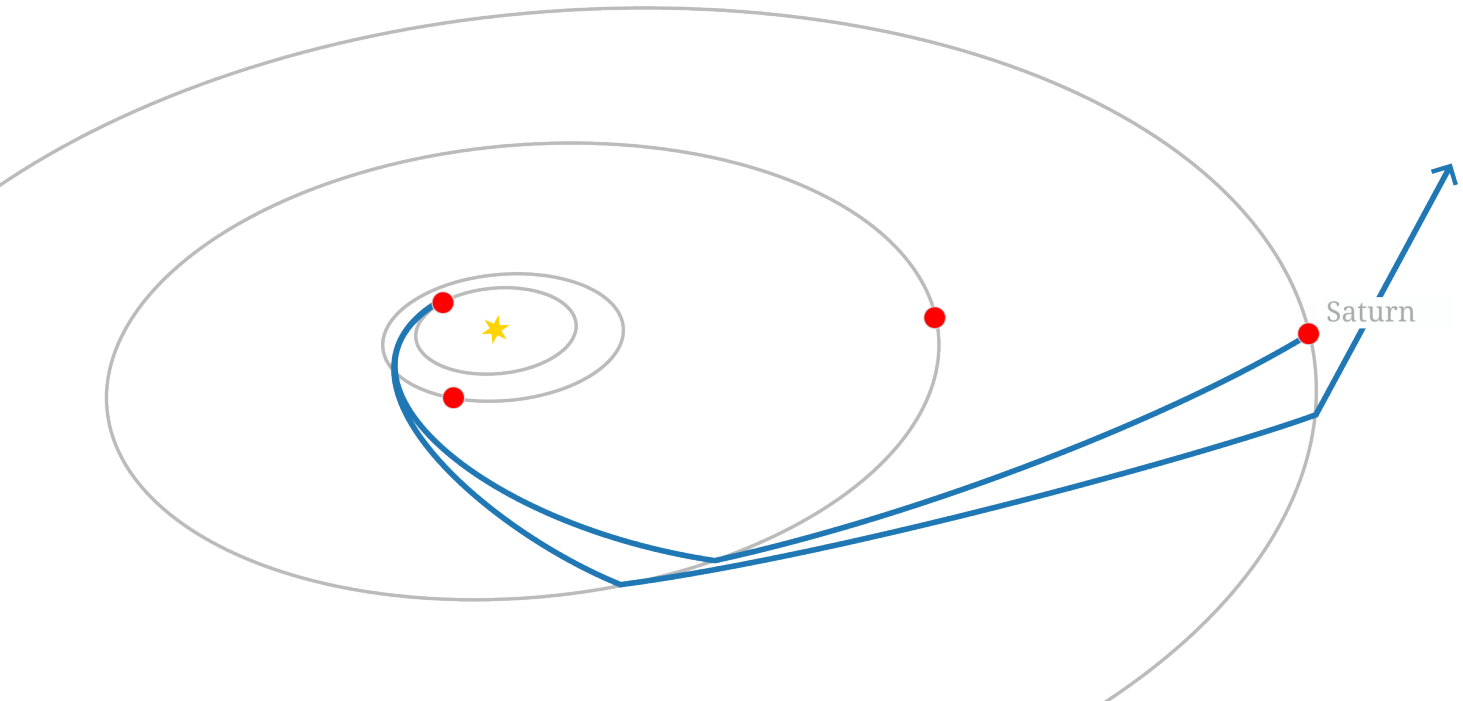
Exploring Interplanetary Travel: Planning and Optimising a Mission from Earth to Saturn's Moon Enceladus

Félix Bommier, 6i

Literargymnasium Rämibühl
Maturitätsarbeit 2025
Supervisor: Manuel Benz



Contents

Contents **1**

List of Figures **2**

1 Introduction **3**
1.1 Historical Background . . . 3
1.2 Personal Motivation . . . 3
1.3 Aim . . . 3

2 Fundamental Concepts **5**
2.1 Planetary Information . . . 5
2.1.1 Ephemerides . . . 5
2.1.2 Mass . . . 6
2.2 Gravity Assist . . . 6
2.2.1 Advantages and Limits . . . 10
2.3 Oberth Effect . . . 11
2.4 Deep-Space Maneuver (DSM) . . . 13
2.5 Delta-v Budget . . . 13
2.6 Lambert's Problem . . . 14
2.7 Optimization . . . 15

3 Implementation of an Existing Mission **17**
3.1 Historical Background . . . 17
3.2 Mission-Specific Considerations . . . 18
3.2.1 Advantages . . . 18
3.2.2 Simplification . . . 18
3.2.3 Python Library `poliastro` . . . 18
3.3 Implementation in Python . . . 18
3.3.1 Calculating Delta-v . . . 20
3.3.2 Shortcomings . . . 22
3.4 Effect of Parameters . . . 23
3.4.1 Removing Flybys . . . 23
3.4.2 Changing Flyby Dates . . . 23

4 Designing a New Mission **25**
4.1 Optimization Complexity . . . 25
4.2 Choosing the Right Software . . . 25
4.2.1 `pykep` and `pygmo` . . . 26
4.2.2 Algorithms Used: COBYLA and MBH . . . 26

4.3 Implementation in Python 27
4.3.1 Different Flyby Sequences 30
4.3.2 Shortcomings 31
4.4 General Limitations of Optimization 32

5 Reflection **33**
5.1 Strengths 33
5.2 Limitations 33
5.3 Encountered Challenges 34
5.4 Improvements and Additions 34
5.5 Final Words 35

6 References **36**

List of Figures

1 Spacecraft's velocity change due to a gravity assist. 7
2 Hyperbolic trajectory of a spacecraft during a gravity assist. 8
3 Effect of the impact parameter (b) on the trajectory and velocity. . 9
4 Possible outcomes of gravity assist maneuvers. 10
5 Relationship between mass ratio (initial mass to final mass) and the change in velocity (Δv) normalised by exhaust velocity (v_e). . 13
6 Application of Lambert's problem for a transfer orbit. 14
7 Cassini's interplanetary flight path. 17
8 Calculated trajectory of the Cassini-Huygens spacecraft. 20
9 Effect of different planetary flyby sequences on the total Δv. . . . 23
10 Effect of shifting flyby dates on the total Δv. 24
11 Best optimised spacecraft trajectory for an $Earth \rightarrow Venus \rightarrow Mars$ flyby sequence. 30
12 Effect of flyby sequence on total Δv for optimised trajectories. . . 31

1 Introduction

The exploration of our solar system is driven by three fundamental goals: the search for life beyond Earth, the identification of valuable resources to support future endeavors and humanity's curiosity to understand the universe and our place within it. One of the most promising candidates in the search for extraterrestrial life is Enceladus, a moon of Saturn that has captured the attention of scientists and space agencies. Located far from the Sun, Enceladus experiences extremely cold temperatures and receives very little solar energy. Despite these harsh conditions, the European Space Agency (ESA) has identified Enceladus as a key target for future missions. They believe that the moon could host life, due to a subsurface ocean with a powerful source of chemical energy that could potentially fuel living organisms [18]. Furthermore, the plumes ejected from its icy crust are rich in organic compounds.

1.1 Historical Background

Enceladus was discovered in 1789 by Sir William Herschel. It is one of Saturn's moons, has a diameter of approximately 504 km and orbits Saturn at a distance of about 238 000 km [27]. Previous interplanetary missions, such as Voyager 1 and 2, conducted flybys of Saturn, which provided valuable data about the planet and its moons [27]. The Cassini-Huygens mission further revealed the mysteries of Enceladus through detailed observations and data collection during its close encounters with the moon [24]. Given this historical context, planning a journey to this celestial body is not only realistic but also a natural progression in the ongoing quest to understand our solar system.

1.2 Personal Motivation

My personal motivation for this paper is to deepen my understanding of the mathematics, physics and computational principles underlying interplanetary travel. This exploration will not only enhance my understanding of scientific concepts, such as gravitational forces and the dynamics of multibody systems, but also allow me to appreciate the engineering marvels that make space travel possible. Engaging with this topic will provide me with valuable insights into the real-world applications of theoretical principles in the field of aerospace engineering.

1.3 Aim

The objective of this paper is to plan a journey from Earth to Enceladus. This plan will incorporate the use of multiple gravity assists (MGA), the Oberth effect and deep space maneuvers (DSMs). First, I will calculate the trajectory of an

existing mission, where the departure time, flybys and arrival details are already known. Then, I will design a new trajectory for a future mission to Enceladus, determining the optimal departure time, identifying suitable flybys and establishing a timeline for arrival. This trajectory will be optimised to minimise the total change in velocity (Δv) required for the mission. With this paper, I aim to contribute to the ongoing dialogue about the feasibility of future missions to Enceladus.

2 Fundamental Concepts

To successfully carry out this work, it is essential to understand the following fundamental concepts behind interplanetary space travel.

2.1 Planetary Information

2.1.1 Ephemerides

Knowing the positions and velocities of celestial bodies at any given time is crucial for planning flybys and determining arrival points.

While planets follow elliptical orbits described by Kepler's laws [22], these orbits are influenced by several perturbations:

- **Gravitational Interactions:** The gravitational pull from other planets, moons and stars alters a planet's orbit and thus causes small changes in position and velocity. This can lead to phenomena like orbital resonances, where bodies' orbits become synchronised. This amplifies their gravitational effects [5].
- **Uneven Mass Distribution:** Planets with non-uniform mass distributions (e.g., mountains or oceans) experience slight orbital shifts due to gravitational imbalances, known as the J2 perturbation, especially in planets like Earth with an oblate shape [5].
- **Relativistic Effects:** According to general relativity, massive objects warp space-time, causing slight changes in a planet's orbit. For example, Mercury's orbit experiences precession due to relativistic effects from the Sun's gravity [36].

These perturbations complicate the calculations and require more advanced models than Kepler's laws for higher accuracy. Entire books, like *Astronomical Algorithms* by Jean Meeus [5], span hundreds of pages solely dedicated to these complex calculations.

Fortunately, existing libraries handle these calculations for us by providing ephemerides, which offer the position and velocity of a celestial body at any given time. In practical terms, ephemerides can be viewed as a function that provides the position R and velocity V of a celestial body at a specific point in time, expressed as

$$[R, V] = \text{Ephemerides}(b, t)\,, \tag{1}$$

where b is the identifier of the body and t is the epoch.[1] In this paper, we will use

[1] In astronomy, an epoch is a **specific point in time** used as a reference for tracking changing astronomical measurements. It is important because celestial coordinates and orbital elements vary over time due to gravitational influences.

the Jet Propulsion Laboratory (JPL) ephemerides provided by NASA [7], which are based on real observations, with interpolation for times between data points and with extrapolation for future predictions.

While the ephemerides provide a reliable foundation for planetary positions and velocities, they only partially simplify my task. I still needed to perform the majority of the work, including trajectory assembly, optimization and all associated calculations.

2.1.2 Mass

In addition to determining the position and velocity of celestial bodies, it is essential to account for the gravitational pull exerted by each planet, as described by Newton's law of universal gravitation [32]. The gravitational force is given by the equation

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

where F is the force, m_1 and m_2 are the masses of the objects interacting, r is the distance between the centers of the masses and G is the gravitational constant [32]. Thus, the gravitational force depends on the masses of both interacting bodies and knowing the mass of the Sun and each planet is therefore crucial. Fortunately, Python libraries provide easy access to mass data and thus simplify the process.[2]

2.2 Gravity Assist

A gravity assist, commonly referred to as a slingshot maneuver or unpowered flyby, is a vital technique used in interplanetary travel to gain speed and change trajectory without expending additional fuel. The fundamental principle behind a gravity assist involves the conservation of energy and momentum during a spacecraft's flyby of a celestial body [28].

The principle of conservation of energy asserts that the total energy in a closed system remains constant over time [25]. Specifically, for a spacecraft in motion, this means that the incoming kinetic energy (E_{in}) at the start of an interaction is equal to the outgoing kinetic energy (E_{out}) once the interaction has occurred, assuming no energy is lost to external forces. Mathematically, this can be expressed as

$$E_{\text{in}} = E_{\text{out}}\,. \tag{3}$$

Thus, the speed of the spacecraft relative to the celestial body is the same before and after the encounter. However, because the celestial body also moves

[2] For example the `poliastro` library [13].

in respect to the Sun, the spacecraft's velocity in respect to the Sun also changes [8], illustrated in Figure 1.

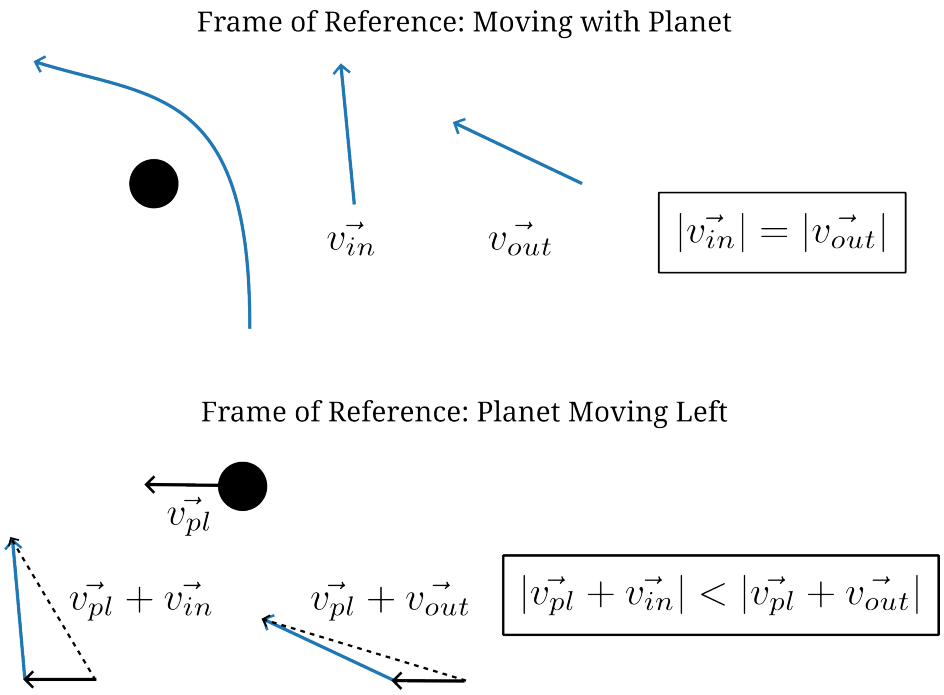


Figure 1: Spacecraft's velocity change due to a gravity assist. In the *Moving with Planet* frame, the spacecraft's incoming and outgoing velocities (v_{in} and v_{out}) have the same magnitude, as the gravitational interaction only changes the direction of the spacecraft's velocity. Considering the principle of conservation of energy, this behavior is logical since $E_{\text{in}} = E_{\text{out}}$ and thus $v_{\text{in}} = v_{\text{out}}$. In the *Planet Moving Left* frame, where the planet is considered to be moving, the spacecraft's outgoing velocity relative to the frame is greater than its incoming velocity due to the additional velocity contribution from the planet's motion. The planet's own velocity (v_{pl}) is simply added to the outgoing velocity of the spacecraft relative to the planet (v_{out}).

This process can easily be explained mathematically, since the trajectory of the spacecraft relative to the planet is a hyperbolic trajectory, as shown in Figure 2. This is the case because the spacecraft approaches the planet from a large distance with a velocity high enough to escape the planet's gravitational influence. As the spacecraft passes near the planet, the planet's gravity bends its trajectory, changing its direction and speed. However, because the spacecraft's velocity is above the escape velocity of the planet, it is not captured into orbit but instead follows an open, hyperbolic path. The gravitational interaction does not result in a bound elliptical or circular orbit, but results in a hyperbolic trajectory where the spacecraft exits the gravitational influence after the flyby [30].

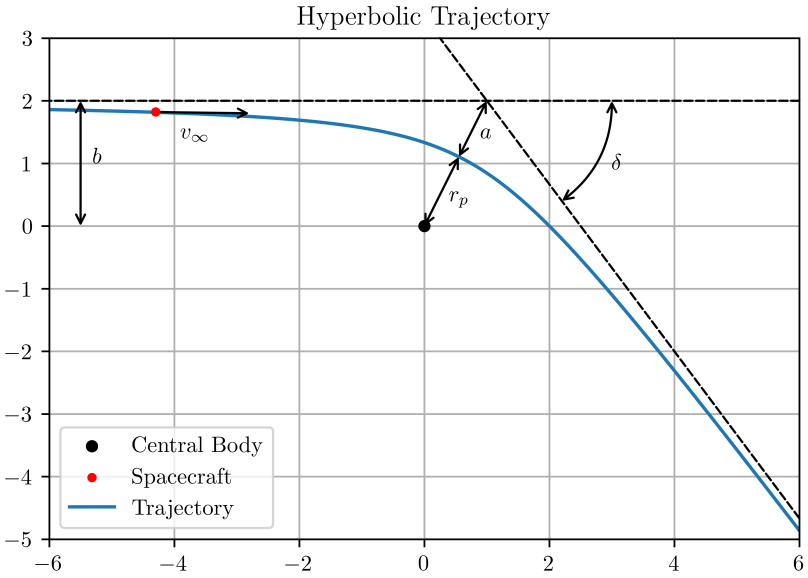


Figure 2: Hyperbolic trajectory of a spacecraft during a gravity assist [30]. The spacecraft's approach velocity (v_∞, also called hyperbolic excess velocity because it represents the spacecraft's velocity relative to a celestial body at infinite distance, where the gravitational influence of the body is negligible) is marked along the initial horizontal trajectory and the impact parameter (b) is indicated as the perpendicular distance between the trajectory and the central body. The angle δ represents the deflection angle of the spacecraft as it follows its hyperbolic path. The central body is shown as a black dot at the origin, while the spacecraft's initial position is marked as a red dot. The dashed line represents the asymptotes of the hyperbolic trajectory and the solid blue curve illustrates the spacecraft's actual path.

Therefore, by knowing the impact parameter b (which indicates how closely the spacecraft would pass by the planet in the absence of gravitational influence, see Figure 2) and the spacecraft's initial velocity v_∞ (known as the hyperbolic excess velocity, see Figure 2 for further details), one can very easily calculate all the necessary parameters, such as the semi-major axis a, the eccentricity e, the periapsis distance r_p and the deflection angle δ, to describe this hyperbolic trajectory [30].

The semi-major axis a (shown in Figure 2) can be calculated with this formula

$$a = -\frac{Gm}{v_\infty^2}\,, \tag{4}$$

the eccentricity e is

$$e = \sqrt{1 + \frac{b^2}{a^2}}\,, \tag{5}$$

the periapsis distance r_p is

$$r_p = -a(e-1) \tag{6}$$

and the deflection angle δ is

$$\delta = 2 \arcsin(\frac{1}{e}) \,. \tag{7}$$

Following these calculations, Figure 3 illustrates the effect of the impact parameter on the spacecraft's path and velocity as and Figure 4 the outcomes of gravity assist maneuvers based on different approach configurations.

This figure, along with all other graphs and associated calculations in this paper, was created by me using custom `python3` scripts and the `matplotlib` library, which I developed specifically for this project.

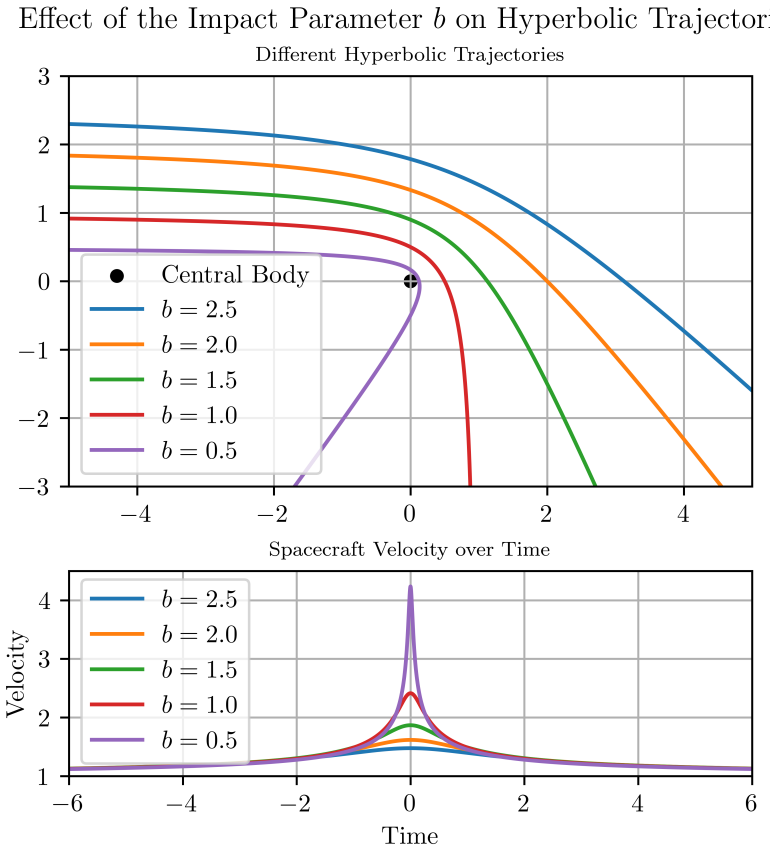


Figure 3: Effect of the impact parameter (b) on hyperbolic trajectories and spacecraft velocity. A smaller b (e.g., $b = 0.5$) results in a sharper turn due to a stronger gravitational interaction, while larger b values produce more gradual curves. As the spacecraft approaches the central body, its velocity peaks due to gravitational acceleration. Smaller impact parameters result in higher peak velocities as the gravitational forces are stronger due to the smaller distance between both bodies. Note that the graph is unitless because it is intended to visualise general trends and outcomes rather than specific examples. The focus is on illustrating the qualitative effects of varying the impact parameter b on the trajectory and velocity, without requiring precise numerical values. The graph is also presented in the frame moving with the planet, as if the planet were stationary.

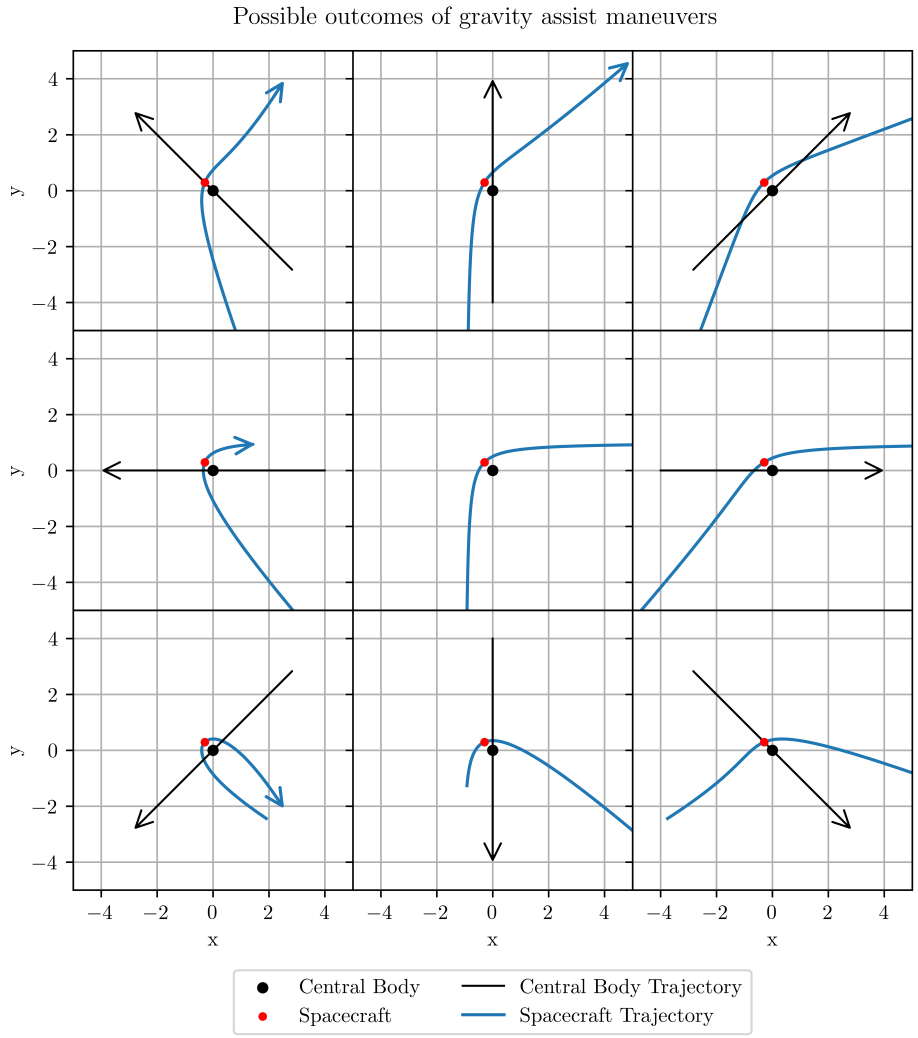


Figure 4: Possible outcomes of gravity assist maneuvers depending on the velocity vector and flyby position of the incoming spacecraft. The black lines represent the planet's trajectory and the blue lines depict the spacecraft's trajectory before and after the maneuver. The black dot indicates the position of the planet and the red dot marks the spacecraft's position at closest approach (at $t = 0$). The hyperbolic excess velocity v_∞ is 1 and the impact parameter b is also 1. The time frame for the simulation is $[-4, 4]$ (with the trajectories of the planet and spacecraft plotted from $t = -4$ to $t = 4$). The various configurations show how changes in approach angle and flyby distance affect the spacecraft's exit path. Depending on the velocity vector of the planet, the gravity assist can either speed up or slow down the spacecraft, both of which are useful in interplanetary travel. Additionally, the change in direction can vary significantly based on these parameters. Note that the configuration in the middle shows the absence of movement, corresponding to the scenario in Figure 3.

2.2.1 Advantages and Limits

Gravity assists provide a highly effective way of increasing a spacecraft's velocity without expending fuel. This conserves its propellant resources and is particularly advantageous in interplanetary missions where the delta-v budget (the total

change in velocity required, which is directly related to fuel consumption) is critical. We will discuss this in more detail later, in Section 2.5.

The primary constraint is the necessity of perfect alignments of planets, which can be infrequent.[3] Atmospheric drag also poses a challenge [28]. While closer approaches can yield more kinetic energy, excessive atmospheric entry can lead to energy loss due to drag. Additionally, as the periapsis (the distance between the spacecraft and the central body) decreases, the forces acting on the spacecraft increase exponentially (see Equation 2 and Figure 3 for more). This could ultimately damage or destroy the spacecraft. Finally, the planet has a physical volume, which establishes a minimum periapsis. If the spacecraft approaches any closer, it will crash into the planet.

2.3 Oberth Effect

The Oberth effect is a phenomenon in space travel that increases the efficiency of a spacecraft's engine when operating at high speeds, particularly when the spacecraft is near a massive celestial body, such as a planet [33]. When a spacecraft approaches a planet, it speeds up due to gravity. At that moment, the spacecraft can gain even more speed by firing its engines. The increase in kinetic energy is much greater than if the engines were fired when the spacecraft was moving slowly.

The basis of the Oberth effect is that kinetic energy increases with the square of velocity, but a rocket burn provides the same Δv regardless of the spacecraft's current speed. This means that a small increase in velocity at high speeds results in a much larger increase in kinetic energy.

The change in kinetic energy (ΔE_{kin}) during a burn can be expressed as

$$\Delta E_{\text{kin}} = \frac{1}{2} m (v_f^2 - v_i^2) \,, \tag{8}$$

where m is the mass of the spacecraft, v_f the final velocity after the burn and v_i initial velocity before the burn [31].

Since kinetic energy is proportional to the square of velocity ($E_{\text{kin}} = \frac{1}{2} m v^2$), a small speed increase results in a larger energy gain at higher speeds. Therefore, Oberth maneuvers or powered flybys allow spacecrafts to use their fuel more effectively, especially when flying close to planets, e.g. during gravity assists.

[3] For instance, the *Grand Tour* alignment of Jupiter, Saturn, Uranus and Neptune, which enabled the Voyager missions, will not occur again until the mid-22nd century [28].

Example

Let's assume the spacecraft has a mass of $m = 1000\,\mathrm{kg}$, a hyperbolic excess velocity of $v_\infty = 10\,000\,\mathrm{m/s}$ and its highest velocity during the gravity assist is $v_{\max} = 15\,000\,\mathrm{m/s}$. The spacecraft will perform a burn that will increase its velocity by $\Delta v = 2000\,\mathrm{m/s}$. The burn will either occur at the closest approach (periapsis) or at a point far from the planet and we will compare the energy change in both cases.

Performing the burn at the closest approach (periapsis): At closest approach, the spacecraft's velocity is high due to gravitational acceleration. Therefore, we calculate the change in kinetic energy using Equation 8 and substitute the given values:

$$\Delta E_{\mathrm{kin}} = \frac{1}{2} m (v_f^2 - v_i^2) \tag{9}$$

$$\Delta E_{\mathrm{kin}} = \frac{1}{2} \times 1000\,\mathrm{kg} \times ((15\,000 + 2000)^2 - 15\,000^2)\mathrm{m}^2/\mathrm{s}^2 \tag{10}$$

$$\Delta E_{\mathrm{kin}} = 32\,000\,000\,000\,\mathrm{J} \tag{11}$$

So, the spacecraft gains 32 billion joules of kinetic energy when the burn is performed performed at periapsis.

Performing the burn far from the planet: Now, let's assume the spacecraft performs the burn far from the planet, where its velocity is lower ($v_\infty = 10\,000\,\mathrm{m/s}$). We calculate the change in kinetic energy again by substituting the values:

$$\Delta E_{\mathrm{kin}} = \frac{1}{2} m (v_f^2 - v_i^2) \tag{12}$$

$$\Delta E_{\mathrm{kin}} = \frac{1}{2} \times 1000\,\mathrm{kg} \times ((10\,000 + 2000)^2 - 10\,000^2)\mathrm{m}^2/\mathrm{s}^2 \tag{13}$$

$$\Delta E_{\mathrm{kin}} = 22\,000\,000\,000\,\mathrm{J} \tag{14}$$

So, the spacecraft gains 22 billion joules of kinetic energy when the burn is performed performed far from the planet.

Conclusion: In this example, performing the burn at the closest approach (periapsis) results in a larger increase in kinetic energy (32 billion joules) compared to performing the burn far from the planet (22 billion joules). This demonstrates the advantage of conducting burns at high velocities, because the Oberth effect leads to a more efficient use of fuel and a greater energy gain when the spacecraft is moving quickly.

2.4 Deep-Space Maneuver (DSM)

Deep Space Maneuvers (DSMs) are impulsive propulsion maneuvers employed between flybys to fine-tune a spacecraft's direction and velocity [34]. These maneuvers use the spacecraft's propulsion system to make necessary adjustments, ensuring that it remains on its intended trajectory.

2.5 Delta-v Budget

The delta-v budget refers to the total velocity change (in km/s) that a spacecraft needs to achieve its mission objectives. It includes maneuvers such as launches, orbit insertions, transfers and landings. The delta-v budget directly correlates to the fuel consumption of a mission [26].

It should be noted that fuel consumption in space travel follows an exponential trend, as described by the Tsiolkovsky rocket equation in Figure 5 [37].

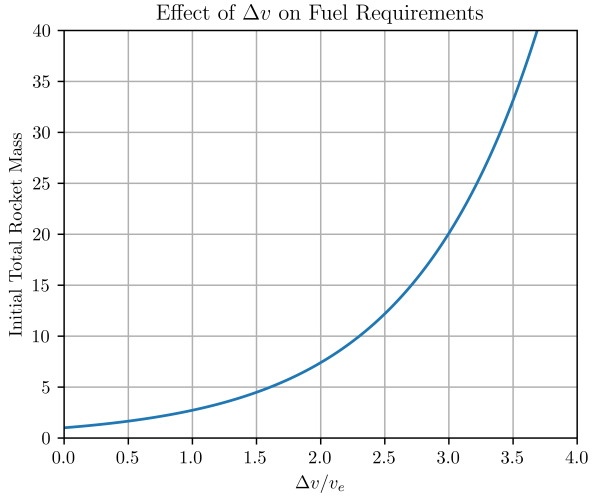


Figure 5: Relationship between mass ratio (initial mass to final mass) and the change in velocity (Δv) normalised by exhaust velocity (v_e), as described by the Tsiolkovsky rocket equation [37]. It states that $\frac{m_0}{m_f} = e^{\frac{\Delta v}{v_e}}$, where m_0 is the initial mass of the spacecraft (including fuel), m_f is the final mass of the spacecraft (excluding fuel), Δv is the total change in velocity and v_e is the effective exhaust velocity, or the speed at which the exhaust leaves the rocket. The steep exponential increase shows that even small increases in Δv require a significantly larger mass ratio. It emphasises the importance of optimising spacecraft trajectories to minimise Δv in order to reduce fuel requirements and improve mission efficiency.

As the spacecraft must carry additional fuel to support subsequent maneuvers, the amount required increases significantly. Therefore, effectively managing the delta-v budget by minimising DSMs and powered flybys is essential to reduce overall fuel needs at launch. This ultimately improves the mission's payload capacity.

2.6 Lambert's Problem

Lambert's problem in celestial mechanics involves determining an orbit based on two known position vectors and the time taken to travel between them [23]. This problem is crucial for mission planning, especially for interplanetary trajectories, where a spacecraft moves from point P_1 to point P_2 while being influenced by a gravitational body C. The objective is to calculate the start and end velocities for a spacecraft travelling between these two points.

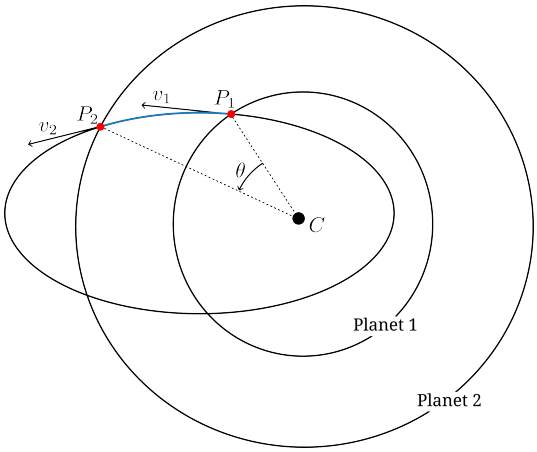


Figure 6: Application of Lambert's problem for a transfer between two planetary orbits [17]. It determines the orbit connecting two positions (P_1 and P_2) around a central body C within a specified time. The figure shows a possible transfer orbit between Planet 1 and Planet 2, with initial and final velocity vectors (v_1 and v_2), and the angle (θ) between the two position vectors. The blue line represents the calculated trajectory of the spacecraft, which satisfies the conditions of traveling from P_1 to P_2 within a specific time frame and under the influence of the central body C. Solving Lambert's problem is key to planning efficient interplanetary trajectories.

Thus, Lambert's problem plays a vital role in trajectory planning, especially when considering the segments between flybys. It helps determine the velocities of a spacecraft as it departs from one planet (P_1) and arrives at another (P_2), while accounting for the gravitational influence of the Sun (C) during the journey between the two planets (see Figure 6).

Solving Lambert's problem involves the application of differential equations related to the two-body problem [23]. Specifically, it establishes the relationship between the time of flight, the distances from the gravitational body and the characteristics of the orbit (like the semi-major axis).

The underlying mathematics to solve this problem are very complicated and therefore out of the scope of this paper [1]. Fortunately, modern Python libraries effectively handle these calculations for us. However, it should be noted that these calculations are very computationally intensive. This imposes a significant constraint when attempting to optimise trajectories.

2.7 Optimization

With all those fundamental concepts explained, we can compute a spacecraft's trajectory when specific parameters such as the initial direction and velocity, the timings for all the flybys and powered maneuvers and the timing or the arrival are provided. This computation is relatively straightforward. However, the difficulty arises when we aim to achieve a predetermined trajectory from point A to point B without knowing all these variables. In this case, we need to determine the optimal parameters, which include:

- Departure date
- Flyby dates
- Arrival date
- Deep Space Maneuvers
- Powered flybys

These parameters p can be collectively represented as a list, for example

$$p = [p_1, p_2, p_3, \ldots] \,. \tag{15}$$

The goal of optimization in this context is to minimise the spacecraft's total change in velocity, or Δv. As mentioned in Section 2.5, Δv is a critical factor in mission design because it directly correlates with fuel consumption and mission feasibility.[4]

To formalise this optimization process, we define a function

$$f(p) = \Delta v \,. \tag{16}$$

The objective is to minimise Δv by adjusting the input parameters. This process is essentially a reversal of the initial problem where we calculated trajectory based on given conditions.

However, optimization presents its complexities. These interconnections make it challenging to achieve an ideal balance, as small adjustments can lead to significant changes in the overall mission plan [21]. This sensitivity to changes can result in complex, sometimes unpredictable scenarios. For instance, a slight modification in the departure date can change the required flyby dates, which then affects arrival timing and necessitates further adjustments to the spacecraft's velocity. This makes the optimization of multiple gravity assist and deep space

[4] In some scenarios, we may also aim to minimise flight time or strike a balance between minimising both Δv and flight duration.

maneuver (MGA-DSM) trajectories a non-trivial task.[5]

Another critical aspect of optimization is the selection of flybys. Different sequences of planetary flybys can yield various outcomes in terms of Δv and overall mission profile. As such, the choice of which planets to use for gravity assists can greatly influence mission efficiency.

Luckily, there are numerous complex algorithms to optimise MGA-DSM trajectories, each operating fundamentally differently [20]. These include:

- Genetic Algorithms (GA)
- Differential Evolution (DE)
- Particle Swarm Optimization (PSO)
- Multi-Start (MS)
- Monotonic Basin Hopping (MBH)

The DE (Differential Evolution) and GA (Genetic Algorithm) algorithms are part of the broader class of Evolutionary Algorithms (EAs). They are optimization techniques inspired by the process of natural selection and biological evolution. These algorithms typically involve populations of candidate solutions that evolve over generations, applying mechanisms such as mutation, crossover and selection. DE focuses on perturbing candidate solutions using the differences between individuals, while GA uses genetic operators like crossover and mutation to explore the search space [20].

On the other hand, PSO (Particle Swarm Optimization) is categorised as an agent-based algorithm, where multiple individual "particles" move through the solution space based on their own experience and that of their neighbors. These particles adjust their positions according to certain rules, inspired by social behaviors such as bird flocking or fish schooling [20].

In contrast, MS (Multistart) and MBH (Monotonic Basin Hopping) algorithms rely on performing multiple local searches. These methods begin by exploring the solution space from different starting points and then use gradient-based methods to refine solutions in the surrounding area of those starting points. The goal is to avoid getting trapped in local minima by leveraging multiple independent searches across the problem space. These algorithms are often used when the problem is highly nonlinear or contains many local optima. Therefore, they are well-suited for complex optimization tasks, such as MGA-DSM trajectory optimization [20].

As we design our own mission, we will return to these algorithms and use them in Section 4.1 and 4.2, in order to navigate the complex landscape of trajectory optimization.

[5]In fact, the problem is so complex that the European Space Agency (ESA) has organised Global Trajectory Optimization Problems (GTOP) competitions, where scholars from different universities compete to find the best possible trajectories [19].

3 Implementation of an Existing Mission

Now that the fundamental concepts have been explained, it is time to apply them to an example of an existing mission. For this purpose, the Cassini-Huygens mission has been selected.

3.1 Historical Background

The Cassini-Huygens mission, launched in 1997, was a joint effort between NASA, ESA and ASI (Italian Space Agency), designed to explore Saturn and its moons [24]. The mission followed a complex interplanetary trajectory, using multiple powered gravity assists (MGA), including flybys of Venus, Earth and Jupiter, to reach Saturn in 2004, as illustrated in Figure 7.

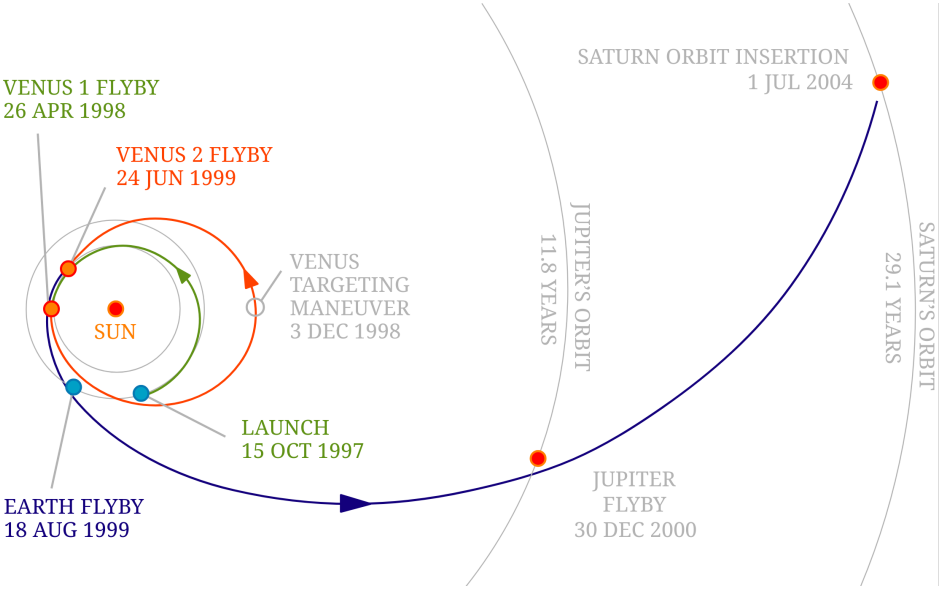


Figure 7: Cassini's interplanetary flight path [24]. It started with its launch from Earth on October 15, 1997, followed by a series of gravity assist flybys: Venus (April 26, 1998 and June 21, 1999), Earth (August 18, 1999) and Jupiter (December 30, 2000). Cassini arrived at Saturn on July 1, 2004. The gravity assist flybys increase the spacecraft's velocity relative to the Sun, enabling it to reach Saturn. The use of the *Earth* → *Venus* → *Venus* → *Earth* → *Jupiter* → *Saturn* trajectory allowed Cassini to complete its journey to Saturn in 6.7 years, without using too much fuel.

Once there, the Cassini orbiter studied Saturn's atmosphere, rings and magnetic field, while the Huygens probe descended to the surface of Titan, Saturn's largest moon.

Cassini-Huygens is an ideal example for this work, because it successfully utilised core principles of space trajectory design. These include gravity assists and DSMs, which were essential for navigating the spacecraft to reach Saturn and its moons.

3.2 Mission-Specific Considerations

3.2.1 Advantages

This mission offers notable advantages, as we can simply reuse the documented sequence of planetary flybys and their dates to recreate the trajectory. This means that we can skip the optimization process, as it has already been effectively handled.

3.2.2 Simplification

The spacecraft used a DSM on December 3, 1998, between its two flybys of Venus. For simplicity, we will disregard this DSM and focus solely on analysing the trajectory in terms of powered flybys.

3.2.3 Python Library `poliastro`

`poliastro` is a python library designed for interactive astrodynamics and orbital mechanics [13]. It provides an accessible way to calculate and plot trajectories. We will especially make use of functions for calculating ephemerides, based on NASA's JPL data. Additionally, it features an integrated solver for the Lambert problem (`poliastro.iod.izzo.lambert`), which immensely simplifies our task [14].

However, a significant drawback is that the library is no longer maintained, requiring an older version of Python (`python3.10`) to function properly. Additionally, `poliastro` only assists with providing planetary positions and solving the Lambert problem. The majority of the work, including assembling the overall trajectory and calculating the total Δv, still requires custom scripts and additional calculations.

3.3 Implementation in Python

We first start by importing all the necessary libraries, as shown in the box below.

```
# Handle astronomical time for calculations
from astropy.time import Time
# Handle astronomical units for calculations
from astropy import units as u
# Represents celestial bodies for trajectory calculations
from poliastro.bodies import Sun, Earth, Saturn, Venus, Jupiter
# Computes planetary positions and velocities (ephemerides)
from poliastro.ephem import Ephem
# Solves Lambert's problem for orbital transfers
from poliastro.iod import izzo
# Creates and manages spacecraft orbits
```

```
from poliastro.twobody import Orbit
# Calculates vector magnitudes, here for velocities
from poliastro.util import norm
```

The next step is to define the launch and arrival dates and flybys at each planet.

```
# Defining the sequence of planetary flybys
# Each planet is represented by its JPL Heliocentric position
sequence = [
    # Start at Earth (at 1997-10-15)
    [Earth, Time("1997-10-15", scale="utc").tdb],
    # First flyby at Venus (at 1998-04-26)
    [Venus, Time("1998-04-26", scale="utc").tdb],
    # Second flyby at Venus (at 1999-6-24)
    [Venus, Time("1999-6-24", scale="utc").tdb],
    # Third flyby at Earth (at 1999-8-18)
    [Earth, Time("1999-8-18", scale="utc").tdb],
    # Fourth flyby at Jupiter (at 2000-12-30)
    [Jupiter, Time("2000-12-30", scale="utc").tdb],
    # Arrival at Saturn (at 2004-07-01)
    [Saturn, Time("2004-07-01", scale="utc").tdb]
]
```

Then, we can calculate the trajectory.

```
# Loop through each pair of consecutive planets in the sequence
for i in range(len(sequence)-1):
    # Get the position (r0) and velocity (v0) of the current
    ↪  planet at the specified time
    r0, v0 = Ephem.from_body(sequence[i][0],
    ↪  sequence[i][1]).rv()

    # Get the position (r1) and velocity (v1) of the next planet
    ↪  at the specified time
    r1, v1 = Ephem.from_body(sequence[i+1][0],
    ↪  sequence[i+1][1]).rv()

    # Solve Lambert's problem between the two planets to find
    ↪  the transfer orbit vectors
    l0, l1 = izzo.lambert(Sun.k, r0[0], r1[0],
    ↪  sequence[i+1][1]-sequence[i][1])

    # Create an orbit object based on the initial position and
    ↪  transfer orbit velocity
    orbit = Orbit.from_vectors(Sun, r0[0], l0, sequence[i][1])
```

Continuing from the previous steps, once we have calculated the trajectory using the Lambert solution, we can visualise the results by plotting the trajectory.

The `poliastro` library provides simple functions for plotting orbits and flybys (`StaticOrbitPlotter.plot_ephem`) [15]. The resulting plot is shown in Figure 8.

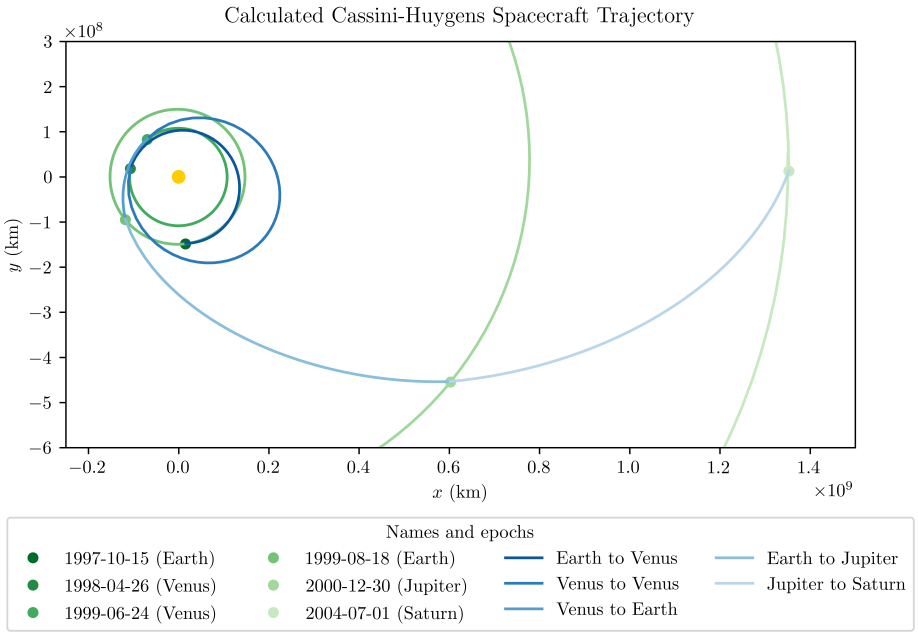


Figure 8: Calculated trajectory of the Cassini-Huygens spacecraft. It shows its gravitational assists and interplanetary journey from Earth to Saturn. The plot depicts key flyby events at Earth, Venus and Jupiter between 1997 and 2004, with colour-coded segments for each phase of the journey. The spacecraft's transfer orbits for each leg were calculated using Lambert's problem to determine the optimal path between planetary flybys.

3.3.1 Calculating Delta-v

The next step in our analysis is to calculate the total Δv. As discussed in Section 2.5, minimising Δv is essential for optimising fuel efficiency, which in turn affects the mission's overall feasibility, cost and payload capacity.

In this context, the Δv of a single flyby is defined as the increase in the spacecraft's velocity relative to the planet responsible for the flyby. The incoming velocity is given by $v_{\text{in}} - v_{\text{planet}}$, while the outgoing velocity is $v_{\text{out}} - v_{\text{planet}}$. Therefore, Δv is the difference in the magnitudes of these two velocities. Mathematically, this can be expressed as

$$\Delta v_{\text{flyby}} = \left|v_{\text{out}} - v_{\text{planet}}\right| - \left|v_{\text{in}} - v_{\text{planet}}\right|. \tag{17}$$

While Δv_{flyby} quantifies the difference in velocity relative to each planet, we may introduce Δv_{total} or simply total Δv, which quantifies the total change in velocity of a mission.

When calculating the total Δv, it is also necessary to account for the initial Δv when the spacecraft needs to accelerate when departing from Earth (Δv_{dep}), and the final Δv, when the spacecraft needs to slow down at the arrival at Saturn (Δv_{arr}).

The total Δv can therefore be expressed as

$$\Delta v_{\text{total}} = \Delta v_{\text{dep}} + \sum \Delta v_{\text{flyby}} + \Delta v_{\text{arr}} \,. \tag{18}$$

Note that while v_{in}, v_{out} and v_{planet} are vectors, Δv_{total}, Δv_{flyby}, Δv_{dep} and Δv_{arr} are numbers.

As mentioned in Section 2.7, our goal is ultimately to minimise the Δv_{total} (which will be addressed later in Section 4). Intuitively, adding flybys might not seem beneficial, as they only increase $\sum \Delta v_{\text{flyby}}$. However, this is not the case. The theory behind it is that while powered flybys do require fuel (and thus increase the Δv_{total}), the Δv_{dep} and Δv_{arr} become smaller. In the end, the increase in Δv_{flyby} is more than offset by the resulting decrease in Δv_{dep} and Δv_{arr}, as powered flybys are highly efficient because they also leverage the planet's velocity. Thus, adding flybys ultimately reduces the Δv_{total}.[6] Figure 9 illustrates well, how omitting flybys increases the Δv_{total}.

Here is my Python code that implements the Δv_{total} calculation based on the principles discussed above.

```
# Variable to store the outgoing velocity from the prev. flyby
last_l1 = None
# Set total delta-v to zero, with units of km/s
total_dv = 0 * u.km/u.s

# Loop through each pair of planets in the flyby sequence
for i in range(len(sequence)-1):
    # Get the position (r0) and velocity (v0) of the current
    ↪  planet at the specified time
    r0, v0 = Ephem.from_body(sequence[i][0],
    ↪  sequence[i][1]).rv()

    # Get the position (r1) and velocity (v1) of the next planet
    ↪  at the specified time
    r1, v1 = Ephem.from_body(sequence[i+1][0],
    ↪  sequence[i+1][1]).rv()

    # Solve Lambert's problem between the two planets to find
    ↪  the transfer orbit vectors
    l0, l1 = izzo.lambert(Sun.k, r0[0], r1[0],
    ↪  sequence[i+1][1]-sequence[i][1])
```

[6] Note that in the case of an unpowered flyby, $\Delta v_{\text{flyby}} = 0$.

```
    # If it is the first leg of the journey, calculate the
    ↪  departure delta-v from Earth
    if not i:
      total_dv += norm(l0 - v0[0]) # Add the delta-v required to
      ↪  leave Earth's orbit
    # For flybys, calculate the delta-v for each maneuver
    if i:
      inc_vel = norm(last_l1 - v0[0]) # Incoming velocity
      ↪  relative to the planet
        out_vel = norm(l0 - v0[0]) # Outgoing velocity relative
        ↪  to the planet
        total_dv += out_vel - inc_vel # Add the delta-v for the
        ↪  flyby maneuver

    # Store the outgoing velocity (l1) for use in the next
    ↪  iteration
    last_l1 = l1

# Add the final delta-v required to slow down when arriving
total_dv += norm(l1 - v1[0])
```

Running this script results in a Δv_{total} of `12.524 km/s`.

3.3.2 Shortcomings

One of the main shortcomings of this approach is the lack of precise orbital insertion calculations, particularly for Saturn and its moon Enceladus. The method simply calculates transfer trajectories between planets but does not account for the critical maneuvers required for Saturn orbit insertion, which are crucial to actually enter Saturn's gravitational sphere and remain in orbit, rather than just fly by.

Additionally, while the method uses gravity assists to calculate flybys, it does not account for important parameters such as the deflection angle (δ) and the direction of the spacecraft after each gravity assist. The direction of the spacecraft's velocity vector after the flyby is crucial, as it determines the subsequent trajectory and future orbital dynamics. In real missions, both of these factors are critical for precisely adjusting the spacecraft's path, as they influence the timing and location of subsequent maneuvers. Ignoring them leads to a simplified, less accurate model that may not reflect the exact conditions necessary for achieving mission objectives.

Moreover, this approach does not include any DSMs. The actual Cassini mission, for example, had a significant DSM between its Venus flybys [24]. Excluding DSMs leads to a simplified trajectory that does not reflect the full complexity of real interplanetary missions.

3.4 Effect of Parameters

Understanding the impact of various parameters on the spacecraft's trajectory is critical for optimising interplanetary missions.

3.4.1 Removing Flybys

Flybys help save fuel by boosting velocity without using propellant. Removing some flybys increases the total Δv, requiring more fuel and potentially extending mission duration, as illustrated in Figure 9. Simulating this effect highlights the importance of each gravity assist in reducing fuel needs.

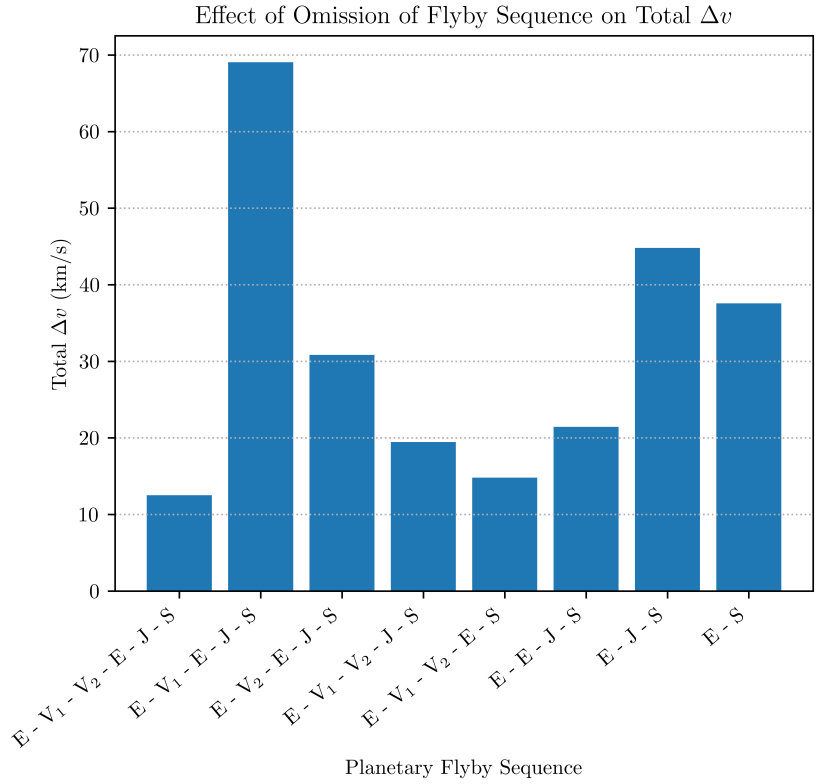


Figure 9: Effect of different planetary flyby sequences on the total change in velocity (Δv) for a spacecraft. The sequence used by NASA for the Cassini-Huygens mission (E-V_1-V_2-E-J-S) results in the lowest total Δv, indicating it was the most efficient trajectory. In contrast, omitting flybys leads to significantly higher total Δv values. This is less desirable, as they require more energy and fuel and make the mission less efficient.

3.4.2 Changing Flyby Dates

Timing is critical for flybys. Even small changes in flyby dates affect the spacecraft's trajectory and increase Δv due to altered planetary positions. Adjusting dates shows how sensitive missions are to timing and how unintuitive the effects

might be. This makes optimization a complex task, as mentioned in Section 2.7. Figure 10 illustrates this by plotting the total Δv with respect to the date of the second flyby at Venus.

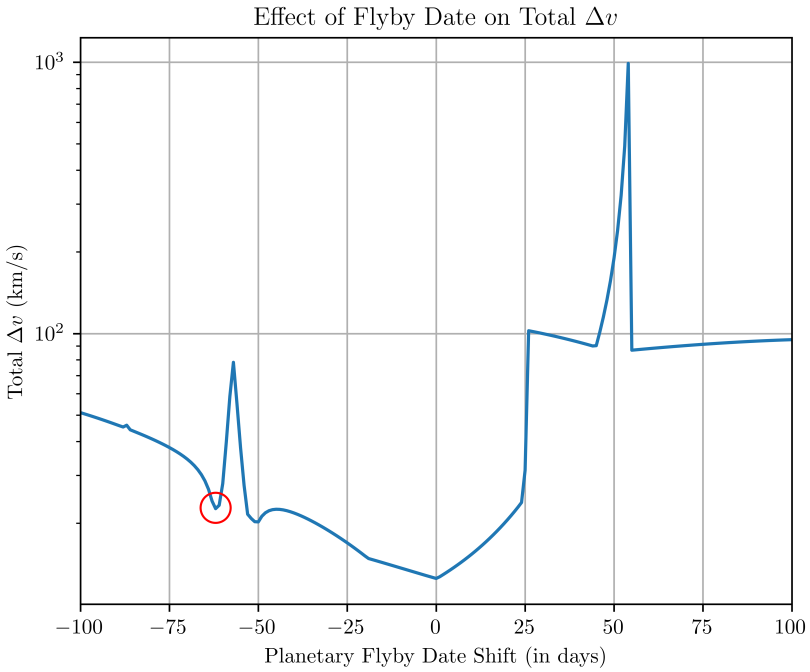


Figure 10: Effect of shifting the second Venus flyby date (from NASA's original date of June 24, 1999) on the total change in velocity (Δv) for the mission. The x-axis represents the shift in days, while the y-axis (logarithmic scale) shows the resulting total Δv. The plot demonstrates that even small changes in the flyby date can lead to unintuitive and highly variable effects on Δv, making optimization complex. The red circle marks a local minimum, which could very well trap an optimization algorithm, as it might mistakenly conclude that the global minimum has been reached. However, the true global minimum lies at $x = 0$. This illustrates the challenges faced in MGA-DSM trajectory optimization, where optimization algorithms must carefully navigate such local minima to find the optimal solution.

4 Designing a New Mission

In this section, we focus on designing and optimising our own interplanetary mission, which uses multiple gravity assists and one DSM per leg of the journey.

4.1 Optimization Complexity

Optimising a mission from scratch is complex and computationally intensive. Unlike the Cassini mission, where the flyby sequence and timing were predetermined, we start with no fixed path, making the choice of planets, flyby dates and DSMs open for exploration. This lack of a starting point highlights some key difficulties:

- **Lack of Predefined Trajectories:** With no established sequence of flybys or maneuvers, we must optimise the entire trajectory from scratch, introducing numerous variables. Both the flyby dates and the selection and order of planets for the flybys are entirely unconstrained.
- **Parameter Interdependence:** As discussed earlier in Section 2.7 and 3.4.2, the interdependencies among key mission parameters significantly complicate the optimization process. For instance, adjusting the departure date can lead to cascading changes in flyby timings, arrival dates and the required velocities at various points. This interconnectedness results in high sensitivity to even minor adjustments, resulting in complex, non-linear and chaotic behaviour [21], as illustrated in Figure 10.
- **Mathematical Complexity:** Optimising an interplanetary mission involves tackling advanced trajectory and optimization problems, which can be mathematically intensive. This area of research is often at the forefront of aerospace studies, as evidenced by competitions like ESA's Global Trajectory Optimization Competition, which highlight the significant challenges involved [19].
- **Computational Requirements:** Identifying the optimal trajectory entails navigating an extensive search space of potential paths. Evaluating countless possibilities demands advanced algorithms and substantial processing resources, making the optimization process both non-trivial and time-consuming.

Given these complexities, we rely on existing software to handle the heavy lifting.

4.2 Choosing the Right Software

While NASA provides advanced numerical software tools for trajectory optimization, they are often costly or unavailable for widespread use:

- **GMAT [9]:** Open-source but lacks the advanced optimization capabilities.
- **Copernicus [6]:** Developed by NASA, but is not publicly accessible.[7]
- **SNOPT7 [16]:** A robust optimization solver, but prohibitively expensive for most users.

On the other hand, the European Space Agency (ESA) is much more open regarding software accessibility. They provide open-source alternatives that are also well documented, such as `pykep` and `pygmo` [12][10].

4.2.1 `pykep` and `pygmo`

For this project, we use `pykep` and `pygmo`, open-source libraries provided by ESA.

- `pykep`: This library specialises in trajectory design, including MGA missions with DSMs. It also includes solvers for Lambert's problem. We rely on the `pykep.trajopt.mga_1dsm` method for missions with a single DSM between flybys [11]. Essentially, this library allows us to define the function $f([p_1, p_2, p_3, \ldots]) = \Delta v$, which needs to be optimised by tweaking the parameters p_i.
- `pygmo`: This package allows us to apply advanced optimization algorithms like the COBYLA solver and a Monotonic Basin Hopping algorithm (MBH) to solve the problem efficiently and find the parameters p_i, which ultimately minimise the total Δv of the mission.

4.2.2 Algorithms Used: COBYLA and MBH

In this project, two optimization algorithms (COBYLA and MBH) are used in combination to efficiently solve the trajectory problem.

- **COBYLA** (Constrained Optimization By Linear Approximations) [3] is a local optimization algorithm that improves solutions step by step within a limited area. It does this by creating a linear approximation of the objective function near the current point, helping to decide where to evaluate next. The algorithm focuses on "trust regions," which means it only explores changes within a certain distance from the current solution. COBYLA is especially useful when evaluating the objective function is costly or when the function is not smooth (non-differentiable). It can also handle constraints, making it versatile for different types of problems.

[7] I even wrote an email to NASA to request access to the Copernicus software. Unfortunately, they responded that the "Copernicus software is only available with a U.S. government contract and purpose."

- **Monotonic Basin Hopping** (MBH) [4] is a stochastic global optimization method. It runs multiple local optimizations from different starting points, allowing the search to "hop" out of local minima and explore more of the solution space to find the global minimum, or the best overall solution.

By combining COBYLA to fine-tune local solutions and MBH to explore globally, the approach helps avoid suboptimal solutions and increases the likelihood of finding the most efficient trajectory. For this project, I wrote custom scripts to adapt these algorithms for MGA-DSM trajectory optimization, configuring them to suit the problem's constraints and integrating them with my trajectory models.

4.3 Implementation in Python

First, we import all the necessary libraries.

```
# Provides tools for trajectory optimization and astrodynamics
import pykep as pk
# Provides algorithms for complex optimization problems
import pygmo as pg
# Handle astronomical time for calculations
from astropy.time import Time
```

Then, we define the planetary sequence for the departure, flybys and arrival.

```
# Defining the sequence of planetary flybys
# Each planet is represented by its JPL Heliocentric position
sequence = [
    pk.planet.jpl_lp('earth'),  # Start at Earth
    pk.planet.jpl_lp('venus'),  # Flyby at Venus
    pk.planet.jpl_lp('mars'),   # Flyby at Mars
    pk.planet.jpl_lp('saturn')  # Arrival at Saturn
]
```

The next step is to define the MGA-DSM trajectory, that needs to be optimised. Essentially, this can be seen as the function $f([p_1, p_2, p_3, \ldots]) = \Delta v$, which needs to be optimised.

```
# Creating a MGA-1DSM trajectory optimization problem
↪  (User-Defined Problem, UDP)
udp = pk.trajopt.mga_1dsm(
    seq=sequence, # The flyby sequence defined earlier
    t0=[ # The launch window: start and end dates
        Time("2025-01-01").mjd-51544.5, # Earliest launch date
        Time("2035-01-01").mjd-51544.5  # Latest launch date
    ],
```

```
    tof_encoding="alpha", # Encoding for time of flight: "alpha"
    ↪  means flexible segment duration
    tof=[3*365, 10*365], # Total time of flight range (in days)
    vinf=[10., 15.], # Range (min and max) for hyperbolic excess
    ↪  velocity (in km/s)
    add_vinf_dep=True, # Use dep. velocity for optimization
    add_vinf_arr=True, # Use arr. velocity for optimization
    orbit_insertion=True,  # Perform orbit insertion at Saturn
    e_target=0, # Target eccentricity for the orbit at arrival
    rp_target=240e6, # Target periapsis radius at Saturn (in km)
    multi_objective=False  # Single-objective optimization
    ↪  (minimising delta-v)
)

# Defining a problem object using the UDP created above
prob = pg.problem(udp)

# Setting a very small constraint tolerance for optimization
# Tight constraint tolerance for precise optimization
prob.c_tol = 1e-8
```

Then, we can use the `pygmo` package to run the optimization algorithm on our problem. This will give us parameters (flyby dates, timings for DSMs,...) which return a minimal total Δv.

```
# Create an instance of the NLopt 'COBYLA' algorithm
↪  (Constrained Optimization BY Linear Approximations)
uda = pg.nlopt("cobyla")

# Set the relative tolerance for the decision variables
# If the change in variables is below this, the algorithm stops
uda.xtol_rel = 1e-8

# Set the relative tolerance for the objective function
uda.ftol_rel = 1e-8

# Set the maximum number of function evaluations allowed to
↪  prevent infinite loops or excessive runtime
uda.maxeval = 100_000

# Monotonic Basin Hopping (MBH) approach to avoid local minima
uda2 = pg.mbh(uda, 5, 0.05)

# Create the new optimization algorithm object using MBH
algo = pg.algorithm(uda2)

# Create a population of 100,000 individuals
# A large population helps improve the chances of finding the
↪  global optimum by covering more of the search space.
```

```
# The decision vectors are randomly initialised
pop = pg.population(prob, 100_000)

# Run the evolutionary optimization algorithm on the population
pop = algo.evolve(pop)
```

Running this script gives us the optimised parameters (`pop.champion_x`) and the resulting Δv_{total} when using those parameters (`pop.champion_y`). In this case, Δv_{total} is `26846.012 m/s` or `26.846 km/s` and the parameters for the function f(parameters) are:

```
First Leg: earth to venus
Departure: 2027-May-17 09:22:38.654340 (9998.390725165971 mjd2000)
Duration: 864.0828467861795days
VINF: 10.833576266715246 km/sec
DSM after 372.12526143385725 days
DSM magnitude: 1279.1130871062674m/s

leg no. 2: venus to mars
Duration: 356.1282801539626days
Fly-by epoch: 2029-Sep-27 11:21:56.616666 (10862.47357195215 mjd2000)
Fly-by radius: 2.286171655195602 planetary radii
DSM after 45.60161564820116 days
DSM magnitude: 272.4243475528733m/s

leg no. 3: mars to saturn
Duration: 1904.793916107756days
Fly-by epoch: 2030-Sep-18 14:26:40.021968 (11218.601852106112 mjd2000)
Fly-by radius: 5.501264486243985 planetary radii
DSM after 245.14135554836133 days
DSM magnitude: 8282.146055463349m/s

Arrival at saturn
Arrival epoch: 2035-Dec-06 09:29:54.373678 (13123.39576821387 mjd2000)
Arrival Vinf: 5956.903720514249m/s
Insertion DV: 6178.752281491581m/s
Total mission time: 8.555797516900475 years (3125.005043047898 days)
```

Using those parameters results in the trajectory shown in Figure 11.

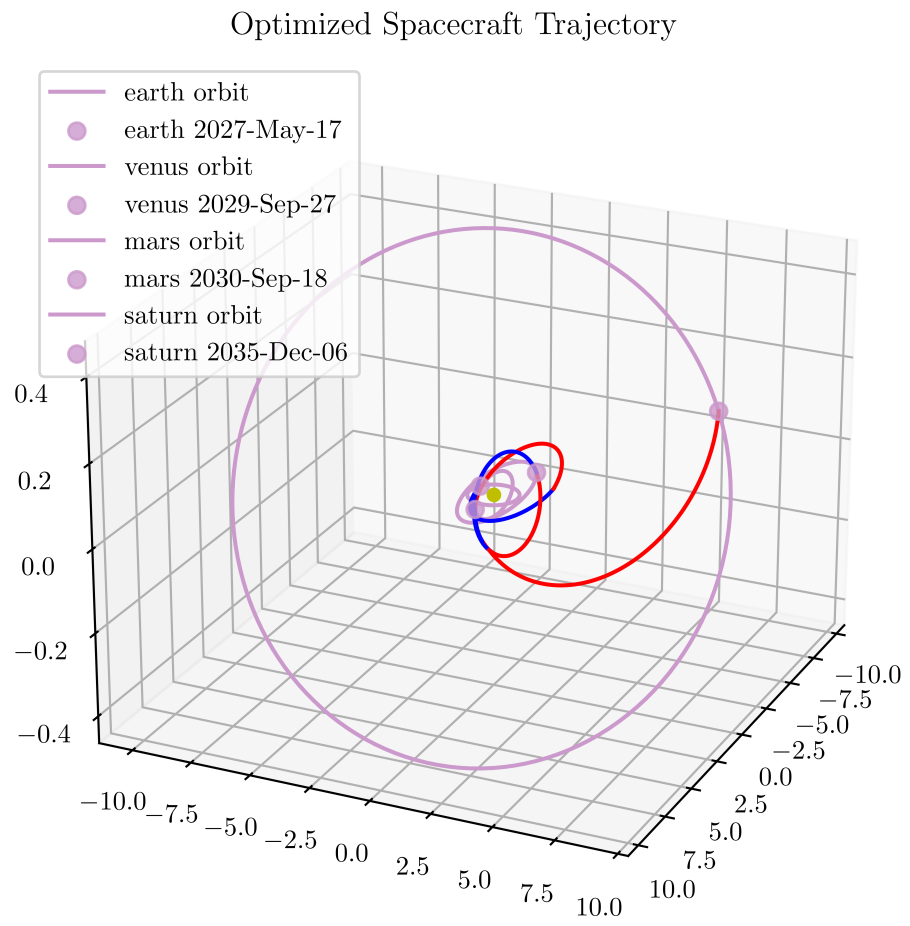


Figure 11: Best optimised spacecraft trajectory for an *Earth* → *Venus* → *Mars* flyby sequence. This 3D plot displays the best optimised trajectory identified for a spacecraft travelling from Earth to Saturn using gravity assists at Venus and Mars. The spacecraft departs Earth on May 17, 2027, performs a Venus flyby on September 27, 2029, followed by a Mars flyby on September 18, 2030, before reaching Saturn on December 6, 2035. One DSM is performed between each planetary flyby. Planetary orbits are shown in purple, while the spacecraft's path is highlighted in red and blue. This trajectory minimises the total Δv required for the mission.

4.3.1 Different Flyby Sequences

Now let us optimise trajectories using different flyby sequences. An *Earth* → *Venus* → *Mars* → *Saturn* sequence might not be the best option. The Cassini-Huygens mission for example used a *Earth* → *Venus* → *Venus* → *Earth* → *Jupiter* → *Saturn* sequence to effectively reach Saturn. Figure 12 illustrates the comparison of various trajectory options, highlighting the potential benefits of more complex flyby paths in optimising mission efficiency.

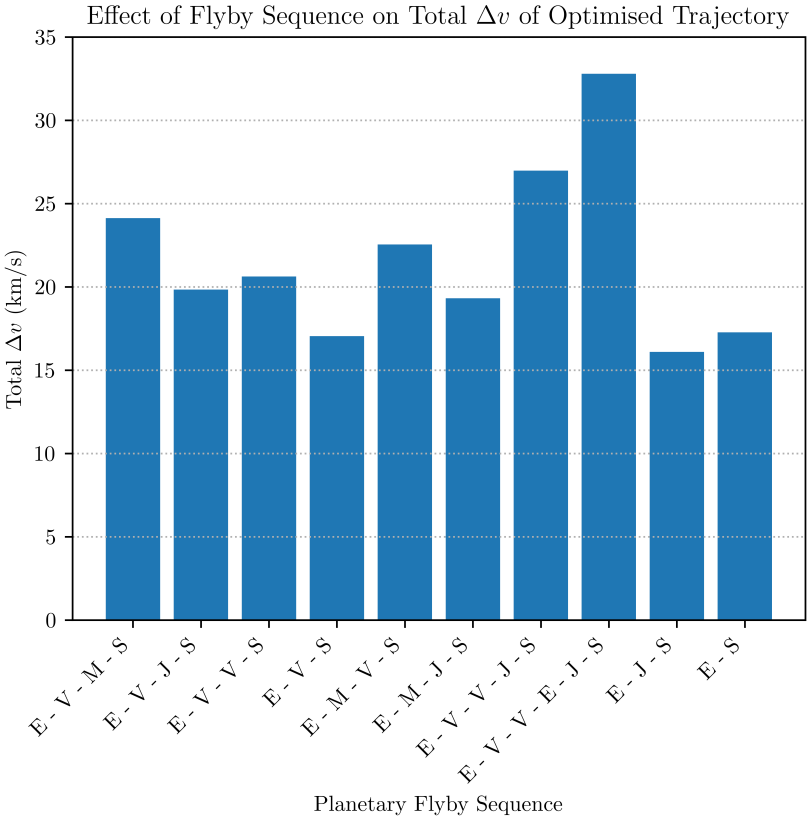


Figure 12: Effect of flyby sequence on total Δv for optimised trajectories. This bar chart illustrates the total Δv required for various planetary flyby sequences in a mission from Earth to Saturn. The sequences include combinations of gravity assists at Venus (V), Mars (M), Jupiter (J) and Earth (E), with the final destination being Saturn (S). They have been manually defined, relying on our intuition of which flyby sequences might make sense. However, this approach may overlook other but potentially more efficient sequences. Among the sequences we have chosen, E-J-S and E-V-S show very low total Δv values, while E-V-V-E-J-S has the highest. Therefore, an Earth-Jupiter-Saturn sequence might be the most fuel-efficient option for optimising the trajectory to Saturn.

4.3.2 Shortcomings

There are some limitations in our approach:

- **No Powered Flybys:** We assume that all flybys are passive, which may not be entirely accurate in real missions using Oberth maneuvers.
- **Deep Space Maneuvers:** While our approach uses DSMs, it simplifies the process by considering only one DSM per leg.
- **Flyby Sequences:** We manually define the flyby sequence, relying on our intuition to select potentially logical sequences. However, this approach may overlook less obvious but more efficient flyby sequences.

Despite these limitations, our methodology provides reasonable estimates for mission planning and calculating total Δv.

4.4 General Limitations of Optimization

Even with advanced tools, optimization has inherent challenges. The search space is vast and local minima can trap optimization algorithms, preventing them from finding the true best solution. Additionally, practical issues such as hardware limitations, algorithm settings and time constraints impact the results.

Furthermore, solving MGA-DSM trajectories requires the use of iterative optimization methods which approximate the optimal solution [2]. Thus, this method has no guarantee of giving the best trajectory and can propose different solutions for the same input settings.

5 Reflection

5.1 Strengths

One of the key strengths of this project was the successful simulation of an existing mission that reached Saturn. By employing gravity assists and DSMs, we replicated the fundamental elements of the Cassini-Huygens mission. We demonstrated an understanding of core concepts, such as gravity assists, the Oberth effect and optimization in interplanetary trajectories. These core concepts form the backbone of space mission planning.

Another strength was the development of our own feasible mission plan from Earth to Saturn. We managed to optimise a trajectory and incorporate key mission elements such as flyby sequences and DSMs, which are crucial for reducing fuel consumption and achieving realistic mission timelines.

Additionally, the use of ESA's optimization tools `pykep` and `pygmo` helped guide the mission planning, enabling us to implement realistic solutions and optimise the MGA-DSM trajectory efficiently.

5.2 Limitations

However, there were several limitations to this project. First, while the mission plan successfully reached Saturn, it did not extend to Enceladus, such as performing an orbit insertion around the moon or landing on it, which was the primary goal. We were only able to simulate an orbit insertion at Saturn, although with an orbit radius similar to that of Enceladus. Therefore, a close encounter with the moon, without requiring too many impulsive maneuvers to alter the spacecraft's trajectory, should be possible. Nonetheless, this could still be considered an incomplete realisation of the project's objective.

Other limitations were the ephemerides used for the optimization (`pk.planet.jpl_lp('<planet>')`). These only provide accurate extrapolations of planetary positions up to 2050. Beyond this point, the calculations become too imprecise. As a result, we were forced to restrict the launch window to a narrow timeframe (2025 - 2035), potentially overlooking better planetary alignments for gravity assists. It is important to note that the success of such trajectories heavily depends on perfect planetary alignments, which occur only infrequently. Thus, a broader launch timeframe would likely have resulted in identifying a more efficient trajectory.

The manual comparison of different flyby sequences was also suboptimal. Due to the complexity of this optimization, it was not possible to automate the testing of all possible sequences efficiently. The manual nature of this comparison relied on our intuition and restricted our ability to explore more diverse and potentially

more efficient sequences.

Additionally, the project heavily relied on a third-party optimization algorithm provided by the ESA. While this allowed us to simulate a feasible mission, the limitation here was our lack of deep understanding of how the optimization algorithm functioned internally. This left us reliant on a "black box" approach, where we input data without fully controlling the mechanics of how the outputs were generated.

On top of that, the settings of the optimization algorithm were chosen somewhat arbitrarily, as I did not have a deep understanding of the underlying mechanics of the algorithm. Parameters such as the *relative tolerance for the decision variables*, the *relative tolerance for the objective function*, the *population size* and the *maximum number of evaluations* were selected based on available defaults or trial and error, rather than a detailed, theoretical understanding of their impact on the optimization process. As a result, the configuration of the algorithm may not have been ideal for the problem at hand, which could have led to suboptimal results. Specifically, the algorithm might have missed other potentially better trajectories due to these imperfect settings.

Lastly, there were inconsistencies in the use of powered flybys and DSMs. We used powered flybys and no DSMs in the implementation of the Cassini-Huygens mission, but used no powered flybys but DSMs in our own mission. Therefore, this inconsistency resulted in an uneven comparison between the two missions.

5.3 Encountered Challenges

One of the greatest challenges was the lack of accessible software. Several NASA-developed tools like Copernicus and GMAT are either expensive or not publicly accessible. Instead, we had to rely on ESA's tools, which, while functional, had their own limitations in terms of user-friendliness and documentation.

Moreover, the topic itself was extremely complex, especially the optimization of MGA-DSM trajectories. Therefore, the fundamental functioning of these optimization algorithms as well as how to solve Lambert's problem could not be explained, as they would have gone beyond the scope of this work.

5.4 Improvements and Additions

There are several ways this project could be improved. One could be the use of different optimization algorithms or even combine multiple ones to further increase the efficiency of the mission. Existing research has attempted this and demonstrated the positive effects it can have on algorithm efficiency [20].

Another improvement would be implementing our own optimization algorithm, allowing for a deeper understanding of the optimization process and pro-

viding full control over trajectory design. This approach would eliminate the reliance on third-party tools and also address the issue of imperfect algorithm settings. With a custom algorithm, we could better understand the impact of these algo settings and, consequently, make more informed and effective choices.

Using different ephemerides that extrapolate planetary positions beyond 2050 would also be beneficial, as it would enable a broader launch window and increase the likelihood of identifying more efficient trajectories.

In addition, we could include other celestial bodies as candidates for flybys. For example, NASA's Galileo mission (1989) used flybys of asteroids 243 Ida and 951 Gaspra during its journey to Jupiter [28]. Introducing asteroids as potential flyby targets could add more possible trajectories and further minimise fuel consumption.

Furthermore, we could calculate the optimal launch location on Earth and the direction for the rocket's launch.[8]

5.5 Final Words

Reflecting on this project, I find it truly remarkable how NASA accomplished similar missions in the 1970s. For example, Pioneer 10 (launched in 1972) became the first spacecraft to use a gravitational slingshot effect to reach escape velocity and leave the Solar System [35]. The computational and technological limitations they faced compared to today's powerful tools and software highlight just how groundbreaking these early missions were. This makes the success of early missions like Pioneer 10 and Voyager 1 and 2 even more impressive.

Interplanetary travel reflects our innate curiosity, a drive to explore the unknown and answer profound questions about existence. As discussed in the introduction, missions to Enceladus, with its potential for life, embody this quest. They challenge us to find out if we are truly alone in the universe and redefine our place within it.

Ultimately, astrophysics is not only about advancing science and technology but also about expanding our collective understanding of the universe. By building on past achievements, we continue to push the boundaries of exploration, seeking knowledge that could reshape how we view ourselves and our place in the cosmos.

[8] Launching near the equator would be advantageous, as it takes advantage of the Earth's rotational speed to provide a centrifugal boost, thereby increasing the rocket's initial velocity. A prime example of this is the *Guiana Space Centre* in French Guiana, which is used by the ESA. This facility's location allows rockets to gain additional momentum from the Earth's rotation, making launches more efficient and reducing fuel requirements [29].

6 References

[1] David de la Torre Sangrà and Elena Fantino. Review of lambert's problem. https://core.ac.uk/download/pdf/41829276.pdf, 2015. [Online; accessed October 18, 2024].

[2] Krafpy. Multiple gravity assist trajectory planner for ksp. https://krafpy.github.io/KSP-MGA-Planner/, 2024. [Online; accessed October 18, 2024].

[3] Entropica Labs Pte Ltd. Gradient-free optimizers. https://openqaoa.entropicalabs.com/optimizers/gradient-free-optimizers/#cobyla, 2022. [Online; accessed October 20, 2024].

[4] Steven L McCarty, Laura M Burke, and Melissa McGuire. Parallel monotonic basin hopping for low thrust trajectory optimization. In 2018 Space Flight Mechanics Meeting, page 1452, 2018.

[5] Jean Meeus. Astronomical Algorithms. Willlmann-Bell, Inc., 2 edition, 1998.

[6] NASA. Copernicus trajectory design and optimization system. https://www.nasa.gov/general/copernicus/, 2024. [Online; accessed October 18, 2024].

[7] NASA Jet Propulsion Laboratory. Jpl planetary and lunar ephemerides. https://ssd.jpl.nasa.gov/planets/eph_export.html, 2020. [Online; accessed October 18, 2024].

[8] NASA Science. Basics of spaceflight: A gravity assist primer. https://science.nasa.gov/learn/basics-of-space-flight/primer/, 2024. [Online; accessed October 17, 2024].

[9] NASA Software Catalog. General mission analysis tool (gmat) v.r2016a(gsc-17177-1). https://software.nasa.gov/software/GSC-17177-1, 2020. [Online; accessed October 18, 2024].

[10] pagmo development team. pygmo — pygmo 2.19.6 documentation. https://esa.github.io/pygmo2/, 2021. [Online; accessed October 18, 2024].

[11] pykep Development Team. The trajopt module — pykep documentation. https://esa.github.io/pykep/documentation/trajopt.html#pykep.trajopt.mga_1dsm, 2020. [Online; accessed October 18, 2024].

[12] pykep Development Team. Welcome to pykep. https://esa.github.io/pykep/index.html, 2020. [Online; accessed October 18, 2024].

[13] Luis Cano Rodríguez and Poliastro Development Team. Poliastro — astrodynamics in python. https://docs.poliastro.space/en/stable/index.html, 2013. [Online; accessed October 18, 2024].

[14] Luis Cano Rodríguez and Poliastro Development Team. poliastro.iod.izzo — izzo's algorithm for lambert's problem. https://docs.poliastro.space/en/stable/autoapi/poliastro/iod/izzo/index.html?highlight=lambert#poliastro.iod.izzo.lambert, 2013. [Online; accessed October 18, 2024].

[15] Luis Cano Rodríguez and Poliastro Development Team. poliastro.plotting.static — plots ephem object over its sampling period. https://docs.poliastro.space/en/stable/autoapi/poliastro/plotting/static/index.html?highlight=plot_ephem#poliastro.plotting.static.StaticOrbitPlotter.plot_ephem, 2013. [Online; accessed October 18, 2024].

[16] Stanford Businss Software Inc. Snopt. http://www.sbsi-sol-optimize.com/asp/sol_product_snopt.htm, 2021. [Online; accessed October 18, 2024].

[17] Peter Swan, C. Swan, michael Fitzgerald, vern hall, james torla, and Matthew Peet. Figure 1.1.5. lambert's problem for a transfer between two planetary orbits. https://www.researchgate.net/figure/5-Lamberts-problem-for-a-transfer-between-two-planetary-orbits_fig2_343657849, 2020. [Online; accessed October 20, 2024].

[18] The European Space Agency. Saturn's moon enceladus top target for esa. https://www.esa.int/Science_Exploration/Space_Science/Saturn_s_moon_Enceladus_top_target_for_ESA, 2024. [Online; accessed October 18, 2024].

[19] The European Space Agency and Advanced Concepts Team. Global trajectory optimisation problems database. https://www.esa.int/gsp/ACT/projects/gtop/, 2019. [Online; accessed October 18, 2024].

[20] Massimiliano Vasile, Edmondo Minisci, and Marco Locatelli. Analysis of some global optimization algorithms for space trajectory design. Journal of Spacecraft and Rockets - J SPACECRAFT ROCKET, 2010.

[21] Matthew Vavrina, Jacob Englander, and Donald Ellison. Global optimization of n-maneuver, high-thrust trajectories using direct multiple shooting. https://ntrs.nasa.gov/api/citations/20160001644/downloads/20160001644.pdf, 2016. [Online; accessed December 8, 2024].

[22] Wikipedia contributors. Kepler orbit — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Kepler_orbit&oldid=1190931001, 2023. [Online; accessed October 18, 2024].

[23] Wikipedia contributors. Lambert's problem — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Lambert%27s_problem&oldid=1186500207, 2023. [Online; accessed October 17, 2024].

[24] Wikipedia contributors. Cassini–huygens — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Cassini%E2%80%93Huygens&oldid=1249537855, 2024. [Online; accessed October 17, 2024].

[25] Wikipedia contributors. Conservation of energy — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Conservation_of_energy&oldid=1261093904, 2024. [Online; accessed December 8, 2024].

[26] Wikipedia contributors. Delta-v budget — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Delta-v_budget&oldid=1198158102, 2024. [Online; accessed October 18, 2024].

[27] Wikipedia contributors. Enceladus — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Enceladus&oldid=1246574234, 2024. [Online; accessed October 17, 2024].

[28] Wikipedia contributors. Gravity assist — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Gravity_assist&oldid=1249768019, 2024. [Online; accessed October 17, 2024].

[29] Wikipedia contributors. Guiana space centre — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Guiana_Space_Centre&oldid=1251842844, 2024. [Online; accessed October 18, 2024].

[30] Wikipedia contributors. Hyperbolic trajectory — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Hyperbolic_trajectory&oldid=1229906981, 2024. [Online; accessed October 17, 2024].

[31] Wikipedia contributors. Kinetic energy — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Kinetic_energy&oldid=1243765531, 2024. [Online; accessed October 18, 2024].

[32] Wikipedia contributors. Newton's law of universal gravitation — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Newton%27s_law_of_universal_gravitation&oldid=1260483051, 2024. [Online; accessed December 8, 2024].

[33] Wikipedia contributors. Oberth effect — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Oberth_effect&oldid=1246559945, 2024. [Online; accessed October 17, 2024].

[34] Wikipedia contributors. Orbital maneuver — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Orbital_maneuver&oldid=1245076274, 2024. [Online; accessed October 17, 2024].

[35] Wikipedia contributors. Pioneer 10 — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Pioneer_10&oldid=1248293070, 2024. [Online; accessed October 18, 2024].

[36] Wikipedia contributors. Special relativity — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Special_relativity&oldid=1259126276, 2024. [Online; accessed December 8, 2024].

[37] Wikipedia contributors. Tsiolkovsky rocket equation — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=Tsiolkovsky_rocket_equation&oldid=1247245962, 2024. [Online; accessed October 18, 2024].