UNIVERSITAT POLITÈCNICA DE CATALUNYA

STUDY OF CASSINI AND NEW HORIZONS TRAJECTORIES USING JPL SPICE LIBRARY

A thesis submitted by Roger Sala Marco for the BSc Degree in Aerospace Vehicle Engineering

June 10, 2019

Directed by:
Manel Soria Guerrero

Co-director:
Enrique García Melendo

Acknowledgements

"I would like to thank Manel Soria for his invaluable support during the course of this project and being always there to solve all problems that could arise. I would also like to congratulate Martí Sierra for the great work he has done, it would not have been possible to achieve the goals of the project without his support."

To my family and friends...

Abstract

The obtaining of scientific data in interplanetary missions is based on a series of complex algorithms that serve to plan and develop the experiments.

At different stages of the mission, due to the fact that the conditions planned to do each observation can end up not being the ones that actually happen, the best timing to carry out each observation must be recalculated.

Therefore, the objective of this project is to understand and implement a subset of the algorithms used to obtain scientific data, using Matlab and SPICE, a library created by JPL, which is a software tool that contains a series of functions that are helpful when designing observations.

In this project, a series of SPICE-based algorithms have been developed to calculate occultations, transit and the illumination of celestial bodies. Specifically, in this project are studied the flybys that were made by the probes: Cassini, Voyager and New Horizons, on the different planets for which they passed by. For the case of when Cassini flies over Jupiter, observations have been designed in two different ways; one considering whether the camera was pointing to the planet and the other without taking it into account.

The results obtained from the algorithms have been compared with the real trajectories already made, obtaining satisfactory results. It has also been proven from navigated images that the algorithms work correctly.

Another algorithm has also been created to calculate the time interval during which the transit of a celestial body can be seen in front of another, from any planet. In particular, it has been tested with the transit that Mercury is going to make on November 11, 2019, achieving the expected result.

Contents

List of Figures

List of Algorithms

Part I

General Introduction

Chapter 2

Overview

The obtention of scientific data in interplanetary missions such as Voyayer, Cassini or New Horizons involves a set of complex computations to plan and develop the experiments. The experimental data to be obtained includes images of celestial bodies in different wavelengths, gravity science [13] and radio science data obtained during the occultations [21, 28].

Such computations are carried out at different mission development stages. During the mission preliminary analysis, an initial sequence of observations has to be designed so that the spacecraft trajectory or even the design of the instruments can be redefined if needed, for instance, to improve the resolution of the images. During the spacecraft operation the final sequence has to be designed and tested with software simulators. Finally, once the data has been obtained, the actual spacecraft position and orientation of the different instruments has to be computed and used in order to obtain, for instance, planetary images where the latitude and longitude of each pixel is precisely known.

JPL library SPICE [40, 22] is a software tool widely used not only by NASA but also by other space agencies. While it has a wide range of functions, it is only a library and needs an external program in order to obtain the necessary data to compute the aforementioned computations. JPL has developed other software layers to help scientists and engineers, such as the Spatial Operator Algebra (SOA) [47]. The ESA has developed another software apart from the one developed by the JPL, called Solar System Science Laboratory (SOLab) [40].

The aim of this TFG is to understand and implement a subset of the algorithms used to design the interplanetary observations, using Matlab and SPICE.

The goals have been to understand SPICE library and some of algorithms used by higher level software such as SOA to calculate occultations, transits and illumination of the celestial bodies.

Specifically, the TFG has been focused on flyby missions, such as Cassini flyby of Jupiter. For a given flyby, the actual trajectory data (available in the form of SPICE kernels) has been considered as a design trajectory and, from it, opportunities to obtain images and radio science data have been identified with the

algorithms developed.

It is important to note that a preliminary tool to manage the SPICE kernels from Matlab in different computer platforms (Windows/Unix) has been developed, but graphic user interfaces for the software developed have been considered out of scope.

2.1 Aim

The aim of this project is to analyze the trajectory of different existing spacecraft as well as their attitude during the flybys of different solar system bodies observed in order to be capable to implement a subset of algorithms to calculate the best timing for the observations that probe had to make.

In future projects, this work will allow the users to design planetary observations of future probes.

2.2 Scope

This project is devoted to the development of some algorithms based on NASA's JPL SPICE library. These must be capable of analyzing the trajectories made by different spacecraft as well as the attitude of the probe during the flyby of a planet in order to design the observations that probe had to make.

In this project, the goal is to develop a preliminary tool to manage the SPICE kernels from Matlab in different computer platforms (Windows/Unix) must be developed, but graphic user interfaces for the software developed have been considered out of scope.

2.3 Requirements

These MATLAB algorythms will be used in the future by other students to study the design phase of different spacecraft and to study their trajectories and their attitude during its lifetime, so the code must be intelligible and clear so that everyone can use it. As it is based in the SPICE library, a brief introduction to this matter must be done so that everyone who wants to use this solver gets easily familiar with this library.

2.4 Justification

The NASA and the ESA have their own private software to design the observations that must be done with any spacecraft to any planet, but as said above they are

private software, so in this project, what was intended was to understand the algorithms that are used to design the observations and develop some algorithms based on the NASA's SPICE library and, extracting the data from the SPICE kernels, be able to find the best timing to design these observations.

2.5 Collaboration

In order to compare the science observation opportunities predicted by this created software with the actual observations made by spacecraft, RAW images by Voyager and Cassini have been used. The processing software has been developed by Martí Sierra Salvadó and the relevant information can be found in his TFG.

Chapter 3

Introduction to interplanetary missions

Before reading this project, a brief introduction of each of the missions that are studied on it is done.

3.1 Voyager

The goal of the Voyager mission was to study the outer Solar System by using two probes, the Voyager 1 and the Voyager 2 [9]. Their launch was in 1977 to take advantage of a favourable alignment of Jupiter, Saturn, Uranus and Neptune. The primary mission was to study only Jupiter and Saturn, but Voyager 2 finally continued to Uranus and Neptune, becoming the only probe to study these two planets [36].

On 25 August 2012, Voyager 1 became the first human-made object to enter interstellar space [10]. On 5 November 2018, Voyager 2 also entered interstellar space[16].

Thanks to Voyager data and images, unknown details have been revealed about each of the four giant planets systems. At Jupiter, images from the spacecraft discovered volcanic activity on Jupiter's moon Io and also charted Jupiter's complex cloud forms, winds and storm systems [32]. While orbiting Saturn, the spacecraft found that the rings have enigmatic braids, folds and radii, and also that they are accompanied by innumerable "curls" [33, 44]. During the Uranus flyby, Voyager 2 found out a substantial magnetic field around the planet and ten more moons[45]. Its flyby of Neptune uncovered three rings and six unknown moons [46, 37], a planetary magnetic field and complex, widely distributed auroras.

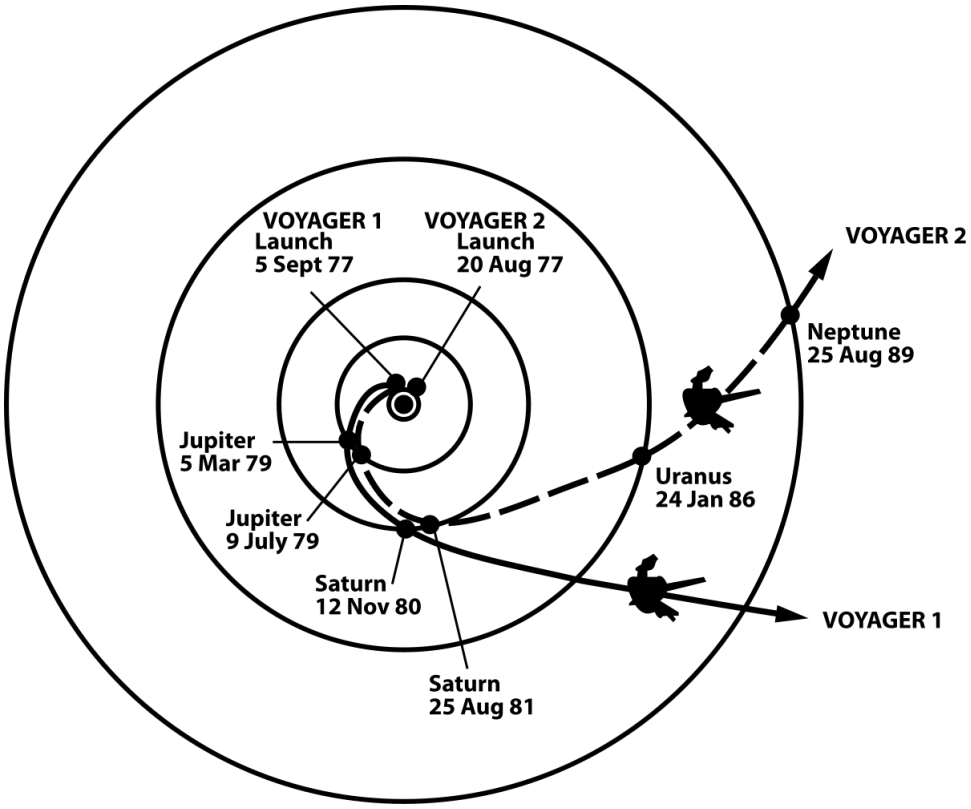

Figure 3.1: Voyager trajectory [12]

3.2 Cassini-Huygens

The Cassini-Huygens mission consisted of the Cassini Saturn Orbiter, provided by the NASA and the Huygens Titan Probe, supplied by ESA. The Huygens Titan Probe was targeted for entry into the atmosphere of Saturn's largest moon, Titan, which is a planet-sized satellite with a dense, veiling atmosphere.

The primary goal of Cassini/Huygens was to conduct an in-depth exploration of the Saturnian System [15], as well as a an in-depth study of Titan moon. Not only would this new mission study the individual objects (i.e., planet, satellites, rings, and magnetosphere) but it also would study the interactions between and among them [35].

Its launch was on October 15, 1997 to arrival at Saturn's orbit in July 1, 2004 [20]. Once in orbit about Saturn, Huygens was released from the orbiter and descended through Titan's atmosphere by parachute while measuring the properties of it. The rings, the magnetosphere and Saturn itself were all studied as well as the interactions among them [39]. The Cassini/Huygens mission made both in situ measurements, and remote sensing observations of targets under favourable geometric and temporal conditions that are not available from Earth (e.g., range and angles of illumination and emission; events such as occultations and eclipses)[15].

While Huygens was studying Titan, the Cassini Orbiter released a four-year

mission which included more than seventy orbits of Saturn and its moons. Cassini's primary mission ended on June, 2008. After that, NASA decided to carry out a two-year mission until September, 2010 to study the Saturn's equinox, which took place on August, 2009. This mission was called the Cassini Equinox Mission [1]. When it ended, they decided to extend the Cassini's lifetime until the Saturn summer solstice that took place on May, 2017. This mission was named Cassini Solstice Mission [14].

The final mission of the Cassini Orbiter was to make orbits of very high eccentricity to finally make an approach to Saturn, burning in its atmosphere on September 15, 2017. This last mission was called Grand Finale [29].

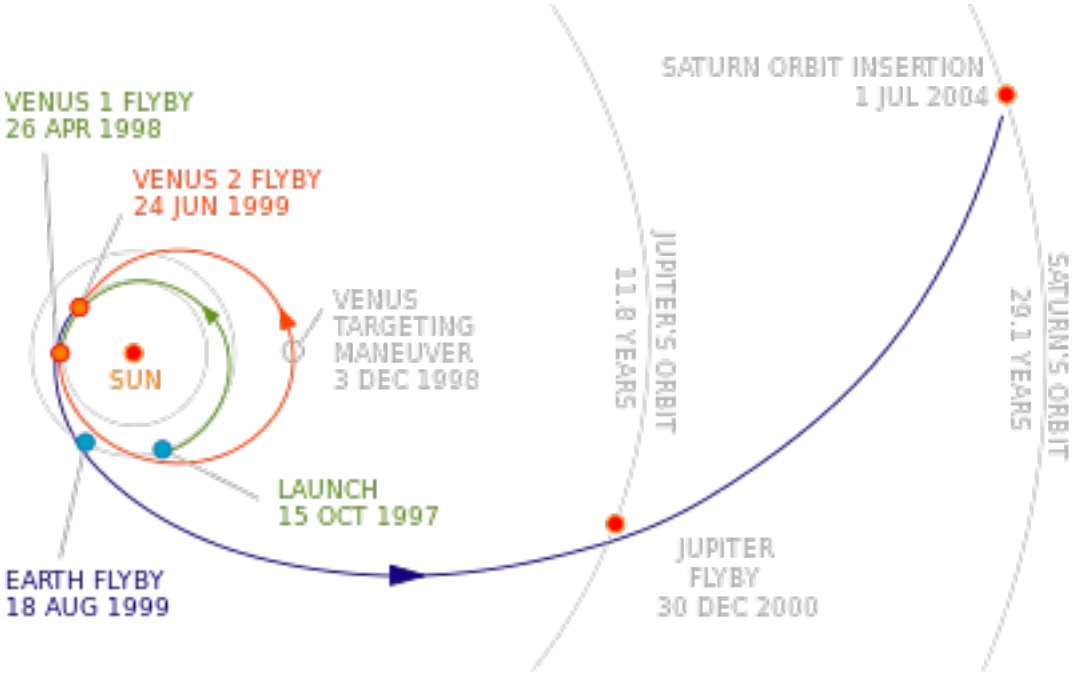

Figure 3.2: Cassini trajectory [11]

3.3 New Horizons

New Horizons is an interplanetary space probe that was launched on January 19, 2006 from Cape Canaveral Air Force Station as a part of NASA's New Frontiers program [25]. Its primary mission was to perform a fly by study of the Pluto system in 2015. New Horizons' secondary mission was to fly by and study one or more other Kuiper Belt Objects (KBOs) in the decade to follow [42]. It is the fifth space probe to achieve the escape velocity needed to leave the Solar System[34].

New Horizons made its closest approach to Jupiter on February 28, 2007. The Jupiter fly by provided a gravity assist that increased New Horizons' speed. Furthermore, the flyby enabled a general test of New Horizons' scientific capabilities, returning data about the planet's atmosphere, moons, and magnetosphere[24, 41, 38].

After the Jupiter encounter, the spacecraft hibernated to preserve on-board systems, except for brief annual checkouts. On December 6, 2014, New Horizons was awaken from its hibernation to get ready for the Pluto encounter. On January 15, 2015, the spacecraft began its approach phase to Pluto[43].

On July 14, 2015 it flew above the surface of Pluto, making it the first spacecraft to explore the dwarf planet[23]. On October 25, 2016 the last of the recorded data from the Pluto fly by was received from New Horizons. After completing its primary mission, New Horizons manoeuvred for a fly by of Ultima Thule, which occurred on January 1, 2019 [30].

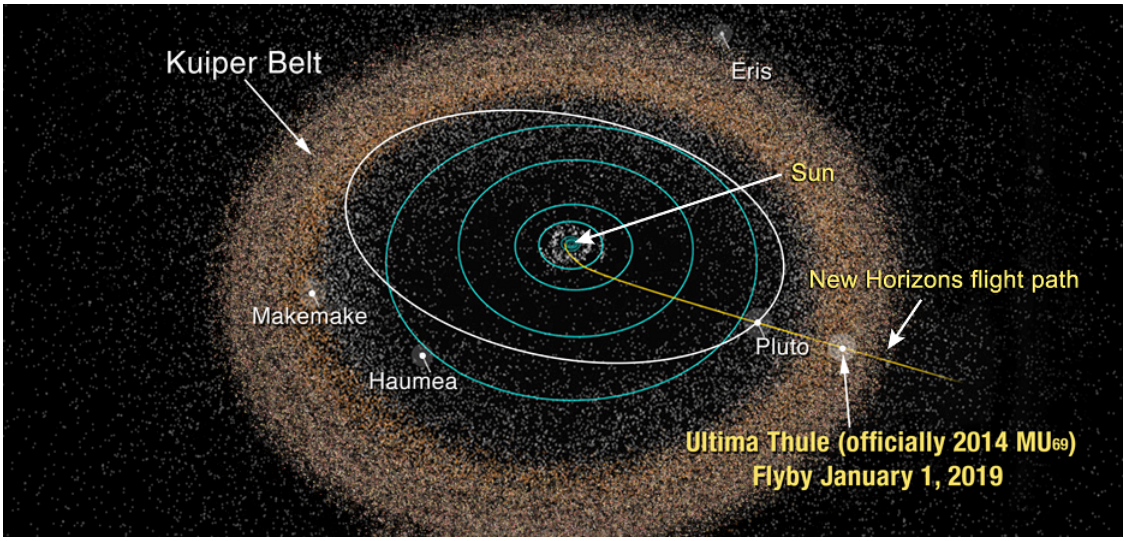

Figure 3.3: New Horizons trajectory [7]

Part II

MATLAB algorithms: development, results and validation

Chapter 4

Introduction to SPICE

It has been considered that before explaining the program which has been created, the reader should be informed about the SPICE library, in which the entire project has been based.

4.1 SPICE overview

SPICE (Spacecraft Planet Instrument Camera-matrix Events) is the observation geometry system of the NASA for space science missions. It was developed by the Navigation and Ancillary Information Facility (NAIF), located at the Jet Propulsion Laboratory.

The first purpose of SPICE was to prepare science archives and to analyze data during and after mission operations. Nowadays, any person using a software based on SPICE can design and validate a space mission, make a detailed science observation planning and, of course, analyze the data returned from the instruments and prepare the science data archives.

However, SPICE is only a library so, without a code to run its functions the user will not obtain any results. NASA and ESA have each designed a private software based on SPICE to design observations, NASA software is called SOA and ESA software is called SOLAD. The objective of this project to find out how their private software works and to make our own software to design observations.

The SPICE library is available in five different programming languages, but in this project it has been decided to use the MATLAB SPICE library as it is the one the author is most familiar with.

4.2 Functionality

On the one hand, it is important to know what kind of observation geometry parameters can anyone obtain with SPICE. Any person using a program based on

SPICE could compute things such as:

- Position and velocities of planets, satellites, comets, asteroids and spacecraft.
- Size, shape and orientation of planets, satellites, comets and asteroids.
- Orientation of a spacecraft and its various moving structures.
- Instrument field-of-view location on a surface of a planet or atmosphere.

Furthermore, it allows the user to calculate the time when a specified geometric event occurs such as an occultation, the minimum distance between two bodies, the transit (that is when an object is seen in front of another from the spacecraft), etc.

On the other hand, the most important issue of using SPICE is that it can not be ran without having the correct ancillary data files, the kernels.

Kernels are files that have all the geometry data from a spacecraft in a certain time so, for example, if the position of the spacecraft Cassini wants to be calculated on January, 2004, the program must work with all the kernels from that date.

There are many types of kernels, each of them containing different information:

- PCK: orientation, size and shape of the different solar system bodies. It can also contain parameters for gravitational model, atmospheric model or rings model.
- SPK: space vehicle or target body trajectory (ephemeris). More generally, position of something relative to something else.
- IK: instrument field-of-view size, shape and orientation. Possibly it can contain internal timing too.
- CK: spacecraft attitude or orientation of any articulating structure on it relative to a specified reference frame.
- EK: events information such as: science plan, sequence of events or experimenter's notebook.
- FK: reference frame specifications.
- LSK: leap seconds tabulation. They are used to convert time from UTC to TDB (et) and vice versa.
- SCLK: spacecraft clock coefficients. These kernels are used for SCLK to TDB (et) time conversions and vice versa.
- DSK: digital shape models.

These different types of kernels help scientists and engineers determine observation geometry, such as:

- Position of the spacecraft.
- Orientation and size of the spacecraft and its instruments.
- Location, size, shape and orientation of the target observed.
- Surface point where the instrument was pointing.

Of all the types of existing kernels, the most important to carry out this project is the SPK (spacecraft and planet ephemeris kernel) because it contains the position and velocity data (usually called "state") for a spacecraft and for the different bodies of the solar system with respect to a certain reference system. That is why this kernel is so important, because without it it would be impossible to know the position of the spacecraft and the solar system bodies that are being studied with respect to a certain frame, which would make it impossible to design the observations presented in this project.

Another type of kernel important in this work when designing observations considering the camera attitude is the CK (Camera-Matrix kernel). It contains the orientation of the spacecraft and the various instruments of this. Without this kernel it would not have been possible to study the moments when the camera was pointing to the planet[1].

[1]To amplify the knowledge about this kernel, read the project of the TFG's author partner Martí Sierra Salvadó

Chapter 5

Loading kernels

The first thing that should be done when working with a program based on the SPICE library is to load the kernels that contain the data of the time interval which is going to be studied. This may seem basic, but it is not.

These kernels are downloaded directly from the JPL website. Within the web are distributed on the one hand according to the different missions carried out and on the other hand by a section called general kernels where information of the different solar system bodies can be found.

In order to work with these kernels in the program, the user must have them downloaded in the computer to be able to load them, but a single mission can contain millions of different kernels covering different time intervals of the lifetime of the spacecraft. Therefore, the problem lies in how to know which kernels must be downloaded.

It has not yet found by this teamwork the way to download only the necessary kernels for the time interval of interest (unless manually looking at the time interval covered by each kernel of the web) but it has been able to load into the program only the necessary kernels. In Section 5.2, the concept of necessary kernels will be developed.

5.1 Mission

Each mission has its own kernel package that can contain thousands of them. In order to distinguish between different missions, structures with the kernels of each mission have been created in order to quickly know which kernels are from each spacecraft. These structures have two columns, the first column is the URL from where the kernel has to be downloaded if needed, and the second column is the name of the kernel.

Hence, the program will only have to find what kernels are needed according to the list of kernels of the mission that wants to be studied. For example, if the Cassini mission wants to be studied, the program will look for the kernel structure

of Cassini and then search within this list for the kernels that satisfy the time interval and that contain information about the planet that wants to be studied.

5.2 Necessary kernels

In this section, which kernels are the necessary kernels will be explained as well as how to recognize them.

As already said before, not all kernels are useful when it comes to studying a certain time interval, solar system body and mission, since each kernel covers a different time interval and contains information for a specified mission on a certain planet system. Thus, only the kernels which cover the mission studied at the time interval of interest must be loaded, as well as the ones that have the data of the bodies related to the planet system which is being studied.

Thus, the kernels to be loaded into the program must satisfy at least one of the two conditions shown below:

- **Time interval:** The time interval that the kernel covers is within the time interval studied.
- **Planet:** The kernel has the data of the planet system studied.

5.2.1 Time interval

All types of kernels can be classified into two large groups, those that depend on time and those that do not. The time dependant kernels are the SPK and the CK kernels, all the other types of kernels do not depend on time.

Each time dependent kernel has a coverage window that indicates the first and the last ephemeris time of the time interval that it covers. This coverage window is very useful when trying to load only the necessary kernels. Thus, the program may have a look at the coverage window of each kernel and find out if it satisfies the time interval to be studied and, if it does, load it.

To do so, the SPICE library has two functions called *cspice_ckcov* in the case it is a CK and *cspice_spkcov* in the case it is a SPK[1], but to be able to run these functions the program must have loaded two different kernels before, a LSK and a SCLK. These two kernels, as said before, do not depend on the time interval studied, so they are valid for all time intervals. Furthermore, all the other types of kernels that are not time dependent must be loaded since they are valid on every time interval of the lifetime of a spacecraft.

Even though, these time independent kernels have more than one version so the program must have the newest version loaded as it is the most accurate version.

[1]To see more detailed information about the operation of the two SPICE functions mentioned see Appendix **??** and **??**

This is done manually as it has not yet found how to know automatically which kernel is the most actual.

In this way, all kernels satisfying the time interval would be loaded in the program, but the problem is that, in some cases, there are two different time dependent kernels (CK or SPK) covering the same time interval and one of them must not be loaded.

There are two types of CK and SPK kernels that must be considered:

- *Predicted kernel:* this kernel is made with the design data of the mission and it is not the trajectory and the attitude the spacecraft really had.
- *Reconstructed kernel:* this kernel is made after studying the data provided by the spacecraft during the mission. Thus, all the data within this kernel is better than the data from the predicted kernel.

Therefore, only the reconstructed kernels must be loaded into the program. It has not yet found how to load automatically only the reconstructed kernels to the program so it is done manually erasing all predicted kernels from the kernel list.

It is important to keep in mind that, as stated above in Section 4.1, only spacecraft have camera kernels (CK), while not only spacecraft have SPK but the different bodies of the solar system too. That is why, when looking for the coverage window of a SPK kernel, it first has to be checked whether it is a kernel related to the spacecraft or it is a generic kernel.

5.2.2 Planet

Each general kernel contains a list of the bodies of the solar system of which it contains information. This way it can be known if the kernel contains information about the planet that wants to be studied. If so, the kernel would have to be loaded into the program, but if it does not contain information about the planet, the kernel should not be loaded.

Even so, if the kernel contains information about the Earth, it should be loaded since without the pertinent information to the Earth the occultations can not be calculated as they are between the planet studied and the Earth.

5.2.3 *initSpiced.m*

To load only the necessary kernels and download them from the web in the case that the user does not have them downloaded in the computer, a function has been created.

The pseudo code of the algorithm developed is shown below.

Algorithm 1: *initSpiced.m*

Input data: UTC time (*utctim*), number of days (*NDAYS*), mission name (*missionname*) and target (*target*);
METAKR ← Cell structure of size Mx2, where M is the number of kernels it contains. It is obtained using the function *CreateMETAKR* (explained in the next section). The first column corresponds to the URL of the kernel and the second column is the name of the kernel;
for $i \leftarrow 1$ **to** *size(METAKR,1)* **do**
    Check if the kernel is downloaded on the computer. If not, download the kernel from the web using the *websave* function;
    if *kernel file type* = *LSK* | *SCLK* **then**
        Load the kernel using *cspice_furnsh* function (see Appendix **??**);
    end
end
et0 ← Convert the *utctim* to ephemeris time using *cspice_str2et* function (see Appendix **??**);
et ← Ephemeris time vector calculated by adding the *linspace* of the *NDAYS* converted to seconds to the *et0*;
for $i \leftarrow 1$ **to** *size(METAKR,1)* **do**
    if *kernel type=CK* **then**
        id ← Index values of the objects of which the kernel has information. They are calculated using the function *cspice_ckobj* (see Appendix **??**);
        Get the coverage window using *cspice_ckcov* (see Appendix **??**);
        Load the kernel if the time interval covered by the kernel is within the time interval defined using the function *cspice_furnsh*;
    end
    if *kernel type=SPK* **then**
        id ← Index values of the objects of which the kernel has information. They are calculated using the function *cspice_spkobj* (see Appendix **??**);
        ids ← Index value of the *missionname* calculated using the function *cspice_bodn2c* (see Appendix **??**);
        If the kernel which is being studied is a generic kernel, check if it has information either from the *target* or from the Earth. If it is so, load it using the function *cspice_furnsh*;
        If the kernel which is being studied has information about the spacecraft, get the coverage window using *cspice_spkcov* (see Appendix **??**);
        Load the kernel if the time interval covered by the kernel is within the time interval defined using the function *cspice_furnsh*;
    end
    If the kernel type is not an SPK nor a CK, load the kernel using *cspice_furnsh* function;
end
return *METAKR;*

5.2.4 *CreateMETAKR.m*

Algorithm 2: *CreateMETAKR.m*

1 **Input data:** mission name (*missionname*);
2 *missname* ← Cell structure having all the four possible mission names to be studied (i.e. {'CASSINI','VOYAGER 1','VOYAGER 2','NEW HORIZONS'});
3 *id* ← Position of the *missname* where the *missionname* is located. It is calculated using the function *strcmp* for every name of the *missname* cell;
4 *METAKR* ← Mx2 cell structure, where M is the number of kernels it contains. The first column contains the URL of the kernel and the second column, the name of the kernel;
5 *METAKERNEL* ← Cell structure containing the four possible *METAKR*, each one corresponding at its mission name;
6 *METAKER* ← Store the *METAKERNEL{id}*. It is the *METAKR* cell of the spacecraft which is being studied **return** *METAKER*;

Chapter 6

Observations design

In this chapter, the design process of the observations that would have to be made by different interplanetary missions considering different variables is described, as well as the code developed to get a first estimation of the moment to do these observations.

For the Cassini mission, two different ways of designing observations have been considered to be able to compare them, one taking into account the attitude of the camera during the trajectory of the probe and the other regardless of whether the camera was pointing to the target or not. Thus, a comparison between them will be done[1].

For the New Horizons and the Voyager mission, the observations have only been designed without considering the camera attitude.

6.1 Introduction

To design the observations, the first thing that has to be done is to decide with what conditions they have to be done.

In this project it has been decided to design an observation with each of the conditions that can be seen below. Always keeping in mind that at the moment in which one of these events occurs, the probe is not occulted from the Earth.

- **Occultation:** it occurs when the planet on which the flyby is being made is between the spacecraft and the Earth. In this way, the spacecraft is hidden by the planet. This is a problem when designing observations since the images captured by the spacecraft are sent to the Earth by radio and if there is a planet between them, the radio signal doesn't reach the Earth. Therefore, while some type of occultation occurs, no images would have to be captured

[1]It was intended to do the same for the two Voyager probes, but it could not because the camera kernels (CK) of the Voyager missions do not provide us with enough information, as explained in the TFG of Martí Sierra Salvadó.

since they will not be able to be received on Earth.

- **Minimum distance:** when a flyby is being made, there is a moment when the distance between the spacecraft and the planet reaches a minimum. This distance is the *minimum distance*.
- **Maximum illumination:** during a flyby, the angle of incidence between the Sun's rays and the planet seen from the spacecraft changes. This angle reaches a maximum at some point of the flyby. This maximum angle of incidence is what in this project is called *maximum illumination*.
- **Transit:** it happens when a satellite of the planet on which the flyby is being made is between the spacecraft and the planet. Thus, if a picture of the planet is made at the moment of a transit, the satellite would also be seen on the image.

These events will be studied more closely by separate and the developed code to study each of them will be presented.

6.2 Developed software

As said before, two different ways to design the observations have been considered for the already completed mission Cassini. Therefore, there will be two different times to take a picture of the target with the maximum illumination, one considering whether the camera was pointing to the target and the other without taking it into account. This will also happen with the transits and with the minimum distance.

On the other hand, for the other two missions (New Horizons and Voyager), there will only be one moment designed to do each of the observations. Thus, there will only be one time to take a picture of the target with the maximum illumination, with a transit and from the minimum distance.

The general algorithm used to calculate the optimum moment for each of the observations explained in Section 6.1 is presented below. Keep in mind that the optimum moment is that in which the event occurs without occultation, i.e. if the maximum illumination occurs during an occultation, an observation will not be designed at that moment.

Algorithm 3: *ObservationsDesign.m*

1 Define the mission name (*missionname*), the target (*target*), the moon to calculate the transit with (*satellite*), the reference frame (*frame*), the aberration correction to be applied (*abcorr*), the barycenter of the observer from which the flyby is going to be plotted (*barycenter*) and the instrument name (*instname*, only in the case of the Cassini mission);
2 Define the data (*data*) and the number of days (*NDAYS*) that are going to be studied;
3 Load the necessary kernels using the function *initSpiced* (see Section 5.2.3);
4 Define the object name and type of all the solar system bodies related with *target* and the spacecraft using the function *getOBJ* (see Section 6.2.8);
5 Get the state of each object using the function *getSPS2* (the variables contained in the OBJ structure are explained later in Section 6.2.1);
6 **if** *missionname=='CASSINI'* **then**
7 Plot the two flybys with the most important events, one considering the camera attitude and the other without taking it into account with the function *PlotFlybyCam* (see Section 6.2.2);
8 Get the time to do the different observations with and without the camera with the function *ObservationTimeCam* (see Section 6.2.3);
9 Show the different times to do the observations using the MATLAB function *fprintf*;
10 **else**
11 Plot the flyby and the most important events using the function *PlotFlyby* (see Section 6.2.2);
12 Get the time to do the different observations using the function *ObservationTime* (see Section 6.2.3);
13 Show the different times to do the observations using the MATLAB function *fprintf*;
14 **end**
15 Unload the kernels using the function *cspice_kclear* (see Appendix **??**);

6.2.1 Object structure

First of all, to be able to calculate all the conditions aforementioned in Section 6.1, the state of the solar system objects and the different planetary constants of them during the time interval studied have to be calculated, as well as the state of the spacecraft.

Consequently, a function called *getSPS2* has been designed. This function returns an object cell structure that contains the variables shown below:

- **OBJ.r:** SPICE matrix of position. It is a Mx3 matrix where M is the length of the ephemeris time vector and the 3 columns correspond to the 3 different dimensions (x,y,z). The position is calculated in [km].

- **OBJ.v:** SPICE matrix of velocity. It is a Mx3 matrix where M is the length of the ephemeris time vector and the 3 columns correspond to the 3 different dimensions (v_x,v_y,v_z). The velocity is calculated in [km/s].
- **OBJ.lt:** SPICE light time vector.
- **OBJ.target:** SPICE ID code.
- **OBJ.type:** Type of body. 'S' for spacecrafts, 'P' for planets, 'M' for moons, 'E' for Earth and '0' for Sun. This variable is introduced manually.
- **OBJ.name:** SPICE name.
- **OBJ.RADII:** Vector of 3 radius (ellipsoid) as SPICE takes all solar bodies as ellipsoids unless a DSK is loaded[2].
- **OBJ.GM:** SPICE GM extracted from the PCK kernels.

The algorithm of the code developed to get the object cell structure is the following:

[2]In this project DSK kernels haven't been loaded since all solar system bodies have been computed as ellipsoids

Algorithm 4: *getSPS.m*

Input data: object SPICE name (*name*), object type (*type*, the different types of bodies are explained above), ephemeris time vector (*et*), reference frame (*frame*), aberration correction (*abcorr*) and the barycenter of the observer from which the flyby is going to be plotted (*observer*);
SPS ← Structure which will store all the variables said above;
SPS.name ← Store the *name* of the object in the extension *.name*;
target ← Get the SPICE ID of the object using the SPICE function *cspice_ bodn2c* (see Appendix **??**);
SPS.target ← Store the variable *target* of the object in the extension *.target*;
SPS.type ← Store the variable *type* of the object in the extension *.type*;
Get the state of the object within the time interval defined using the SPICE function *cspice_ spkezr* (see Appendix **??**);
d ← State vector(s) containing the position and the velocity at every time step relative to the *barycenter*. The first three rows are the position in the three axis and the next three rows are the velocity in their respective axis;
lt ← One-way light time between the object and the *barycenter*;
SPS.lt ← Store the variable *lt* in the extension *.lt*;
SPS.r ← Store the position vector(s) of the object in the extension *.r*;
SPS.v ← Store the velocity vector(s) of the object in the extension *.v*;
if *SPS.type=='S'* **then**
    Leave the extension *.RADII* void;
else
    Get the three-axis radius vector using the function *cspice_ bodvrd* (see Appendix **??**) and store it in the extension *.RADII*;
end
return *SPS;*

The object name and the object type given as input data are obtained using the function *getOBJ* (see Section 6.2.8).

6.2.2 Flyby

To be able to make a plot of a flyby, the most important thing is the position of the different solar system objects in the time interval during which the flyby happens. Thus, the most important kernels in this chapter will be the SPK. This doesn't mean that all the other kernels don't have to be loaded. In order to compute all the characteristics of the flyby, all types of kernels will be needed and, as said before, the information that an SPK kernel contains can not be obtained without having other types of kernels loaded in the program.

The flyby plot is an important tool when designing observations. On it, apart from seeing the spacecraft's trajectory around the planet and the orbit of all the planet's satellites during that period of time, the different events of which an observation wants to be made will be marked in order to have a clearer idea of

their sequence.

Two different functions have been designed in order to plot the flyby considering the camera (*PlotFlybyCam.m*) and without taking it into account (*PlotFlyby.m*). These two different functions are explained below.

A. PlotFlybyCam.m

As said before, for the Cassini mission the observations are designed in two different ways. Thus, for the Cassini mission, there will be two different flyby plots, one with the events calculated considering the attitude of the camera and the other without taking it into account. In addition, a plot comparing the two flyby plots will be done.

The code developed to make these three different plots is shown below.

Algorithm 5: *PlotFlybyCam.m*

1 **Input data:** ephemeris time vector (*et*), reference frame (*fixref*), objects structure (*OBJ*), mission name (*missionname*), moon of transit (*satellite*), the target (*target*) and the instrument name (*instname*);
2 Get the index of the *OBJ* structure where the spacecraft, the earth, the sun and the target are located and keep them into *sc*, *ei*, *si* and *trgt*, respectively using the function *getIndex* (see Section 6.2.8);
3 Plot the spacecraft trajectory in 3d using the function *plot3* and the position data from the spacecraft OBJ structure. It is *OBJ{sc}.r* (see Section 6.2.1);
4 Find the time interval when the spacecraft is occulted from Earth by the *target* using the *FindOccultations* function (see Section 6.2.4) and plot it on the trajectory as a black line;
5 Find the moment at which the illumination is maximum considering the camera attitude using the function *IlluminationCam* and the moment at which it is maximum without taking it into account using the function *FindIllumination* (see Section 6.2.6) and plot them on the trajectory as a black point and a red point, respectively;
6 Find all the time intervals when there is a transit from the *satellite* between the *target* and the spacecraft (without considering the camera attitude) using the *Transit* function (see Section 6.2.7) and plot them on the trajectory as blue lines;
7 Find all the time intervals when there is a transit from the *satellite* between the *target* and the spacecraft taking the camera attitude into account using the function *TransitCam* and without considering it with the function *FindTransit* (see Section 6.2.7) and plot them on the trajectory as red lines and blue lines, respectively;
8 Find the moment at which the distance between the spacecraft and the *target* is the minimum considering the camera attitude using the *MinDistCam* function and without taking it into account with the function *MinDist* (see Section 6.2.5) and plot them on the trajectory as a green cross and a magenta cross, respectively;
9 **for** $i \leftarrow 1$ **to** *length(OBJ)* **do**
10 Check if the OBJ is a moon or a planet and if it is so, plot its trajectory and find the time when the minimum distance between them and the spacecraft occurs using the *MinDist* function;
11 *mdi* $\leftarrow$ Index of the *et* vector when the minimum distance occurs;
12 Plot the moons and the planets as ellipsoids at *et(mdi)* using the MATLAB functions *ellipsoid* and *surf*;
13 **end**
14 **for** $i \leftarrow 1$ **to** *length(OBJ)* **do**
15 **if** *OBJ{i}.type=='P'* **then**
16 Calculate the rotation axis and plot it on the flyby plot (the method used to calculate the rotation axis is explained later);
17 Calculate the vectors pointing from the planet to the Sun and from the planet to the Earth at the minimum distance between the spacecraft and the *target* and plot them using the MATLAB function *quiver*;
18 **end**
19 **end**

The algorithm shown above only creates the flyby plot that serves to compare the moments in which the events mentioned previously in Section 6.1 occur considering the camera and without considering it. To make the plot but with only the events studied considering the camera, the code would be the same but when looking for the moments when the events occur (lines 5, 6 an 7 of *PlotFlybyCam.m*), the functions that may have to be called would only be the ones that take into account the attitude of the camera. These are: *IlluminationCam*, *TransitCam* and

MinDistCam.

B. PlotFlyby.m

To plot the flyby and the events mentioned on Section 6.1 of a specified planet from a determined spacecraft without taking the camera attitude into account, a function has been made. The pseudo-code of the function is shown below.

In this project, this function is used when the missions studied are the New Horizons or the Voyager.

Algorithm 6: *PlotFlyby.m*

Input data: ephemeris time vector (*et*), reference frame (*fixref*), objects structure (*OBJ*), mission name (*missionname*), moon of transit (*satellite*) and the target (*target*);
Get the index of the *OBJ* structure where the spacecraft, the earth, the sun and the target are located and keep them into *sc*, *ei*, *si* and *trgt*, respectively using the function *getIndex* (see Section 6.2.8);
Plot the spacecraft trajectory in 3d using the function *plot3* and the position data from the spacecraft OBJ structure. It is *OBJ{sc}.r* (see Section 6.2.1);
Find the time interval when the spacecraft is occulted from Earth by the *target* using the *FindOccultations* function (see Section 6.2.4) and plot it on the trajectory as a black line;
Find the moment at which the illumination is maximum using the *FindIllumination* (see Section 6.2.6) function and plot it on the trajectory as a red point;
Find all the time intervals when there is a transit from the *satellite* between the *target* and the spacecraft using the *Transit* function (see Section 6.2.7) and plot them on the trajectory as blue lines;
Find the moment at which the distance between the spacecraft and the *target* is the minimum using the *MinDist* function (see Section 6.2.5) and plot it on the trajectory as a magenta cross;
for $i \leftarrow 1$ **to** *length(OBJ)* **do**
    Check if the OBJ is a moon or a planet and if it is so, plot its trajectory and find the time when the minimum distance between them and the spacecraft occurs using the *MinDist* function;
    mdi $\leftarrow$ Index of the *et* vector when the minimum distance occurs;
    Plot the moons and the planets as ellipsoids at *et(mdi)* using the MATLAB functions *ellipsoid* and *surf*;
end
for $i \leftarrow 1$ **to** *length(OBJ)* **do**
    if *OBJ{i}.type=='P'* **then**
        Calculate the rotation axis and plot it on the flyby plot using the MATLAB function *quiver*;
        Calculate the vectors pointing from the *target* to the Sun and from the *target* to the Earth at the minimum distance between the spacecraft and the *target* and plot them using the MATLAB function *quiver*;
    end
end

Rotation axis

To calculate the rotation axis of the target, the first thing that has to be defined is the body-fixed frame of it. Once we have the frame of the planet, we have to define the vector that is pointing towards the north pole of it (i.e. [0 0 1]) and convert it to the reference system with which we are calculating the positions of all the objects using a transformation matrix that is achieved using the SPICE function cspice_pxform. Then, the angle between the two vectors has to be calculated. This angle is the the angle of the axis of rotation with respect to the reference system in which the position of all the objects is defined.

In order to plot the rotation axis in the flyby plot what has to be done is to double the vector pointing to the north of the planet transformed into the reference system in which the plot is defined and to plot it using the MATLAB function *quiver*. To see the rotation axis coming out both in the north and in the south of the planet, the method has to be repeated but with the vector in the opposite direction.

6.2.3 Observation time

Once the flyby plot is done, it remains to calculate the best moment to make each observation. The best moment to do each of the observations has to be understood as that in which there is no occultation. The code to do the flyby plot calculates the moments for which the events mentioned in Section 6.1 occur, but it does not take into account if at those moments there was an occultation or not.

As said before, three different observations want to be designed, one with the maximum illumination (i.e. the minimum phase angle), one from the minimum possible distance and one of the transit of a moon of the studied planet decided by the user, seen from the minimum possible distance.

That is why, during the time interval for which there is occultation, the values of the three variables that determine the best time to do each observation have to be defined as the opposite value to the one that determines it.

Another thing that has been considered is if, at the time of designing the observations from the minimum distance and of a transit, the phase angle is greater than the one defined by the user. This way, the user can restrict the minimum illumination with which the observations will be made.

As also happens with flyby plots, there will be two different functions, one to calculate the best moment for each observation considering the attitude of the camera (*ObservationTimeCam*) and another function that will not take it into account (*ObservationTime*).

A. *ObservationTimeCam.m*

To calculate the best moment to do each of the observations considering the camera attitude, apart from discarding the time interval at which there is occultation, all the moments at which the camera was not pointing towards the target must also be discarded.

To calculate whether the camera was pointing to the planet or not, the name of the instrument with which the observation wants to be made has to be defined. The SPICE function that computes it is the *cspice_sincpt* (see Appendix **??** for more information about this function).

The algorithm of the function is shown below.

Algorithm 7: *ObservationTimeCam.m*

1 **Input data:** ephemeris time vector (*et*), objects structure (*OBJ*), spacecraft name (*missionname*), target name (*target*), name of the satellite with which the transit wants to be calculated (*satellite*) and the instrument name (*instname*);
2 Get the index of the *OBJ* structure where the spacecraft, the earth, the sun and the target are located and keep them into *sc*, *ei*, *si* and *trgt*, respectively using the function *getIndex* (see Section 6.2.8);
3 *ocultid:* Matrix Mx2, where M is the number of occultations, having on the first column the index of the *et* vector when the occultation begins and on the second column the index when it ends. It is calculated using the function *FindOccultations* (see Section 6.2.4). If there is no occultation during the trajectory, *ocultid* is a single number (it is important to understand line 16);
4 Find illumination parameters considering the camera attitude using the function *Illumination* (see Section 6.2.6);
5 *fnd:* Logical values vector of the same length as *et* indicating whether the camera was pointing to the *target*;
6 *phase:* Vector of the same length as *et* containing the phase angle at each moment of the trajectory considering the camera attitude;
7 *idminfasang:* Index of the *phase* vector where the minimum phase angle is located,considering the camera attitude;
8 *distcam:* Vector of the same length as *et* containing the distance at each moment of the trajectory computed using the function *getD* (see Section 6.2.8);
9 *transtcam:* Logical values vector indicating whether there is transit. It is calculated using the function *TransitCam*(see Section 6.2.7);
10 *maxangacc:* Define a maximum phase angle since otherwise nothing would be seen on the picture;
11 **for** $i \leftarrow 1$ **to** *length(et)* **do**
12 **if** *phase(i)>maxangacc* ***or*** *fnd==0* **then**
13 Set *distcam(i)* as the maximum value of *distcam* and set *transtcam(i)* as a 0;
14 **end**
15 **end**
16 **if** *size(ocultid,2)=2* **then**
17 **for** $i \leftarrow 1$ **to** *size(ocultid,1)* **do**
18 Set distcam(ocultid(1,1):ocultid(1,2)) as the maximum value of *distcam*, set phase(ocultid(1,1):ocultid(1,2)) as the maximum value of *phase* and set transtcam(ocultid(1,1):ocultid(1,2)) as 0;
19 **end**
20 Get the index of the *et* vector when the minimum distance and the minimum phase angle happen using the function *min* of the vectors *distcam* and *phase* respectively;
21 Get all time intervals at which there is a transit of the *satellite* using the function *getInterval* (see Section 6.2.8) and keep the one where the distance is the minimum;
22 Get the index of the *et* vector doing the average between the index when the transit starts and the index when it ends. This way, the *satellite* will be at the middle of the transit;
23 **else**
24 Get the index of the *et* vector when the minimum distance and the minimum phase angle happen using the function *min* of the vectors *distcam* and *phase* respectively;
25 Get all time intervals at which there is a transit of the *satellite* using the function *getInterval* (see Section 6.2.8) and keep the one where the distance is the minimum;
26 Get the index of the *et* vector doing the average between the index when the transit starts and the index when it ends. This way, the *satellite* will be at the middle of the transit;
27 **end**
28 **return** *timeMaxIlluminCam,timeMinDistCam,timeTransitCam;*

B. *ObservationTime.m*

To calculate the best moment to do each of the observations without taking the camera attitude into account, only the time interval during which there is occultation will be discarded, it does not matter whether the camera was pointing to the target.

Another thing that has been considered, as in the case of *ObservationTimeCam* is if, at the time of designing the observations from the minimum distance and of a transit, the phase angle is greater than the one defined by the user. This way, the user can restrict the minimum illumination with which the observations will be made.

Below, the algorithm of this function is shown.

Algorithm 8: *ObservationTime.m*

1 **Input data:** ephemeris time vector (*et*), object structure (*OBJ*), spacecraft name (*missionname*), target name (*target*) and name of the satellite with which the transit wants to be calculated (*satellite*);
2 Get the index of the *OBJ* structure where the spacecraft, the earth, the sun and the target are located and keep them into *sc*, *ei*, *si* and *trgt*, respectively using the function *getIndex* (see Section 6.2.8);
3 *ocultid:* Matrix Mx2, where M is the number of occultations, having on the first column the index of the *et* vector when the occultation begins and on the second column the index when it ends. It is calculated using the function *FindOccultations* (see Section 6.2.4). If there is no occultation during the trajectory, *ocultid* is a single number (it is important to understand line 15);
4 Find illumination parameters considering the camera attitude using the function *FindIllumination* (see Section 6.2.6);
5 *fasang:* Vector of the same length as *et* containing the phase angle at each moment of the trajectory considering the camera attitude;
6 *idmaxilumin:* Index of the *phase* vector where the minimum phase angle is located,considering the camera attitude;
7 *dist:* Vector of the same length as *et* containing the distance at each moment of the trajectory computed using the function *getD* (see Section 6.2.8);
8 *transt:* Logical values vector indicating whether there is transit. It is calculated using the function *Transit*(see Section 6.2.7);
9 *maxangacc:* Define a maximum phase angle since otherwise nothing would be seen on the picture;
10 **for** $i \leftarrow 1$ **to** *length(et)* **do**
11 **if** *fasang(i)>maxangacc* **then**
12 Set *dist(i)* as the maximum value of *dist* and set *transt(i,1)* as a 0;
13 **end**
14 **end**
15 **if** *size(ocultid,2)==2* **then**
16 **for** $i \leftarrow 1$ **to** *size(ocultid,1)* **do**
17 Set dist(ocultid(1,1):ocultid(1,2)) as the maximum value of *dist*, set fasang(ocultid(1,1):ocultid(1,2)) as the maximum value of *phase* and set transt(ocultid(1,1):ocultid(1,2)) as 0;
18 **end**
19 Get the index of the *et* vector when the minimum distance and the minimum phase angle happen using the function *min* of the vectors *dist* and *fasang* respectively;
20 Get all time intervals at which there is a transit of the *satellite* using the function *getTrans* (see Section 6.2.8) and keep the beginning and the ending index from the one where the distance is the minimum;
21 Get the index of the *et* vector doing the average between the index when the transit starts and the index when it ends. This way, the *satellite* will be at the middle of the transit;
22 **else**
23 Get the index of the *et* vector when the minimum distance and the minimum phase angle happen using the function *min* of the vectors *distcam* and *phase* respectively;
24 Get all time intervals at which there is a transit of the *satellite* using the function *getTrans* (see Section 6.2.8) and keep the starting and the ending index from the one where the distance is the minimum;
25 Get the index of the *et* vector doing the average between the index when the transit starts and the index when it ends. This way, the *satellite* will be at the middle of the transit;
26 **end**
27 **return** *timeMinDist,timeMaxIlumin,timeTransit;*

6.2.4 Occultation

An occultation is an event that occurs when one object is hidden by another object that passes between it and the observer [17], in this project, when the planet on which the flyby is being made is between the spacecraft and the Earth. In this way, the spacecraft is hidden by the planet. This is a problem when designing observations since the images captured by the spacecraft are sent to the Earth by radio waves and if there is a planet between them, the radio signal doesn't reach the Earth. Therefore, while some type of occultation occurs, no images would have to be captured since they will not be able to be seen on Earth.

To study the time interval on which an occultation occurs, there is a SPICE routine named *cspice_ occult* that determines the occultation condition (not occulted, partially, etc.) of one target (in this case it is always the Earth) relative to another target (the planet on which the flyby is being made) as seen by an observer (the spacecraft) at a given time (see Appendix **??** for more information). The targets can be modelled as points, ellipsoids, or digital shapes (DSK)[3].

The pseudo-code to find the occultation condition is the following:

[3]In this project all planets are modelled as ellipsoids as not all planets have their own DSK

Algorithm 9: *FindOccultations.m*

Input data: ephemeris time vector *et*, object structure of spacecraft (OBJ{sc}), target (OBJ{trgt}) and Earth (OBJ{ei});
ocltid ← vector of the same length as the ephemeris time vector that stores the occultation ID of each time step;
for $i \leftarrow 1$ **to** *length(et)* **do**
    Get the occultation ID for each time using the *cspice_ occult* function (see Appendix **??**) and store it in *ocltid(i)* (the different numbers *cspice_ occult* returns are explained later; to understand the following code, it has to be kept in mind that a number different from 0 means an occultation);
end
if *max(abs(ocltid))>0* **then**
    Find all series of consecutive numbers different from zeros, they correspond to all the occultations. This id done with the function *getInterval* (see Section 6.2.8);
    ocultid ← Mx2 matrix, where M is the number of occultations. The value of first column is the first index of the occultation and the value of the second column is the last index of the occultation;
else
    ocultid=0, it is a single number, not a matrix with two columns. It is important to understand the next command;
end
if *size(ocultid,2)==2* **then**
    for $i \leftarrow 1$ **to** *size(ocultid,1)* **do**
        Convert the time when the occultation starts and finishes to UTC time using the *cspice_ et2utc* function (see Appendix **??**);
        Show on the monitor the initial and the final data of the occultation;
    end
else
    Convert the initial and the final time of the ephemeris time vector to UTC time using the *cspice_ et2utc* function and store them on *UTC1* and *UTCend*;
    Show on the monitor the following phrase: There is no occultation from *UTC1* to *UTCend*.
end
return *ocultid,ocltid;*

The function *cspice_ occult* returns a number in the range [-3,3]. The meaning of each number is explained in the table below:

ID code	Meaning
-3	Total occultation of first target by second
-2	Annular occultation of first target by second
-1	Partial occultation of first target by second
0	No occultation
1	Partial occultation of second target by first
2	Annular occultation of second target by first
3	Total occultation of second target by first

Table 6.1: ID codes for occultations.

6.2.5 Minimum distance

The minimum distance is the one at which the planet and the spacecraft are as close as possible. This occurs during the flyby of the target. Thus, if the distance at every position of the spacecraft's flyby trajectory is calculated, it is possible to know the moment of the flyby when this event occurs. Therefore, an observation of the planet from the closest distance can be designed.

As with all other events, two different functions have been created, one to calculate the moment when the minimum distance occurs considering whether the camera pointed to the target (*MinDistCam.m*) and another to calculate it without taking that into account (*MinDist.m*).

A. *DistCam.m*

To calculate the minimum distance taking into account the attitude of the camera, what has been done is to define all the distance values during which the camera was not pointing to the planet as the maximum distance value between the spacecraft and the target in the time interval studied.

This is done because the distance we are looking for is the minimum, thus, defining all the distances in which the planet was not targeted as the maximum among the possible ones, surely there will not be a minimum distance when the camera does not point to the planet.

This is achieved by giving as an input the *fnd* vector of the *IlluminationCam* function (see Section 6.2.6) which is a vector of logical values where the value 1 indicates that the camera was pointing to the planet and the value 0 indicates the opposite.

The algorithm of the code developed is the following.

Algorithm 10: *MinDistCam.m*

1 **Input data:** ephemeris time vector (*et*), object structure of spacecraft (*OBJ{sc}*) and the body with which the minimum distance wants to be calculated (*OBJ{trgt}*), and *fnd* vector from *IlluminationCam* function (see Section 6.2.6);
2 *distcam* ← Calculate the distance between the spacecraft and the target body at every time using the function *getD* (see Section 6.2.8);
3 *maxdistcam* ← Calculate the maximum distance that occurs during the time interval studied using the MATLAB function *max*;
4 **for** $i \leftarrow 1$ **to** *length(et)* **do**
5 | If the camera is not pointing to the planet (i.e. *fnd(i)=0*), set the distance at that moment as *maxdistcam* (i.e. *distcam(i)=maxdistcam*);
6 **end**
7 Calculate the minimum distance (*mindistcam*) and the position of the vector *et* where it is located (*idmindistcam*) using the MATLAB function *min* of the vector *dist*;
8 **return** *distcam,mindistcam,idmindistcam;*

B. *MinDist.m*

To calculate the minimum distance without considering whether the camera was pointing to the target, the method is the same as with *MinDistCam.m* but without setting the distance of the moments at which the camera was not pointing to the maximum distance. In this way, the minimum distance may occur at a time when the camera was not pointing towards the target.

Algorithm 11: *MinDist.m*

1 **Input data:** ephemeris time vector (*et*), object structure of spacecraft (*OBJ{sc}*) and the body with which the minimum distance wants to be calculated (*OBJ{trgt}*);
2 *dist* ← Calculate the distance between the spacecraft and the target body at every time using the function *getD* (see Section 6.2.8);
3 Calculate the minimum distance (*mindist*) and the position of the vector *et* where it is located (*idmindist*) using the MATLAB function *min* of the vector *dist*;
4 **return** *dist,mindist,idmindist;*

6.2.6 Maximum illumination

While a flyby is being carried out, the phase angle varies. It is the angle between the vector pointing from the observer to a specified surface point and the vector pointing from that point to the Sun, To understand it better, the figure 6.1 shows the three different angles formed at the point of the surface that is being observed

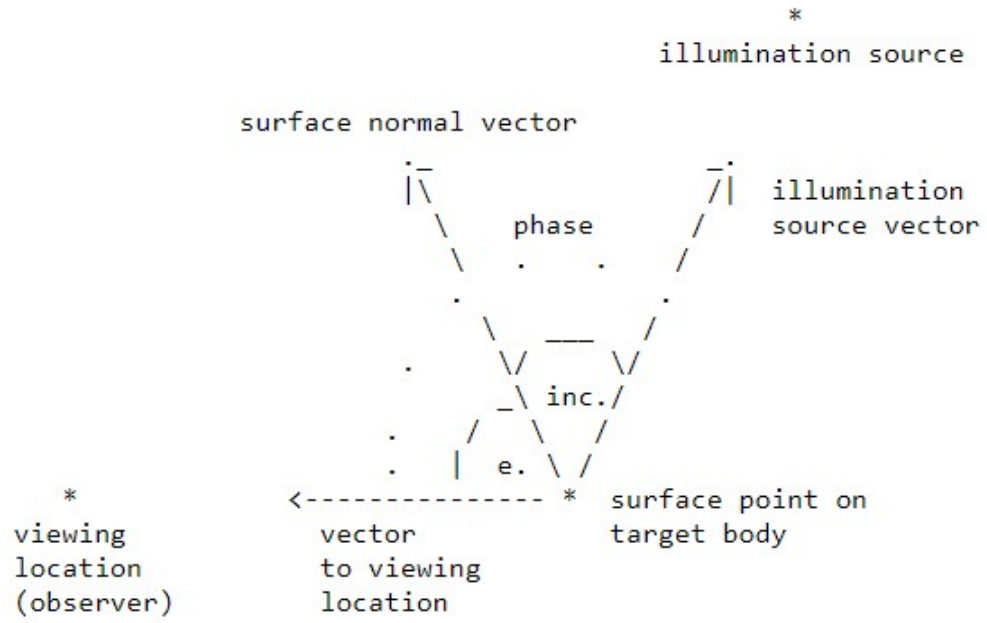

Figure 6.1: Phase, emission and incidence angles [2]

between the vector that points from the spacecraft to the point and the vector that points between the sun and the point.

If the phase angle is calculated at every moment of the flyby, it will reach a minimum at some point of it. This minimum phase angle means that the Sun is illuminating the maximum surface that can be seen from the spacecraft.

As said before, the maximum illumination is taken in this project as a condition to do an observation. Therefore, if the moment of the trajectory of the spacecraft when the planet is most illuminated is known, it will be feasible to make an observation with the maximum illumination possible during the flyby.

In this project, the moment to do an observation with the maximum illumination has been calculated by two different ways. The first does not take into account the attitude of the camera during the trajectory, since it is about designing the observation without the restrictions of the camera, i.e. without considering whether at the moment of maximum illumination the camera was pointing to the planet or not (*FindIllumination.m*). In the second method, the attitude of the camera has been considered to know when it would have been possible to make an observation knowing the moments in which the camera pointed to the planet (*IlluminationCam.m*).

A. *FindIllumination.m*

On the one hand, the maximum illumination has been calculated only taking into account the position of the observer, the target and the Sun. This is due to the fact that, as first approach, the attitude of the camera during the trajectory of the spacecraft has not been considered.

In this way, the phase angle has been calculated as the minimum angle between the vectors pointing from the observer and from the Sun to the surface point, which is the intersection between the vector pointing from the spacecraft to the center of the target and the surface of the planet.

The algorithm of the code developed to calculate the maximum illumination (i.e. the minimum phase angle) and the time it happens without the camera restriction is shown below.

Algorithm 12: *FindIllumination.m*

Input data: ephemeris time vector (*et*), Object structure of the spacecraft (*OBJ{sc}*), the target (*OBJ{trgt}*) and the sun (*OBJ{si}*);
for $i \leftarrow 1$ **to** *length(et)* **do**
    scpos ← Position of the spacecraft at *et(i)*;
    trgtpos ← Position of the target at *et(i)*;
    sctrgtv ← Calculate the vector pointing from the spacecraft to the target as *trgtpos-scpos*;
    Get the surface point where *sctrgtv* intersects the surface of the target using the SPICE function *cspice_ surfpt* (see Appendix **??**);
    point ← Surface point where *sctrgtv* intersects with the surface of the target;
    found ← Logical values vector of the same length as *et* indicating whether *sctrgtv* intersects the surface of the target or not;
    if *found==1* **then**
        sunpointv ← Vector pointing from the Sun to *point*;
        scpointv ← Vector pointing from the spacecraft to *point*;
        phase ← Minimum angle between *sunpointv* and *scpointv* calculated using the MATLAB function *acos*;
        fase ← Mx2 matrix, where M is the number of positions when *sctrgtv* intersects the surface of the target (i.e. when *found=1*). The first column of the matrix is the value of *phase* and the second column is the index of the *et* vector where it is located;
    end
end
Find the minimum phase angle *maxilumin* and the index value of the vector *fase* when it happens *idfase* using the MATLAB function *min*;
Get the index value of the vector *et* when the maximum illumination occurs as *idmaxilumin=fase(idfase,2)*;
return *found, fasang, maxilumin, idmaxilumin*

B. *IlluminationCam.m*

On the other hand, the maximum illumination has been computed considering, in addition to the variables of the method shown above, the attitude of the camera during the trajectory of the spacecraft.

This has been done to compare the moment of maximum illumination considering whether the camera was pointing to the planet or without taking it into account. The results will be discussed later on Chapter 8.

In this way, the illumination is only computed when the boresight vector of the camera intersects with the surface of the target. The algorithm developed to do so is shown below.

Algorithm 13: *IlluminationCam.m*

Input data: ephemeris time vector (*et*), object structure of the spacecraft (*OBJ{sc}*) and the target (*OBJ{trgt}*) and instrument name (*instname*);
Get the instrument SPICE ID code using the function *cspice_ bodn2c* (see Appendix **??**);
instid ← Instrument SPICE ID code;
found ← Logical value indicating whether the instrument name is found on the kernel pool or not;
if *found==0* **then**
    Display the following error: Unable to determine ID code for: *instname*;
end
Get the Field-of-View (FOV) of the instrument using the function *cspice_ getfov* (see Appendix **??**);
boresight ← Boresight vector of the *instname*;
for *i* ← 1 **to** *length(et)* **do**
    Get the surface interception point between the *boresight* and the planet using the *cspice_ sincpt* function (see Appendix **??**);
    spoint ← Surface point where the boresight vector intersects the planet surface;
    fnd ← Vector of the same length as *et* containing logical values indicating whether the boresight vector intersects the planet surface at an specified *et*;
    srfvec ← Vector pointing from the observer to the *spoint*;
    if *fnd==1* **then**
        Calculate the illumination angles (in radians) using the *cspice_ ilumin* function (see Appendix **??**);
        phase ← Angle between the vector *Illumination source-spoint* and the vector *srfvec*;
        emission ← Angle between the vector *srfvec* and the surface normal vector at *spoint*;
        incidence ← Angle between the surface normal vector at *spoint* and the vector *Illumination source-spoint*;
        fasang ← Mx2 matrix, where M is the number of positions when the boresight vector emanating from the spacecraft intersects the planet surface (i.e. when *fnd==1*). The first column of the matrix is the value of *phase* and the second column is the index of the *et* vector where it is located;
        emisang ← Same as *fasang* but with the *emission* angle;
        incdncang ← Same as *fasang* but with the *incidence* angle;
    end
end
Find the minimum phase angle *minfase* and the index value of the vector *fasang* when it happens *idfase* using the MATLAB function *min*;
Get the index value of the vector *et* when the maximum illumination occurs as *idminfasang=fase(idfase,2)*;
return *fond,fasang,emisang,incdncang,minfasang,idminfasang;*

6.2.7 Transit

In astronomy, a transit occurs when a celestial body passes directly between a larger body and the observer[18]. In this project, the transit condition has been considered when a satellite of the planet on which the flyby is being made is between the spacecraft and the planet. Thus, if an observation of the planet is made at the moment of a transit, the satellite would also be seen on the image.

During a flyby of a planet, more than one transit of the selected moon will occur. That is why, in this project, it has been decided that the transit from which the observation is going to be designed is the one that occurs when the spacecraft is closest to the planet, that is, once all the transits are known, the one that meets the minimum distance condition mentioned above will be chosen as the one to make the picture of.

As with all the other events of which an observation wants to be designed in this project, it has been designed by two different ways.

The first one does not take into account the camera attitude during the time interval studied (*FindTransit.m*), i.e. it does not consider whether the camera was pointing to the target at the moment when the transit of which the observation wants to be made occurs.

In the second method, the attitude of the camera has been considered to know when it would have been possible to make an observation knowing the moments in which the camera pointed to the planet (*TransitCam.m*).

A. *FindTransit.m*

As said above, the first method has been computed without considering the camera attitude during the trajectory of the spacecraft. This way, it does not matter whether the camera is pointing towards the planet at the moment to calculate if there is a transit or not.

The algorithm of the code developed to calculate all time intervals when there is a transit of the moon chosen by the user is shown below.

Algorithm 14: *FindTransit.m*

Input data: ephemeris time vector (*et*), object structure (*OBJ*), spacecraft name (*missionname*), target name (*target*) and satellite name (*satellite*);
Get the index of the *OBJ* structure where the spacecraft, the planet and the satellite are located and keep them into *sc*, *trgt* and *sat*, respectively using the function *getIndex* (see Section 6.2.8);
fixref ← Body-fixed reference frame of the target;
for *i* ← 1 **to** *length(et)* **do**
    Transform the position of the spacecraft and the satellite into the *fixref* frame multiplying them by the transformation matrix calculated using the *cspice_pxform* function (see Appendix **??**);
    scsatv ← Vector pointing from the spacecraft to the satellite in the *fixref* frame coordinates;
    Calculate the planet surface point where the *scsatv* intersects using the *cspice_surfpt* function (see Appendix **??**);
    spoint ← Surface intersecting point calculated before;
    found ← Logical value indicating whether the *scsatv* vector intersects with the planet surface or not;
    Calculate the distance between the spacecraft and the planet (*distscplanet*) and the distance between the spacecraft and the satellite (*distscsat*);
    transt ← Logical values vector indicating whether there is a transit;
    if *found==1* ***and*** *distscplanet>distscsat* **then**
        transt(i)=1;
        pclong ← Calculate the planetocentric longitude of *spoint* using the *cspice_reclat* function (see Appendix **??**);
        pclatit ← Calculate the planetocentric latitude of *spoint* using the *cspice_reclat* function (see Appendix **??**);
        The longitude and the latitude are calculated in order to compare them with the ones of the images navigated by Martí Sierra Salvadó in his TFG;
    end
end
trans ← Mx2 matrix, where M is the number of transits that occur within the time interval studied. In the first column there is the index of the *et* vector when the transit starts and in the second column the index when it ends. These are calculated using the function *getInterval* (see Section 6.2.8);
return *transt,trans,pclong,pclatit;*

B. *TransitCam.m*

The second method to calculate all time intervals when there is a transit of the moon which has been chosen by the user takes into account the camera attitude.

This way, all transits that occur when the camera is not pointing towards the planet will be rejected, leaving only those in which the camera points to the target.

The algorithm of the code developed to calculate these transits and reject all transits in which the camera is not pointing to the planet is shown below.

Algorithm 15: *TransitCam.m*

Input data: ephemeris time vector (*et*), object structure (*OBJ*), spacecraft name (*missionname*), target name (*target*), satellite name (*satellite*) and instrument name (*instname*);
Get the index of the *OBJ* structure where the spacecraft, the planet and the satellite are located and keep them into *sc*, *trgt* and *sat*, respectively using the function *getIndex* (see Section 6.2.8);
fixref ← Body-fixed reference frame of the target;
Get the instrument SPICE ID code using the function *cspice_ bodn2c* (see Appendix **??**);
instid ← Instrument SPICE ID code;
found ← Logical value indicating whether the instrument name is found on the kernel pool or not;
if *found==0* **then**
    Display the following error: Unable to determine ID code for: *instname*;
end
Get the Field-of-View (FOV) of the instrument using the function *cspice_ getfov* (see Appendix **??**);
boresight ← Boresight vector of the *instname*;
for *i* ← 1 **to** *length(et)* **do**
    fnd ← Vector of the same length as *et* containing logical values indicating whether the *boresight* vector intersects the planet surface at an specified *et* using the function *cspice_ sincpt* (see Appendix **??**);
    Transform the position of the spacecraft and the satellite into the *fixref* frame multiplying them by the transformation matrix calculated using the *cspice_ pxform* function (see Appendix **??**);
    scsatv ← Vector pointing from the spacecraft to the satellite in the *fixref* frame coordinates;
    Calculate the planet surface point where the *scsatv* intersects using the *cspice_ surfpt* function (see Appendix **??**);
    spoint ← Surface intersecting point calculated before;
    found ← Logical value indicating whether the *scsatv* vector intersects with the planet surface or not;
    Calculate the distance between the spacecraft and the planet (*distscplanet*) and the distance between the spacecraft and the satellite (*distscsat*);
    transtcam ← Logical values vector indicating whether there is a transit;
    if *found==1* ***and*** *distscplanet>distscsat* ***and*** *fnd(i)==1* **then**
        transtcam(i)=1;
        pclongcam ← Calculate the planetocentric longitude of *spoint* using the *cspice_ reclat* function (see Appendix **??**);
        pclatitcam ← Calculate the planetocentric latitude of *spoint* using the *cspice_ reclat* function (see Appendix **??**);
        The longitude and the latitude are calculated in order to compare them with the ones of the images navigated by Martí Sierra Salvadó in his TFG;
    end
end
transcam ← Mx2 matrix, where M is the number of transits that occur within the time interval studied. In the first column there is the index of the *et* vector when the transit starts and in the second column the index when it ends. These are calculated using the function *getInterval* (see Section 6.2.8);
return *transtcam,transcam,pclongcam,pclatitcam;*

6.2.8 Complementary functions

In addition to the codes mentioned above, some functions have also been developed to streamline the process of designing the observations. These will be explained below.

A. getD.m

This function gets the distance between two bodies at all specified ephemeris time. The ephemeris time can be defined as a vector or as a simple value.

Algorithm 16: *getD.m*

1 **Input data:** first body (A), second body (B);
2 $rAB \leftarrow$ Get the vector pointing from B to A using the function *getR*;
3 *dist* $\leftarrow$ Get the distance from B to A doing the square root of the vector rAB using the MATLAB function *sqrt*;
4 **return** *dist;*

B. getR.m

This function gets the vector pointing from a body to another body.

Algorithm 17: *getR.m*

1 **Input data:** object structure of the first body (A) and the second body (B);
2 $rAB \leftarrow$ Vector pointing from B to A calculated as the position of B minus the position of A;

C. getOBJ.m

This function creates a structure that contains the name and type of object of each body relate that wd to tants the targeto be studied. In addition, it also contains the name of the spacecraft.

To do so, you have to enter the target that wants to be studied and the mission to work with.

Algorithm 18: *getOBJ.m*

1 **Input data:** mission name (*missionname*) and target name (*target*);
2 *targets* ← cell of strings that contains the different names of *target* that can be entered (the different targets that can be studied and all the solar system bodies related with them are explained before);
3 *OBJ* ← cell that contains the same number of structures as *targets*. Each structure has two columns. In the first column there is the name of the body and in the second column it can be found its type (see Section 6.2.1);
4 *id* ← get the index of the cell *targets* where the name of the *target* is located;
5 *OBJECT* ← structure of the cell *OBJ* located in the position *id*, i.e. the structure containing the names and types of the bodies related to *target*;
6 **return** *OBJECT*;

In this project, all codes have been made considering that there may be three different targets. According to the missions studied, these targets are: Jupiter, Saturn and Pluto.

The different names and the types of object (see Section 6.2.1) of each body of the solar system related to each target is shown on Table 6.2.

Jupiter		**Saturn**		**Pluto**	
'SUN'	'0'	'SUN'	'0'	'SUN'	'0'
'EARTH'	'E'	'EARTH'	'E'	'EARTH'	'E'
Spacecraft	'S'	*Spacecraft*	'S'	*Spacecraft*	'S'
'JUPITER'	'P'	'SATURN'	'P'	'PLUTO'	'P'
'IO'	'M'	'MIMAS'	'M'	Charon	'M'
'EUROPA'	'M'	'ENCELADUS'	'M'		
'GANYMEDE'	'M'	'TETHYS'	'M'		
'CALLISTO'	'M'	'DIONE'	'M'		
'AMALTHEA'	'M'	'RHEA'	'M'		
'THEBE'	'M'	'TITAN'	'M'		
'ADRASTEA'	'M'	'HYPERION'	'M'		
'METIS'	'M'				

Table 6.2: Names and types of objects

D. getIndex.m

This function, given a name and the object structure, finds the position where the data of that body are stored. This is done to have located, within the object structure, where are the data related to the satellite or the target, among others.

Algorithm 19: *getIndex.m*

1 **Input data:** name of the object (*name*) and object structure (*OBJ*);
2 **for** $i \leftarrow 1$ **to** *length(OBJ)* **do**
3 | Compare the *name* with the variable *OBJ{i}.name* using the MATLAB function *strcmp* (see Section 6.2.1);
4 | If the *name* is the same as the *OBJ{i}.name*, save the index where it happens to *id*;
5 **end**
6 If there is no match for any index, send an error message saying that the object has not been entered;
7 **return** *id*;

E. getInterval.m

This function, given a vector of logical values or similar, creates a matrix Mx2, where M is the number of series of numbers different from 0 that the vector contains. The first column of the matrix is the first index of the series and the second column is the last index of the series.

Algorithm 20: *getInterval.m*

1 **Input data:** vector of logical values (*transt*);
2 *trans* $\leftarrow$ Matrix Mx2, where M is the number of series of numbers different from 0 that *transt* contains;
3 **if** *max(abs(transt)) =0* **then**
4 | Find all series of numbers different from 0 and save the first index of the series on the first column of the *trans* matrix and the last index on the second column of the same matrix;
5 **else**
6 | trans=0;
7 **end**
8 **return** *trans;*

Chapter 7

Mercury transit

In this chapter, an explanation of the algorithm that has been made to calculate the time interval during which the Mercury transit will take place, will be made.

7.1 Introduction

The Mercury transit is expected to begin on November 11 in 12:34 UTC and end on 18:04 UTC [27, 19]. Then, in Chapter 8, this time will be compared with the time calculated for the algorithm that has been created.

A transit of Mercury across the Sun happens when the planet Mercury is located between the Sun and a larger planet, becoming visible against the solar disk.

As seen from Earth, only transits of Mercury and Venus are possible since they are closer to the Sun. Transits of Mercury with respect to Earth are much more frequent than transits of Venus, with about 13 or 14 per century, in part because Mercury is closer to the Sun and orbits it more rapidly [31].

7.2 Developed software

To calculate the beginning and the ending of the Mercury transit that, theoretically, will occur on November of 2019, an algorithm has been developed.

This algorithm calculates the transit from the vector that points from the center of the Earth to the center of Mercury. Thus, when this vector intersects with the surface of the Sun, it means that Mercury will be doing a transit. Therefore, if this condition is calculated for all time instants, the time interval during which the transit occurs will be obtained.

Algorithm 21: *MercuryTransit.m*

1 Define the mission name as (*'MERCURY TRANSIT'*) and the target *target*, in this case (*'MERCURY'*);
2 Define the UTC time *utctim* and the number of days *NDAYS* that are going to be studied;
3 Load all necessary kernels using the function *initSpiced* (see Chapter 5.2.3);
4 Define the reference frame in which the solar system bodies position will be studied (*frameref*) and the body-fixed reference frame of the body from which the transit wants to be seen (*frame2*), in this case (*'IAU_EARTH'*);
5 Define the barycenter of the Sun (*barycenter*);
6 Get the position of Sun, Mercury and Earth in the *fixref* using the function *cspice_spkezr* (see Appendix **??**);
7 Get the three different radius of the Sun, as it is modelled as an ellipsoid, using the function *cspice_bodvrd* (see Appendix **??**);
8 **for** $i \leftarrow 1$ **to** *length(et)* **do**
9 | *earthmercvec* ← Vector pointing from the Earth to Mercury at every time step;
10 | Find if the *earthmercvec* intersects with the surface of the Sun using the function *cspice_surfpt* (see Appendix **??**);
11 | *found* ← Logical values vector indicating whether the vector *earthmercvec* intersects the surface of the Sun or not;
12 **end**
13 *trans* ← Mx2 matrix, where M is the number of transits that occur during the time interval studied. In the first column, there is the index of the *et* vector when the transit starts. In the second column, there is the index of the *et* vector when the transit ends. This matrix is obtained using the function *getInterval* (see Section 6.2.8);

Chapter 8

Results

In this chapter, the results obtained with the different algorithms developed are presented.

The observations have been designed for four different spacecraft, that is why the results of each mission will be discussed separately.

The observations of each spacecraft have been computed for each planet in which the probe made a flyby, with the exception of Voyager 2, for which only the observations on Jupiter and Saturn have been designed, leaving the observations in the planets Uranus and Neptune.

The results obtained with the algorithm *MercuryTransit* are also shown and discussed.

8.1 Observations design

In this section, the different moments to do the various observations obtained with the different spacecraft are shown, as well as a discussion of them. Also, a comparison between the trajectory obtained with the algorithm created and the real trajectory is done.

First of all, the flyby plot will be shown with the most important events marked on it. The most important events are:

- Minimum distance: it is marked with a magenta cross in the flyby plot.
- Transit: all time intervals during which there is a transit are identified in the flyby plot by a blue line.
- Maximum illumination: the moment of maximum illumination is marked with a red dot.
- Occultation: the time interval during which there is occultation is identified by a black line.

Furthermore, all these events will be studied in detail.

The best time to make each observation will be shown taking into account the occultations and, also, that there is at least a certain illumination of the planet in order to see it in the image, as if there is no illumination or it is very poor, the observation would not be useful[1].

Each mission is studied separately in order to study all the results in more detail.

8.1.1 Cassini to Jupiter

For the case of the Cassini mission doing a flyby of Jupiter, the time interval studied has been from '2000 NOV 07 00:00:00' to '2001 JAN 22 00:00:00'.

As already said before in Chapter 6, for the Cassini mission studied in the Jupiter system, the observation time to do each of the observations has been computed with and without considering the camera attitude. That is why, in this section, there will be two different times to do each observation, one computed taking into account the camera attitude and the other without considering it. In addition, a comparison between these two different times will be made.

Therefore, for this particular case in which Jupiter is being studied as seen from the Cassini, there will be three different flyby plots. One with the events calculated without taking into account whether the camera pointed to the planet Figure 8.1, one taking it into account Figure 8.2 and another at the end comparing the previous two Figure 8.3. It is important to note that, for the case in which the attitude of the camera is taken into account, instead of having the events marked in the aforementioned way, they are marked following the way that is shown below:

- The minimum distance is shown as a green circle.
- The maximum illumination is marked by a black dot.
- The transits are represented by red lines.

[1]In this project, the minimum illumination has been set to a phase angle of 30 degrees. If the phase angle is greater than this one, no observation will be designed at that moment.

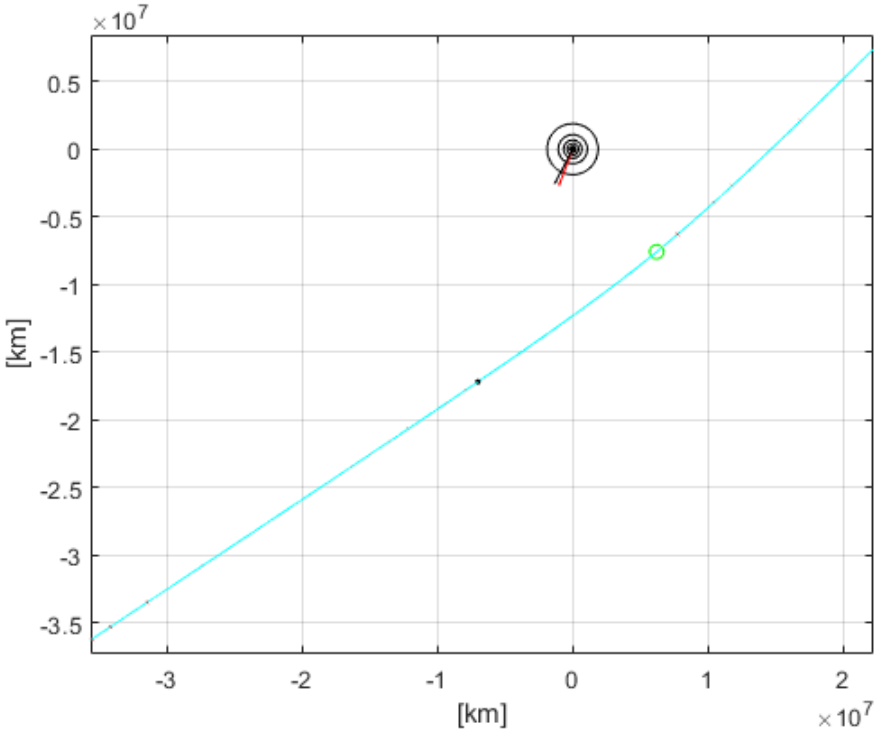

Figure 8.1: Cassini flyby of Jupiter with camera

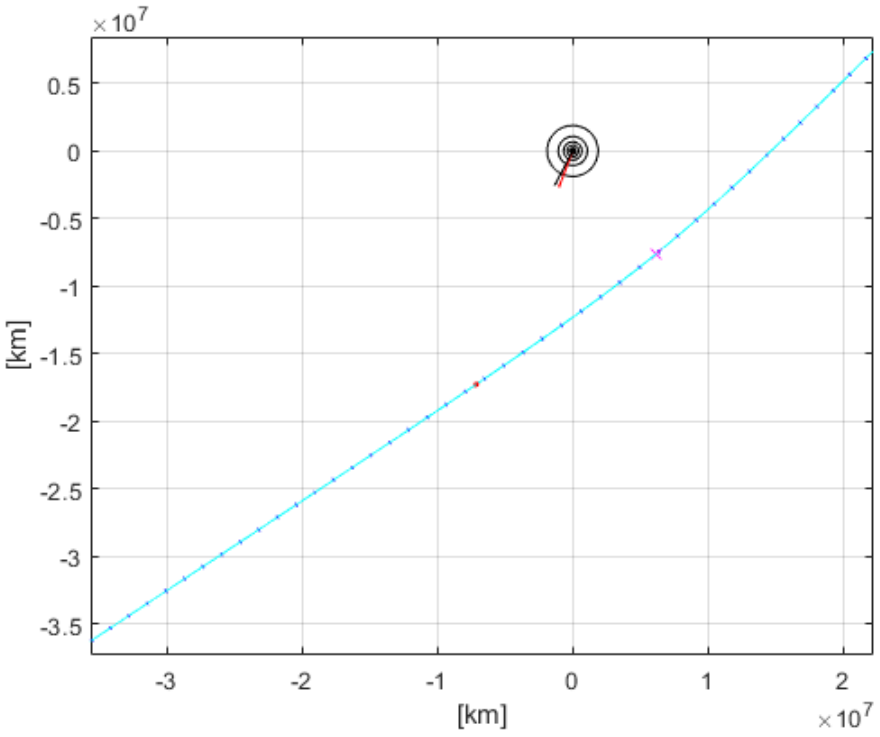

Figure 8.2: Cassini flyby of Jupiter without camera

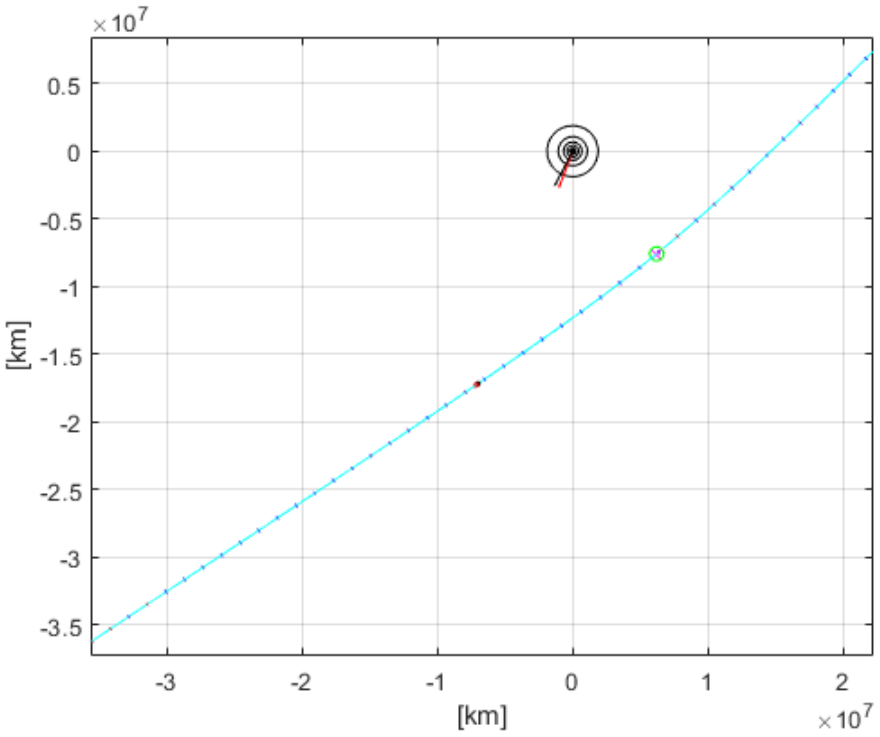

Figure 8.3: Comparison between the two plots

If a comparison between the Figure 8.1 and the Figure 8.2 is done, it can be seen how, taking into account the attitude of the camera, the amount of transits that are observed is much less than the number of transits that would be observed if the camera always points to the planet, fact that is quite logical.

It can also be observed how the moment of maximum illumination taking into account the attitude of the camera is a bit of the same moment calculated without taking into account the camera, as can be seen in Figure 8.3.

The moment when the minimum distance happens, seems to be the same for the case in which the camera attitude is considered and the case without taking it into account. It will be studied more closely later.

Furthermore, it can be seen on the figures below that there is not occultation on the flyby of Cassini to Jupiter.

The next image (Figure 8.4) shows the real path that the probe Cassini followed when doing the flyby of Jupiter. It can be seen that, although Figure 8.4 is seen from a different angle, the vector pointing to the Earth makes almost the same angle with the trajectory path of the Cassini probe.

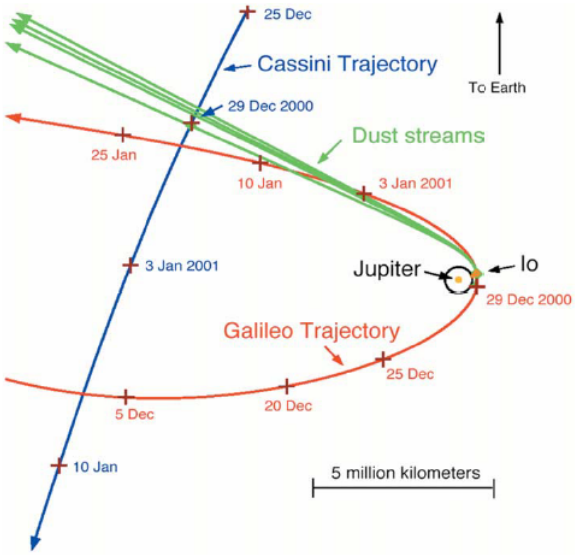

Figure 8.4: Real trajectory of Cassini approach to Jupiter[26]

B. *Minimum distance*

In this section, the results obtained from calculating the minimum distance between Cassini and Jupiter will be shown.

The moment of the trajectory when the minimum distance occurs is shown in Figure 8.5. The image below is a zoom from the Figure 8.3 at the moment when the minimum distance happens in order to see more clearly the position of the probe at the moment when this event occurs.

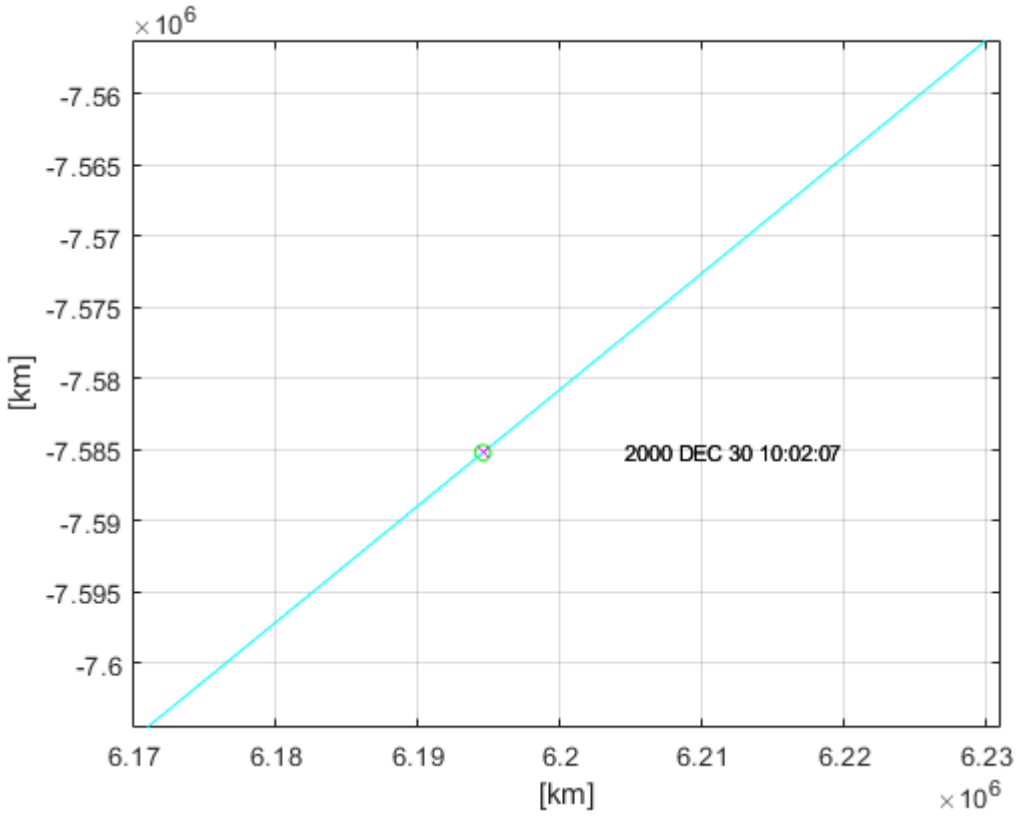

Figure 8.5: Zoom to the minimum distance between Cassini and Jupiter

It can be seen that the minimum distance calculated considering the camera attitude occurs in the same position as the one calculated without taking it into account.

The exact moment in which the minimum distance occurs, calculated according to the algorithm created, is at '2000 DEC 30 10:02:07', as can be seen in Figure 8.5.

This obtained time makes sense with the times in the Figure 8.4, where it can be observed that the minimum distance has to happen at some point between the dates '2000 DEC 29' and '2000 JAN 3'.

As the planet is illuminated at a phase angle smaller than 30^{o} and there is no occultation, the observation at the minimum distance will be designed at '2000 DEC 30 10:02:07.451'.

C. *Maximum illumination*

In this section, the results of calculating the optimum moment to make an observation with the maximum illumination are shown.

The best time to make an observation has been computed without taking into account the camera attitude and considering it.

Below there is a zoom of the Figure 8.3 with the two different times to do an observation with the maximum illumination. The red one corresponds to the time without taking into account the camera attitude and the black one corresponds to the time considering the camera attitude.

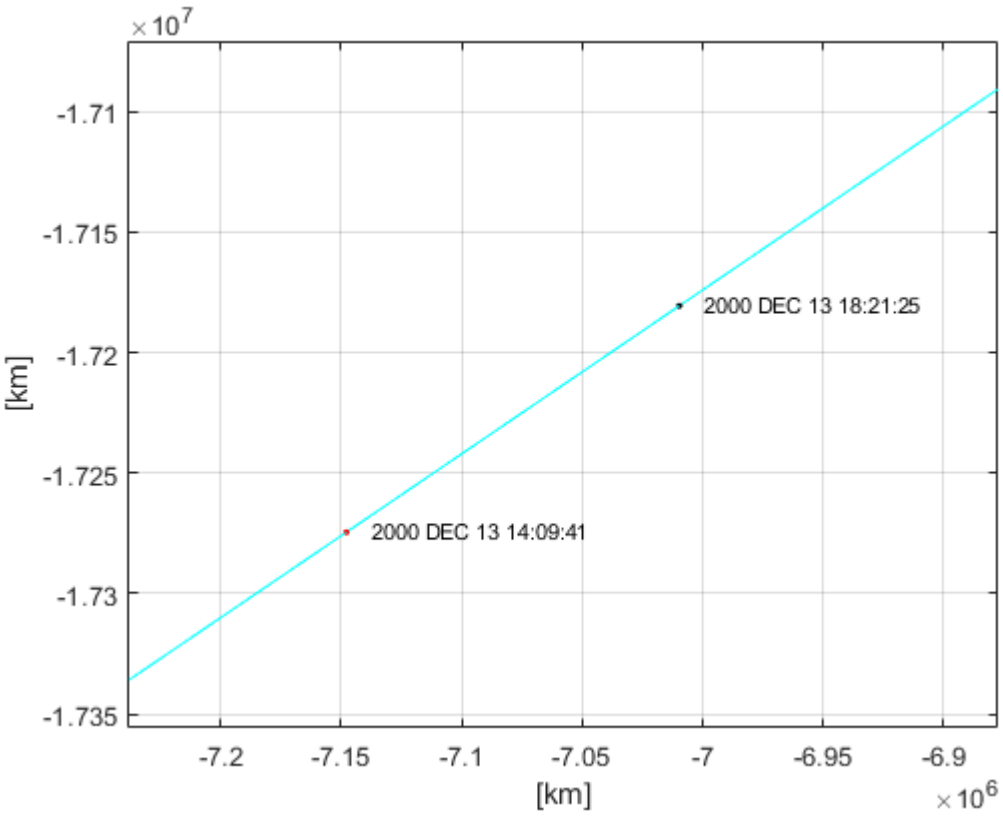

Figure 8.6: Zoom to the maximum illumination of Jupiter as seen from the Cassini

As can be seen in Figure 8.6, the optimum time to do an observation considering the camera attitude is at '2000 DEC 13 14:09:41' and the optimum time to do an observation considering the camera attitude is at '2000 DEC 13 18:21:25', four hours later. This means that the spacecraft was not pointing to the planet at the moment of maximum illumination. It could be because the maximum illumination condition was not considered as a requirement to do an observation.

Therefore, the moment to do an observation of the maximum illumination is set at UTC time: '2000 DEC 13 14:09:41' for the case without considering the camera attitude and at UTC time: '2000 DEC 13 18:21:25' for the case taking it into account.

D. *Transits*

In this section, the results of calculating the different time intervals at which there is a transit of Io between Jupiter and Cassini will be shown.

Below, in Figure 8.7, it can be seen all the values of the transit vector throughout the different ephemeris time studied. Remember that a 0 means that there is no transit and that a value equal to 1 means that there is transit at that ephemeris time as seen from the spacecraft.

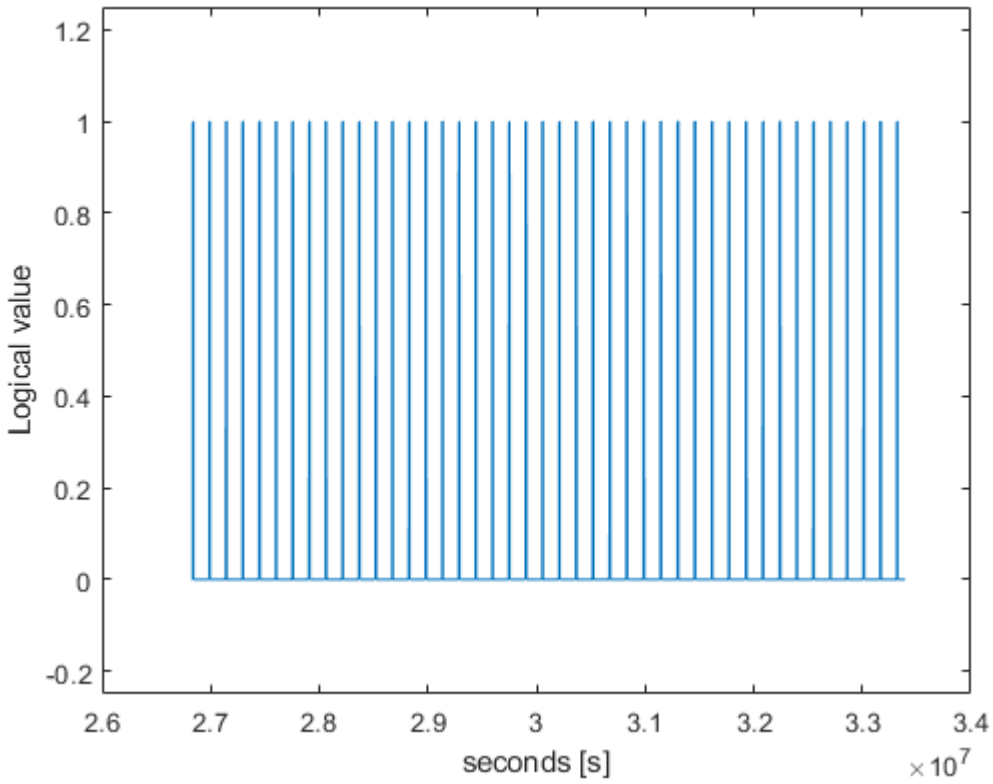

Figure 8.7: Transit values vs. et

It is important to note that the time interval chosen to make the observation is the one that occurs when Cassini is at the minimum distance from Jupiter.

Below it can be seen in Figure 8.8, the time interval during which the transit at the minimum distance of Cassini and Jupiter occurs without considering the camera attitude.

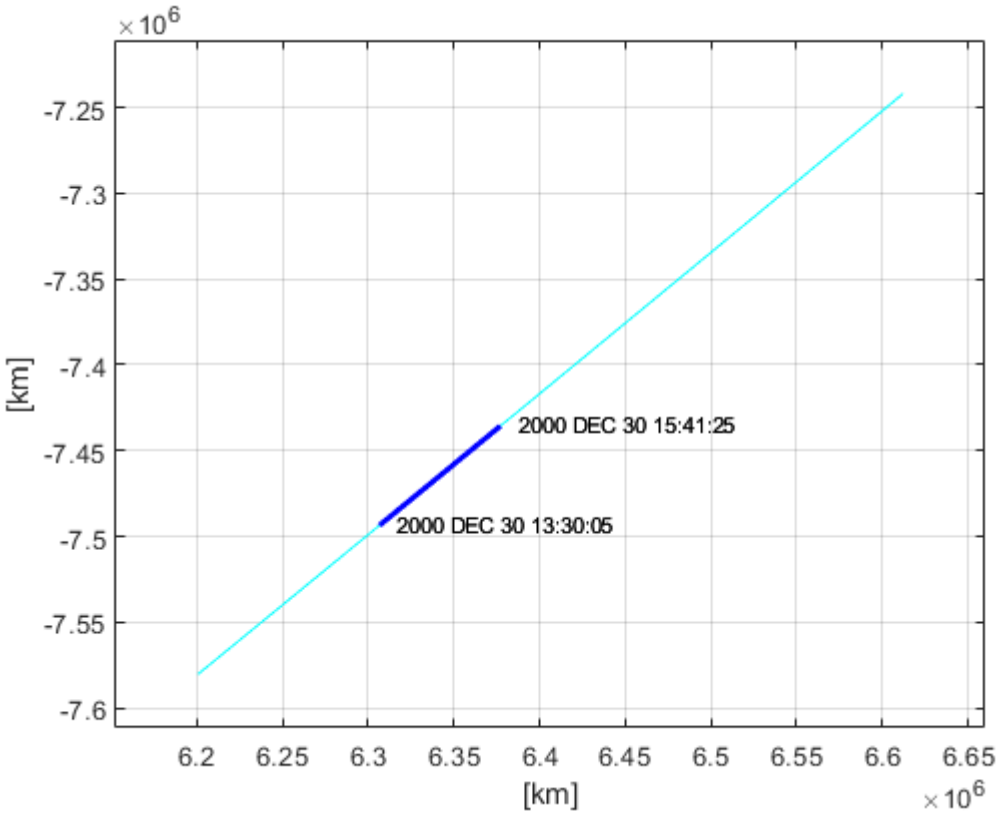

Figure 8.8: Zoom to the transit of Io between Jupiter and Cassini

The transit starts on '2000 DEC 30 13:30:05' and ends on '2000 DEC 15:41:25'. Therefore, as the observation of the transit has been set at the middle of the transit, this transit observation is set at UTC time: '2000 DEC 30 14:35:60'.

Below, in Figure 8.9, it can be seen the time interval during which there is transit of Io between Cassini and Jupiter taking into account if the camera was pointing to the planet or not.

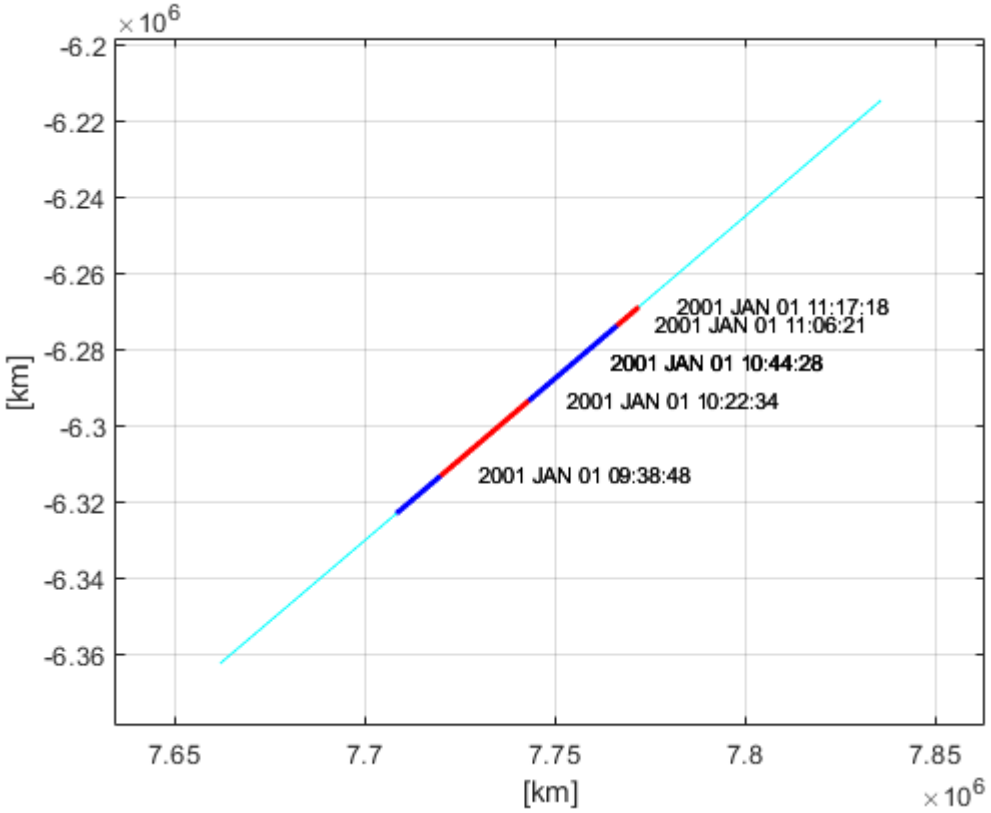

Figure 8.9: Zoom to the transit of Io between Jupiter and Cassini without camera

It can be seen that the transit considering the camera attitude does not take place on the same time interval as the transit without taking the camera attitude into account. The time interval of this transit starts on '2001 JAN 01 09:38:48' and ends on '2001 JAN 01 10:22:34'. Almost a day after the transit without considering the camera attitude.

This could mean that the transit of Io as seen from the minimum distance possible, was not a condition to do an observation neither.

The observation of the transit as seen from the camera is set at UTC time: '2001 JAN 1 09:59:59.998', the middle of the time interval during which there is transit.

In order to validate the algorithm *FindTransit* and prove that it works properly, the times of two images from the same transit of Io between the Cassini and Jupiter taken by the Cassini and navigated by the TFG's author partner Martí Sierra Salvadó have been taken [2].

To validate it, the time interval between the two images has been set as the time interval studied by the algorithm and has been proved whether the algorithm calculates that there was a transit of Io during that time interval.

The images taken by the Cassini are shown below (Figure 8.10 and Figure 8.11).

[2]It is recommended to read his TFG in order to expand the knowledge about image navigation.

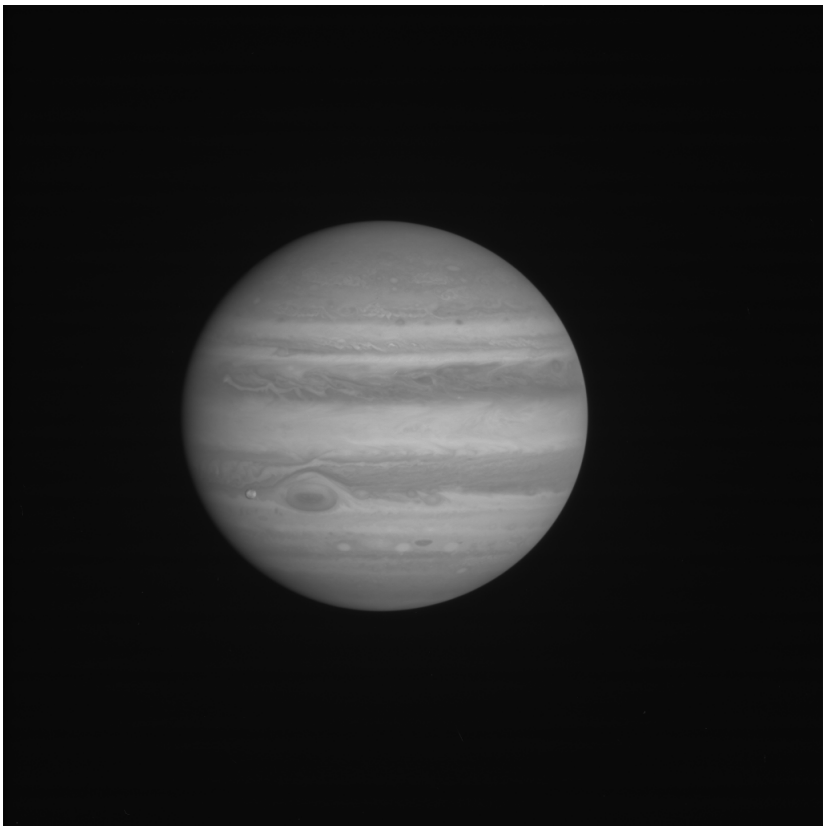

Figure 8.10: Transit of Io as seen from the Cassini at '2000 NOV 12 09:38:18.440'[4]

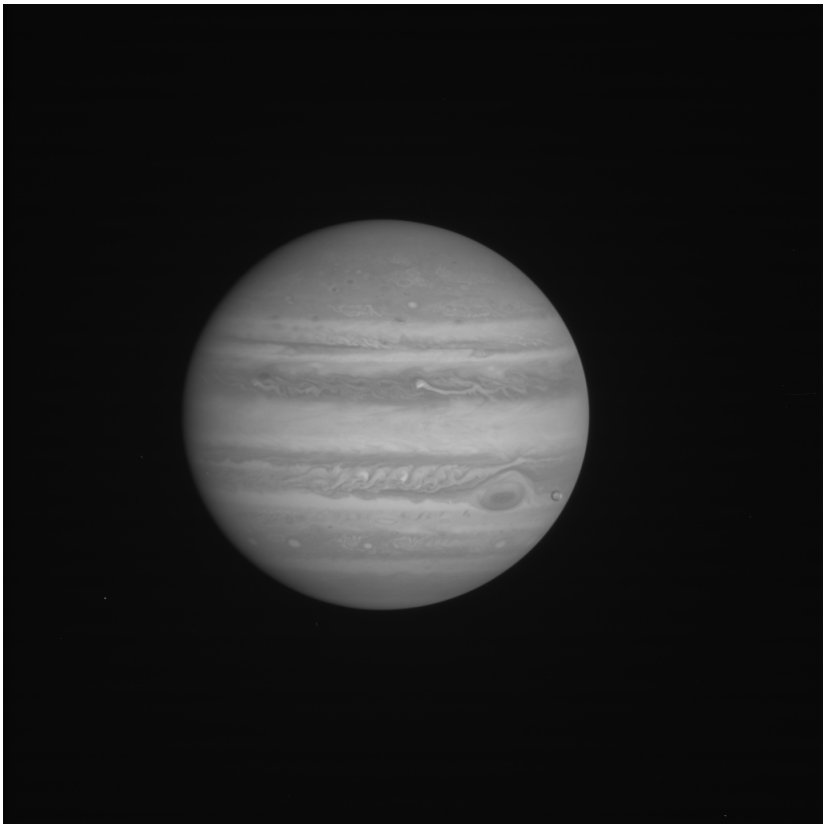

Figure 8.11: Transit of Io as seen from the Cassini at '2000 NOV 12 11:18:18.399'[5]

As it can be seen on the images, in the first one (Figure 8.10), Io is starting the transit at the time '2000 NOV 12 09:38:18.440', and in the second one, Io is at the end of its transit at the time '2000 NOV 12 11:18:18.399'.

Below, a plot of the transit values given by the algorithm during the time interval of the images above is shown. Note that a value equal to 1 means that there is transit of Io and a value equal to 0 means the opposite.

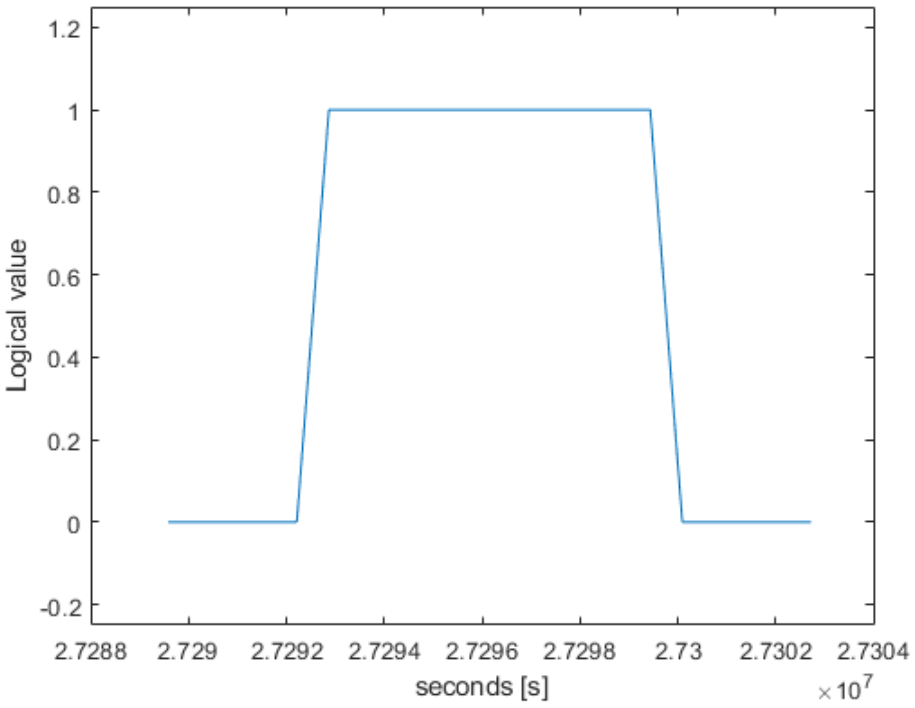

Figure 8.12: Transit of Io given by the algorithm

It can be seen in Figure 8.12 that during the time interval studied, the algorithm calculates that there is a transit of Io, fact that proves that the algorithm works properly.

In can also be seen that the transit starts at an ephemeris time of 2.72925e+07, approximately, which is the equivalent in UTC time of '2000 NOV 12 09:13:55.817' and it ends at an approximated ephemeris time of 2.73e+07 which, in UTC time, is at '2000 NOV 12 11:18:55.817'.

Thus, since the time of the first image (Figure 8.10) is very close to the time in which the algorithm indicates that the transit begins, the moon should be seen quite close to the limit of the planet. Indeed, we can verify that this happens.

In addition, it can also be seen in the second image (Figure 8.11) that the traffic is finishing, and if the time of this image is compared with the time in which, according to the algorithm, the transit finishes, it can be see that they are similar.

8.1.2 New Horizons to Jupiter

In this section, the results of the study of the trajectory of the New Horizons probe flyby to Jupiter will be shown.

The time interval that has been studied for this particular case starts on '2007 FEB 15 00:00:00' and ends on '2007 MAR 17 00:00:00'.

A. *Flyby plot*

First of all, a comparison between the flyby plot obtained with the algorithm *PlotFlyby* and the real trajectory is done.

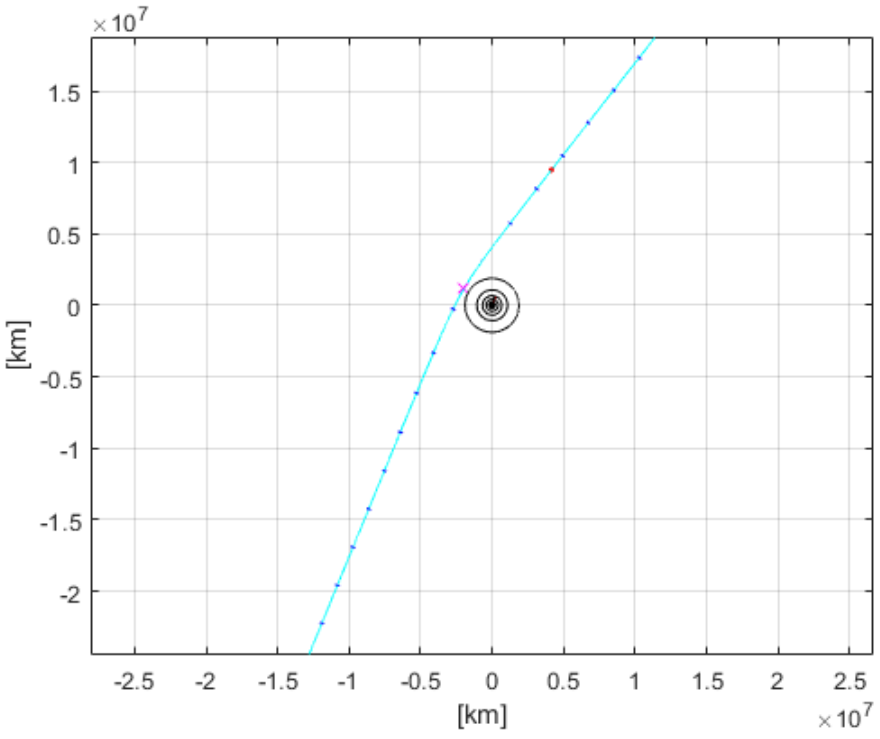

Figure 8.13: New Horizons flyby of Jupiter

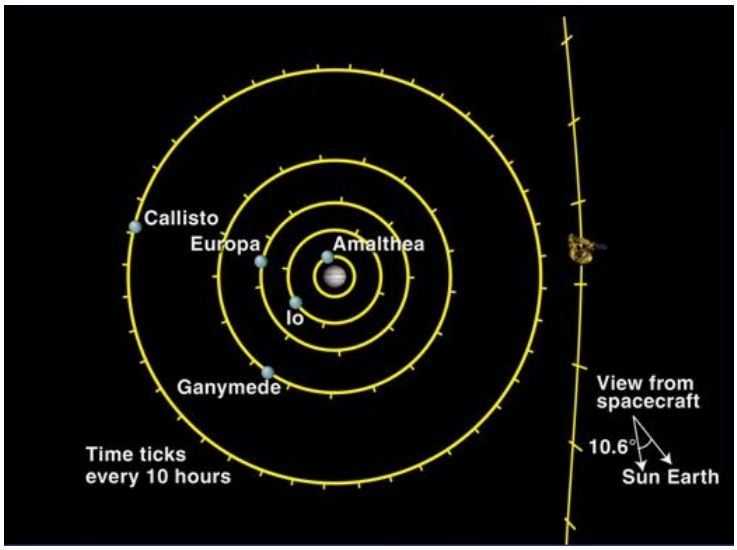

Figure 8.14: Real trajectory of New Horizons approach to Jupiter[6]

In Figure 8.13 it can be seen the trajectory obtained with the algorithm, with the events of which an observation wants to be made marked on it. It is important to highlight that for the time interval studied there is not occultation.

If a comparison between this image and the Figure 8.14 is done, and taking into account that they are not seen from the same angle, it can be seen that the two figures are very similar.

Also, it can be seen that the vector pointing to the Earth (black vector in Figure 8.13) and the vector pointing to the Sun (red vector in Figure 8.13) are pointing to the same direction as the ones seen in Figure 8.14.

B. *Minimum distance*

In this section, the results obtained from calculating the minimum distance between New Horizons and Jupiter will be shown.

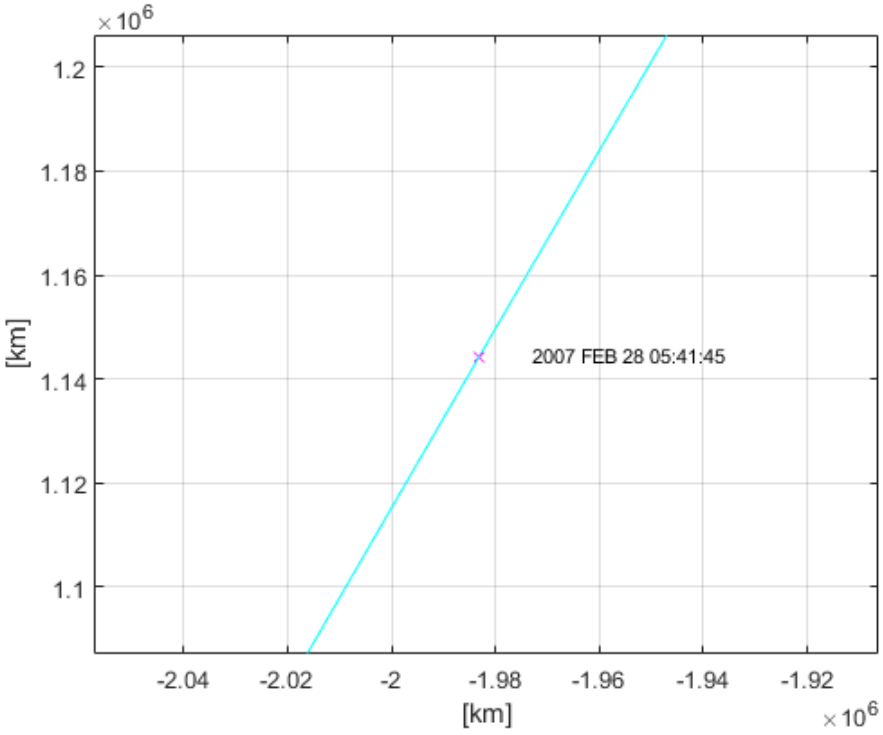

Figure 8.15: Zoom to the minimum distance between New Horizons and Jupiter

As it can be seen in the figure above, the minimum distance occurs on '2007 FEB 28 05:41:45'.

As there is no occultation at this time and the phase angle is smaller than the one defined by the user, the observation from the minimum distance is set at UTC time '2007 FEB 28 05:41:45'.

C. *Maximum illumination*

In this section, the best moment to do an observation of Jupiter with the condition of maximum illumination, i.e. the minimum phase angle, is presented.

Below, in Figure **??** a zoom of the Figure 8.13 at the moment when the maximum illumination occurs can be seen.

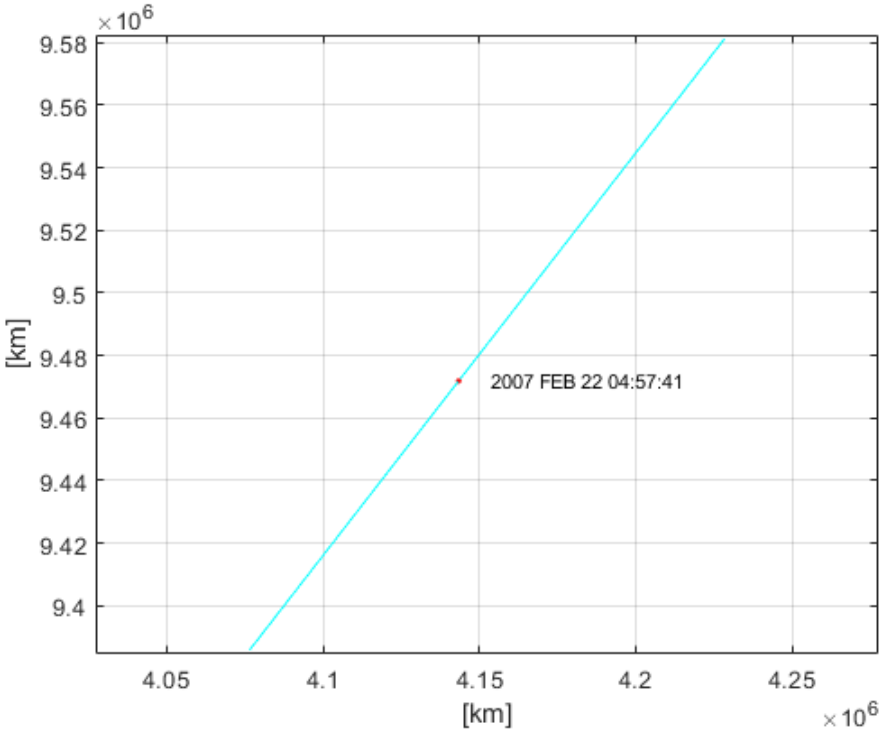

Figure 8.16: Zoom to the maximum illumination of Jupiter as seen from the New Horizons

As it is shown on the figure above, the moment when the maximum illumination occurs is at UTC time: '2007 FEB 22 04:57:41'. As there is not occultation at that time, the time to do this observation is set at this same moment.

D. *Transits*

The results obtained of the transits of Io calculated for the case when New Horizons makes a flyby of Jupiter are shown below.

In the figure below it is shown a zoom of the Figure 8.13 to the moment when the transit at the minimum distance happens.

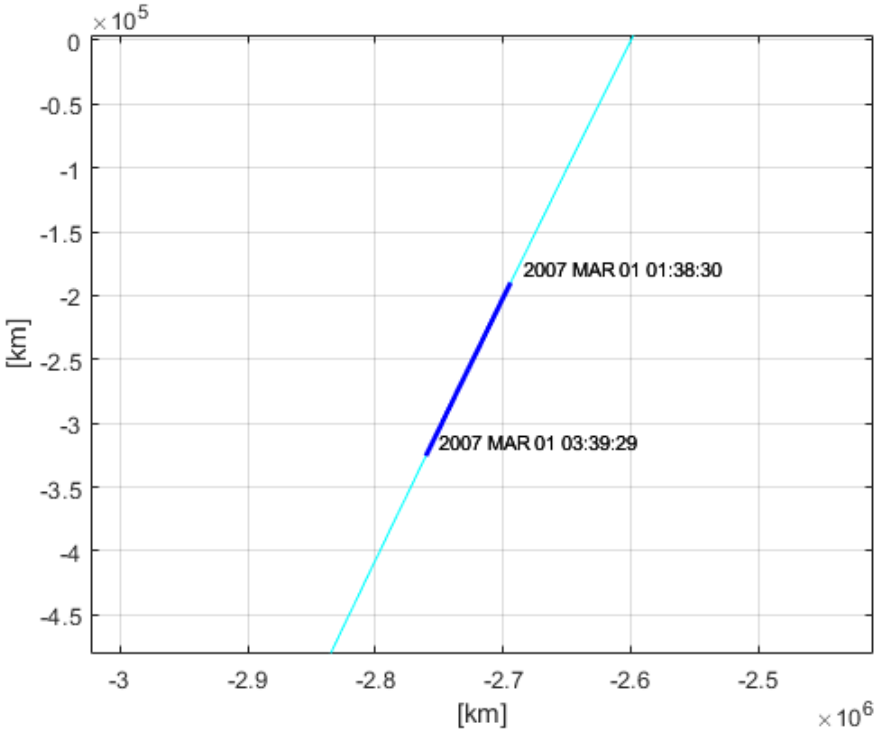

Figure 8.17: Zoom to the transit of Io between Jupiter and New Horizons

It can be seen in the figure above that the transit of the minimum distance starts at '2007 MAR 01 01:38:30' and ends at '2007 MAR 01 03:39:29'. As there is not occultation during this time interval and the phase angle is smaller than the one defined as the maximum phase angle to do an observation. Therefore, the time to do an observation of the transit of Io as seen from the New Horizons is set at UTC time: '2007 MAR 01 02:38:59'.

8.1.3 Voyager 1 to Jupiter

In this section, the results of the study of the trajectory of the Voyager 1 probe flyby to Jupiter will be shown.

The time interval that has been studied for this particular case starts on '1979 FEB 22 00:00:00.000' and ends on '1979 MAR 14 00:00:00.000'.

A. *Flyby plot*

In this section, a comparison between the flyby plot obtained with the algorithm *PlotFlyby* and the real trajectory is done.

In addition, a figure representing only the spacecraft trajectory and the most important events on it (Figure 8.20) is shown in order to see more clearly this events as in the Figure 8.18 they can not be seen clearly.

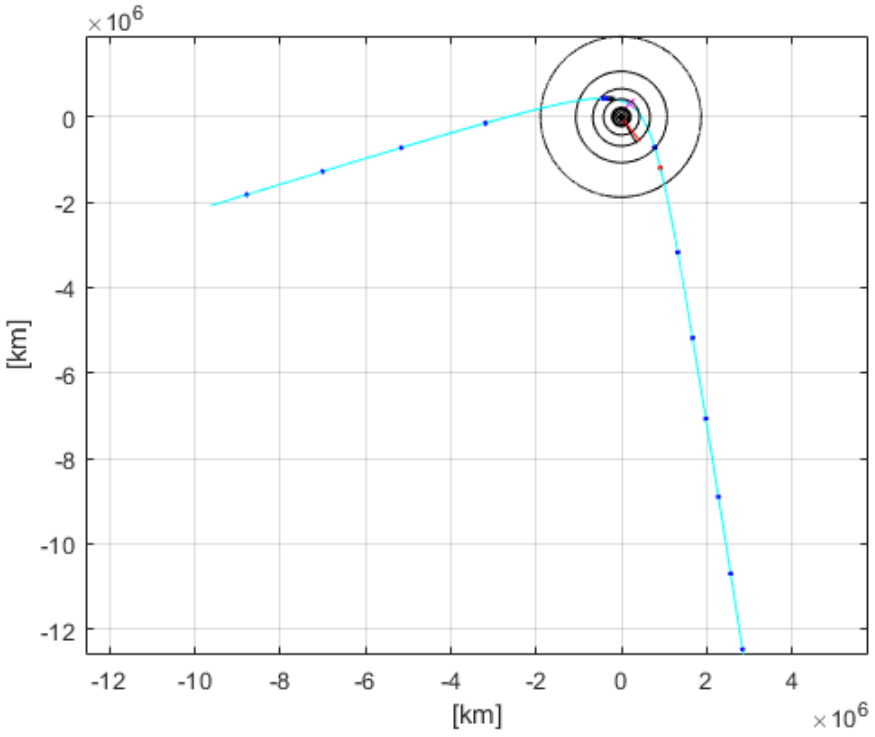

Figure 8.18: Voyager 1 flyby of Jupiter

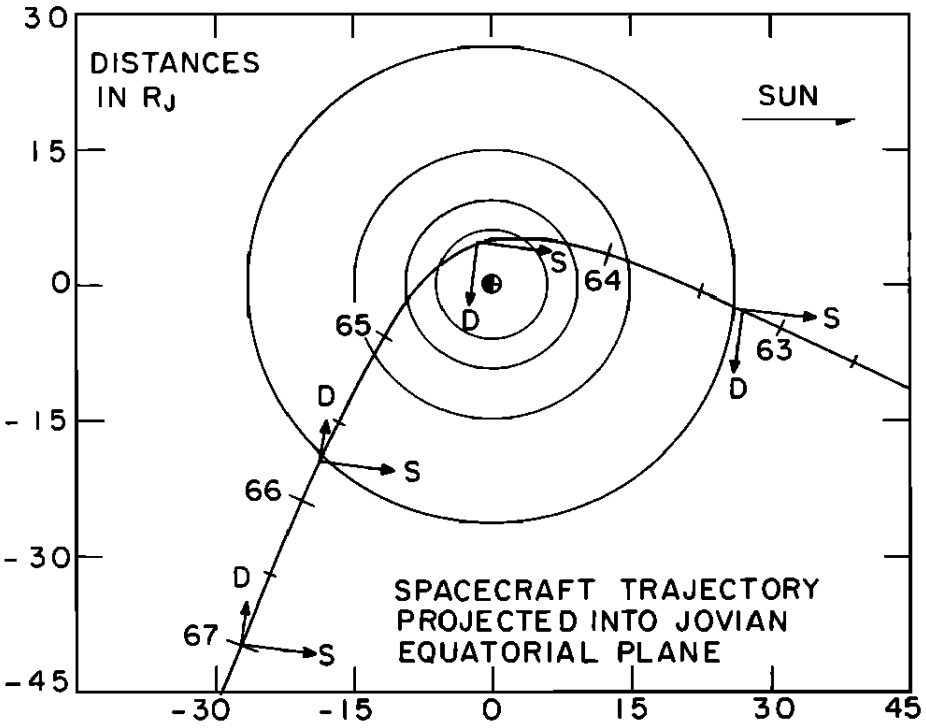

Figure 8.19: Real trajectory of Voyager 1 approach to Jupiter[12]

As it can be seen, the figures above, considering that there is a rotation between the two images, are very similar. Also, the vector pointing to the Sun (black vector in Figure 8.18) is pointing to the same direction.

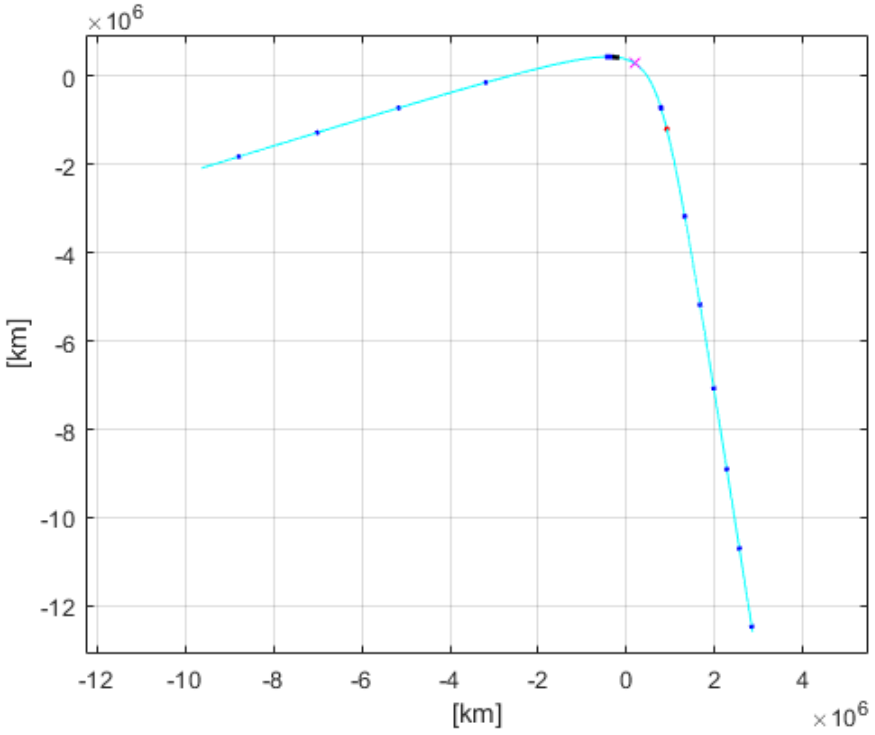

Figure 8.20: Voyager 1 flyby of Jupiter without planets

In the figure above, the most important events are shown more clearly than in the Figure 8.18. It can be seen that, in this case, there is occultation.

B. *Occultation*

In this section, the time interval during which there is occultation is studied.

In the figure below, it can be seen the time interval during which there is occultation.

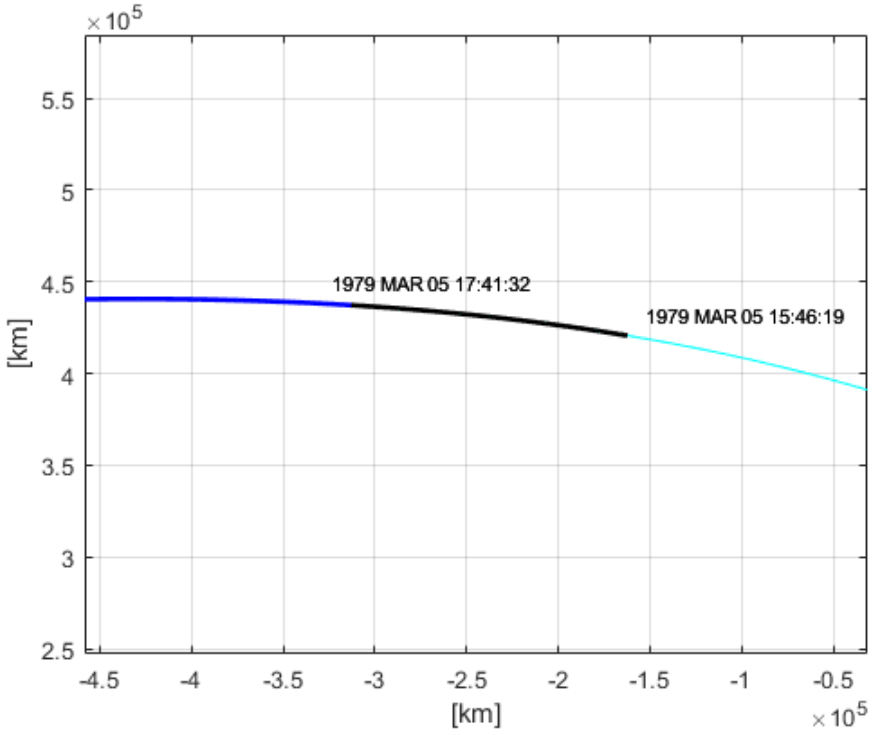

Figure 8.21: Zoom to the occultation of Voyager 1 at Jupiter

The figure above shows that the occultation starts on '1979 MAR 05 15:48:19' and ends on '1979 MAR 05 17:41:32'. Thus, during this time interval, no images should be done.

B. *Minimum distance*

In this section, the optimum moment to do the observation from the minimum distance possible from the Voyager 1 to Jupiter will be studied.

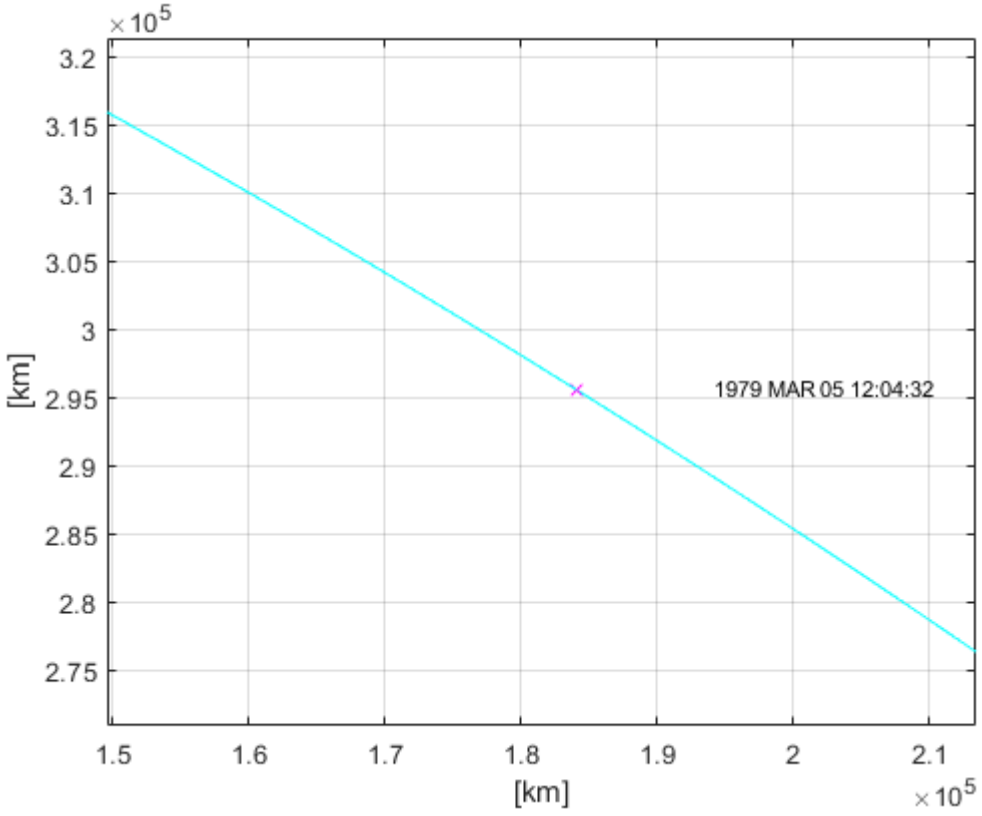

Figure 8.22: Zoom to the minimum distance between Voyager 1 and Jupiter

In the figure above, it can be seen that the minimum distance happens on '1979 MAR 05 12:04:32'.

As the occultation, as seen before, is not happening at this moment, and the phase angle is smaller than the one set as the maximum phase angle to do an observation, the time to do the observation from the minimum distance is set at UTC time: '1979 MAR 05 12:04:32'.

C. *Maximum illumination*

In this section, the best moment to do an observation of Jupiter as seen from the probe Voyager 1 with the condition of maximum illumination will be presented.

In the figure below, a zoom of the Figure 8.18 at the moment when the maximum illumination occurs can be seen.

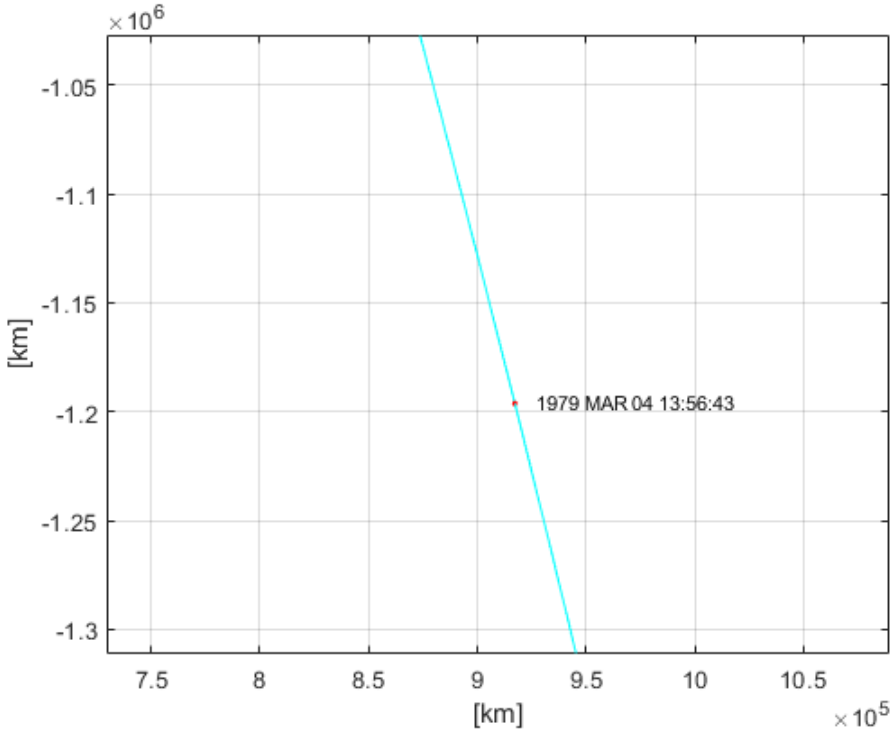

Figure 8.23: Zoom to the maximum illumination of Jupiter as seen from the Voyager 1

As it can be seen on the figure above, the moment when the maximum illumination happens is at UTC time: '1979 MAR 04 13:56:43'. As the occultation does not occur at this time, the observation of Jupiter at its maximum illumination as seen from the Voyager 1 is set at UTC time: '1979 MAR 04 13:56:43'.

D. *Transits*

The results obtained of the transits of Io calculated for the case when Voyager 1 makes a flyby of Jupiter are presented below.

In the figure below it is shown a zoom of the Figure 8.18 to the moment when the transit at the minimum distance happens.

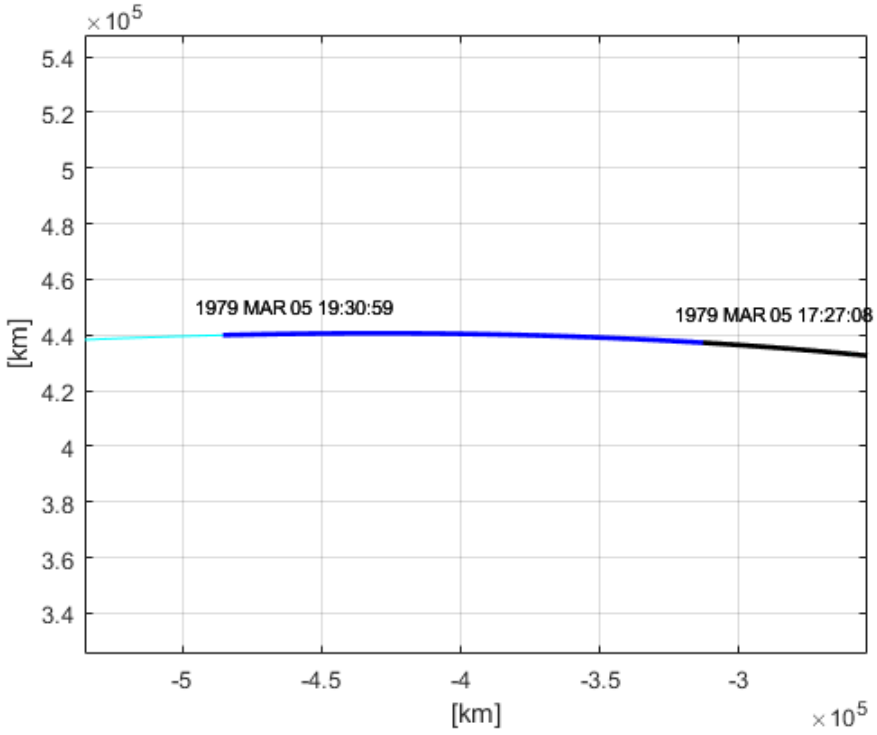

Figure 8.24: Zoom to the transit of Io between Jupiter and Voyager 1

It can be seen on the image above that the transit starts at UTC time: '1979 MAR 05 17:27:08' and ends at UTC time: '1979 MAR 05 19:30:59'. Thus, as the condition to design the observation of the transit is to the moon to be in the middle of its transit, the time to do the observation of the transit of Io as seen from the Voyager 1 is set at UTC time: '1979 MAR 05 18:39:08.155'.

8.1.4 Voyager 2 to Jupiter

In this section, the results of the study of the trajectory of the Voyager 2 probe flyby to Jupiter will be shown.

The time interval that has been studied for this particular case starts on '1979 JUN 30 00:00:00' and ends on '1979 JUL 19 00:00:00'.

A. *Flyby plot*

In this section, a comparison between the flyby plot obtained with the algorithm *PlotFlyby* and the real trajectory is done.

In addition, a figure representing only the spacecraft trajectory and the most important events on it (Figure 8.27) is shown in order to see more clearly this events as in the Figure 8.25 they can not be seen clearly.

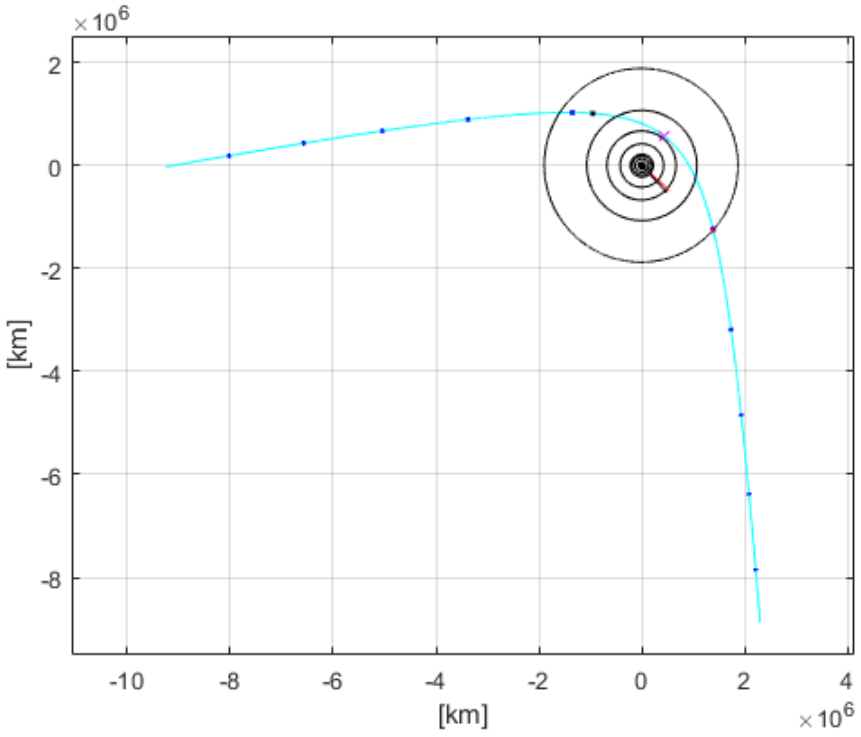

Figure 8.25: Voyager 1 flyby of Jupiter

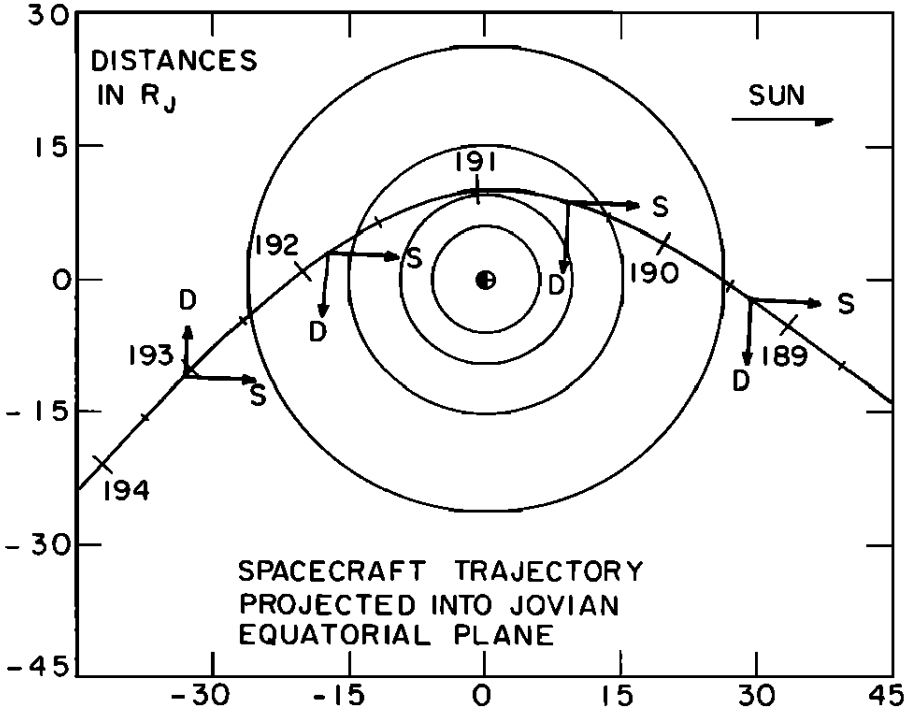

Figure 8.26: Real trajectory of Voyager 2 approach to Jupiter[12]

As it can be seen, the figures above, taking into account that there is a rotation between them, are very similar. Also, the vector pointing to the Sun (black vector in Figure 8.25) is pointing to the same direction.

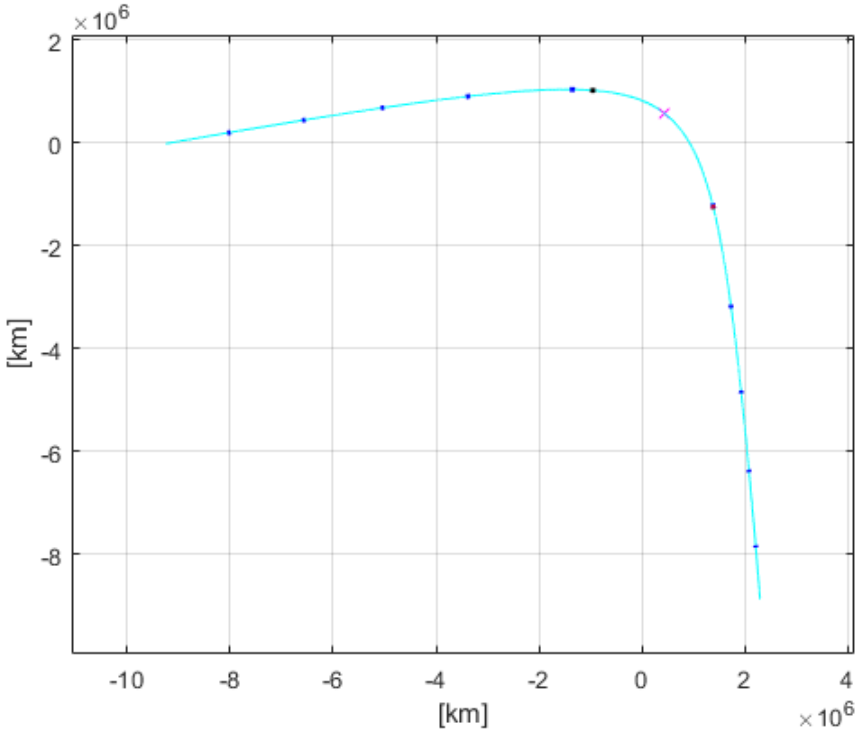

Figure 8.27: Voyager 2 flyby of Jupiter without planets

In the figure above, the most important events are shown more clearly than in the Figure 8.25. It can be seen that, in this case, there is occultation.

B. *Occultation*

In this section, the time interval during which there is occultation is studied.

In the figure below, it can be seen the time interval during which there is occultation.

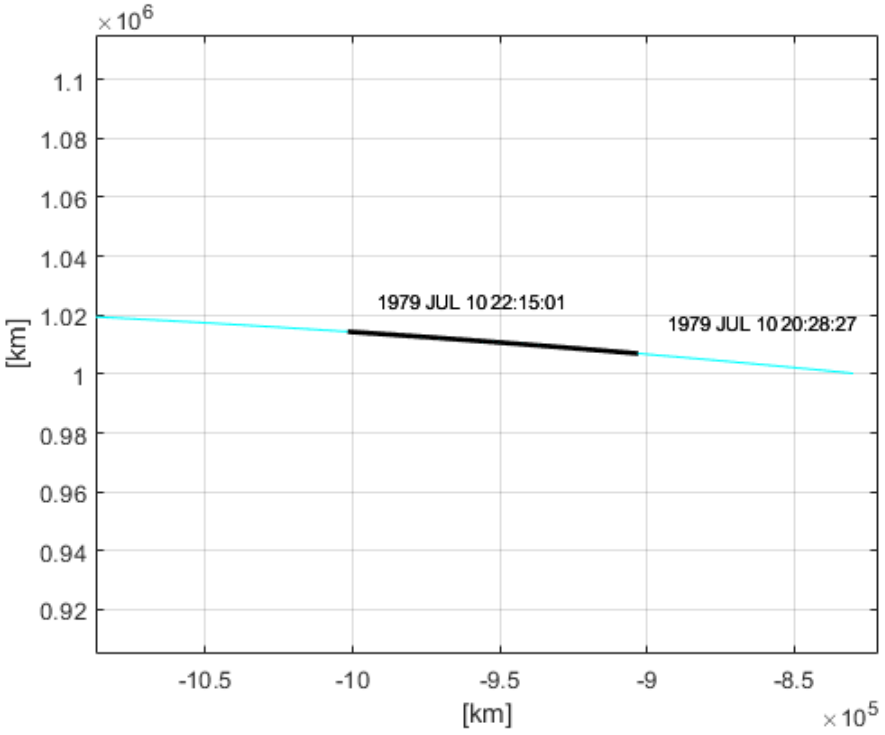

Figure 8.28: Zoom to the occultation of Voyager 1 at Jupiter

The figure above shows that the occultation starts on '1979 JUL 10 20:29:36' and ends on '1979 JUL 10 22:16:19'. It is important to note that the occultation only occurs during two hours so, no images can be taken during this two hours.

B. *Minimum distance*

In this section, the optimum moment to do the observation from the minimum distance possible from the Voyager 2 to Jupiter will be studied.

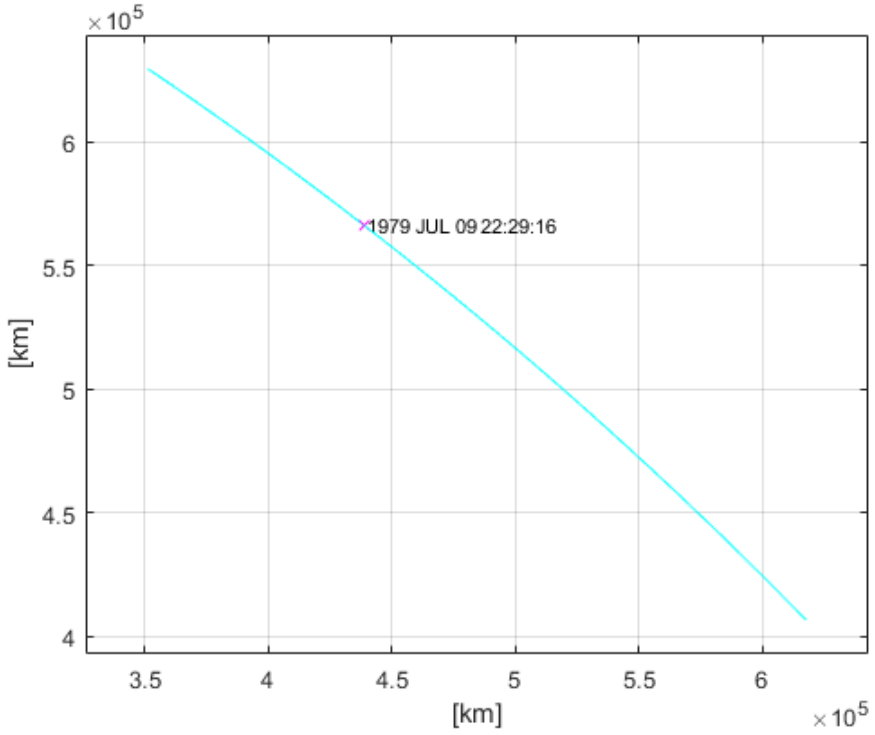

Figure 8.29: Zoom to the minimum distance between Voyager 2 and Jupiter

In the figure above, it can be seen that the minimum distance happens on '1979 JUL 09 22:27:58'.

As the occultation, as seen before, is not happening at this moment, and the phase angle is smaller than the one set as the maximum phase angle to do an observation, the time to do the observation from the minimum distance is set at UTC time: '1979 JUL 09 22:27:58'.

C. *Maximum illumination*

In this section, the best moment to do an observation of Jupiter with the condition of the maximum illumination will be presented.

In the figure below, a zoom of the Figure 8.25 at the moment when the maximum illumination occurs can be seen.

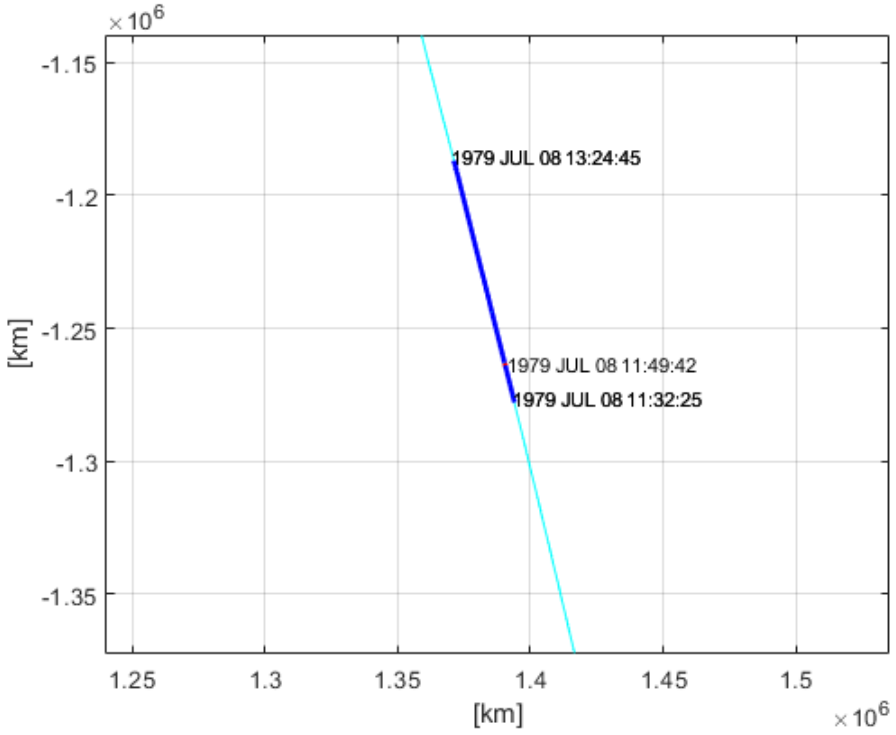

Figure 8.30: Zoom to the maximum illumination of Jupiter as seen from the Voyager 2

As it can be seen on the figure above, the moment when the maximum illumination happens is at UTC time: '1979 JUL 08 11:51:09'. As the maximum illumination occurs during a transit, this observation will capture both events.

As the occultation does not occur at this time, the observation of Jupiter at its maximum illumination as seen from the Voyager 2 is set at UTC time: '1979 JUL 08 11:51:09'.

D. *Transits*

The results obtained of the transits of Io calculated for the case when Voyager 2 makes a flyby of Jupiter are presented below.

In the figure below it is shown a zoom of the Figure 8.25 to the moment when the transit at the minimum distance happens.

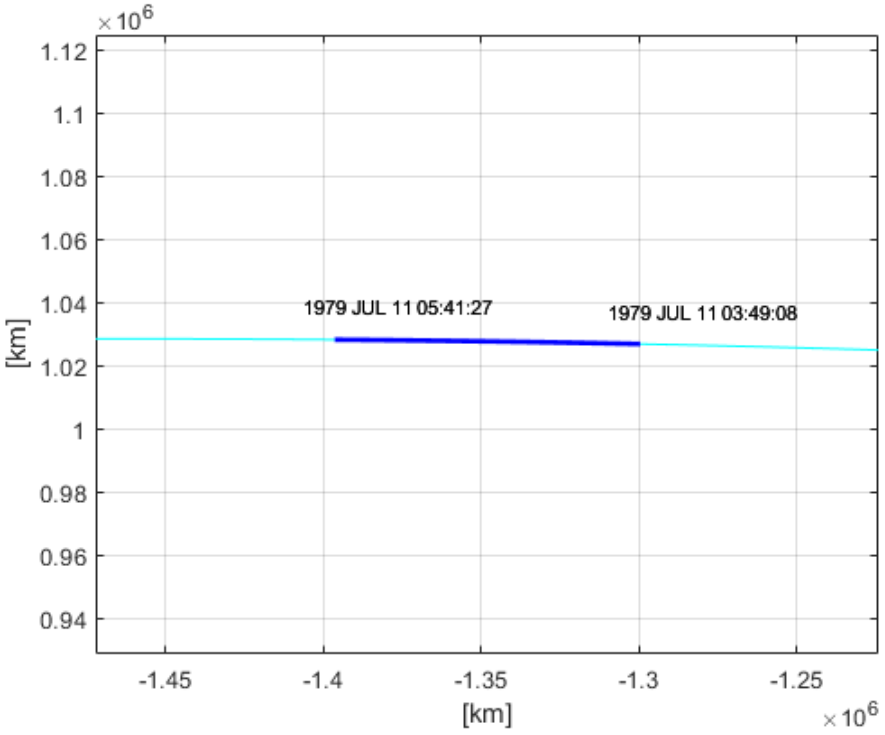

Figure 8.31: Zoom to the transit of Io between Jupiter and Voyager 2

It can be seen on the image above that the transit starts at UTC time: '1979 JUL 11 03:50:08' and ends at UTC time: '1979 JUL 11 05:42:19'. As there is not occultation during the time interval when the transit occurs, and as the condition to design the observation of the transit is to the moon to be in the middle of its transit, the time to do the observation of the transit of Io as seen from the Voyager 2 is set at UTC time: '1979 JUL 11 04:47:35.806'.

8.1.5 New Horizons to Pluto

In this section, the results of the study of the trajectory of the New Horizons probe flyby to Pluto will be shown.

The time interval that has been studied for this particular case starts on '2015 JUL 14 10:48:07' and ends on '2015 JUL 14 13:12:07'.

A. *Flyby plot*

In this section, a comparison between the flyby plot obtained with the algorithm *PlotFlyby* and the real trajectory is done.

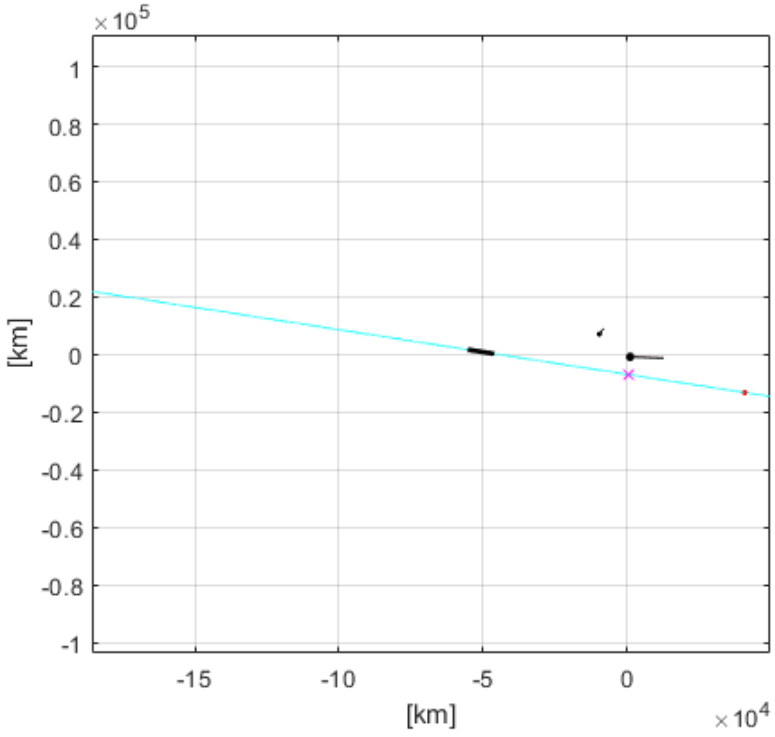

Figure 8.32: New Horizons flyby of Pluto

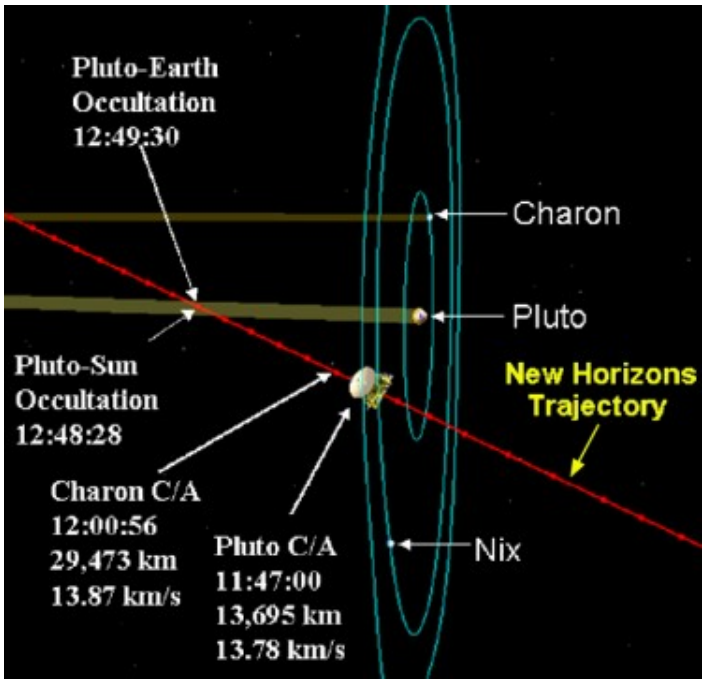

Figure 8.33: Real trajectory of New Horizons approach to Pluto[8]

As it can be seen, the figures above (Figure 8.32 and 8.33), are very similar. It is important to note that for the Figure 8.32 the trajectory of Charon and Pluto have only been plotted for time interval studied, that is why Pluto and Charon are seen as a simple ellipsoid, because they almost do not move.

B. *Occultation*

In this section, the time interval during which there is occultation is studied.

In the figure below, it can be seen the time interval during which there is occultation.

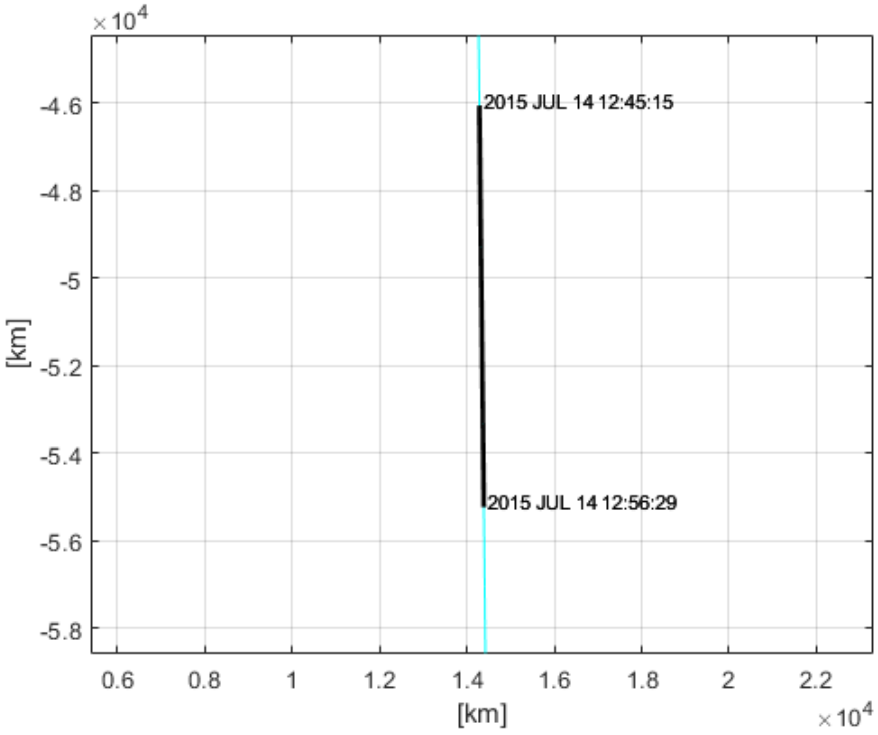

Figure 8.34: Zoom to the occultation of Pluto at Pluto

The figure above shows that the occultation starts on '2015 JUL 14 12:45:15' and ends on '2015 JUL 14 12:56:30'.

B. *Minimum distance*

In this section, the optimum moment to do the observation from the minimum distance possible from the New Horizons to Pluto will be studied.

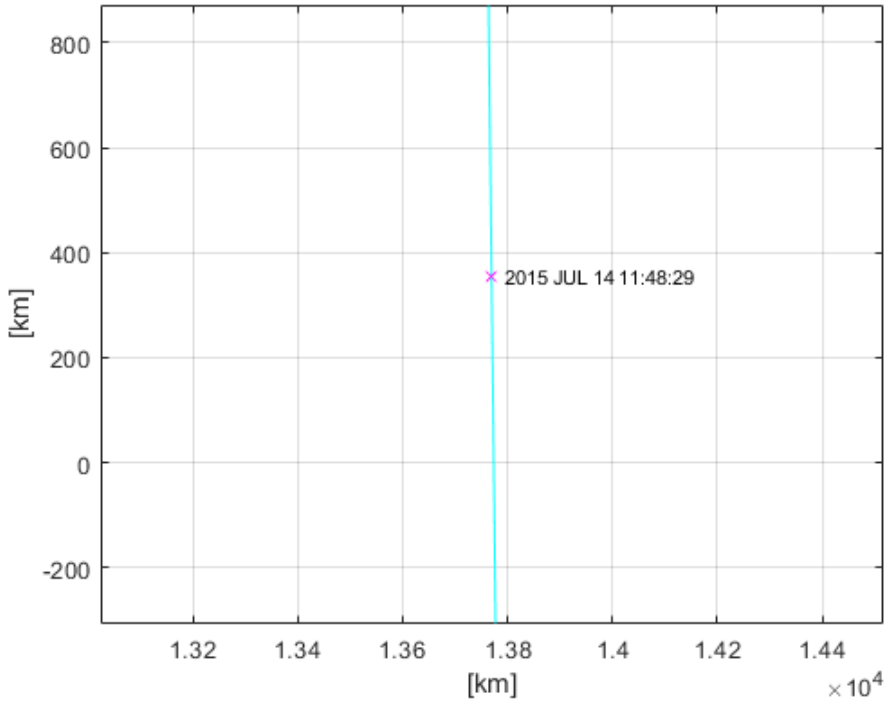

Figure 8.35: Zoom to the minimum distance between New Horizons and Pluto

In the figure above, it can be seen that the minimum distance happens on '2015 JUL 14 11:48:29'.

Time to do an observation with the maximum illumination: 2015 JUL 14 11:36:52.061 As the occultation, as seen before, is not happening at this moment, and the phase angle is smaller than the one set as the maximum phase angle to do an observation, the time to do the observation from the minimum distance is set at UTC time: '2015 JUL 14 11:48:29'.

C. *Maximum illumination*

In this section, the best moment to do an observation of Jupiter with the condition of the maximum illumination will be presented.

In the figure below, a zoom of the Figure 8.32 at the moment when the maximum illumination occurs can be seen.

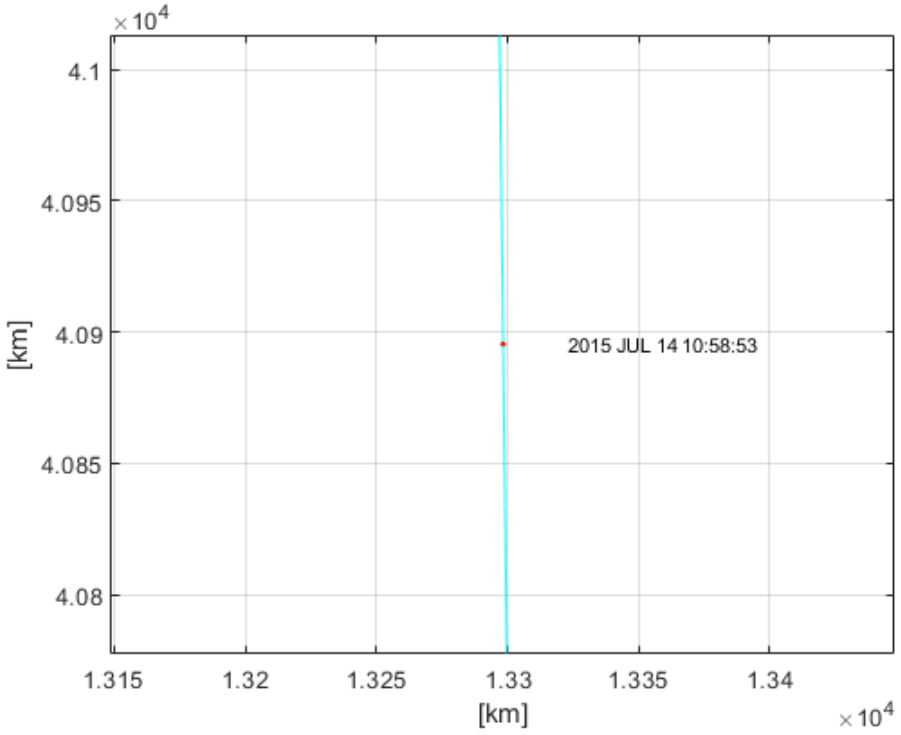

Figure 8.36: Zoom to the maximum illumination of Pluto as seen from the New Horizons

As it can be seen on the figure above, the moment when the maximum illumination happens is at UTC time: '2015 JUL 14 10:58:53'.

As the occultation does not occur at this time, the observation of Pluto at its maximum illumination as seen from the New Horizons is set at UTC time: '2015 JUL 14 10:58:53'.

D. *Transits*

There is no transit of Charon between the New Horizons and Pluto at the time interval studied.

8.1.6 Voyager 1 to Saturn

The results of the study of the trajectory of the Voyager 1 probe flyby to Saturn will be shown.

The time interval that has been studied for this particular case starts on '1980 NOV 07 00:00:00.000' and ends on '1980 NOV 19 00:00:00.000'.

A. *Flyby plot*

A comparison between the flyby plot obtained with the algorithm and the real trajectory will be done in this section.

Furthermore, a figure representing only the Voyager 1 trajectory and the most important events on it (Figure 8.39) is shown in order to see more clearly this events as in the Figure 8.37 they can not be seen clearly.

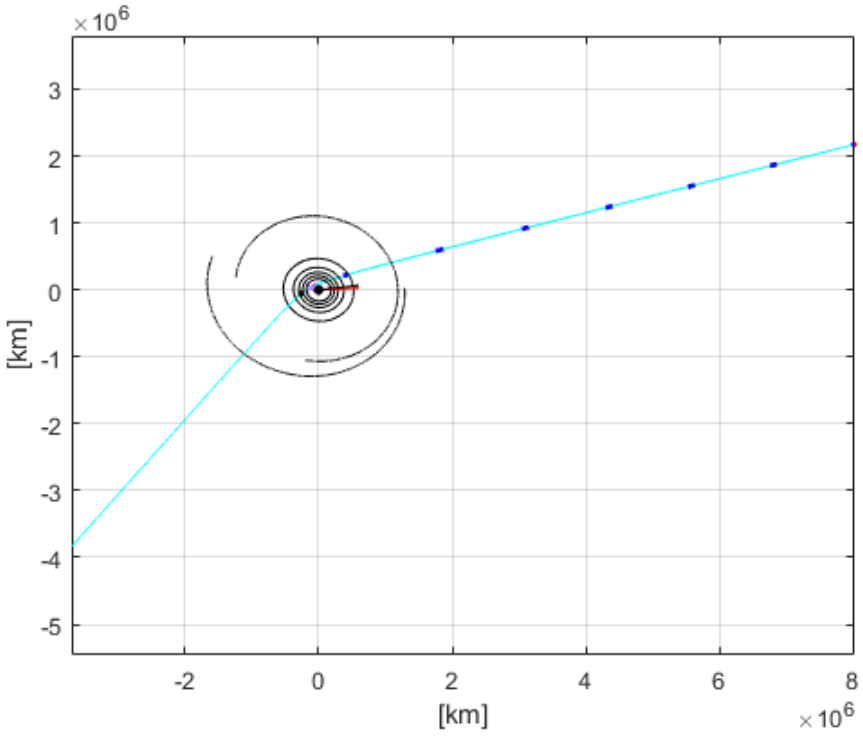

Figure 8.37: Voyager 1 flyby of Saturn

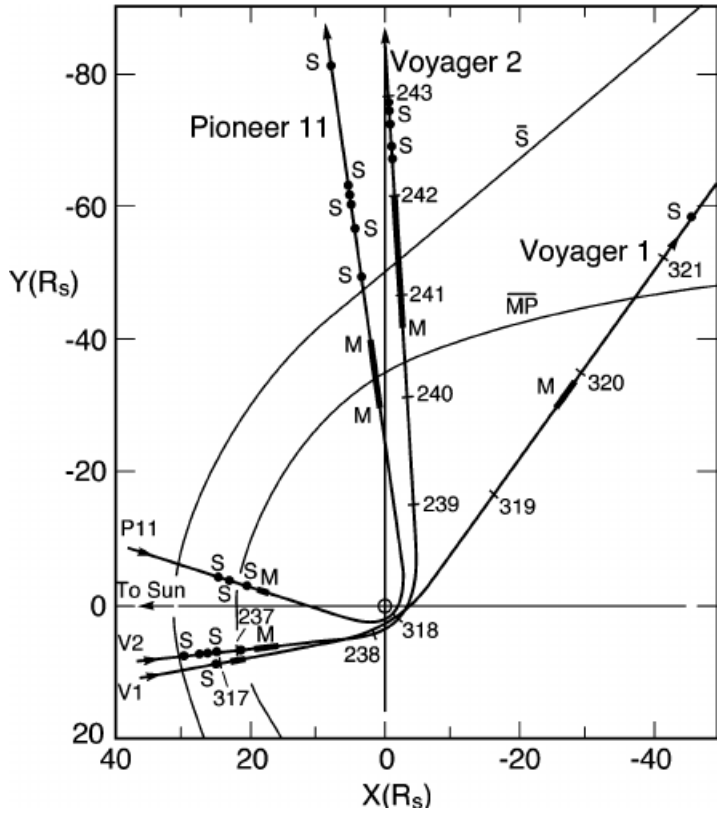

Figure 8.38: Real trajectory of Voyager 1 and Voyager 2 approach to Saturn[3]

As it can be seen if the two figures above are compared, the trajectory obtained with the algorithm adjusts very well to the real trajectory that Voyager 1 made when flying over Saturn. It should be taken into account that there is a rotation between the two images.

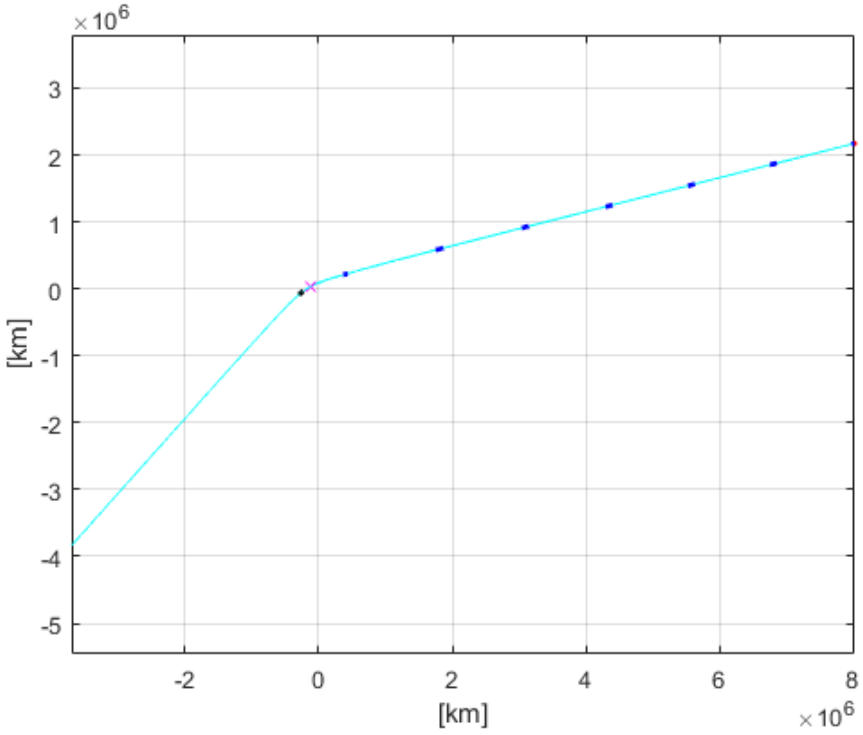

Figure 8.39: Voyager 1 flyby of Saturn without celestial bodies

In the figure above, the most important events are shown more clearly than in the Figure 8.37. It can be seen that there is occultation during a certain time interval while doing the flyby of Saturn.

B. *Occultation*

The time interval when there is occultation calculated with the algorithm will be studied in this section.

The figure below shows a zoom of the Figure 8.37 at the time interval when there is occultation in order to see more clearly when it starts and when it ends.

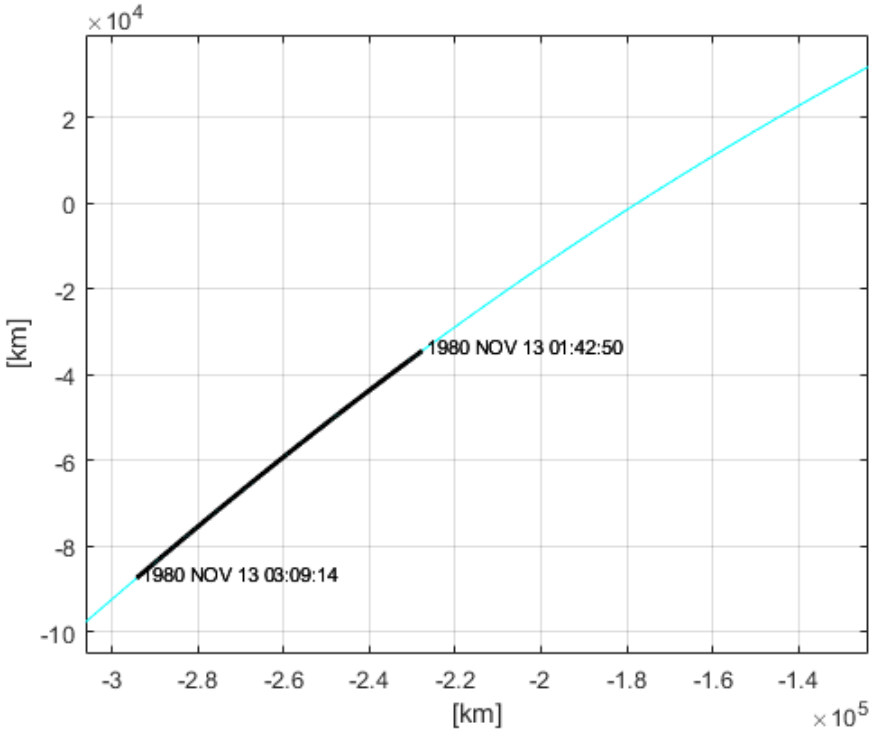

Figure 8.40: Zoom to the occultation of Voyager 1 at Saturn

The figure above shows that the occultation of the Voyager 1 at Saturn starts on '1980 NOV 13 01:42:50' and ends on '1980 NOV 13 03:09:14'.

B. *Minimum distance*

The best moment to do the observation from the minimum distance possible from the Voyager 1 to Saturn will be studied in this section.

In the figure below (Figure **??**), it can be seen a zoom of the Figure 8.37 at the moment when the minimum distance happens.

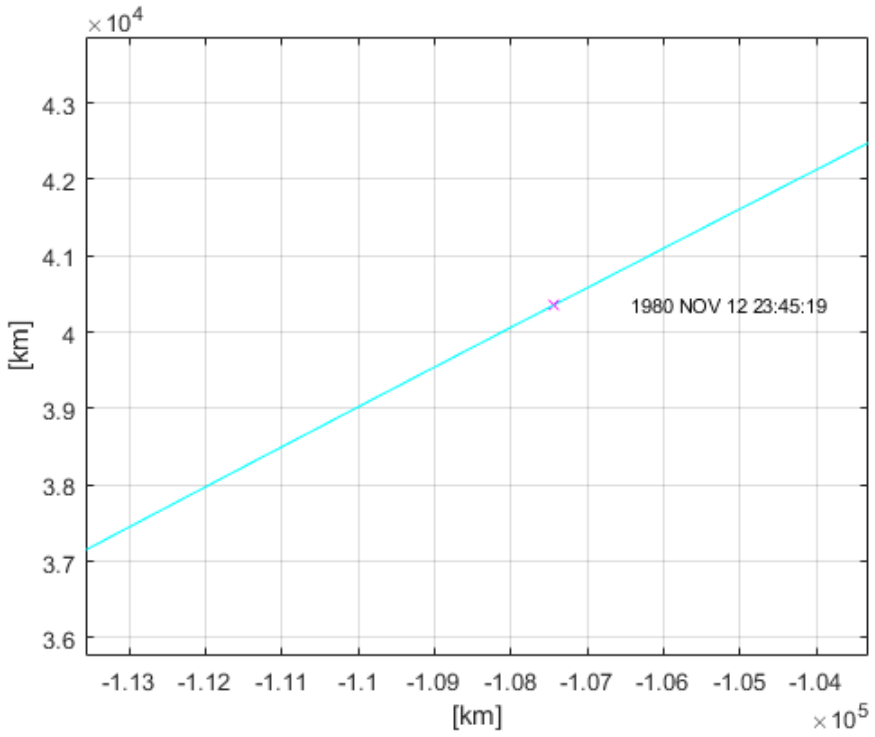

Figure 8.41: Zoom to the minimum distance between Voyager 1 and Saturn

As it can be seen on the figure above, the minimum distance occurs on '1980 NOV 12 23:45:19'.

As there is not occultation at this moment, and the phase angle is smaller than the one set as the maximum phase angle to do an observation, the time to do the observation from the minimum distance is set at UTC time: '1980 NOV 12 23:45:19'.

C. *Maximum illumination*

The maximum illumination, as it can be seen on the Figure 8.37, happens at the start of the time interval studied for the case of the Voyager 1 doing a flyby of Saturn.

This is because the phase angle, as seen from the Voyager 1, increases when the spacecraft gets closer to Saturn. Thus, the maximum illumination will always be the first moment of the time interval studied.

D. *Transits*

In this section, the results obtained of the transits of Titan calculated for the case when Voyager 1 makes a flyby of Saturn are presented below.

In the figure below (Figure 8.42), it is shown a zoom of the Figure 8.37 to the moment when the transit at the minimum distance occurs.

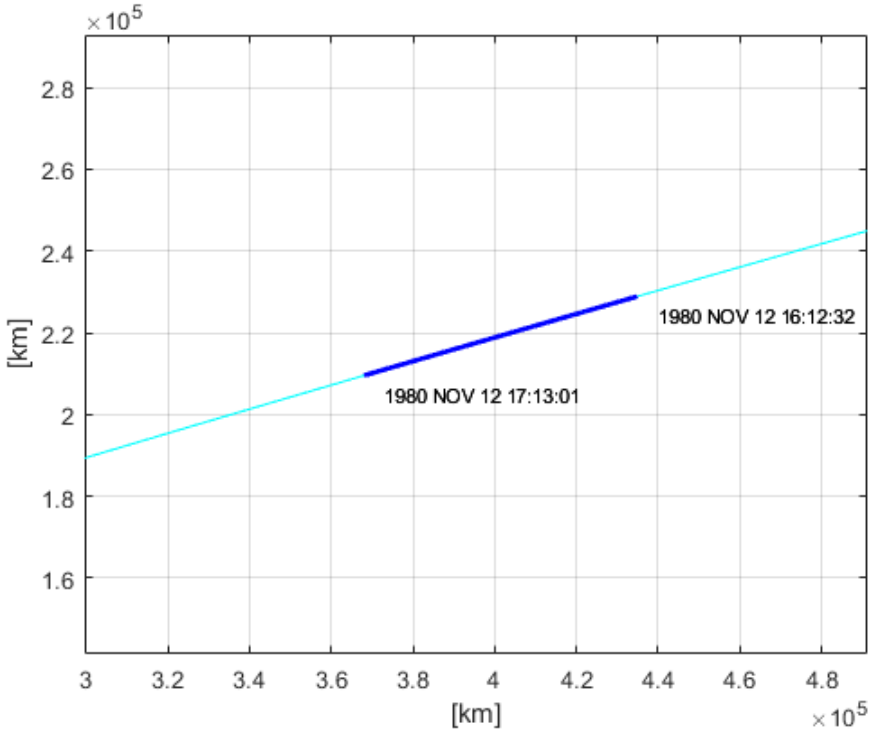

Figure 8.42: Zoom to the transit of Mimas as seen from the Voyager 1

In the image above, it can be seen that the transit as seen from the minimum distance starts at UTC time: '1980 NOV 12 16:12:32' and ends at UTC time: '1980 NOV 12 16:12:32'. As there is not occultation during the time interval when the transit occurs, and as the condition to design the observation of the transit is to the moon to be in the middle of its transit, the time to do the observation of the transit of Io as seen from the Voyager 1 is set at UTC time: '1980 NOV 12 16:43:38.182'.

8.1.7 Voyager 2 to Saturn

The results of studying the trajectory of the Voyager 2 probe flyby to Saturn will be shown.

The time interval that has been studied for this particular case starts on '1981 AUG 17 00:00:00' and ends on '1981 SEP 01 00:00:00'.

A. *Flyby plot*

In this section, a comparison between the flyby plot obtained with the algorithm *PlotFlyby* and the real trajectory is done.

In addition, a figure representing only the spacecraft trajectory and the most important events on it (Figure 8.45) is shown in order to see more clearly this events as in the Figure 8.43 they can not be seen clearly.

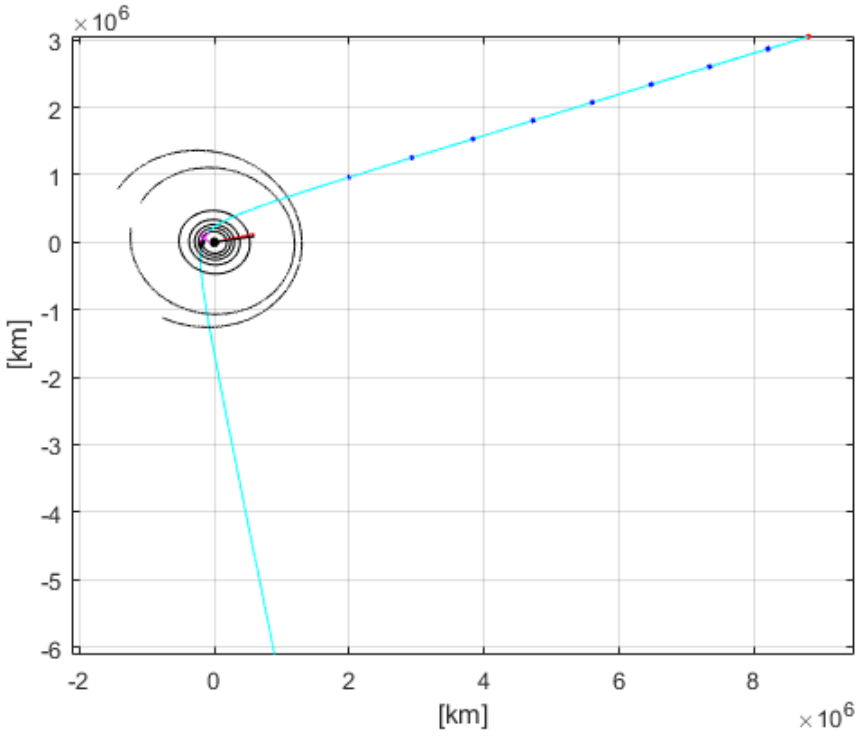

Figure 8.43: Voyager 1 flyby of Saturn

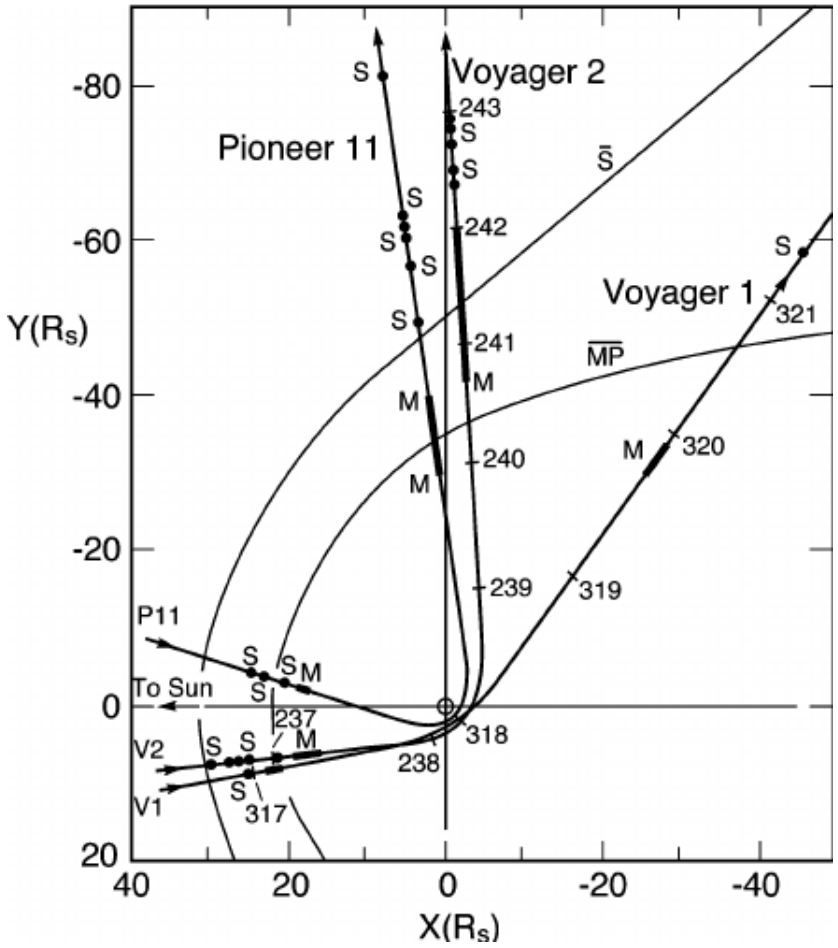

Figure 8.44: Real trajectory of Voyager 1 and Voyager 2 approach to Saturn[3]

As it can be seen if the two figures above are compared, the trajectory obtained with the algorithm adjusts very well to the real trajectory that Voyager 2 made when flying over Saturn. It should be taken into account that there is a rotation between the two images.

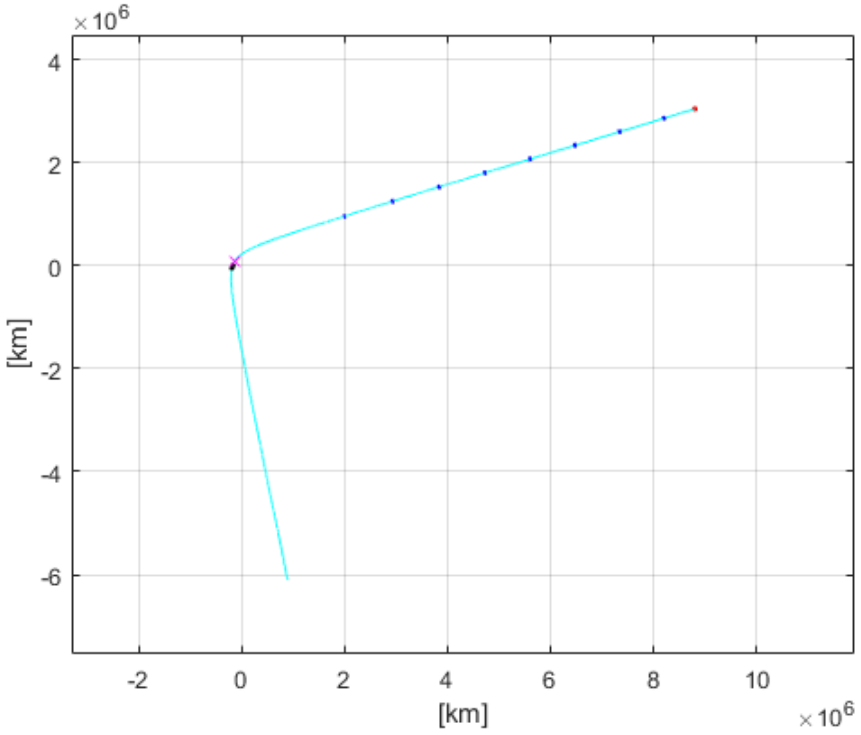

Figure 8.45: Voyager 2 flyby of Saturn without celestial bodies

In the figure above, the most important events are shown more clearly than in the Figure 8.43. It can be seen that there is occultation during a certain time interval while doing the flyby of Saturn.

B. *Occultation*

The time interval when there is occultation calculated with the algorithm will be studied in this section.

The figure below shows a zoom of the Figure 8.43 at the time interval when there is occultation in order to see more clearly when it starts and when it ends.

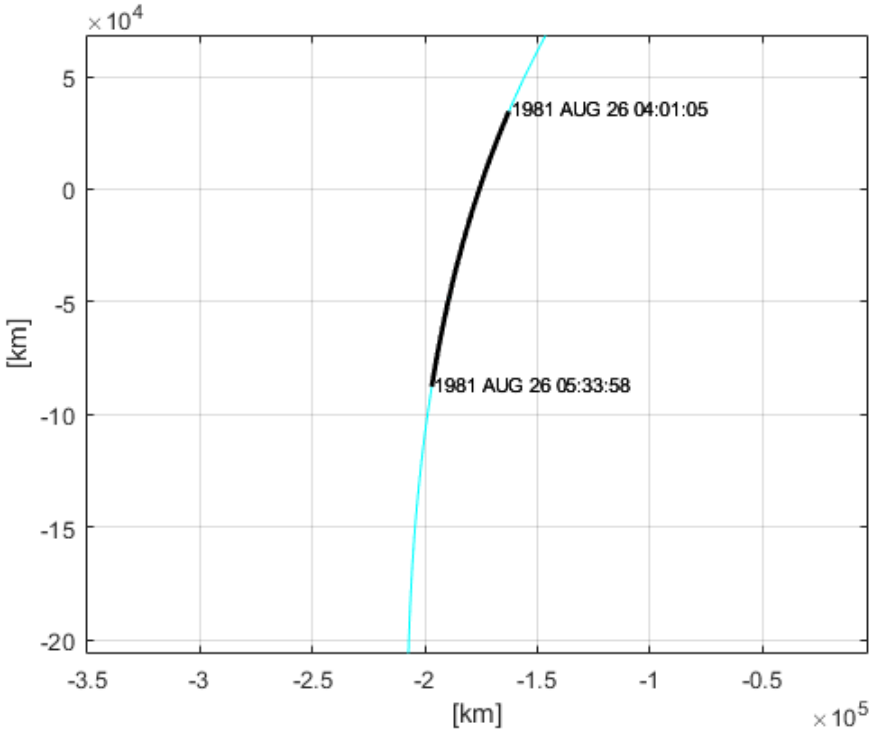

Figure 8.46: Zoom to the occultation of Voyager 2 at Saturn

The figure above shows that the occultation of the Voyager 2 at Saturn starts on '1981 AUG 26 04:01:13' and ends on '1981 AUG 26 05:33:24'.

B. *Minimum distance*

The best moment do the observation from the minimum distance possible from the Voyager 2 to Saturn will be studied in this section.

In the figure below (Figure 8.47), it can be seen a zoom of the Figure 8.43 at the moment when the minimum distance happens.

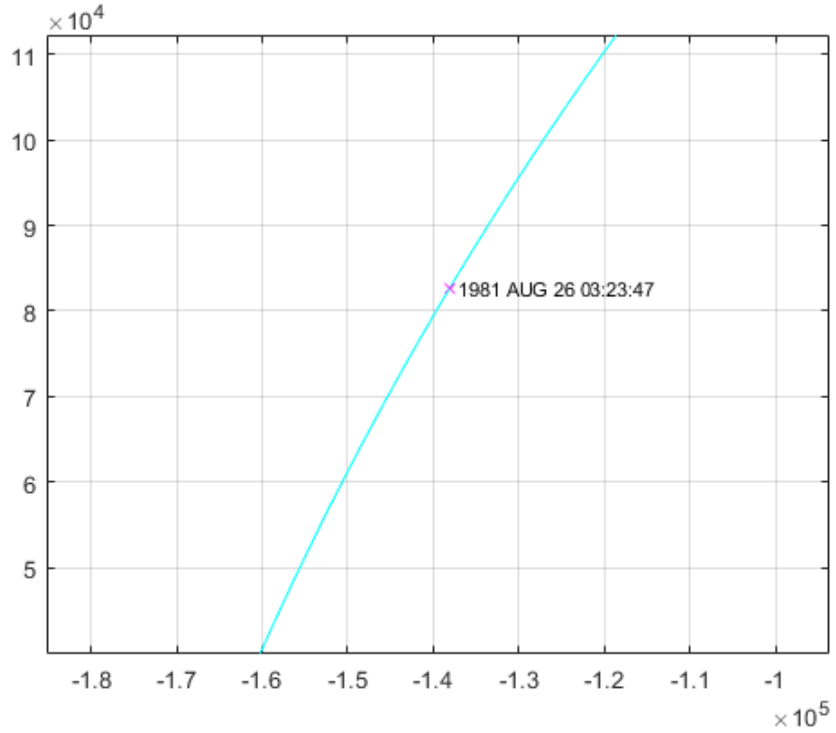

Figure 8.47: Zoom to the minimum distance between Voyager 2 and Saturn

As it can be seen on the figure above, the minimum distance occurs on '1981 AUG 26 03:23:47'.

As the occultation, as seen before, is not happening at this moment, and the phase angle is smaller than the one set as the maximum phase angle to do an observation, the time to do the observation from the minimum distance is set at UTC time: '1981 AUG 26 03:23:47'.

C. *Maximum illumination*

As with the case seen before at which the Voyager 1 is doing a flyby of Saturn, the moment of maximum illumination occurs at the start of the time interval studied. This is due to the same fact as in the case mentioned before; the phase angle increases as the spacecraft is approaching the planet so the maximum illumination will always happen at the point more far away from the planet. Thus, the observation of the maximum illumination can not be designed in this case.

D. *Transits*

In this section, the results obtained of the transits of Titan calculated for the case when Voyager 1 makes a flyby of Saturn are presented below.

In the figure below (Figure **??**), it is shown a zoom of the Figure 8.43 to the moment when the transit at the minimum distance occurs.

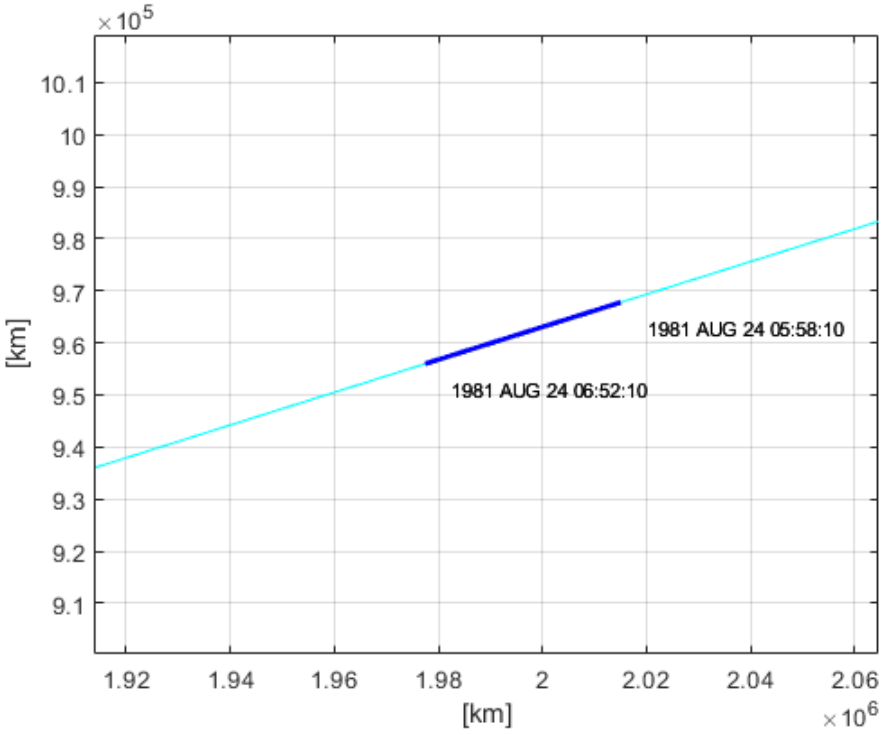

Figure 8.48: Zoom to the transit of Mimas as seen from the Voyager 2

In the image above, it can be seen that the transit starts at UTC time: '1981 AUG 24 05:57:35' and ends at UTC time: '1981 AUG 24 06:52:19'. As there is not occultation during the time interval when the transit occurs, and as the condition to design the observation of the transit is to the moon to be in the middle of its transit, the time to do the observation of the transit of Io as seen from the Voyager 2 is set at UTC time: '1981 AUG 24 06:24:57.030'.

8.1.8 Cassini to Saturn

In this section, the results of the study of the trajectory of the first flyby of the Cassini probe to Saturn will be shown.

The time interval that has been studied for this particular case is from '2004 JUN 23 00:00:00.000' to '2004 JUL 13 00:00:00.001'.

A. *Flyby plot*

In this section, a comparison between the flyby plot obtained with the algorithm *PlotFlyby* and the real trajectory is done.

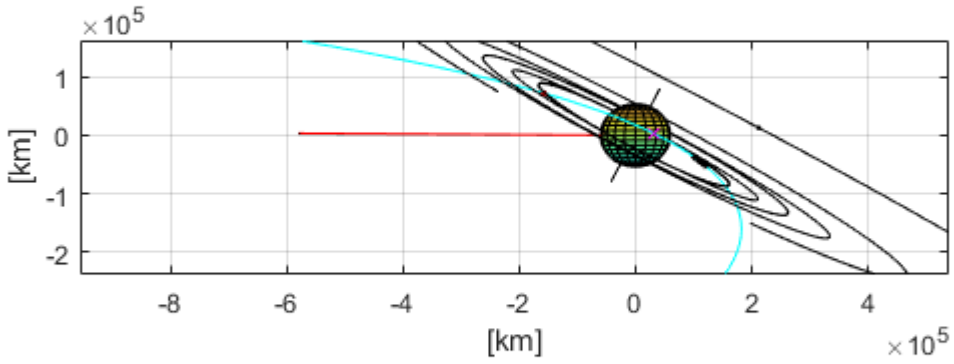

Figure 8.49: Cassini first approach to Saturn

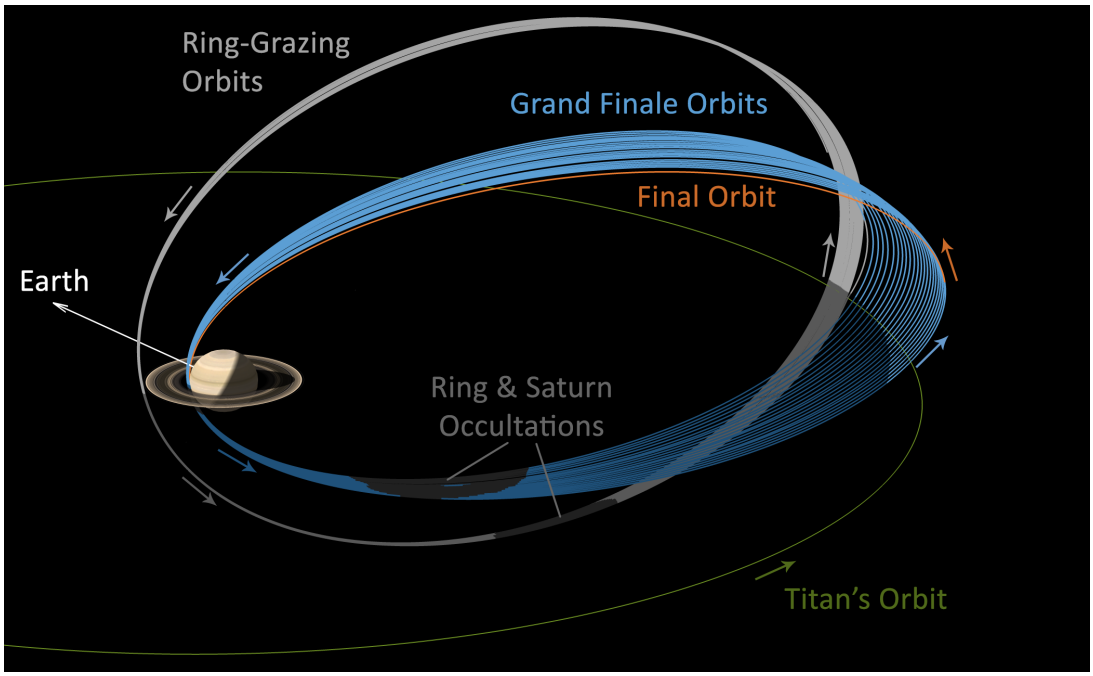

Figure 8.50: Real trajectory of Cassini orbiting Saturn[39]

As it can be seen if the two figures above are compared, although one is rotated with respect to the other, the trajectory obtained with the algorithm is quite faithful to the real trajectory that Cassini made when flying over Saturn.

B. *Occultation*

The time interval when there is occultation calculated with the algorithm will be studied in this section.

The figure below shows a zoom of the Figure 8.49 at the time interval when there is occultation in order to see more clearly when it starts and when it ends.

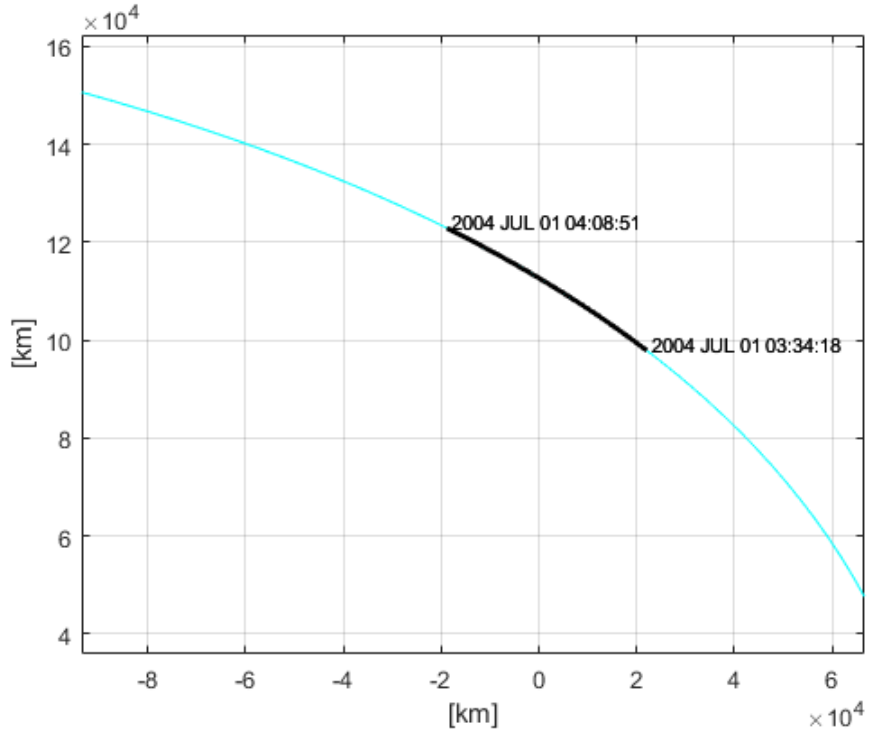

Figure 8.51: Zoom to the occultation of Cassini at Saturn

The figure above shows that the occultation of the Cassini at Saturn starts on '2004 JUL 01 03:34:18' and ends on '2004 JUL 01 04:08:51'.

B. *Minimum distance*

The optimum moment to do an observation from the minimum distance possible from the Cassini to Saturn will be studied in this section.

In the figure below (Figure 8.52), it can be seen a zoom of the Figure 8.49 at the moment when the minimum distance happens.

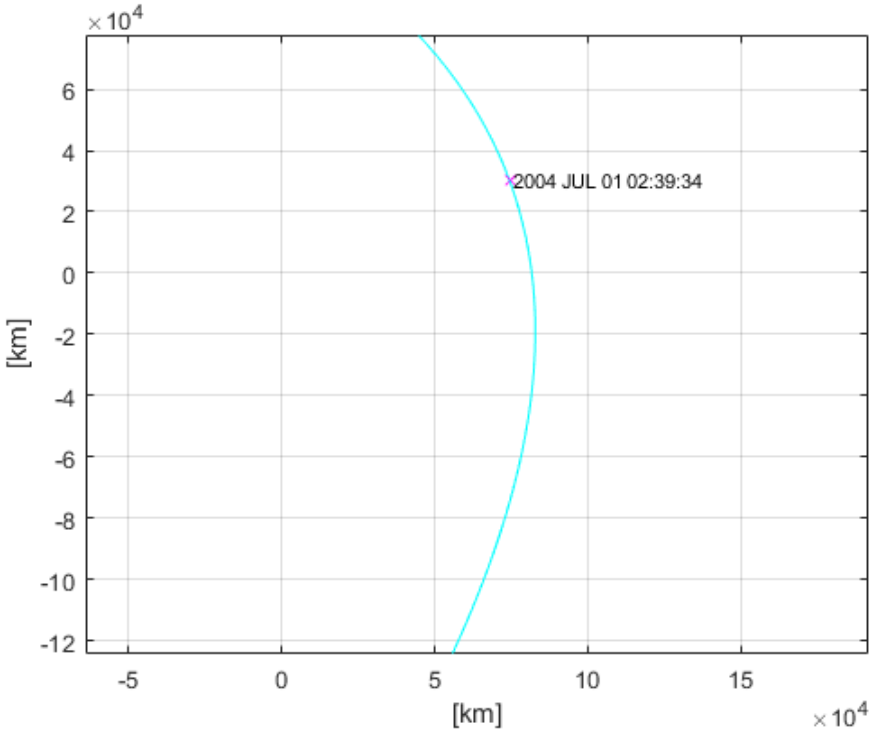

Figure 8.52: Zoom to the minimum distance between Cassini and Saturn

As it can be seen on the figure above, the minimum distance occurs on '2004 JUL 01 02:39:34'.

As the occultation does not happen at this moment, and the phase angle is smaller than the one set as the maximum phase angle to do an observation, the time to do the observation from the minimum distance is set at UTC time: '2004 JUL 01 02:39:34'.

C. *Maximum illumination*

In this section, the best moment to do the observation of Saturn with the condition of the maximum illumination will be shown.

In the figure below, a zoom of the Figure 8.49 at the time when the maximum illumination happens can be seen.

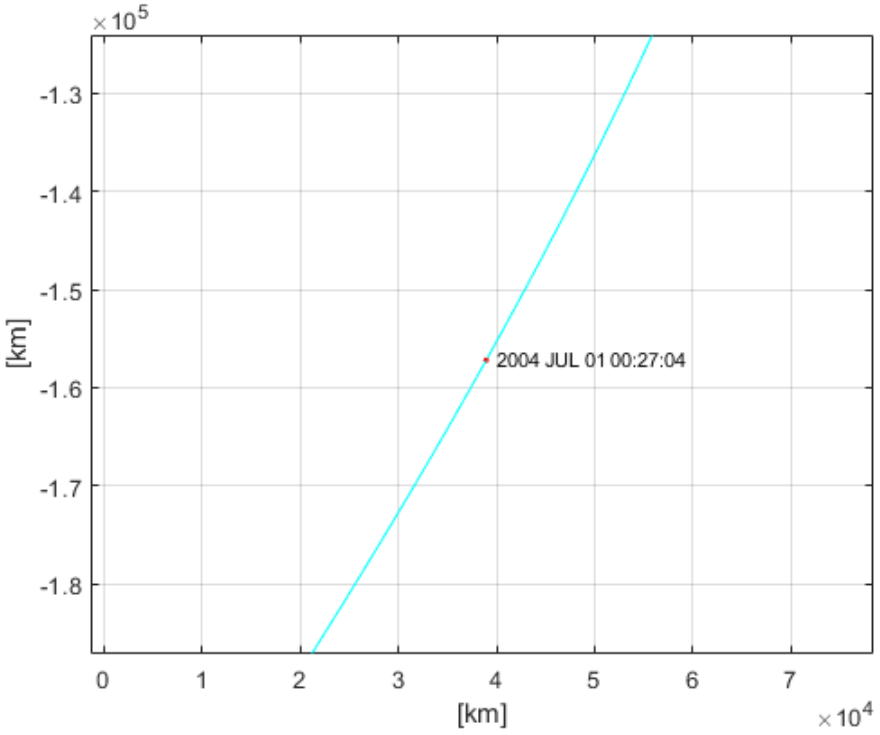

Figure 8.53: Zoom to the maximum illumination of Saturn as seen from the Cassini

As it can be seen in the the figure above, the moment when the maximum illumination happens is at UTC time: '2004 JUL 01 00:27:04'.

As there is not occultation at this time, the observation of Saturn at its maximum illumination as seen from the Cassini is set at UTC time: '2004 JUL 01 00:27:04'.

D. *Transits*

There is no transit of the moon Mimas as seen from the Cassini as its trajectory does not coincide with the orbit of the moon.

8.2 Mercury transit

In this section, the results obtained with the algorithm *MercuryTransit* are shown.

The time interval that has been studied in order to obtain the moments when the transit starts and ends has been from '2019 NOV 01 00:00:00.000' to '2019 NOV 30 23:59:59.999'.

In the figure below (Figure 8.54), it can be seen that there is a time interval during which there is transit of Mercury. Note that the trajectory of Mercury is represented by a blue line, the trajectory of the Earth by a green line and the transit by a black line.

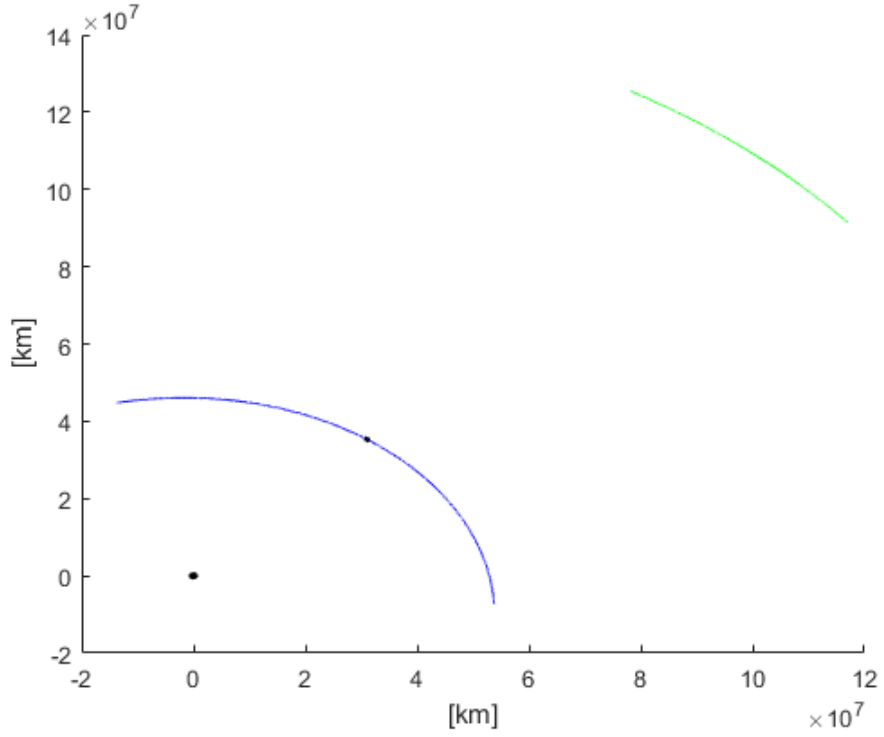

Figure 8.54: Mercury, Earth and Sun trajectories with the Mercury transit marked

In order to study more closely and see more clearly the time interval during which the transit of Mercury happens a zoom of the Figure 8.54 has been made. The result is the Figure 8.55 shown below.

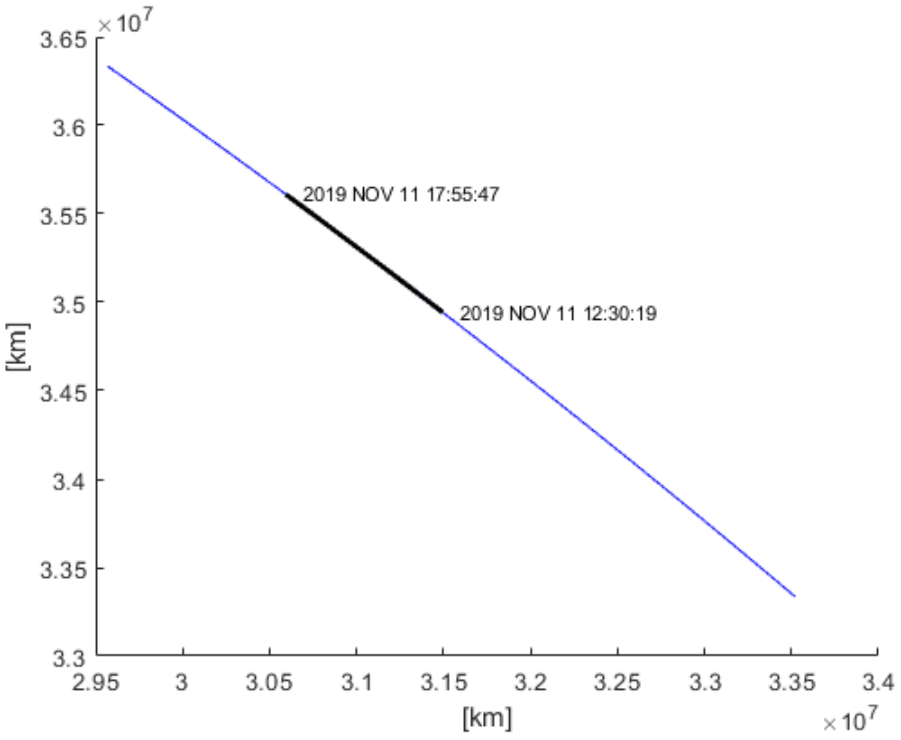

Figure 8.55: Zoom of the time interval during which there is a transit of Mercury

As it can be seen on the figure above, the algorithm calculates that the transit of Mercury, as seen from the center of the Earth, starts at UTC time: '2019 NOV 11 12:31:45' and ends at UTC time: '2019 NOV 11 17:55:47'.

If a comparison between the start computed by the algorithm and the real start when it is expected to occur presented in Chapter **??**, it can be seen that there is an error of 3 minutes. For the case of the ending, the error is of 9 minutes.

These two errors could be due to some inaccuracy in the position kernels of Mercury since this planet has been poorly studied.

It could also be because the kernels that were used to calculate this transit were not the ones that would have been better approximated to the real time. Even so, it will have to wait until November 11 to discover at what exact moment the Mercury transit will eventually happen.

Chapter 9

Environmental impact

Regarding the environmental impact of this project, it could be said that it is practically null since whenever there has been a displacement, public transport has been used to pollute less. Nevertheless, there is one thing that has contaminated, the energy consumed by computers for writing the code and memory of this project.

The pollution caused by the use of the author's laptop can be calculated thanks to an article from the University of Pennsylvania which estimates that a laptop with a power of 80kWh would consume 0.05kg of CO_2 per hour. Since the author's laptop consumes 33.33 kWh, and using a simple conversion, it can be calculated a total consumption of 11.56 kg of CO_2.

Chapter 10

Conclusion

Once the project is finished, it can be affirmed that the objectives have been achieved, among them, the functioning of the SPICE library has been understood, as well as the development of a subset of algorithms based in this library that are capable of designing the best timing to do the observations during an interplanetary mission.

This project is just a first approach to what can be achieved by developing software based on the SPICE library. There are many actions that could be carried out in order to improve the usability and performance of the different algorithms, as well as expand them to calculate the trajectory that a spacecraft should take, taking into account the observations that are desired to do.

As can be seen in the Chapter 8, specifically those from Cassini to Jupiter, the events that have been chosen in this project as a condition for making an observation may not be the ones that are really a condition for the observations. Thus, for future projects, it would be necessary to study what conditions have to be fulfilled in order for an observation to be made and, in this way, using the algorithms created in this work, be able to design the observations.

Another possibility that in this work could not be carried out due to the lack of time, but could be considered as a future project, would be to calculate the ignitions of the engines that are necessary in order to make a flyby from a planet based on certain conditions imposed by the user. In this way, if the spacecraft had a certain initial trajectory, but to be able to do the observations planned the trajectory would have to be altered, the acceleration needed to do so and be able to meet the requirements could be calculated. Thus, the amount of fuel and the direction and the moment in which this ignition of the motor would be needed to correct the path of the probe could be known.

Finally, an improvement of the algorithms created in this work, more specifically the algorithm that calculates the Mercury transit, would be to calculate this transit but instead of doing so from the center of the Earth, to get it from any point of the surface of the Earth, such as from Barcelona. Thus, with this algorithm, it could be calculated at what time interval, an object located at a point on the surface of

a planet, such as a Rover on the surface of Mars, would see a transit from another planet.

Bibliography

[1] Cassini Equinox Mission. http://sci.esa.int/cassini-huygens/43181-cassini-tour-equinox-mission. [Accessed: 2019-06-05].

[2] cspice_ilumin. https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/MATLAB/mice/cspice_ilumin.html. [Accessed: 2019-06-08].

[3] Equatorial view of the three spacecraft flybys of Saturn. On the... | Download Scientific Diagram. https://www.researchgate.net/figure/Equatorial-view-of-the-three-spacecraft-flybys-of-Saturn-On-the-trajectories-the-black{_}fig4{_}225973303. [Accessed: 2019-06-06].

[4] N1352713767_1.IMG. https://pds-rings.seti.org/viewmaster/volumes/COISS_1xxx/COISS_1002/data/1352683583_1352743767/N1352713767_1.IMG. [Accessed: 2019-06-08].

[5] N1352719767_1.IMG. https://pds-rings.seti.org/viewmaster/volumes/COISS_1xxx/COISS_1002/data/1352683583_1352743767/N1352719767_1.IMG. [Accessed: 2019-06-08].

[6] New Horizons : The Path to Pluto and Beyond. http://pluto.jhuapl.edu/Mission/The-Path-to-Pluto-and-Beyond.php{#}Pluto-Flyby. [Accessed: 2019-06-06].

[7] New Horizons: 2019 Onward. http://pluto.jhuapl.edu/Mission/2019-Onward.php. [Accessed: 2019-06-05].

[8] New Horizons: Navigating to Pluto - Sky & Telescope. https://www.skyandtelescope.com/astronomy-news/new-horizons-navigating-to-pluto-040320154/. [Accessed: 2019-06-06].

[9] Voyager - Mission Overview. https://voyager.jpl.nasa.gov/mission/. [Accessed: 2019-06-05].

[10] Voyager 1 entró en el espacio interestelar - Archivo Digital de Noticias de Colombia y el Mundo desde 1.990 - eltiempo.com. https://www.eltiempo.com/archivo/documento/CMS-13060839. [Accessed: 2019-06-05].

[11] Cassini Trajectory | NASA Solar System Exploration. https://solarsystem.nasa.gov/resources/11776/cassini-trajectory/, 1997. [Accessed: 2019-06-05].

[12] Voyager trajectory - Our Planet. https://ourplnt.com/12-years-new-horizons-launch/voyager-trajectory/, 2018. [Accessed: 2019-06-05].

[13] Asmar, S. W., Bolton, S. J., Buccino, D. R., Cornish, T. P., Folkner, W. M., Formaro, R., Iess, L., Jongeling, A. P., Lewis, D. K., Mittskus, A. P., Mukai, R., and Simone, L. The Juno Gravity Science Instrument. *Space Science Reviews 213*, 1-4 (2017), 205–218.

[14] Barnett, A., Shekhtman, L., and Thompson, J. R. Cassini Solstice Mission | NASA Solar System Exploration, 2018. [Accessed: 2019-06-05].

[15] Barnett, A., Shekhtman, L., and Thompson, J. R. Overview | Cassini – NASA Solar System Exploration. https://solarsystem.nasa.gov/missions/cassini/overview/, 2019. [Accessed: 2019-06-05].

[16] Brown, D., and Fox, K. Voyager - NASA's Voyager 2 Probe Enters Interstellar Space. https://voyager.jpl.nasa.gov/news/details.php?article{_}id=112. [Accessed: 2019-06-05].

[17] contributors, W. Occultation — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Occultation&oldid=891748222, 2019. [Accessed: 2019-06-05].

[18] contributors, W. Transit (astronomy) — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Transit_(astronomy)&oldid=899087558, 2019. [Accessed: 2019-06-05].

[19] contributors, W. Transit of mercury — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Transit_of_Mercury&oldid=899174090, 2019. [Accessed: 2019-06-05].

[20] Corum, J. Mapping Saturn's Moons - The New York Times. https://www.nytimes.com/interactive/2015/12/18/science/space/nasa-cassini-maps-saturns-moons.html, 2015. [Accesed: 2019-06-05].

[21] Eshleman, V. R., Hinson, D. P., Lindal, G. F., and Tyler, G. L. Past and future of radio occultation studies of planetary atmospheres. *Advances in Space Research 7*, 12 (1987), 29–32.

[22] Facility, A. I. An Overview of SPICE NASA ' s Ancillary Data System.

[23] Gladstone, G. R., Stern, S. A., Ennico, K., Olkin, C. B., Weaver, H. A., Young, L. A., Summers, M. E., Strobel, D. F., Hinson, D. P., Kammer, J. A., Parker, A. H., Steffl, A. J., Linscott, I. R., Parker, J. W., Cheng, A. F., Slater, D. C., Versteeg, M. H.,

Greathouse, T. K., Retherford, K. D., Throop, H., Cunningham, N. J., Woods, W. W., Singer, K. N., Tsang, C. C. C., Schindhelm, E., Lisse, C. M., Wong, M. L., Yung, Y. L., Zhu, X., Curdt, W., Lavvas, P., Young, E. F., Tyler, G. L., and New Horizons Science Team, t. N. H. S. The atmosphere of Pluto as observed by New Horizons. *Science (New York, N.Y.) 351*, 6279 (mar 2016), aad8866.

[24] Grundy, W. M., Buratti, B. J., Cheng, A. F., Emery, J. P., Lunsford, A., McKinnon, W. B., Moore, J. M., Newman, S. F., Olkin, C. B., Reuter, D. C., Schenk, P. M., Spencer, J. R., Stern, S. A., Throop, H. B., Weaver, H. A., and New Horizons Team. New horizons mapping of Europa and Ganymede. *Science (New York, N.Y.) 318*, 5848 (oct 2007), 234–7.

[25] Guo, Y., and Farquhar, R. W. New horizons mission design. *New Horizons: Reconnaissance of the Pluto-Charon System and the Kuiper Belt*, January 2007 (2009), 49–74.

[26] Hansen, C. J., Bolton, S. J., Matson, D. L., Spilker, L. J., and Lebreton, J.-P. The Cassini–Huygens flyby of Jupiter. *Icarus 172*, 1 (nov 2004), 1–8.

[27] James, S. E. *OBSERVER'S HANDBOOK.* ROYAL ASTRONOMICAL SOC, 2018.

[28] Kliore, A. J., Anderson, J. D., Armstrong, J. W., Asmar, S. W., Hamilton, C. L., Rappaport, N. J., Wahlquist, H. D., Ambrosini, R., Flasar, F. M., French, R. G., Iess, L., Marouf, E. A., and Nagy, A. F. Cassini radio science. *Space Science Reviews 115*, 1-4 (2005), 1–70.

[29] Maize, E., Edgington, S. G., and Pitesky, J. E. The "Grand Finale" of the Cassini Mission | Conde Duque Madrid. `https://www.condeduquemadrid.es/en/activities/grand-finale-cassini-mission`. [Accessed: 2019-06-05].

[30] Moore, J. M., McKinnon, W. B., Cruikshank, D. P., Gladstone, G. R., Spencer, J. R., Stern, S. A., Weaver, H. A., Singer, K. N., Showalter, M. R., Grundy, W. M., Beyer, R. A., White, O. L., Binzel, R. P., Buie, M. W., Buratti, B. J., Cheng, A. F., Howett, C., Olkin, C. B., Parker, A. H., Porter, S. B., Schenk, P. M., Throop, H. B., Verbiscer, A. J., Young, L. A., Benecchi, S. D., Bray, V. J., Chavez, C. L., Dhingra, R. D., Howard, A. D., Lauer, T. R., Lisse, C. M., Robbins, S. J., Runyon, K. D., and Umurhan, O. M. Great Expectations: Plans and Predictions for New Horizons Encounter With Kuiper Belt Object 2014 MU 69 ("Ultima Thule"). *Geophysical Research Letters 45*, 16 (aug 2018), 8111–8120.

[31] NASA. *Seven Century Catalog of Mercury Transits: 1601 CE to 2300 CE*. NASA.

[32] Natha, A., and Espinoza, L. Voyager - Jupiter Approach. `https://voyager.jpl.nasa.gov/mission/science/jupiter/`. [Accessed: 2019-06-05].

[33] Natha, A., and Espinoza, L. Voyager - Saturn Approach. `https://voyager.jpl.nasa.gov/mission/science/saturn/`. [Accessed: 2019-06-05].

[34] Overview, A., and Flight, I. *Deep space craft: an overview of interplanetary flight*, vol. 47. 2013.

[35] Peralta, F., and Flanagan, S. Cassini interplanetary trajectory design. *Control Engineering Practice 3*, 11 (1995), 1603–1610.

[36] Ping, C. U. I., and Ang, S. H. Voyager Mission Description. *Space Science Reviews 21*, 2 (1977), 77–101.

[37] REITSEMA, H. J., HUBBARD, W. B., LEBOFSKY, L. A., and THOLEN, D. J. Occultation by a Possible Third Satellite of Neptune. *Science 215*, 4530 (jan 1982), 289–291.

[38] Reuter, D. C., Simon-Miller, A. A., Lunsford, A., Baines, K. H., Cheng, A. F., Jennings, D. E., Olkin, C. B., Spencer, J. R., Stern, S. A., Weaver, H. A., and Young, L. A. Jupiter cloud composition, stratification, convection, and wave motion: a view from new horizons. *Science (New York, N.Y.) 318*, 5848 (oct 2007), 223–5.

[39] Schilling, K. The Cassini/Huygens space mission to explore the Saturnian system. *IFAC Proceedings Volumes (IFAC-PapersOnline) 16* (2005), 151–156.

[40] Sitjà, M. C., Moinelo, A. C., Frew, D., and Arviset, C. SOLAR SYSTEM GEOMETRY TOOLS WITH SPICE FOR ESA ' S PLANETARY MISSIONS Telespazio VEGA UK SL for ESA (Contact : marc.costa@esa.int) SERCO for ESA. 2–7.

[41] Spencer, J. R., Stern, S. A., Cheng, A. F., Weaver, H. A., Reuter, D. C., Retherford, K., Lunsford, A., Moore, J. M., Abramov, O., Lopes, R. M. C., Perry, J. E., Kamp, L., Showalter, M., Jessup, K. L., Marchis, F., Schenk, P. M., and Dumas, C. Io volcanism seen by new horizons: a major eruption of the Tvashtar volcano. *Science (New York, N.Y.) 318*, 5848 (oct 2007), 240–3.

[42] Stern, S. A. The new horizons pluto kuiper belt mission: An overview with historical context. *New Horizons: Reconnaissance of the Pluto-Charon System and the Kuiper Belt* (2009), 3–21.

[43] Stern, S. A., Bagenal, F., Ennico, K., Gladstone, G. R., Grundy, W. M., McKinnon, W. B., Moore, J. M., Olkin, C. B., Spencer,

J. R., Weaver, H. A., Young, L. A., Andert, T., Andrews, J., Banks, M., Bauer, B., Bauman, J., Barnouin, O. S., Bedini, P., Beisser, K., Beyer, R. A., Bhaskaran, S., Binzel, R. P., Birath, E., Bird, M., Bogan, D. J., Bowman, A., Bray, V. J., Brozovic, M., Bryan, C., Buckley, M. R., Buie, M. W., Buratti, B. J., Bushman, S. S., Calloway, A., Carcich, B., Cheng, A. F., Conard, S., Conrad, C. A., Cook, J. C., Cruikshank, D. P., Custodio, O. S., Dalle Ore, C. M., Deboy, C., Dischner, Z. J. B., Dumont, P., Earle, A. M., Elliott, H. A., Ercol, J., Ernst, C. M., Finley, T., Flanigan, S. H., Fountain, G., Freeze, M. J., Greathouse, T., Green, J. L., Guo, Y., Hahn, M., Hamilton, D. P., Hamilton, S. A., Hanley, J., Harch, A., Hart, H. M., Hersman, C. B., Hill, A., Hill, M. E., Hinson, D. P., Holdridge, M. E., Horanyi, M., Howard, A. D., Howett, C. J. A., Jackman, C., Jacobson, R. A., Jennings, D. E., Kammer, J. A., Kang, H. K., Kaufmann, D. E., Kollmann, P., Krimigis, S. M., Kusnierkiewicz, D., Lauer, T. R., Lee, J. E., Lindstrom, K. L., Linscott, I. R., Lisse, C. M., Lunsford, A. W., Mallder, V. A., Martin, N., McComas, D. J., McNutt, R. L., Mehoke, D., Mehoke, T., Melin, E. D., Mutchler, M., Nelson, D., Nimmo, F., Nunez, J. I., Ocampo, A., Owen, W. M., Paetzold, M., Page, B., Parker, A. H., Parker, J. W., Pelletier, F., Peterson, J., Pinkine, N., Piquette, M., Porter, S. B., Protopapa, S., Redfern, J., Reitsema, H. J., Reuter, D. C., Roberts, J. H., Robbins, S. J., Rogers, G., Rose, D., Runyon, K., Retherford, K. D., Ryschkewitsch, M. G., Schenk, P., Schindhelm, E., Sepan, B., Showalter, M. R., Singer, K. N., Soluri, M., Stanbridge, D., Steffl, A. J., Strobel, D. F., Stryk, T., Summers, M. E., Szalay, J. R., Tapley, M., Taylor, A., Taylor, H., Throop, H. B., Tsang, C. C. C., Tyler, G. L., Umurhan, O. M., Verbiscer, A. J., Versteeg, M. H., Vincent, M., Webbert, R., Weidner, S., Weigle, G. E., White, O. L., Whittenburg, K., Williams, B. G., Williams, K., Williams, S., Woods, W. W., Zangari, A. M., and Zirnstein, E. The Pluto system: Initial results from its exploration by New Horizons. *Science 350*, 6258 (oct 2015), aad1815–aad1815.

[44] Stone, E. C., and Miner, E. D. Voyager 2 encounter with the saturnian system. *Science (New York, N.Y.) 215*, 4532 (jan 1982), 499–504.

[45] Stone, E. C., and Miner, E. D. The voyager 2 encounter with the uranian system. *Science (New York, N.Y.) 233*, 4759 (jul 1986), 39–43.

[46] Stone, E. C., and Miner, E. D. The voyager 2 encounter with the neptunian system. *Science (New York, N.Y.) 246*, 4936 (dec 1989), 1417–21.

[47] Streiffert, B., Polanskey, C. A., Reilly, T. O., and Colwell, J. Science opportunity analyzer - a multi-mission tool for planning. *Jpl*.