source space visualization software, widely recognized for its performance and its graphical quality [2]. All of Celestia's features described above and others are also available in VTS.

The first part of VTS is the configuration tool. This application is used to define elements of the visualization scene: 3D models, satellite geometries, mobile parts, data sources for position, attitude, rotation angle, etc. It allows configuration of ground stations, onboard sensors, etc. Client applications participating in the visualization session are also defined here. VTS then uses this configuration to start the animation.

The architecture of VTS is designed to allow connected applications to take control of the time flow in the visualization. This way, VTS can be driven by a simulator, a plotting application, etc. Data, such as satellites position or orientation, can be read from constant values, from files, or from the network. In the case of this project, satellites position has been read from local files where the trajectory has previously been generated. The data file format, which has been defined according to the CIC protocol [3], is based on the CCSDS specification, facilitating interoperability with all software dealing with the European standard.

With these characteristics, VTS becomes a helpful tool for all activities implying space data like a graphical validation method, as an educational tool, or as a communication support.

Hereafter, a summarised list of the most relevant characteristics from VTS is presented:

- VTS respects the CCSDS protocol stablished.
- VTS supports different types of trajectory data, from own designed trajectories to standard Keplerian orbits.
- It uses .txt files as input for trajectories' data (describe document format, reference frames, time format etc)
- Simple Keplerian orbits can be generated easily using an integrated Keplerian generator, no need to design some orbits externally
- Possibility to use predefined cameras during the visualization easily, no programming knowledge is needed to change point of view.
- No possibility to change the reference frame. All trajectories need to be expressed in an EME2000 frame.
- Possibility to plot orbit's ground track using an integrated client application.
- Extensive support documentation regarding usage of the software and description of file's format to be used.
- Satellite attitude can also be defined and loaded in files.

2.4 GMAT, Celestlab, Orekit

Other well-known tools in space industry have been taken into account in this analysis like GMAT, Celestlab and Orekit. These tools are really powerful in terms of orbit computation, propagation and calculations in general. However,

when talking about visualization they are not as powerful as Celestia or VTS. In fact, it is clear that they are not designed for this purpose.

Therefore, the conclusion that can be extracted regarding these three tools is that they are no longer a valid option to visualize spacecraft trajectories in the frame of the development of this project.

2.5 Selected tools

Finally, after analysing all tools, VTS and Celestia have been selected to be used depending on the specific needs of each scenario to visualize. It is clear that for those situations where it is useful to have a view of the orbit's ground track, VTS will be used, whereas for those situations where the EME2000 coordinate reference system is not appropriate Celestia will be used. For other situations not applying to these requirements VTS will normally be used due to its simplicity and easy management.

It is interesting to note that the use of the post-process software Camtasia is foreseen in order to include descriptions and other info needed to help with the easy learning and to give visualizations an even more professional view.

3 Trajectories generation and compatibility

Once taken the decision of which visualization tools will be employed, it is time to see with more detail how the studied trajectories are going to be generated and properly imported in order to be compatible with the chosen visualization software, whether VTS or Celestia. The procedure followed is quite similar in both cases but some divergences exist depending on the tool used that will be explained hereafter.

3.1 Compatibility

There is one specification that all selected tools share and this fact will facilitate using one or the other depending on the needs of each particular trajectory. This feature is that both VTS and Celestia use .xyz or .txt files as the main input for visualising trajectories. Although initially it seems that they are different file extensions, the content inside them is the pretty the same in both cases. It is common to deal with .txt files, but a .xyz file can be defined as an ASCII file containing a list of time stamps and position records (or optionally, positions with velocities).

This list is composed of sets of numbers where each set of numbers consists of date information (a TDB Julian date) followed by x, y and z positions measured in kilometres. They are positions within the coordinate system associated with the object. Additionally, if it is wanted that velocities are taken into account, they shall be attached after position values expressed in kilometres per second. Then, with all written states, the trajectory is built using interpolation methods between states so that error is minimized, giving as a result a continuous path.

The states defining the trajectory shall be presented as shown in Table 1, where each row represents one state. Note that bold values are compulsory whilst the others, corresponding to velocity data, are optional.

Table 1 - Trajectory file's format

Date 1	X_1	Y_1	Z_1	Vx_1	Vy_1	Vz_1
Date 2	X_2	Y_2	Z_2	Vx_2	Vy_2	Vz_2
Date 3	X_3	Y_3	Z_3	Vx_3	Vy_3	Vz_3
⋮	⋮	⋮	⋮	⋮	⋮	⋮
Date n-1	X_{n-1}	Y_{n-1}	Z_{n-1}	Vx_{n-1}	Vy_{n-1}	Vz_{n-1}
Date n	X_n	Y_n	Z_n	Vx_n	Vy_n	Vz_n

Regarding date information required to define a new state of the trajectory, there are some differences when setting its format depending on the visualization tool being used, VTS or Celestia, but in both cases are all based on the Julian calendar concept.

Celestia uses Julian Date format to define each state in the list that conforms the trajectory. The Julian Date of any instant is defined as the Julian day number plus the fraction of a day since the preceding noon. On the other hand, VTS uses Modified Julian Date (MJD) format to set each state in the list that conforms the trajectory. The Modified Julian Date is achieved by applying the following relation with the Julian Date [4]:

$$MJD = JD - 240000{,}5$$

It shall be noted that VTS selected format uses two separate fields, a first one that indicates the undivided number of days since the established reference, and a second to express the number of seconds within that day. Note that the second term it shall be smaller than 86400, which are the seconds in a day, whilst the first term it is not bounded.

In Table 2, an example of the date formats used in trajectory data source files for Celestia and VTS is presented. In order to see differences between them, the same date is expressed in a common calendar date format and then in Julian Date format, employed in Celstia, and Modified Julian Date, employed in VTS.

Table 2 - Date formats used

Calendar Date	Julian Date	Modified Julian Date
29 - Nov- 2018 06:00:00	2458451,75	54451 21600

3.2 Generation

Once explained the requirements that files containing trajectory's data must comply with to be compatible with the chosen software, it is time to establish the way these trajectories will be generated.

An advantage of VTS that has already been mentioned in previous sections is that it comes with an integrated Keplerian orbits generator. As it can be observed in Figure 1, by setting values to de various orbital parameters that appear as well as start and end date for the visualization, it generates a .txt file with the described format containing the trajectory data.

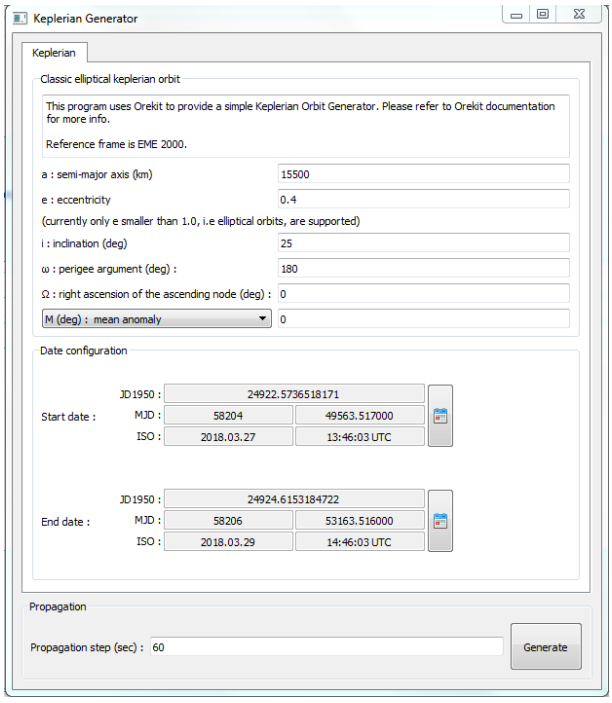

Figure 1 - VTS Keplerian Generator

This generator is a fast and simple way to generate orbits to be visualized. However, it presents some limitations not allowing achieving the goals of this project. The observed limitations are that there is only the possibility to produce simple Keplerian orbits, not being possible to create for example parabolas or hyperbolas. What is also not possible is to do is to generate non-Keplerian orbits, like Halo orbits around Lagrangian points for example.

Therefore, a need arises to solve these limitations, which is to find a way to generate these orbits externally. Due to short-time limitation during this project Matlab has been selected as the tool to generate these trajectories since other software like Orekit or Celestlab maybe could be more powerful but there was not time to be trained to, and Matlab already could solve the needs.

Apart from generating the trajectories itself in Matlab, some code has been generated to export the created trajectories respecting the needed format. It should be noted that two options will be possible taking into account the divergences in date format from VTS and Celestia explained in previous sections.

4 Implementation of Keplerian motion

In this section, the method explained to generate and visualize orbits has been implemented regarding Keplerian motion. It is interesting to note that, as the various orbits to be visualized as part of this section can properly be expressed in an EME2000 frame, all illustrations in this section have been visualized using VTS, due to its practical way to change point of views, simulation time management and dealing with different scenarios. Moreover, when exporting the visualization to a video format there are also advantages in comparison to Celestia exporting mode.

Regarding orbit generation process, the vast majority of them have been generated using the integrated Keplerian generator, but there are some that have been generated using Matlab scripts. In each subsection it will be specified which method has been employed.

4.1 Illustration of Kepler's laws

The first scenarios to illustrate regarding Keplerian motion are related to Kepler's law, concretely first and second laws. In both cases, the orbits have been generated using the integrated VTS Keplerian generator due to its simplicity as mentioned before.

First Kepler's Law

This illustration represents the first law of Kepler that says that the orbit of every planet is an ellipse with the Sun at one of the two foci, but here the planet is represented by a satellite and the Earth represents the Sun located at one of the foci. The orbit then is an elliptical equatorial orbit where it is possible to observe the kinetic momentum perpendicular to the plane of motion. The orbital parameters of the orbit are summarized in Table 3.

Table 3 - Orbital parameters to illustrate Kepler's laws

Semi-major axis	a = 30000 km
Eccentricity	e = 0,7
Inclination	i = 0°

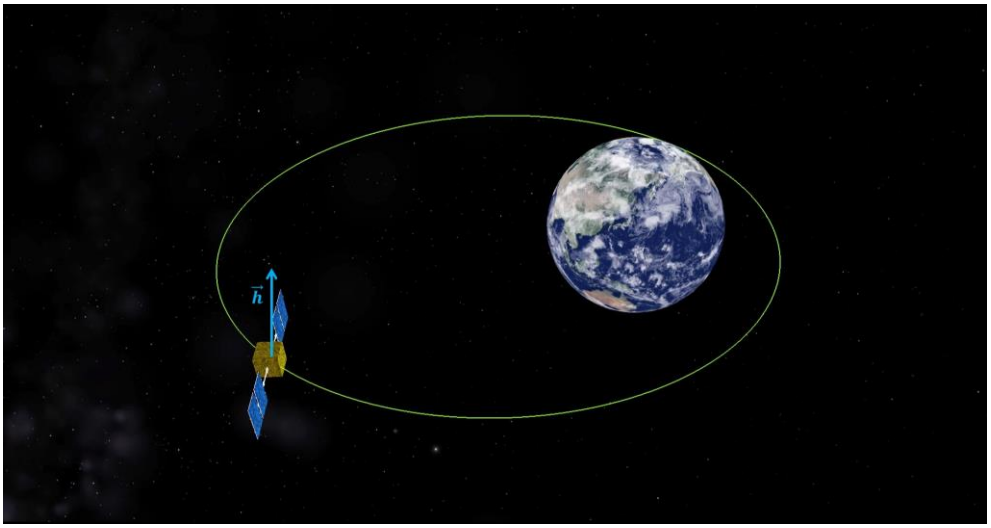

Figure 2 - Illustration of first Kepler's law

Second Kepler’s Law

Regarding the second law of Kepler, the illustrated orbit presents the same orbital parameters as in the illustration of the first law and they have already been presented in Table 3. During the visualization it is possible to evidence the second law of Kepler that says that a line joining the satellite orbiting the Earth and the Earth itself sweeps out equal areas in equal intervals of time. An elliptical orbit with high eccentricity value has been chosen to obtain a more educational result.

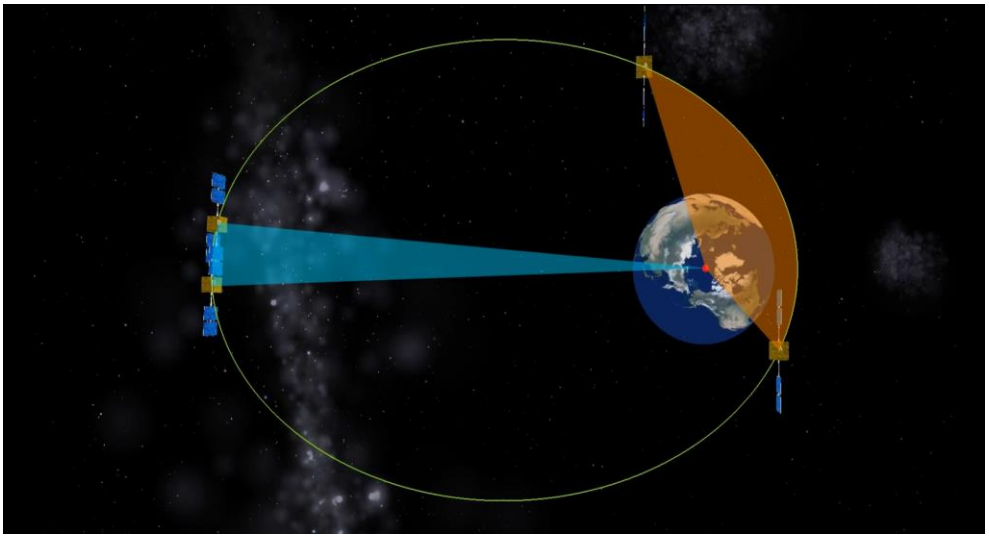

Figure 3 - Illustration of second Kepler's law

As seen in Figure 3, the blue and orange areas are equal and they have been obtained during the same interval of time in both cases. This phenomenon can be really well observed in the produced animation.

4.2 Illustration of the influence of orbital parameters

This section has been done to illustrate the influence of each orbital parameter when designing an orbit around a celestial body. In order to do so, several animations have been created, at least one for each orbital parameter studied. Each animation has all orbital parameters fixed except one which is increased progressively. This allows observing easily the influence of each parameter on the orbit's shape and if considered relevant, the track on the Earth's surface.

Inclination

A single animation has been created with VTS integrated Keplerian generator to illustrate the effect of changing the inclination when designing a satellite orbit around the Earth. The inclination of an orbit is defined as the angle between a reference plane and the orbital plane or axis of direction of the orbiting object. In the case of the Earth, the reference plane is the plane containing the equator's line.

Table 4 - Orbital parameters to visualize inclination's influence

Semi-major axis	a = 12000 km
Eccentricity	e = 0.3
Inclination	i = [0, 90]
Perigee Argument	ω = 180°
Ascending Node	Ω = 0°

In Table 4, the values of the orbital parameters chosen to illustrate the effect of changing the inclination are presented. It consists of a MEO elliptical orbit around the Earth. Its inclination increases progressively from 0º to 90º in intervals of 10ª: In other words the orbit evolves from an equatorial orbit to a polar one. In Figure 4, the effect of changing the inclination is evidenced in both 3D view and surface

view where orbits' ground tracks can be compared. As the orbit's inclination increases, the amplitude of the ground track in vertical direction increases.

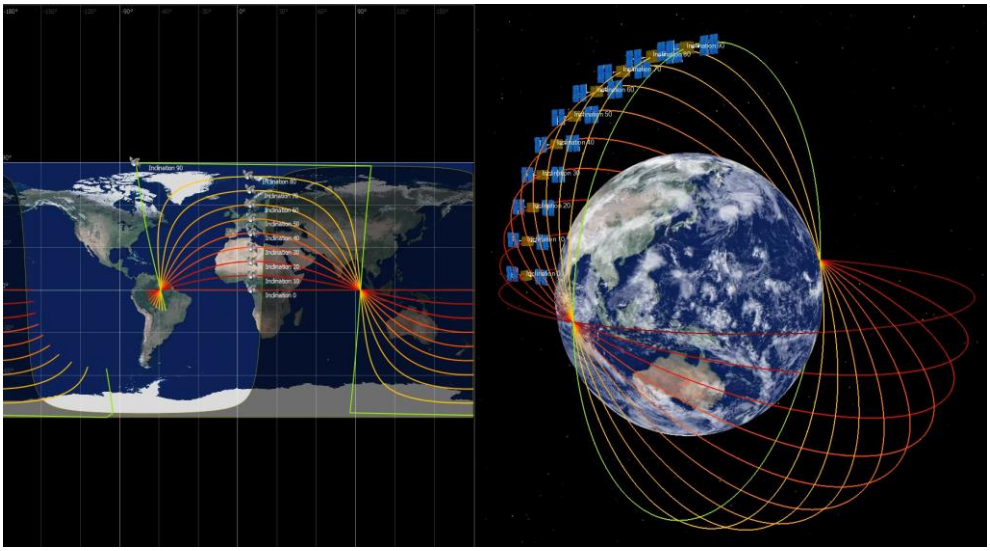

Figure 4 - Inclination influence

Semi-major axis

In order to illustrate the influence of semi-major axis magnitude in the shape of an orbit by keeping other parameters constant, two animations using VTS integrated Keplerian generator have been done. The major axis of an ellipse is its longest diameter found. Therefore the semi-major axis is one half of the major axis, starting from the centre to the perimeter passing through one focus. Note that when talking about circular orbits the semi-major axis is the radius.

Table 5 - Orbital parameters to visualize semi-major axis' influence

Semi-major axis	a = [15500, 29000] km
Eccentricity	e = 0.4
Inclination	i = 25°
Perigee Argument	ω = 180°
Ascending Node	Ω = 0°

In this case, several orbits have been visualized to evidence what changes when modifying the magnitude of semi-major axis. It may be noted that all orbits are coplanar, have same argument of perigee, same longitude of the ascending node but the semi major axis evolves from 15500 km to 29000 km in intervals of 1500 km as shown in Figure 5 and Table 5.

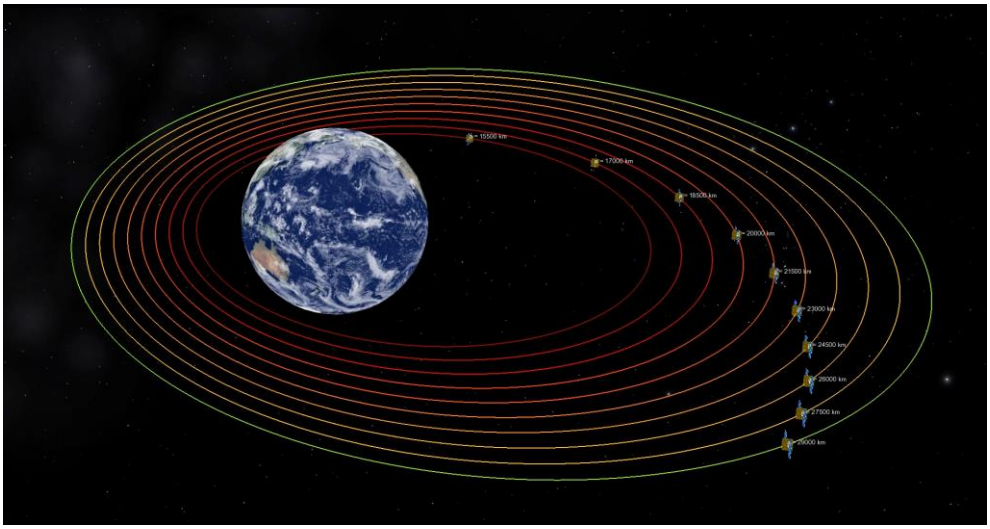

Figure 5 – Semi-major axis influence

Analysing the obtained result where an offset exists between satellites from the different orbits, it is easy to observe that as the semi-major axis increases, the orbital period gets longer. This fact also evidences the third Kepler's law, which states that the square of the orbital period is directly proportional to the cube of the semi-major axis.

Another scenario has been created regarding the effect of changing the semi-major axis of an orbit. In fact, this scenario shows what can be done by combining different values of inclinations and semi-major axis. As shown in Table 6, the visualization is conformed by two MEO and one GEO orbit.

Table 6 - Orbital parameter of MEO and GEO orbits

	MEO 1	**MEO 2**	**GEO**
Semi-major axis	a = 20000 km	a = 20000 km	a = 42157 km
Eccentricity	e = 0	e = 0	e = 0
Inclination	i = 30°	i = 0°	i = 0°

As it can be seen in Figure 6, a geostationary orbit can be achieved by setting the inclination to 0º, in other words having an equatorial orbit, and at same time setting the semi-major axis to the value that makes the period of the orbit of one day. This fact is also reflected in the ground track view of the orbits where the satellite corresponding to the geostationary orbit remains always on the same point of the projection on the Earth surface.

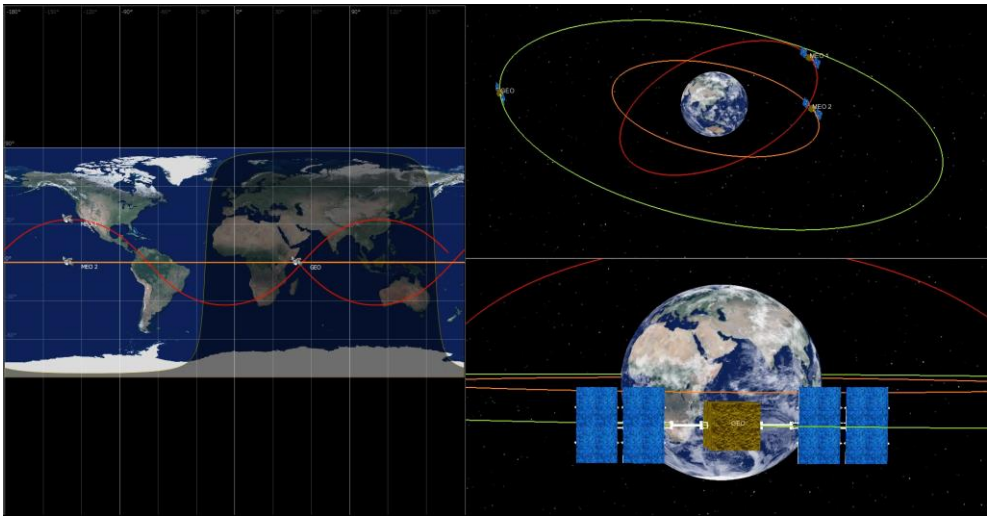

Figure 6 - MEO and GEO orbits

Eccentricity

Two animations have been created to evidence the effect of changing the eccentricity of an orbit by keeping the other parameters constant. The eccentricity of an orbit is defined as a parameter that determines the amount by which its orbit around another body deviates from a perfect circle. So that the more close to zero is the eccentricity value, the more similar to a circle the orbit's shape is.

In the first visualization, the VTS integrated generator has been used to compute the orbits. As shown in Table 7, all orbital parameters are fixed except the eccentricity that evolves from 0 (red curve in Figure 7) to 0.8 (green curve in Figure 7) in intervals of 0.2. In other words, the orbit starts being a circle and as the eccentricity increases, the orbit gets a more elliptical shape.

Table 7 - Orbital parameters to visualize eccentricity's influence

Semi-major axis	a = 40000 km
Eccentricity	e = [0, 0.8]
Inclination	i = 0°
Perigee Argument	ω = 0°
Ascending Node	Ω = 0°

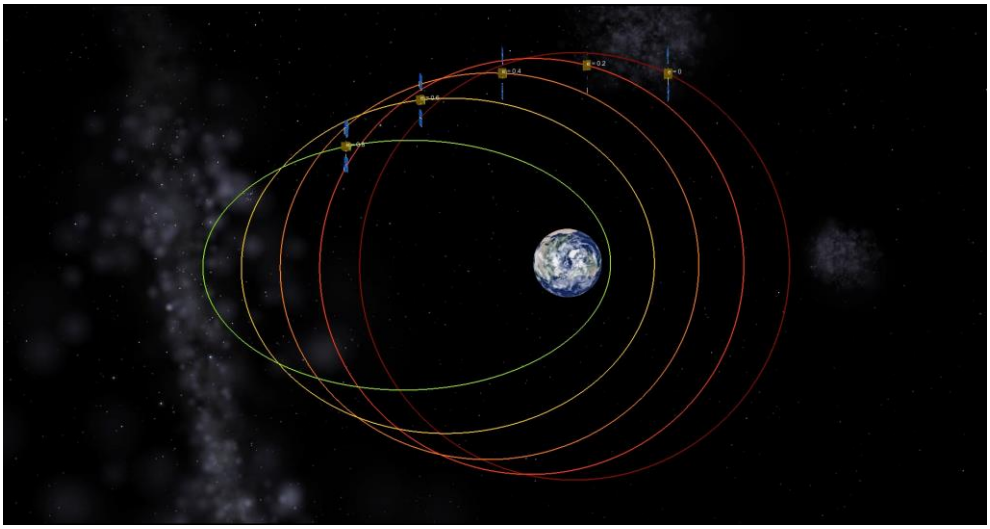

Figure 7 - Eccentricity influence I

A limitation of the VTS Keplerian generator has been found as it does not support eccentricity values equal or greater than 1. Therefore, a need to create another visualization arose, just to illustrate orbits with greater eccentricity values than that. The trajectories of this animation, Figure 8, have been visualized using VTS. However, the files containing the trajectories' data have been generated using Matlab scripts using a mathematical approach to generate the curves. To see more detail on how they are generated see Annex.

The result observed in Figure 8, is achieved by setting the orbital parameters as detailed in Table 8 and keeping fixed the same injection point of the orbit. It should be noted that parabolas only occur when eccentricity is equal to 1, whilst hyperbolas occur when larger values than that.

Table 8 - Orbital parameters to keep the same injection point

	Circle	Ellipse	Parabola	Hyperbola
Semi-major axis	a = 12000 km	a = 30000 km	-	a = 30000 km
Eccentricity	e = 0	e = 0.6	e = 1	e = 1.4
Inclination	i = 0°	i = 0°	i = 0°	i = 0°

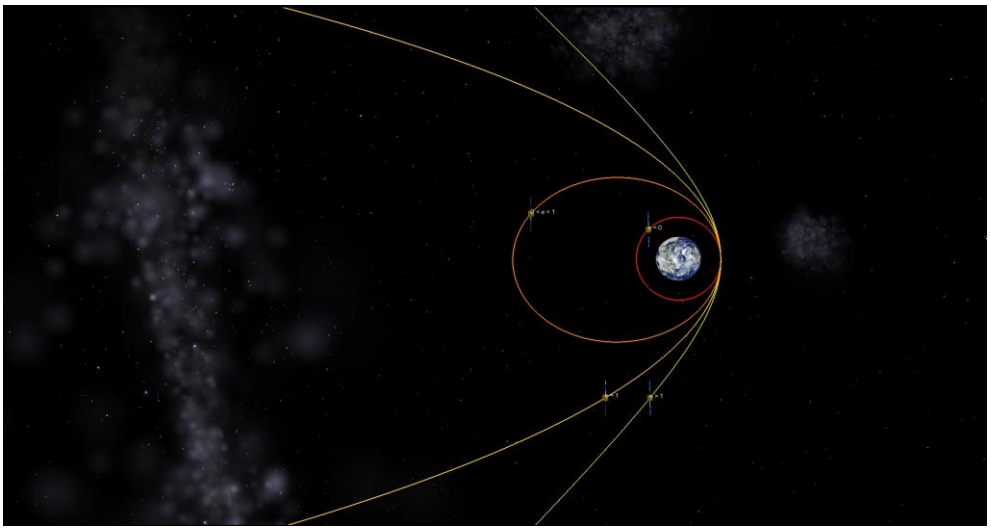

Figure 8 - Eccentricity influence II

Perigee's argument

In order to illustrate the influence of perigee's argument in the shape of an orbit by keeping other parameters constant, a single animation using VTS integrated Keplerian generator has been done. The characteristics of the generated orbits can be observed in Table 9, where all orbital parameters are fixed except the argument of perigee that changes from 0º to 360º.

The argument of perigee is defined as the angle from the body's ascending node to its perigee always being measured in the direction of motion. In other words, an argument of perigee of 0º would mean that the satellite will be at its closest approach to the central body at the same moment that it crosses the equatorial plane from southern to northern hemisphere.

Table 9 - Orbital parameters to visualize perigee argument's influence

Semi-major axis	a = 17000 km
Eccentricity	e = 0.5
Inclination	i = 33.4°
Perigee Argument	ω = [0, 360] °
Ascending Node	Ω = 0°

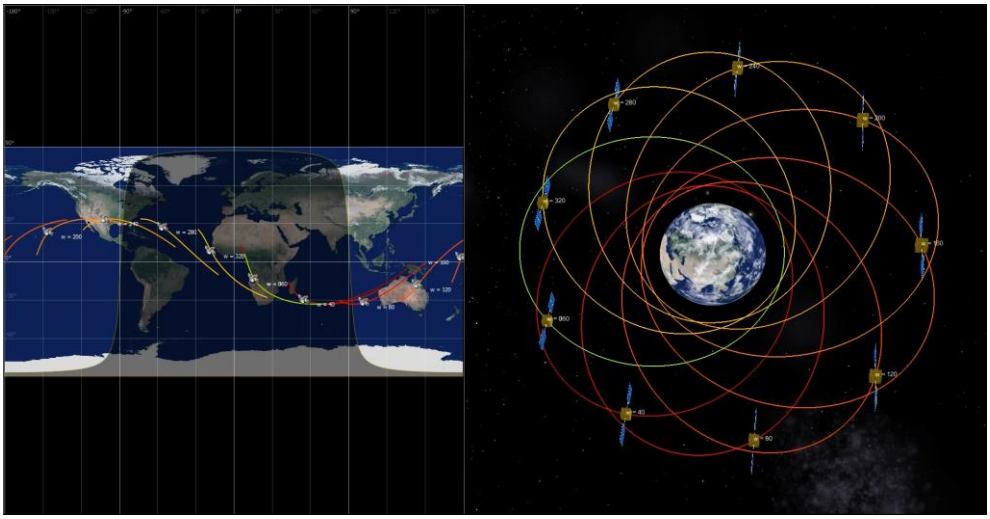

Figure 9 - Perigee argument influence

Longitude of the ascending node

In order to illustrate the influence of the longitude of the ascending node in the shape of an orbit by keeping other parameters constant, a single animation using VTS integrated Keplerian generator has been done. The characteristics of the generated orbits can be observed in Table 10, where all orbital parameters are fixed except the longitude of the ascending node that changes from 0º to 360º.

The longitude of the ascending node is defined as the angle from a reference direction to the direction of the ascending node measured in a reference plane. Note that the ascending node is the point where the orbit of the satellite in this case passes through the plane of reference. It should be highlighted that the angle is measured eastwards.

Table 10 - Orbital parameters to visualize ascending node's influence

Semi-major axis	a = 12000 km
Eccentricity	e = 0
Inclination	i = 20°
Perigee Argument	ω = 0°
Ascending Node	Ω = [0, 360] °

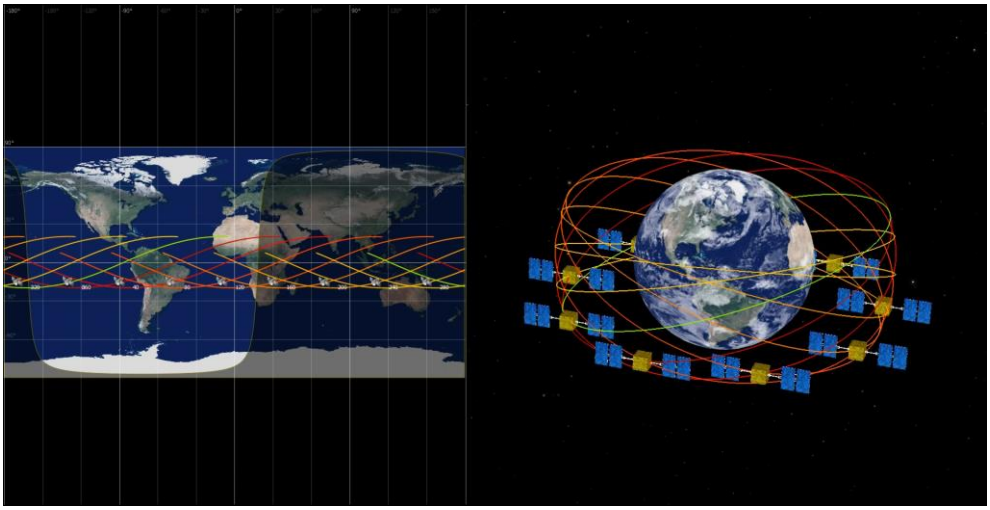

Figure 10 - Longitude of the ascending node variation's effect

This visualization was quite complicated to visualize because the effect was not clearly observed. Therefore, the decision to add the ground track view was taken to help with the comprehension of the phenomenon. The other facts that helped to the clear understanding of what is happening was to implement the visualization using circular orbits with a medium inclination value, since great or small values of inclination create a confusing view.

4.3 Illustration of types of maneuvers

Basic manoeuvre to change an orbit

This illustration is done to represent the effect of applying a coplanar engine impulse given an initial orbit. Therefore, it is possible to observe that a new orbit emerges from just applying a simple impulse. In this particular case, the engine impulse is applied to a circular orbit giving as a result an elliptical orbit. In other words, when applying the engine impulse, the velocity vector is modified in terms of direction and magnitude and consequently the orbit is no longer circular. Values of the main orbital parameters defining both orbits, before and after the impulse, are shown in Table 11.

Table 11 – Orbits characteristics for a basic change of orbit manoeuvre

	Initial Orbit	Final Orbit
Semi-major axis	a = 15000 km	a = 17000 km
Eccentricity	e = 0	e = 0,5
Inclination	i = 0°	i = 0°

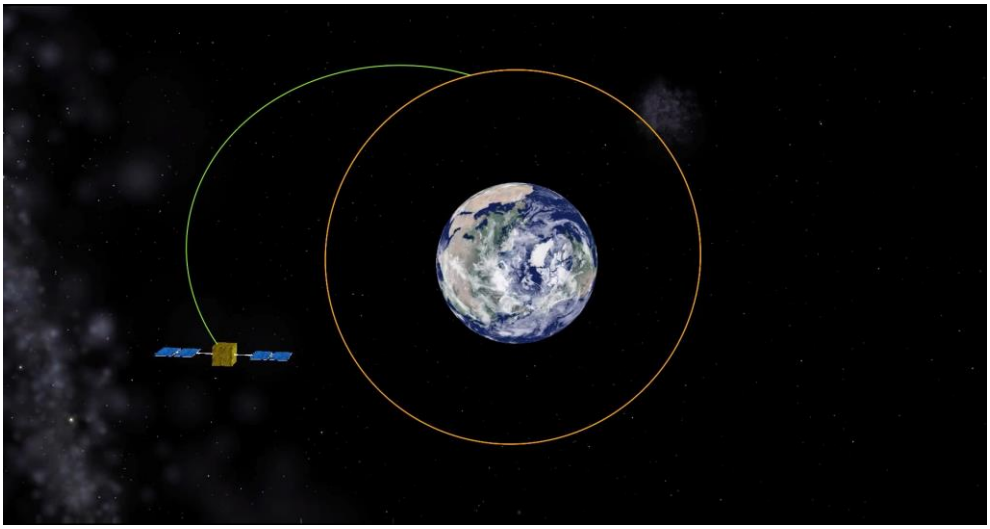

Figure 11 - Basic manoeuvre to change orbit

Hohmann Transfer

It is the most efficient manoeuvre in terms of ΔV magnitude to reach a target circular orbit departing from a coplanar circular orbit. This manoeuvre is composed of two engine impulses, one to put the spacecraft into the transfer orbit and the other to move it from the transfer orbit to the target circular orbit. It is interesting to mention that the transfer orbit is a semi ellipse and the two impulses occur on its perigee and apogee respectively.

The characteristics of all initial, final and transfer orbits are detailed in Table 12. As shown, in this particular case the transfer occurs from a MEO circular equatorial orbit to a geostationary orbit, explained in previous sections, through a semi-elliptical trajectory. The three orbits have been generated with integrated Keplerian generator and visualized using VTS.

Table 12 - Orbits characteristics for Hohmann transfer

	Initial Orbit	Final Orbit	Transfer Orbit
Semi-major axis	a = 12000 km	a = 42164 km	a = 27082 km
Eccentricity	e = 0	e = 0	e = 0.5569
Inclination	i = 0°	i = 0°	i = 0°

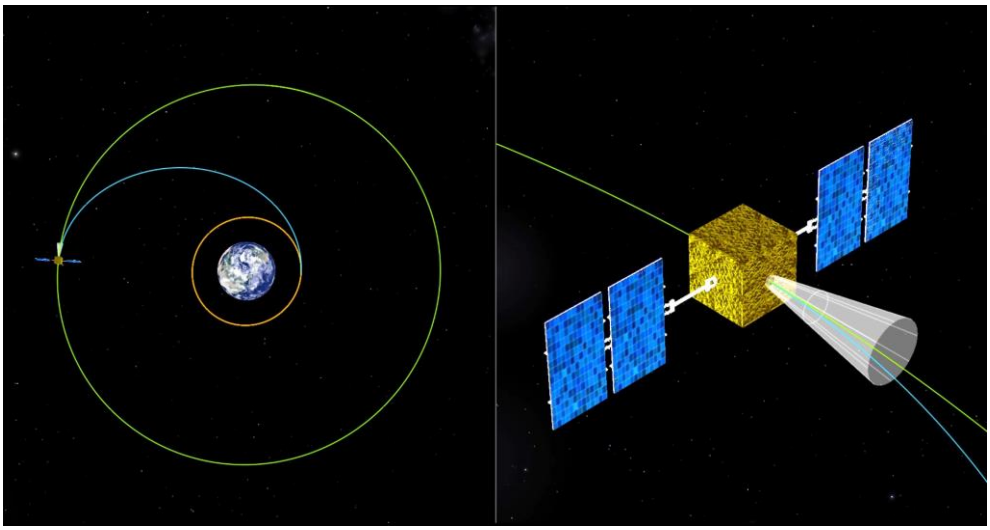

Figure 12 - Hohmann transfer manoeuvre

Manoeuvre to change the orbit's inclination

This particular manoeuvre enables to change the inclination of an orbit around the Earth just by applying a single engine impulse by keeping all the other orbital parameters the same. In order to do so, it may be noted that the impulse must be done when the satellite is crossing the reference plane, in this case the equatorial plane, or in other words when passing through the ascending node.

Regarding generation of orbits, due to simplicity issues they have been generated using integrated Keplerian generator and visualized in VTS software. It has been decided to also add the ground track view to evidence even more the change of inclination.

Table 13 - Orbits characteristics for inclination's change manoeuvre

	Initial Orbit	Final Orbit
Semi-major axis	a = 20000 km	a = 20000 km
Eccentricity	e = 0,5	e = 0,5
Inclination	i = 25°	i = 50°

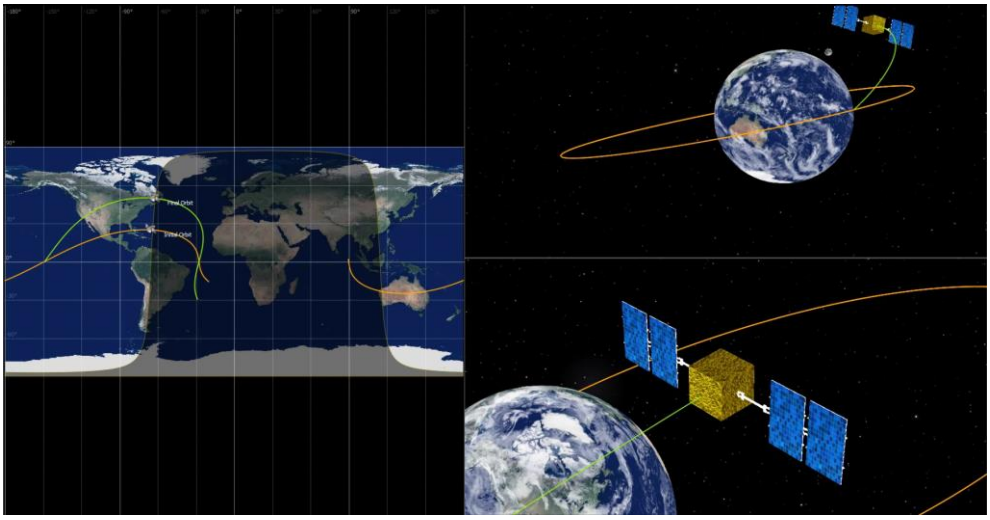

Figure 13 - Change of orbit's inclination manoeuvre

4.4 Illustration of Heliosynchronism

Sun synchronous orbits are type of orbits that correspond to fixing one constraint along one dimension, concretely the rate of change of the RAAN (Right Ascension of the Ascending Node). It depends on the current inclination, but does not affect the current inclination. This can be reflected on the following expression that gives the rotation rate in function of the orbit's inclination and other parameters explained below.

$$\omega_p = -\frac{3R_T^2}{2a^2} J_2 \omega \cos i$$

Where:

ω_p	is the precession rate (rad/s)
R_T	is the equatorial Earth's radius (m)
a	is the orbit's semi-major axis (m)
ω	is the orbit's angular frequency (rad/s)
i	is the orbit's inclination
J_2	1.08262668 10^-3

Therefore, it is possible to see the eclipse line, the orbit track and the equator all crossing at the same point, but the angle between equator and track, which is the inclination by definition, should remain constant whereas the eclipse line changes as the Sun latitude changes.

In fact, the ascending node should slightly move along the equator in a small interval containing the eclipse line node. This is due to the fact that in Sun Synchronous orbits the rate of the RAAN is set to the mean value of Earth revolution rate around the Sun (360 degrees in approximately 365.25 solar days), but as Earth orbit is not circular, at some point in the year the Earth is at its perihelion and goes faster, and in the opposite date of the year it goes slower than its mean rate. So the eclipse line which follows this changing rate will oscillate around the fixed rate of the RAAN.

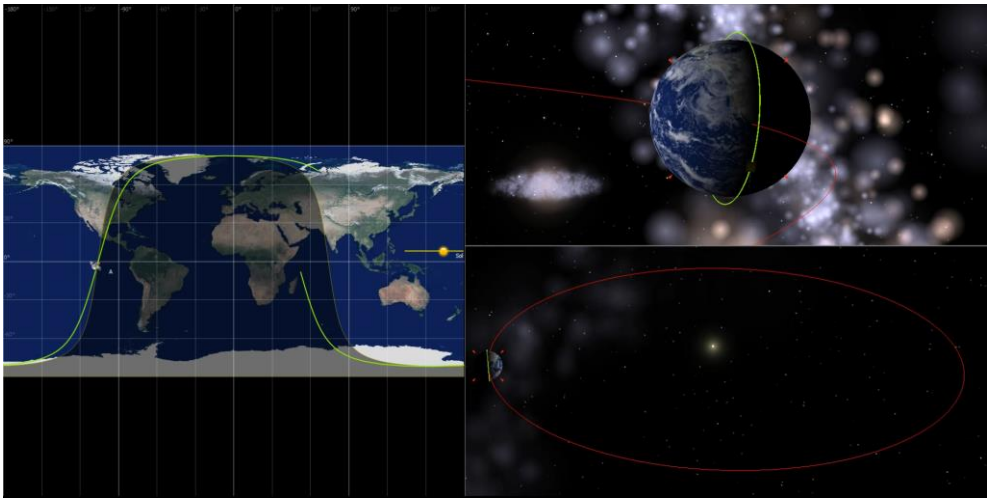

Figure 14 - Heliosynchronism visualization

This visualization has been carried out using VTS since it was interesting to also add the ground track view. However, due to the complexity of calculations it was not possible to generate the Sun Synchronous orbit with the Keplerian generator, so the whole orbit during a full year has been generated using a Matlab script. To see more detail see Annex.

In the Matlab script some calculations have been done to compute orbit's inclination given the input of the semi-major axis, 7000km, and the eccentricity, equal to zero because the desired orbit is circular. Therefore the obtained value for the inclination is 97,87º. This value has been validated with the Figure 15, obtained with Celestlab. In it, it is possible to see eccentricity values in function of eccentricity and semi-major axis in order to obtain a Sun Synchronous orbit.

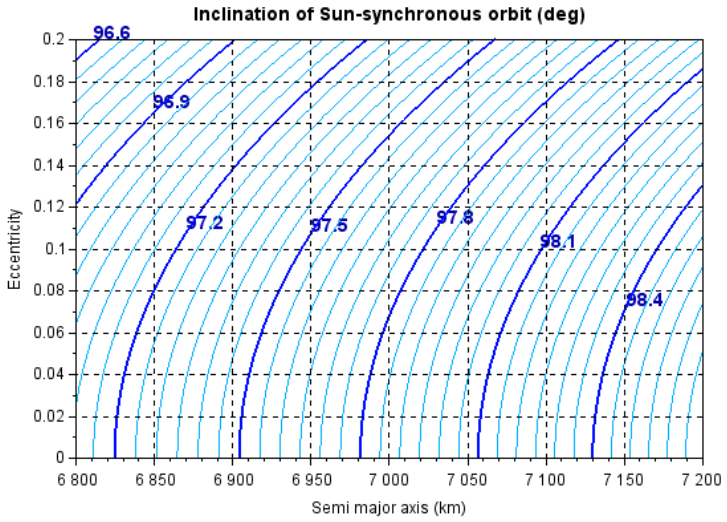

Figure 15 - Inclination vs eccentricity and orbit's radius in heliosynchronous orbits

5 Implementation of non-linear dynamics around Lagrangian points

In this section, the method explained to generate and visualize orbits has been implemented regarding non-linear dynamics around Lagrangian points. It is interesting to note that, orbits in this section are not suitable to the EME2000 reference frame, this is the reason why all illustrations in this section have been visualized using Celestia, just because in VTS it is impossible to customize de reference frame.

Regarding orbit generation process, all of the orbits in this section have been generated using Matlab scripts from previous works done related to the topic [5]. In fact, the developed software has been run in a loop to generate orbits with different elongations and then the computed data has been exported to files using Matlab scripts too. More detail related to this can be found in Annex.

Since the orbits visualized in this section are related to the Circular Restricted 3 Body Problem (CR3BP), first of all a brief introduction to this topic is going to be done referenced in the chosen scenario of Earth-Moon-Satellite system. Let's consider then the motion of an infinitesimal particle, a satellite in our case, under the gravitational attraction of two big masses, the Earth and the Moon.

Due to simplification, the attraction of the infinitesimal particle on the primaries is neglected, so it is possible to affirm that the primaries are describing Keplerian orbits around their common centre of mass. The study of the motion of the satellite under the gravitational effects of the primaries is known as Restricted 3 Body Problem. Moreover, assuming that the primaries are moving in circles around their centre of mass, therefore, this is what it is called Circular Restricted 3 Body Problem [6].

In the following subsections it is possible to observe diverse types of orbits orbiting around Lagrangian points L1 and L2 in the Earth-Moon system.

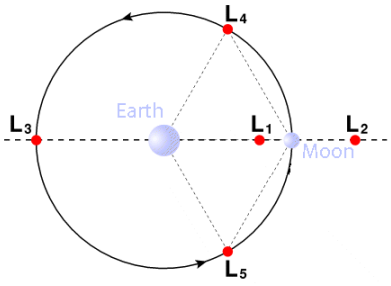

Figure 16 - Representation of Lagrangian points

5.1 Halo Orbits

The first scenario to visualize is the case of Halo orbits around Lagrangian points L1 and L2 in the Earth-Moon system. Orbits are represented so that its elongation increases progressively. In this particular case, the elongation values start at 1000 km (red trajectory) and end at a value of 71000 km elongation (blue trajectory). It is interesting to note that as the elongation increases, the orbit gets more vertical and closer to the moon.

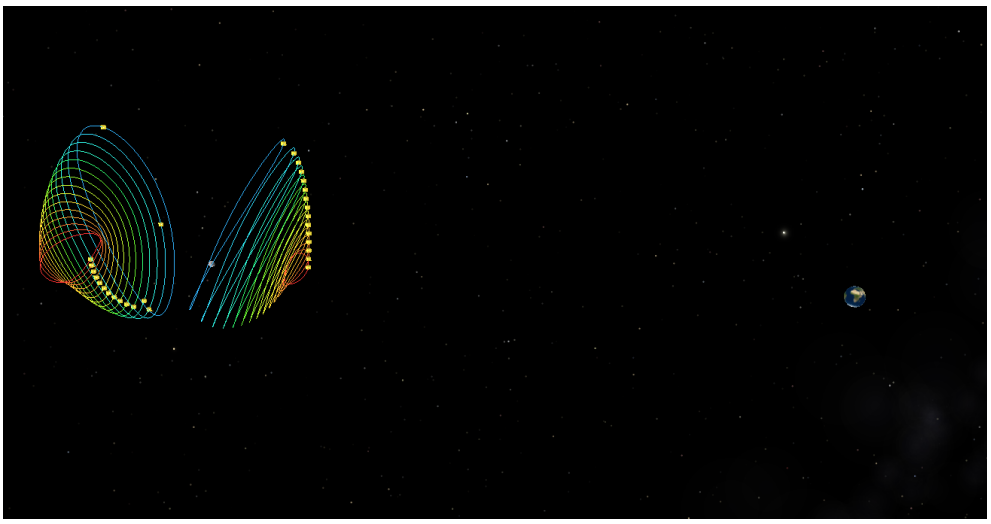

Figure 17 - EML1 and EML2 Halo orbits

5.2 NRO Orbits

The second scenario to visualize is the case of Near Rectilinear Orbits (NRO) around Lagrangian points L1 and L2 in the Earth-Moon system. As well as in Halo orbits, the diverse orbits observed are represented so that its elongation increases progressively. In this particular case, the elongation values start at 72000 km (red trajectory) and end at a value of 94000 km elongation (blue trajectory) for L1. Regarding L2 NRO orbits' elongation values range from 67000 km to 77000 km. It is interesting to note that as the elongation increases, the orbit gets more vertical and closer to the moon. A particularity of NRO orbits is that they can present northern or southern orbits, this is the reason why in Figure 18 there are four sets of orbits.

Figure 18 - EML1 and EML2 NRO orbits

5.3 DRO Orbits

The next scenario to visualize is the case of Direct Retrograde Orbits (DRO) around Lagrangian points L1 and L2 in the Earth-Moon system. It should be observed in Figure 19 that no difference exists between L1 and L2 orbits for DRO orbits. As well as in the other scenarios, the diverse orbits observed are represented so that its elongation increases progressively. In this particular case, the elongation values start at 10000 km (red trajectory) and end at a value of 335000 km elongation (blue trajectory).

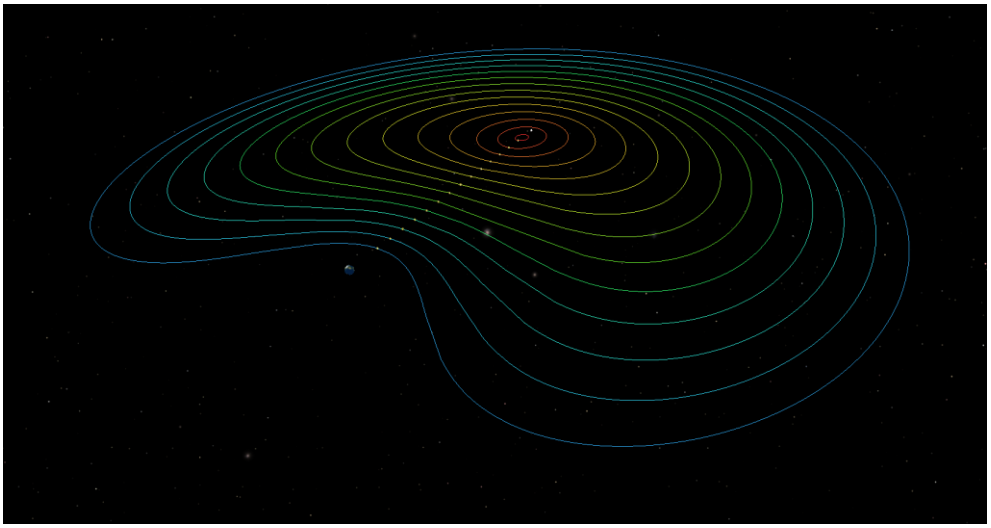

Figure 19 - EML1 and EML2 DRO orbits

5.4 VLYAP Orbits

Finally, the last scenario to visualize is the case of Vertical Lyapunov Orbits around Lagrangian point L2 in the Earth-Moon system. As well as in the other scenarios, the diverse orbits observed are represented so that its verticql elongation increases progressively. In this particular case, the elongation values start at 5000 km (red trajectory) and end at a value of 180000 km elongation (blue trajectory).

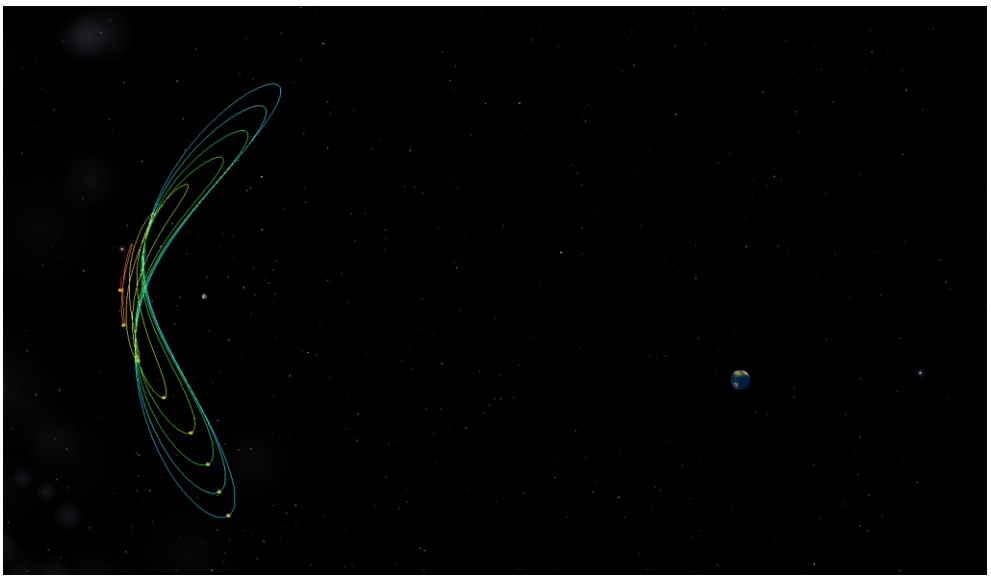

Figure 20 - EML1 and EML2 VLYAP orbits

6 Conclusions and future works

After finishing this project, some conclusions can be extracted regarding the visualization of trajectories in Keplerian motion and in non-linear dynamics around Lagrangian points.

The first conclusion that can be taken into account is that it is needed to choose between VTS or Celestia depending on the specific needs and requirements of each scenario. Thus, if an EME2000 reference frame is not compatible with our trajectory, Celestia will be selected, whilst if the ground track of the orbit wants to be plotted, VTS will be the ideal tool for that.

After all, it has been possible to generate and to export compatible files and trajectories with these two powerful visualization tools. Concretely, there have been no problems when implementing the explained method to illustrate both Keplerian motion and non-linear dynamics around Lagrangian points.

One last thing to evaluate as a conclusion is the importance of post-processing the produced animations, which enables to produce a more professional content and to adapt the illustrations to the reals needs.

Regarding future works on this field, it would be interesting to be aware to future releases of VTS versions just to see if CNES adds the option of changing the reference frame of the orbit as it is possible in Celestia. This would solve the most limiting problem of VTS and maybe it should be valuable to leave Celestia behind and perform all visualizations with VTS, which is more comfortable.

Another aspect to look at, it is the improvement or the automatization of importing scenarios in Celestia. Until now, the generated .xyz and .scc files have been copied to Celestia directory manually and it is not ideal at all.

Finally, a list of scenarios has been created to produce visualizations that illustrate properly each case. The list is the following:

- Rendez-vous trajectory between cargo and ISS
- Interplanetary trajectory
- Deorbitation trajectory
- Launch and in-orbit stationing

7 References

[1] National Aeronautics and Space Administration, “Celestia Users Guide.” 2015.

[2] R. J. Bart and Centre National d’Études Spatiales, “Vts User Manual,” p. 211, 2016.

[3] L. E. G. A. L. Jean-luc, C. Thomas, and L. E. G. A. L. Jean-luc, “Cnes Centre D'Ingénierie Concourante Technical Note Cic Data Exchange Protocol V2 . 0,” 2015.

[4] “Julian Date, Julian Modified Date.” [Online]. Available: http://www.csgnetwork.com/julianmodifdateconv.html . [Accessed: 20-Apr-2018].

[5] B. Le Bihan, “Doctorat de l’Université de Toulouse - Study of dynamics about and between libration points of Sun-Earth-Moon coherent models,” p. 267, 2017.

[6] J. Masdemont and J. M. Mondelo, “I . Dynamics and phase space around the Libration Points Equations of motion and Libration Points,” pp. 1–44.

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeries
Industrial i Aeronàutica de Terrassa

Institut Supérieur de l'Aéronautique et de l'Espace
SUPAERO

Validation and visualization of trajectories computaiton

Escola Superior d'Enginyeria Industrial, Aeroespacial

i Audiovisual de Terrassa

-

Institut Supérieur de l'Aéronautique

et de l'Espace – SUPAERO

Màster Universitari en Enginyeria Aeronàutica

Annex

Author: **Albert Guillén Buisan**

Director: **Stéphanie Lizy-Destrez**

June 2018

1. Sun Synchronous Orbit script

1.1. Time format & orbit discretization

```
a = 7000e3;                                                 %in meters
mu = 3.986e14;
T = 2*pi*sqrt(a^3/mu);                                      % Orbit's period
n = 50000;                                                  % Number of discretization
points
nc = 5500;                                                  % Number of complete orbits
to do
n_unit = n/nc;                                              % Number of points per
complete orbit

ini_date = 58218;                                           % Initial day for the
visualization (JD format)
secs = 240;
inc_day = 0;

timeJL = zeros(n,2);                                        % Matrix storing date in
Julian Date

for i = 1:n
    if secs >= 86400
        inc_day = inc_day + 1;
        secs = secs - 86400;
    end
timeJL(i,1) = ini_date + inc_day;
timeJL(i,2) = secs;
secs = secs + T/n_unit;
end

w = 2*pi/T;                                % angular velocity of satellite orbiting
around earth
wp = 2*pi/31557600;                                         % orbit precession rate
inc = acosd(-(2*wp*a^2)/(3*w*6378137^2*1.08262668e-3));  % Calculation of inclination
```

1.2. Orbit computation

```
t = linspace(0,nc*pi,n);                                    % Orbit definition in a
generic plane
x = cos(t);
y = sin(t);
z = 0*t;
pnts = [x;y;z];                          % Matrix storing coordinates of orbit in the
basic plane

% unit normal for original plane
n0 = [0;0;1];
```

```
n0 = n0/norm(n0);

At = T/n_unit;                  % Delta T between discretized points (constant as
satellite velocioty is constant (e = 0))
newPnts = zeros(3,n);

for i = 1:1:n
% unit normal for plane to rotate into
% plane is orthogonal to n1... given by equation
% n1(1)*x + n1(2)*y + n1(3)*z = 0
n1 = [cos(wp*At+ini_ang);sin(wp*At+ini_ang);cos(inc*pi/180)];
n1 = n1/norm(n1);

% theta is the angle between normals
c = dot(n0,n1) / ( norm(n0)*norm(n1) ); % cos(theta)
s = sqrt(1-c*c);                        % sin(theta)
u = cross(n0,n1) / ( norm(n0)*norm(n1) ); % rotation axis...
u = u/norm(u); % ... as unit vector
C = 1-c;

% the rotation matrix
R = [u(1)^2*C+c, u(1)*u(2)*C-u(3)*s, u(1)*u(3)*C+u(2)*s
    u(2)*u(1)*C+u(3)*s, u(2)^2*C+c, u(2)*u(3)*C-u(1)*s
    u(3)*u(1)*C-u(2)*s, u(3)*u(2)*C+u(1)*s, u(3)^2*C+c];

% Rotated points
rotated_pnts = a*1500/1000*R*pnts;              % Dimensional coordinates of the
trajectory
newPnts(:,i) = rotated_pnts(:,i);

At = At + T/n_unit;
end
```

1.3. Export format adaptation to VTS and file creation

```
CIC = [timeJL, newPnts'];                                                   % Matrix
containing time data together with position data

fileID = fopen('SATELLITE_Heliosync.txt','w');                              % Creation
of .txt file needed to import to VTS
fprintf(fileID,'CIC_OEM_VERS = 2.0\n');
fprintf(fileID,'CREATION_DATE = 2011-12-08T17:24:51.012783\n');
fprintf(fileID,'ORIGINATOR = VTS Propagated Orbit\n\n');

fprintf(fileID,'META_START\n\n');

fprintf(fileID,'OBJECT_NAME = UNKNOWN\n');
fprintf(fileID,'OBJECT_ID = UNKNOWN\n');
fprintf(fileID,'CENTER_NAME = UNDEFINED\n');
fprintf(fileID,'REF_FRAME = EME2000\n');
fprintf(fileID,'TIME_SYSTEM = UTC\n\n');
```

```
fprintf(fileID,'META_STOP\n\n');

fprintf(fileID,'%-5.0f %-5.3f %-12.8f %-12.8f %-12.8f\n',CIC');
fclose(fileID);
```

2. Hyperbola generation script

```
X = zeros(1000,1);
Y = zeros(1000,1);
Z = zeros(1000,1);

a = 30000;
h = 42000;
c = h;
b = sqrt(c^2-a^2);

ini_date = 58204;
n = length(X);
secs = 49563.517;
inc_day = 0;

Y = linspace(-100000,100000,1000)';

for i = 1:n
    X(i) = a*(h/a - sqrt(1+(Y(i)^2)/(b^2)));
end

timeformat = 1; % 1 is compatible with VTS data, 2 with Celestia XYZ files

if timeformat == 1

    timeJL = zeros(n,2);
    for i = 1:n
        if secs >= 86400
            inc_day = inc_day + 1;
            secs = secs - 86400;
        end
    timeJL(i,1) = ini_date + inc_day;
    timeJL(i,2) = secs;
    secs = secs + 60;
    end

    CIC = [timeJL, X, Y, Z];

fileID = fopen('SATELLITE_eHyperbola.txt','w');
fprintf(fileID,'CIC_OEM_VERS = 2.0\n');
fprintf(fileID,'CREATION_DATE = 2011-12-08T17:24:51.012783\n');
```

```
fprintf(fileID,'ORIGINATOR = VTS Propagated Orbit\n\n');

fprintf(fileID,'META_START\n\n');

fprintf(fileID,'OBJECT_NAME = UNKNOWN\n');
fprintf(fileID,'OBJECT_ID = UNKNOWN\n');
fprintf(fileID,'CENTER_NAME = UNDEFINED\n');
fprintf(fileID,'REF_FRAME = EME2000\n');
fprintf(fileID,'TIME_SYSTEM = UTC\n\n');

fprintf(fileID,'META_STOP\n\n');

fprintf(fileID,'%-5.0f %-5.3f %-12.8f %-12.8f %-12.8f\n',CIC');
fclose(fileID);

elseif timeformat == 2

    begin = juliandate(2019,09,15); %Converts date to JulianDate format -
"juliandate(year,month,day)"
    timeJL = zeros(n,1);
    for i = 1:n
    timeJL(i,1) = begin + secs;
    secs = secs + 0.1;
    end

    CIC = [timeJL, X, Y, Z];

fileID = fopen('exp.txt','w');

fprintf(fileID,'%-5.4f %-12.8f %-12.8f %-12.8f\n',CIC');
fclose(fileID);

end
```

3. Parabola generation script

```
X = zeros(1000,1);
Y = zeros(1000,1);
Z = zeros(1000,1);

p = -12000;
h = 12000;

ini_date = 58204;
n = length(X);
secs = 49563.517;
inc_day = 0;
```

```
Y = linspace(-100000,100000,1000)';

for i = 1:n
    X(i) = Y(i)^2/(4*p) + h;
end


timeformat = 1; % 1 is compatible with VTS data, 2 with Celestia XYZ files

if timeformat == 1

    timeJL = zeros(n,2);
    for i = 1:n
        if secs >= 86400
            inc_day = inc_day + 1;
            secs = secs - 86400;
        end
    timeJL(i,1) = ini_date + inc_day;
    timeJL(i,2) = secs;
    secs = secs + 60;
    end

    CIC = [timeJL, X, Y, Z];

fileID = fopen('SATELLITE_eParabola.txt','w');
fprintf(fileID,'CIC_OEM_VERS = 2.0\n');
fprintf(fileID,'CREATION_DATE = 2011-12-08T17:24:51.012783\n');
fprintf(fileID,'ORIGINATOR = VTS Propagated Orbit\n\n');

fprintf(fileID,'META_START\n\n');

fprintf(fileID,'OBJECT_NAME = UNKNOWN\n');
fprintf(fileID,'OBJECT_ID = UNKNOWN\n');
fprintf(fileID,'CENTER_NAME = UNDEFINED\n');
fprintf(fileID,'REF_FRAME = EME2000\n');
fprintf(fileID,'TIME_SYSTEM = UTC\n\n');

fprintf(fileID,'META_STOP\n\n');

fprintf(fileID,'%-5.0f %-5.3f %-12.8f %-12.8f %-12.8f\n',CIC');
fclose(fileID);

elseif timeformat == 2

    begin = juliandate(2019,09,15); %Converts date to JulianDate format -
"juliandate(year,month,day)"
    timeJL = zeros(n,1);
    for i = 1:n
    timeJL(i,1) = begin + secs;
    secs = secs + 0.1;
    end

    CIC = [timeJL, X, Y, Z];
```

```
fileID = fopen('exp.txt','w');

fprintf(fileID,'%-5.4f %-12.8f %-12.8f %-12.8f\n',CIC');
fclose(fileID);

end
```

4. Example generation range of Halo orbits

```
% ce script calcul une orbite dont les paramètres sont entrés par
% l'utilisateur dans les premières lignes. Il affiche cette orbite et garde
% en mémoire les coordonnées de cette orbite.

% initiation des paramètres
init;

for i = 1000000:5000:71000000
% initialisation de l'environnemnt
cr3bp = init_CR3BP('EARTH','MOON',default);

% % choix du point de Lagrange : cr3bp.l1 ou cr3bp.l2
lag_point = cr3bp.l1;
%
% % choix du type d'orbite : 'NRO', 'HALO' ou 'DRO'
type = 'HALO';
%
% % choix de la famille d'orbite (pour les orbites NRO) : 'NORTHERN' ou
% % 'SOUTHERN'
family = 'NORTHERN';
%
% % paramètre de définition de l'orbite : 'Az' pour les orbites de Halo et les NRO,
et 'Ax' pour les orbites DRO
Af = 'Az';
%
% % choix de l'extension de l'orbite (dimensionnée)
% % Attention : les valeurs limites que l'on peut donner à cette extension dépendent
du type d'orbite choisi
Afdim = i;
%
% % initialisation de l'orbite
orbit_init = init_orbit(cr3bp,lag_point,type,family,Afdim,cst);
%
% % calcul complet de l'orbite
orbit = orbit_interpolation(cr3bp,orbit_init,default,cst,Af,Afdim);

coord = orbit.yv(:,1:6)*cr3bp.L;

filename = strcat('HALO_L2_', num2str(i));
```

```
if i == 1000
    t1 = datetime(2018,11,23);
end

t2 = t1 + days(5);
t1 = t2;

exportedFile = Export2Celestia(coord, filename, t2);

end
```

5. Export to Celestia function

```
function [file] = Export2Celestia(coord, name, date)
%
secs = 0;
n = length(coord);
begin = juliandate(date); %Converts date to JulianDate format -
"juliandate(year,month,day)"
timeJL = zeros(n,1);

for i = 1:n
    timeJL(i,1) = begin + secs;
    secs = secs + 0.1;
end

CIC = [timeJL, coord(:, 1:3)];

file = fopen(strcat(name,'.xyz'),'w');

fprintf(file,'%-5.4f %-12.8f %-12.8f %-12.8f\n', CIC');

fclose(file);

file2 = fopen(strcat('Script_',name,'.ssc'),'w');
fprintf(file2, strcat('"',name,'"'));
fprintf(file2, '%s\n', ' "Sol/Earth/Moon"');
fprintf(file2, '%s\n', '{');
fprintf(file2, '%s\n', 'Class "spacecraft"');
fprintf(file2, '%s\n', 'Mesh "Cube_body.3ds"');
fprintf(file2, '%s\n', 'Radius 1000.00');
fprintf(file2, '%s\n', 'Orientation [ 90 0 0 1 ]');
fprintf(file2, '%s\n', '');
fprintf(file2, '%s\n', 'Timeline [');
fprintf(file2, '%s\n', '{');
fprintf(file2, 'Beginning " %s %s %s 12:00:00"\n', num2str(year(date)),
num2str(month(date)), num2str(day(date)));
fprintf(file2, '%s\n', 'Ending    "2023 12 01 12:00:00"');
```

```
fprintf(file2, '%s\n', '');
fprintf(file2, '%s\n', 'OrbitFrame { ');
fprintf(file2, '%s\n', 'TwoVector{');
fprintf(file2, '%s\n', 'Center "Sol/Earth"');
fprintf(file2, '%s\n', 'Primary{');
fprintf(file2, '%s\n', 'Axis "x"');
fprintf(file2, '%s\n', 'RelativePosition{');
fprintf(file2, '%s\n', 'Observer "Sol/Earth"');
fprintf(file2, '%s\n', 'Target "Sol/Earth/Moon"');
fprintf(file2, '%s\n', '}');
fprintf(file2, '%s\n', '}');
fprintf(file2, '%s\n', 'Secondary{');
fprintf(file2, '%s\n', 'Axis "y"');
fprintf(file2, '%s\n', 'RelativeVelocity{');
fprintf(file2, '%s\n', 'Observer "Sol/Earth"');
fprintf(file2, '%s\n', 'Target "Sol/Earth/Moon"');
fprintf(file2, '%s\n', '}');
fprintf(file2, '%s\n', '}');
fprintf(file2, '%s\n', '}');
fprintf(file2, '%s\n', '}');
fprintf(file2, '%s\n', '');
fprintf(file2, '%s\n', strcat('SampledTrajectory { Source "', name,'.xyz','" }'));
fprintf(file2, '%s\n', 'FixedRotation { }');
fprintf(file2, '%s\n', '}');
fprintf(file2, '%s\n', ']');
fprintf(file2, '%s\n', '}');
fprintf(file2, '%s\n', strcat('Modify', ' "',name,'" ', '"Sol/Earth/Moon" {'));
fprintf(file2, '%s\n', 'OrbitColor [ 1 0.2 0.2 ]');
fprintf(file2, '%s\n', '}');



fclose(file2);

end
```