INTERNATIONAL UNIVERSITY LIAISON INDONESIA (IULI)

BACHELOR'S THESIS

A STUDY OF TRAJECTORY DESIGN FROM LEO TO GEO USING IZZO'S LAMBERT-SOLVER

By

Maria Lucia
11201501019

Presented to the Faculty of Engineering
In Partial Fulfilment Of the Requirements for the Degree of

BACHELOR OF ENGINEERING
In
AVIATION ENGINEERING

FACULTY OF ENGINEERING

BSD City 15345
Indonesia
September 2020

APPROVAL PAGE

STUDY OF TRAJECTORY DESIGN FROM LEO TO GEO
USING IZZO'S LAMBERT-SOLVER

MARIA LUCIA
11201501019
Presented to the Faculty of Engineering
In Partial Fulfillment of the Requirements for the Degree of

BACHELOR OF ENGINEERING
In
AVIATION ENGINEERING

FACULTY OF ENGINEERING

Triwanto

Triwanto Simanjuntak, PhD
Thesis Advisor

13/08/2020
Date

Dr. Ir. Prianggada Indra Tanaya, M.M.E.
Dean of Faculty of Engineering

Date

STATEMENT BY THE AUTHOR

Maria Lucia	
Student	Date

ABSTRACT

A Study of Trajectory Design From LEO to GEO
Using Izzo's Lambert-Solver

by

Maria Lucia

Triwanto Simanjuntak, PhD, Advisor

Orbital maneuvering deals with all orbit changes needed to position a satellite from an initial orbit to a selected final orbit. Examples of orbital maneuvering techniques that can be used are Hohmann transfer, Bi-elliptical transfer, and Lambert's problem solver. Unlike the first two techniques, using Lamberts problem for designing orbit transfer, as in this case from LEO to GEO, offers a flexibility for selecting time of flight (tof). Lamberts problem itself basically is a two-point boundary value problem or a determination of an orbit that consists of two positions vector and the time of flight. There are various other methods to solve the problem, and Izzo's solver is one of the available methods and was selected as the main focus of this thesis. In this thesis, the Izzo's solver was implemented numerically using Python language programming and used to study how Δv and tof vary under variation of *initial* orbital elements, such as inclination, right ascension of the ascending node, argument of perigee, and altitude. The results from this thesis can be used as a decision parameter in selecting launching service/site and consideration on how total Δv reduction could save the total launch service that will impact the payload of the satellite.

Keyword: *Orbital maneuvering, Lambert's problem, Izzo's solver, Orbital elements,* Δv

ACKNOWLEDGEMENTS

First and foremost, I would like to thank God Almighty for giving me strength, protection, and spirit, and a chance to finish this dissertation and study. With God in my life, I could believe in myself to keep going and keep fighting to complete my education, even when I decided to give up. He always gives me the strength to get through. Without His blessings, this achievement would not have been possible.

I would like to thank Triwanto Simanjuntak, PhD, for giving me a chance to finish my study and guide me through the end. Without his kindness and guidance, I would never be able to complete my thesis and have motivation during this pandemic. I would also say thank you for your time and patience that you have invested in me despite your busy schedules. I know it is a hard job to supervise a student like me with no background in Astrodynamics. I am also thankful for your understanding of my health problem and realize that my health is my number one priority. After finishing my Bachelor's degree, I hope to follow your footstep to continue my study further.

I wish to express my gratitude to Neno Ruseno S.T., M.Sc., for helping me as the head of the Aviation engineering department. I hope with this dissertation; I could contribute to his department and motivate students to take this path.

With my wholeheartedly, I want to thank my mother, Djunari Tanuwidjaja, for always be by my side since the beginning of my study. She has been my inspiration and gave me strength when I thought of giving up, who continually provide her moral, spiritual, emotional, and financial support. I want to thank my father and my sisters for their support and always believe in me to finish my studies.

Last but not least, I would also thank my IULI's lecturers, Mr. Ananta Widjaja and Dr. Ressa Octavianty, for their effort in teaching me during my study through the years and giving me their support. I wish all of them the best in the future.

Contents

List of Figures

List of Tables

To my Father who is in Heaven, ***Jesus Christ***

To my Parents, Djunari Tanuwidjaja and Teddy Suryanto
To my sisters, Sesilia Noviana and Theresia Leriana

CHAPTER 1
INTRODUCTION

1.1 Background

In this modern era, a lot of people depend on their gadgets to communicate with each other. This type of communication doesn't matter if you are far away or near the intended person, because nowadays people can easily access the internet. We can also call it wireless communication which not only we can communicate easily using the internet, but it could also be used as a media to entertain people whether it comes from television or online provider. All of this could happen because **satellite communications**.

Talking about the satellite itself especially in the area of communication, have so many advantages for industries that would want to develop a satellite. For example, many people may think that the number of users will affect the cost of developing a satellite, but in reality, it doesn't. And whether the satellite is crossing continents or staying local, it also doesn't affect the cost. This gives the industries the benefits to have cost-effectiveness. Superior performance could also give people the benefit of developing satellite communication. This type of satellite can also be used as a broadcasting application like television. The two-way IP network, speed, and consistency have become a bigger role for businesses, governments, and costumers to use. (*Advantages of Satellites | Telesat*, n.d.)

Another area that could be explored is remote sensing. The utilization of remote sensing might include the ability to collect detail data or information through broad geographical areas, characterizing natural or physical properties on the ground, routinely analyzing surface and observe their changes over time, and lastly the ability to accommodate such data with other information to support decision making. (-*REMOTE SENSING DATA: APPLICATIONS AND BENEFITS*, n.d.)

Satellite utilization may also benefit a country in the defense sight. In a specific case that a country may have is the location of a terrorist for an example. The government or military will need to do surveillance of every move the terrorist made and make sure their track is recognizable and keep the data for capturing or possibly attacking.

In the sense of navigation, tracking the military vehicles is very crucial for the safety of the personal. The military base needs to be always informed of the whereabouts of their people, so they will know if an attack is coming in their direction or any danger is in their path. A satellite could also be used on placing a precise location in the case of bombing. The bomb needs to hit the target because if not it might affect others that live near the target.

Early warning that the satellite could detect will help a country from getting massive damages. Early warning means, the military could be alerted when there is a missile strike, tracking a missile if the path is near the country and would be dangerous, and nuclear explosion monitoring if the situation becomes life-threatening.(Otani, Ohkami, Naohiko KOHTAKE, & Sakurai, 2012)

In the development of satellites in Indonesia, especially on placing a satellite in GEO (Geostationary orbit) or in general, Indonesia is still far behind from other countries, such as the USA, Singapore, India, etc. It is a known fact that for the FSS (Fixed Service Satellite) category, the needs of national telecommunication satellites that are met by national capacity have reached 70%, where 30% of it is fulfilled by foreign satellites. For the MSS (Mobile Service Satellite) category, Indonesia still relies entirely on foreign satellites. However, the need for national telecommunications satellites will increase in the next few years. (*PUSAT KAJIAN KEBIJAKAN PENERBANGAN DAN ANTARIKSA*, n.d.)

Developing a national telecommunication satellite and placing them on GEO will give Indonesia several benefits. Firstly, Geosynchronous orbit has the benefit of keeping the satellite in the same location throughout the day, and the antennas can be pointed towards the satellite and stay on track. Telecommunication satellites will be benefited from this because many application including broadcasting is relay on this type of satellite.

In broadcasting, placing the satellite in GEO is important. Because, in this case, it is not practicable to change the direction of the antenna. It needs to remain in the same position as mentioned in the previous passage. (*Satellite Geostationary Satellite Orbit, GEO ż Electronics Notes*, n.d.)

According to CNN Indonesia in July 2019 (evn, n.d.), currently, the National Institute of Aeronautics and Space Indonesia or most people know as LAPAN is not focus on sending an astronaut to moon, but instead, they are mainly focused on technology that surrounds the earth's orbit. That means from developing a satellite until rocket launching will be LAPAN most priority.

LAPAN's head of the organization, Dr. Thomas Djamaluddin, said that they didn't rule out the possibility for Indonesia to contribute to space exploration missions. There are already several universities interested in developing robotic that would like to be involved in space exploration missions.

Recently, in an event called **Rekornas Inderaja 2020**, Dr. Bambang Soemantri Brojonegoro said that "*Satellite is the answer for the future*". That is why LAPAN need to focus and concentrate on mastering the transfer technology of a satellite. In this context, Indonesia is in the need to have or developing more remote sensing satellite.(*Rakornas Inderaja 2020 : Satelit adalah Jawaban Masa Depan*, n.d.)

On January 28, 2019, LAPAN has launched the new update of their LAPAN:FireHotspot (using remote sensing satellite). This update includes the quality of the information in forest fire monitoring and launching the website version and mobile application that are equipped with the location and point of the hot-spot. LAPAN:FireHotspot is still not nearly enough to cover the need for remote sensing satellite and asked more stakeholders to be involved in developing more remote sensing satellite for the greater future in Indonesia. (antaranews.com, 2020)

Launching a new satellite is an important activity. The satellite system's longevity relies on a sequence of successful launches, and each launch represents a significant financial investment. Launching a satellite is a complex procedure that involves many teams from all over the world or specifically the space industry that will work together through ups and downs until the satellite ready to be launched.

Upon finalization, a newly manufactured satellite will be moved for final testing and fueling to the launch site before being fitted to the launch vehicle. The launch is a complex multi-stage process that varies depending on the system used.

To place a satellite in geosynchronous or geostationary orbit, there are many methods to do so. The most common and simplest way is by using *Hohmann Transfer*. At first, the satellite will be placed on a Low Earth Orbit (LEO) with an altitude approximately 220 km. Once the satellite reaches LEO, the rockets are fired to place the satellite on an elliptical orbit with the LEO as the perigee and GEO as the apogee with an altitude around 35786 km. After this process, a rocket or booster is fired again when it reached the final altitude to maintain the satellite in GEO with the correct velocity. (*Launching Satellites | ESOA*, n.d.)

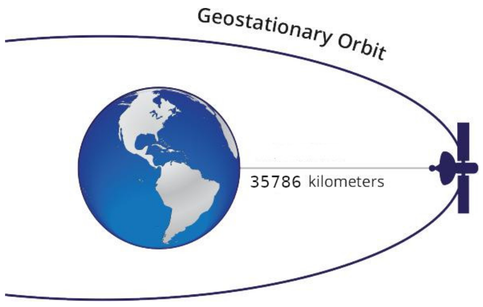


FIGURE 1.1: Geostationary Orbit (*Satellite Orbits | ESOA*, n.d.)

In launching a satellite, it is important to estimate fuel consumption and transfer velocity during the trip. These estimations might as well called fuel budget/propellant budget and transfer budget respectively. According to (*Delta-v budget*, 2019), transfer budget is also known as delta-v budget. This velocity change is needed to determine the estimation of how much propellant for a specific mass and propulsion system is needed per vehicle. Launching a satellite is expensive, especially to GEO. The cost could reach up to 30,000 USD per kg. (*SPACE FREIGHTER | Cannae*, n.d.)

To transfer a heavier satellite from LEO to GEO is needed more fuel than for a lighter one, but at the same time, the delta-v required the same. Delta-v budget can be calculated by using various methods. Among them are Hohmann transfer, bi-elliptical transfer, low energy transfer, etc. Delta-v budget changes with launch time. And course corrections usually required some fuel budget. The propulsion systems cannot necessarily provide the right propulsion in the right direction at all times and there is also confusion in navigation. For correct variations of the optimal trajectory, certain propellants must be reserved.

BRIsat is one of the satellite that placed in GEO. BRI (Bank Rakyat Indonesia) signed an agreement to place a satellite in GEO with Space System/Loral, LLC (SLC) from US and ArianeSpace from France. BRIsat itself is a communication satellite that have 3540 kg of mass that could bring 9 Ku-band transponder and C-band transponder. The total launch cost for BRIsat to reach GEO roughly around 106,200,000 USD. Which is very expensive and it is expected to be that way. But by reducing the total Delta-V, getting a lower launch cost or put more payload to the satellite is very possible. (Page, 2020)

FIGURE 1.2: Launch Vehicle: Ariane-5ECA (Page, 2020)

FIGURE 1.3: BRIsat 1 (Page, 2020)

1.2 Problem Statement

This Thesis mainly will discuss the study to design the orbit transfer from LEO to GEO. Trajectory design is needed in order to solve the problem. Trajectory design itself is the initial data that already given or could find during the study. Using classical orbital element (COE) is the one that used to find the trajectory design in this thesis. The initial LEO itself is depends on the launch service / launch site. Classical orbital elements consists of:

a = semimajor axis	Ω = right ascension of the ascending node
i = inclination	ω = argument of perigee
e = eccentricity	ν = true anomaly

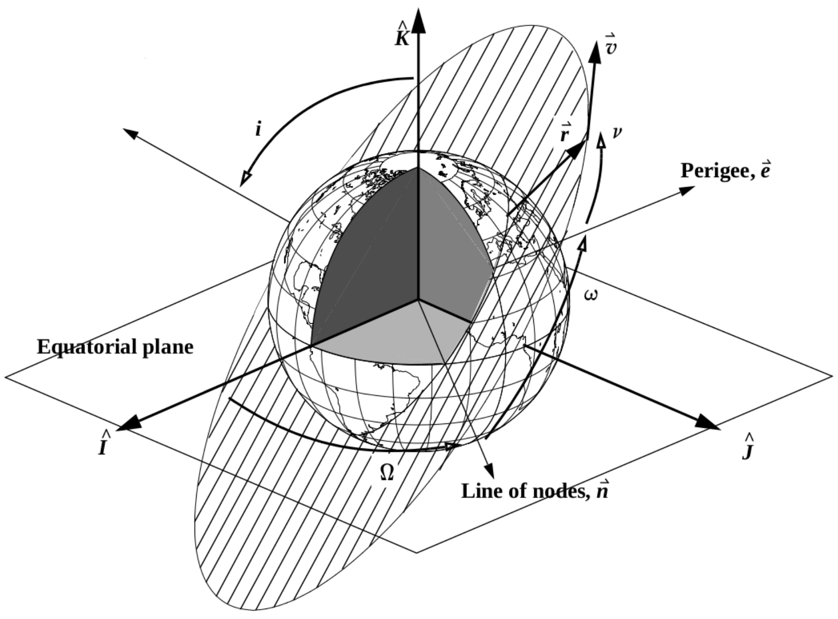


FIGURE 1.4: Classical Orbital Elements (Vallado, 2013)

In this period of time, no more than two impulses is use to do the orbit transfer from LEO to GEO. By doing so, Lambert's problem solver is required. Then, evaluate it on how the variation of initial COE could affect the Total Delta-V.

1.3 Research Goal

The first goal of this thesis is to implementing Lamberts problem solver numerically by Dr. Izzo (Izzo, 2015).

The second is to study how the variations of initial classical orbital elements (COE) affect the result of Total Δv.

1.4 Research Scope

There are two main scopes of this thesis, which are:

- Considering Lamberts problem for a single revolution only;
- Use pure two-body problem dynamics - without any perturbation forces.

It is important to note that this thesis project excludes orbit phasing maneuvers.

1.5 Research Approach

The research approach of this study is to analyze how Total Delta-V varies with time-of-flight (tof) and how it varies for single revolutions in addition to consideration of numerous initial orbital elements. In the end, it is possible to see which one is the best initial orbital elements to produce the maximum Total Δv reduction and how it could save the cost to travel from LEO to GEO. The data on this study will be mostly self-generated by using numerical simulations on the equations of motion that govern the motion of a satellite while orbiting the Earth. To analyze this study, it will be conducted by using computer programming which is Python programming language.

CHAPTER 2
LITERATURE REVIEW

2.1 Introduction to Orbital Maneuvering

Orbital maneuvering, or often could also be called orbit raising, is the use of various techniques that could cover all orbital changes required after a satellite is placed into the intended orbit. There are three techniques to maneuver, which are coplanar, noncoplanar, and fixed Δv. hese techniques are the ones that help us to change numerous orbital elements.

Before going into a deep discussion in orbital maneuvering, several parameters are important in the orbit raising theory. This parameters may also be called the trajectory design that mainly consist of $r_{initial}, r_{final}, v_{initial}$, and v_{final}. In Figure 2.1, it is shown, to do an orbit raising, transfer orbit is needed. It required to have two points, that located near and far the primary body. Each of these points has a velocity change to achieve orbit raising. In astrodynamics, point a is known as periapsis and point b as known as apoapsis.

In coplanar maneuver, the maneuver (initial and final orbit) stays in the same plane. The change happened when talked about the orbit's size (semimajor axis, a) and shape (eccentricity, e), and the apsides location (argument of perigee, ω). Coplanar burns could be tangential or non-tangential. To obtain the maximum efficiency in burning, a zero flight path angle is needed ($\phi_{fpa} \neq 0$), so that tangential burns occur on elliptical orbits only on apoapsis and periapsis. Such burns process allowed to make three coplanar changes, which are:

1. Hohmann transfer (two tangential burns).
2. One-tangent burns (one tangential burn & one non-tangential burn).
3. General transfer (two non-tangential burns).

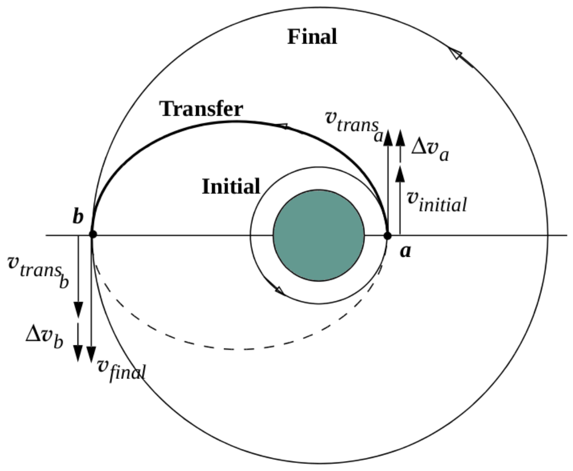


FIGURE 2.1: Orbit Parameters (Vallado, 2013)

The required change needed between inclination, i, and argument of perigee, ω are needed to do an intercept in a noncoplanar transfer. Three possibilities for maneuver in noncoplanar transfer are i only (applying the Δv at a nodal crossing outside the plane), ω only (applying an out-of-plane Δv at a certain point in orbit), or i and ω (applying Δv at any other point). (Vallado, 2013)

Impulse maneuvering is the one that shifts the velocity vector's magnitude and direction instantaneously by rapid firing onboard rocket motors. The change of Δv is happened because of the impulsive maneuver.

Applying a single impulse could make several orbital changes. For example, in a circular orbit, a simple plane change could happen by executing a change of inclination. It is also possible at circular orbit to make an orbit elliptical. During the impulse, the radius and shape will not change, but it will create new velocity. (*Single Impulse Maneuvers in Circular Orbit*, n.d.)

Multiple impulsive or let's just say n-impulse, will not only affect the change of Δv and the time consumed but also the propellants burn will varies with how many impulses will be used during the maneuvering. Take an example from the Hohmann transfer theory. Because it only using two-impulse maneuver for transferring between two coplanar and in this case circular orbits, it produces the most energy-efficient. But the time consumed during this orbital maneuvering is long and it may not meet with the time's goal. (Curtis, 2014)

On providing an optimum propellant burns, low-thrust orbital transfer system may be the solution. With changes in all orbital elements except the true anomaly, it develops control laws that exercised throughout the maneuver (Curtis, 2014; *Simple Control Laws for Low-Thrust Orbit Transfer*, n.d.). Another term of low-thrust is finite burn, where it means "non-zero" or practically over a longer period. The important detail in a finite burn includes mass, center of mass, thruster positions, moment of inertia, fuel consumption, specific impulse, etc. (*Orbital maneuver*, 2019)

2.2 Coplanar Maneuvers

As mentioned in (Vallado, 2013) and the previous passages, coplanar maneuver could have either tangential or non-tangential burn or both. In Figure 2.2, a simple tangential transfer is shown where at the transfer point both orbits are tangent. With this information, the required change in velocity could be easily found by using:

$$\Delta v_a = v_{final} - v_{initial} \tag{2.1}$$

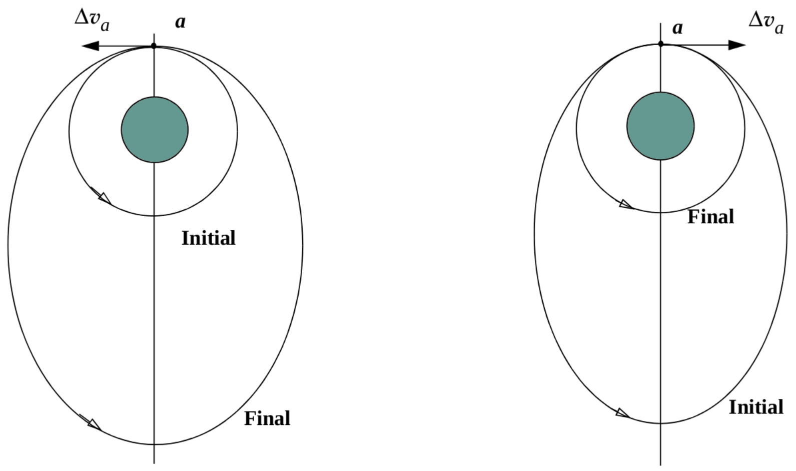


FIGURE 2.2: **Tangential Orbit Transfer** with $\phi_{fpa} = 0°$ (Vallado, 2013)

The nontangential transfer orbit is required ro reach the final orbit when performing a Δv_a maneuver in entering a transfer orbit. It is important to notice in

Figure 2.3, the velocity vector is not parallel anymore, even though this process is still coplanar.

With the introduction of nontangential transfer, it requires a new velocity change formula. In this case, assume knowing the final velocity from the desired final orbit,

$$\begin{gathered}\Delta v_b = \sqrt{v_{trans_b}^2 + v_{final}^2 - 2v_{trans_b}v_{final}\cos(\phi_{trans_b} - \phi_{final})} \\ \Delta\vec{v}_b = \vec{v}_{final} - \vec{v}_{trans_b}\end{gathered} \tag{2.2}$$

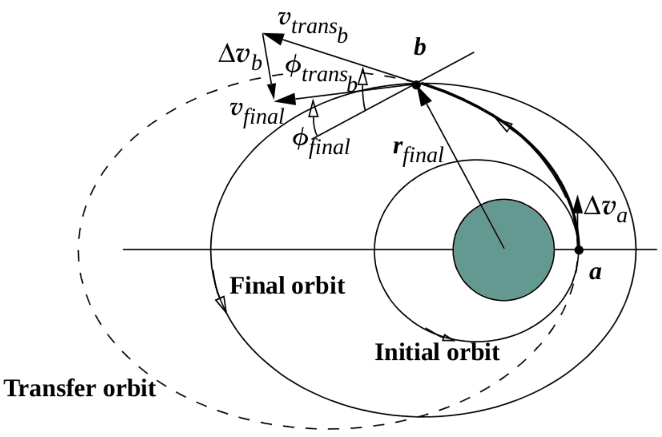


FIGURE 2.3: **Nontangential Orbit Transfer** with $\phi_{fpa} \neq 0$ (Vallado, 2013)

2.2.1 Hohmann Transfer

The Hohmann transfer is the most common and easiest method to be used in terms of orbital maneuvering or orbit raising. It is a two impulse maneuver between two coplanar circular orbits and the most energy-efficient transfer. As illustrated in Figure 2.4, with two circular orbit, the end result of transfer orbit is elliptical (Curtis, 2014). Another important thing to remember is that the flight path angle (ϕ_{fpa}) must be zero.

Originally Hohmann transfer is only considering a transfer between two circular orbit, but there are others that have explored transfer between coaxially even elliptical orbit. Figure 2.5, shows the same understanding as Figure 2.4 but the only difference is the initial and final orbit's shape. Note that elliptical transfer have a negative specific energy ε.

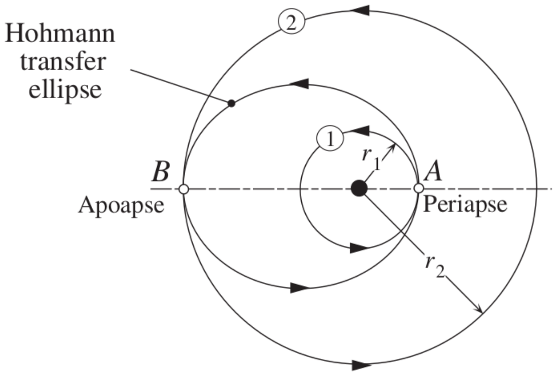


FIGURE 2.4: Hohmann Transfer between Circular Orbits (Curtis, 2014)

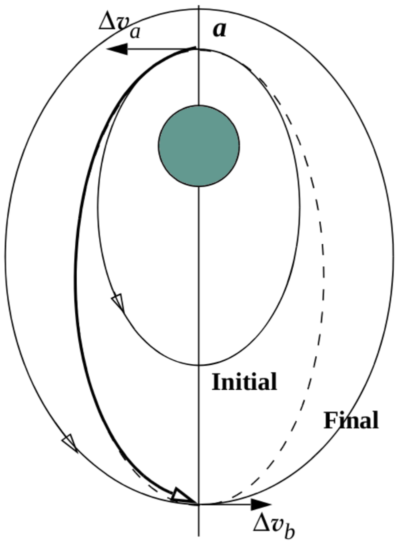


FIGURE 2.5: Hohmann Transfer between Elliptical Orbits (Vallado, 2013)

Semimajor axis a, played an important role in determining the energy of an orbit transfer. To lower the negativity of ε, it requires to reduce its magnitude by increasing the energy. By this definition, it is clear that to have more energy, the semimajor axis need to be larger.

Looking at Figure 2.4, There are point A and B which are periapsis and apoapsis respectively. In order to do an orbital maneuvering, after placing the primary object in the initial orbit, a change of velocity is required. The inital change of velocity at point A or periapsis is also called Δv_A. This allowed the object to do an orbit raising with the help of Hohmann transfer to achieve point B or apoapsis. Then a second velocity change is required to exit the Hohman transfer ellipse then place the object in the final orbit which is symbolized by Δv_B.

The goal of this Hohmann transfer is to see the total delta-v required to do this maneuver. Finding Δv_{total}, there are steps to be fulfilled;
(Notes: μ_{earth} is 398600 km^3/s^2)

1. With knowing the radius of apogee and perigee, finding the semimajor axis is next step to tack.
$$a_{trans} = \frac{r_{initial} + r_{final}}{2} \tag{2.3}$$

2. Find the initial and final velocity.
$$\begin{aligned} v_{initial} &= \sqrt{\frac{\mu}{r_{initial}}} \\ v_{final} &= \sqrt{\frac{\mu}{r_{final}}} \end{aligned} \tag{2.4}$$

3. Then the transfer velocity at point A and B.
$$\begin{aligned} v_{trans_a} &= \sqrt{\frac{2\mu}{r_{initial}} - \frac{\mu}{a_{trans}}} \\ v_{trans_b} &= \sqrt{\frac{2\mu}{r_{final}} - \frac{\mu}{a_{trans}}} \end{aligned} \tag{2.5}$$

4. Calculate the velocity change at periapsis and apoapsis.
$$\begin{aligned} \Delta v_a &= v_{trans_a} - v_{initial} \\ \Delta v_b &= v_{final} - v_{trans_b} \end{aligned} \tag{2.6}$$

5. Lastly find the total Delta-v that required.
$$\Delta v_{total} = |\Delta v_a| + |\Delta v_b| \tag{2.7}$$

Hohmann transfer could also benefit us in finding the transfer time of flight, τ_{trans}, which are

$$\tau_{trans} = \pi\sqrt{\frac{a_{trans}^3}{\mu}} \tag{2.8}$$

2.2.2 Bi-elliptical Hohmann Transfer

The bi-elliptical Transfer implemented two Hohmann transfer in a series. Figure 2.6 proves that this type of transfer is using two Hohmann transfer to achieve the final orbit.

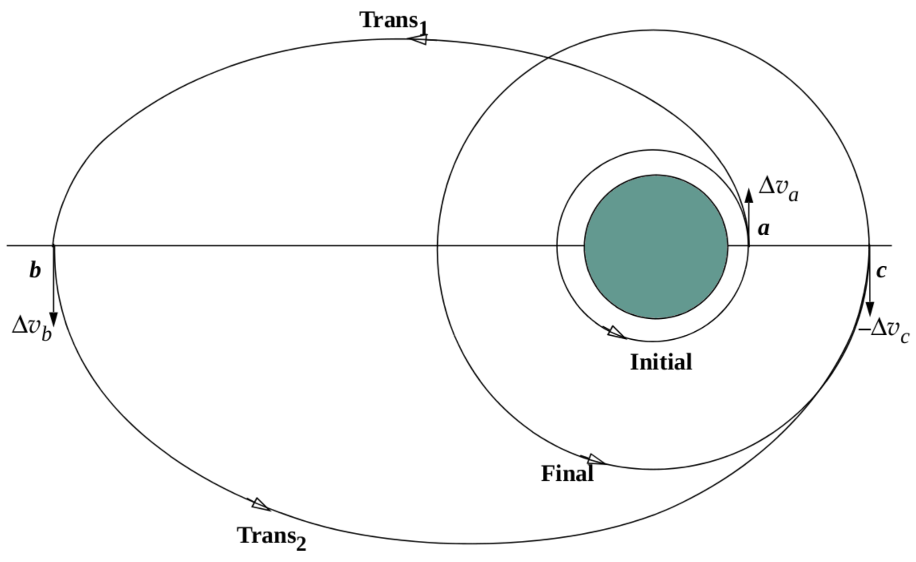


FIGURE 2.6: Bi-elliptical Hohmann Transfer (Vallado, 2013)

The bi-elliptical transfer is more complicated than using a single Hohmann transfer. But it is not as complicated as using other methods like Lambert's Problem that later on will be discussed. This complication rises because it uses three burns to achieve the final orbit. However, the one advantage brought by using bi-elliptical transfer rather than Hohmann transfer is we could save more Δv_{total}.

$$\Delta v_{total(bi-elliptical)} < \Delta v_{total(Hohmann)}$$

This bi-elliptical transfer is worked as the primary object in point a, maneuvered to point b which produces the first transfer ($Trans_1$) and the first Hohmann transfer. Then, the object makes another maneuver to point c and create second transfer ($Trans_2$) or the second Hohmann transfer. And lastly maneuvered it to the final orbit. In the end, these steps are the logical reason that bi-elliptical transfer produces the three burns throughout the orbit raising. Figure 2.6, make it easy to see the velocity change in every maneuver made during the cycle of the bi-elliptical transfer, which are $\Delta v_a, \Delta v_b$, and Δv_c.

Bi-elliptical has the same goals as the Hohmann transfer, which are to find the value of total Delta-v and the time of flight of the transfer with the help of known parameters, like initial radius ($r_{initial}$), transfer radius or intermediate radius at point b (r_b), and final radius (r_{final}). The steps below will explain how to find those goals;

1. Find the two semimajor axis from the two Hohmann transfer.

$$\begin{aligned} a_{trans1} &= \frac{r_{initial} + r_b}{2} \\ a_{trans2} &= \frac{r_b + r_{final}}{2} \end{aligned} \tag{2.9}$$

2. Find all the necessary velocity that needed to be found.

$$v_{initial} = \sqrt{\frac{\mu}{r_{initial}}} \qquad v_{final} = \sqrt{\frac{\mu}{r_{final}}} \tag{2.10}$$

$$v_{trans1_a} = \sqrt{\frac{2\mu}{r_{initial}} - \frac{\mu}{a_{trans1}}} \tag{2.11}$$

$$v_{trans1_b} = \sqrt{\frac{2\mu}{r_b} - \frac{\mu}{a_{trans1}}} \qquad v_{trans2_b} = \sqrt{\frac{2\mu}{r_b} - \frac{\mu}{a_{trans2}}} \tag{2.12}$$

$$v_{trans2_c} = \sqrt{\frac{2\mu}{r_{final}} - \frac{\mu}{a_{trans2}}} \tag{2.13}$$

3. Then calculate all the velocity change during the transfer.

$$\begin{aligned} \Delta v_a &= v_{trans1_a} - v_{initial} \\ \Delta v_b &= v_{trans2_b} - v_{trans1_b} \\ \Delta v_c &= v_{final} - v_{trans2_c} \end{aligned} \tag{2.14}$$

4. Lastly calculate both of the goals withing Δv_{total} and τ_{trans}.

$$\Delta v_{total} = |\Delta v_a| + |\Delta v_b| + |\Delta v_c| \tag{2.15}$$

$$\tau_{trans} = \pi\sqrt{\frac{a^3_{trans1}}{\mu}} + \pi\sqrt{\frac{a^3_{trans2}}{\mu}} \tag{2.16}$$

Comparing Hohmann transfer and bi-elliptical transfer is important to see which of these two is the most useful to use for orbital maneuvering. But it is important to note that each method has its advantage and whether to use Hohmann or bi-elliptical transfer is a choice that has to be made for the individual needs of every organization involved in exploring the orbital maneuvering.

The first thing to do is to find the ratio of the magnitude whether it's from the radius or with the circular orbit semimajor axis.

$$R \equiv r_{final}/r_{initial} = a_{final}/a_{initial} \tag{2.17}$$

For the Hohmann transfer, the formula is

$$\frac{\Delta v_{Hohmann}}{v_{initial}} = \left(1 - \frac{1}{R}\right)\sqrt{\frac{2R}{1+R}} + \sqrt{\frac{1}{R}} - 1 \tag{2.18}$$

Then for the bi-elliptical Hohmann Transfer, there is another parameter that need to be considered. which is $R^* = r_b/r_{initial}$.

$$\begin{aligned}\frac{\Delta v_{bi-elliptic}}{v_{initial}} = &\left|\sqrt{\frac{2R^*}{1+R^*}} - 1\right| + \left|\sqrt{\frac{2}{R^*}}\left(\sqrt{\frac{1}{1+\frac{R^*}{R}}} - \sqrt{\frac{1}{1+R^*}}\right)\right| \\ &+ \left|\sqrt{\frac{1}{R}}\left(\sqrt{\frac{2R^*}{R+R^*}} - 1\right)\right|\end{aligned} \tag{2.19}$$

2.2.3 One-Tangent Burn

Using Hohmann transfer or bi-elliptical transfer requires a long flight time, especially for bi-elliptical because of the three burns method. To overcome this problem, one-tangent burn is the solution. One-tangent burn has both tangential burn and non-tangential burn. From both Figure 2.7 and 2.8, it shows that this method will reduce the transfer time of the previous techniques but the problem is, it will increase the Δv required during the process.

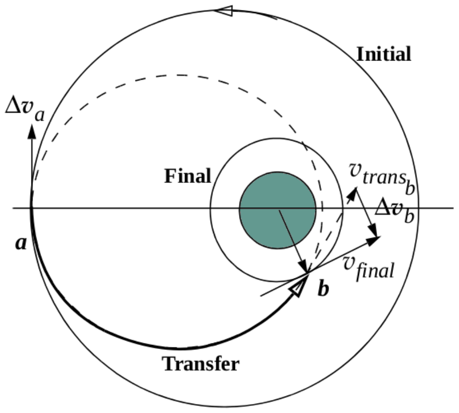


FIGURE 2.7: One-Tangent Burn for Circular Orbit

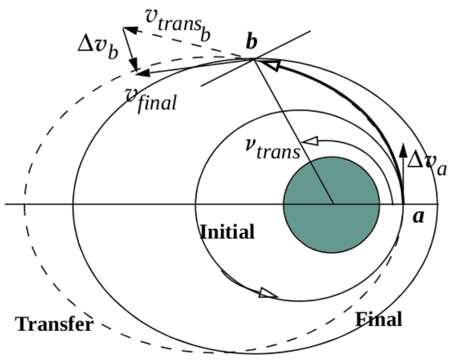


FIGURE 2.8: One-Tangent Burn for Elliptical Orbit

The one-tangent burn uses various transfer orbit's type and shape but mostly uses either the parabolic or hyperbolic types. If we take a look in Figure 2.7, it shows that in the circular orbit, the transfer orbit's type is parabolic and elliptical orbit that shows in Figure 2.8 is using hyperbolic. But it is not a guarantee that all cases will be the same. In reality, it depends on the velocity change capability and time availability.

This method is more complicated to use rather than using Hohmann transfer or bi-elliptical transfer, because of the need to do a non-tangential burn. This type of burn depends on **true anomaly**(semimajor axis or eccentricity), $\boldsymbol{\nu}$, to find the location of the non-tangential burn. Because the values of periapsis and apoapsis is not exist, we could not calculate the transfer semimajor axis using Eq. 2.3.

The solution for using a one-tangent burn is quite difficult but by having the initial and final radius, and the true anomaly of the transfer, we could calculate the total of Delta-v and of course the time of flight. In this moment, implementing one-tangent burn is easier for circular orbit and below is steps to do so;

1. Find the reciprocal of Eq. 2.17.

$$R^{-1} = \frac{r_{initial}}{r_{final}} \tag{2.20}$$

2. Find the eccentricity and semimajor axis of the transfer respectively.

$$e_{trans} = \frac{R^{-1} - 1}{\cos(\nu_{trans_b}) \pm R^{-1}} \qquad a_{trans} = \frac{r_{initial}}{1 \pm e_{trans}} \tag{2.21}$$

$$\begin{matrix} if \nu_{trans_b} < 180°, then \\ if \nu_{trans_b} > 180°, then \end{matrix} \qquad \begin{cases} -periapsis \\ +apoapsis \end{cases}$$

3. The necessary velocity during the process.

$$v_{initial} = \sqrt{\frac{\mu}{r_{initial}}} \qquad v_{final} = \sqrt{\frac{\mu}{r_{final}}} \tag{2.22}$$

$$v_{trans_a} = \sqrt{\frac{2\mu}{r_{initial}} - \frac{\mu}{a_{trans}}} \qquad v_{trans_b} = \sqrt{\frac{2\mu}{r_{final}} - \frac{\mu}{a_{trans}}} \tag{2.23}$$

4. Calculate velocities change from each point, then find the total Delta-v of the sequence.

$$\Delta v_a = v_{trans_a} - v_{initial} \tag{2.24}$$

$$\begin{gathered} \tan(\phi_{trans_b}) = \frac{e_{trans}\sin(\nu_{trans_b})}{1 + e_{trans}\cos(\nu_{trans_b})} \\ \Delta v_b = \sqrt{v_{trans_b}^2 + v_{final}^2 - 2 v_{trans_b} v_{final} \cos(\phi_{trans_b})} \end{gathered} \tag{2.25}$$

$$\Delta v_{total} = |\Delta v_a| + |\Delta v_b| \tag{2.26}$$

5. The last step is to find the time of flight during the transfer.

$$\cos(E) = \frac{e_{trans} + \cos(\nu_{trans_b})}{1 + e_{trans}\cos(\nu_{trans_b})}$$
$$\tau_{trans} = \sqrt{\frac{a^3_{trans}}{\mu}}\{2k\pi + (E - e_{trans}\sin(E)) - (E_o - e_{trans}\sin(E_o))\} \tag{2.27}$$

**Notes: E = Eccentric Anomaly & E_o = zero (starts at periapsis).*

2.3 Non-Coplanar Maneuver

In order to do a non-coplanar maneuver, it is mentioned in (Vallado, 2013) that we might change two elements from the six orbital elements which are inclination, i, and right ascension of the ascending node, Ω. This two elements could be affected when Δv is applied in certain point of the transfer.

The question sometime arise why do we need a non-coplanar maneuver, and there are three answers for that question, which are:

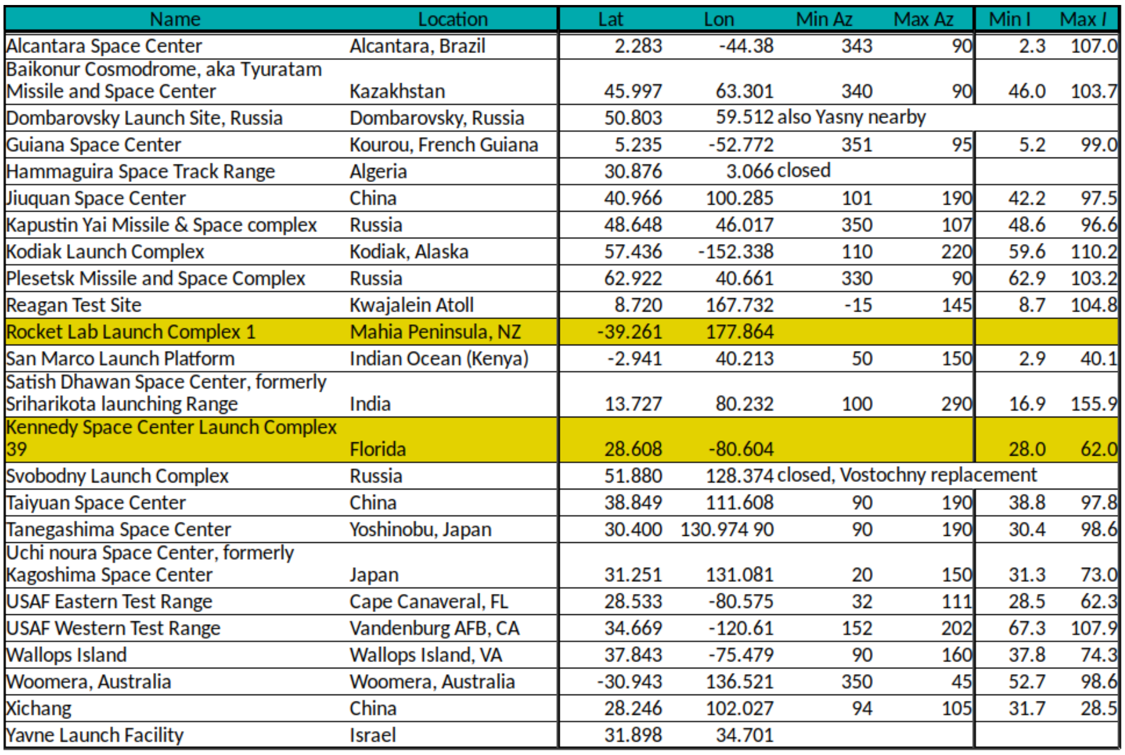
To determine the geometry for launching a satellite

In launching a satellite there is certain a physical limitation, especially for direct launches. That is why the inclination only maneuver is necessary. Direct launch itself is the process of launch from ground and goes directly to the desired orbit. If we don't want to do an inclination maneuver to still have a direct launch, the location site latitude need to be less than or equivalent to the desired inclination. To find the launch window, it involves three steps, which are

- Discovering the launch azimuth, β.
- Finding the auxiliary angle, λ_u.
- Determine the Greenwich Sidereal Time (GMST) or LST.

The expression for the inclination is

$$\cos(i) = \cos(\phi_{gc})\sin(\beta) \tag{2.28}$$



Name	Location	Lat	Lon	Min Az	Max Az	Min I	Max I
Alcantara Space Center	Alcantara, Brazil	2.283	-44.38	343	90	2.3	107.0
Baikonur Cosmodrome, aka Tyuratam Missile and Space Center	Kazakhstan	45.997	63.301	340	90	46.0	103.7
Dombarovsky Launch Site, Russia	Dombarovsky, Russia	50.803	59.512 also Yasny nearby				
Guiana Space Center	Kourou, French Guiana	5.235	-52.772	351	95	5.2	99.0
Hammaguira Space Track Range	Algeria	30.876	3.066 closed				
Jiuquan Space Center	China	40.966	100.285	101	190	42.2	97.5
Kapustin Yai Missile & Space complex	Russia	48.648	46.017	350	107	48.6	96.6
Kodiak Launch Complex	Kodiak, Alaska	57.436	-152.338	110	220	59.6	110.2
Plesetsk Missile and Space Complex	Russia	62.922	40.661	330	90	62.9	103.2
Reagan Test Site	Kwajalein Atoll	8.720	167.732	-15	145	8.7	104.8
Rocket Lab Launch Complex 1	Mahia Peninsula, NZ	-39.261	177.864				
San Marco Launch Platform	Indian Ocean (Kenya)	-2.941	40.213	50	150	2.9	40.1
Satish Dhawan Space Center, formerly Sriharikota launching Range	India	13.727	80.232	100	290	16.9	155.9
Kennedy Space Center Launch Complex 39	Florida	28.608	-80.604			28.0	62.0
Svobodny Launch Complex	Russia	51.880	128.374 closed, Vostochny replacement				
Taiyuan Space Center	China	38.849	111.608	90	190	38.8	97.8
Tanegashima Space Center	Yoshinobu, Japan	30.400	130.974 90	90	190	30.4	98.6
Uchi noura Space Center, formerly Kagoshima Space Center	Japan	31.251	131.081	20	150	31.3	73.0
USAF Eastern Test Range	Cape Canaveral, FL	28.533	-80.575	32	111	28.5	62.3
USAF Western Test Range	Vandenburg AFB, CA	34.669	-120.61	152	202	67.3	107.9
Wallops Island	Wallops Island, VA	37.843	-75.479	90	160	37.8	74.3
Woomera, Australia	Woomera, Australia	-30.943	136.521	350	45	52.7	98.6
Xichang	China	28.246	102.027	94	105	31.7	28.5
Yavne Launch Facility	Israel	31.898	34.701				

TABLE 2.1 Detailed location of several launch services from (Vallado, 2013) with some modification and two new additions.

And the launch azimuth, β,

$$\sin(\beta) = \frac{\cos(i)}{\cos(\phi_{gc})} \tag{2.29}$$

The determination of the direction of launch location

By doing this determination, it could influence how much amount of velocity, the booster should apply. By using angular-velocity relations, we may find the influence of the launch's site velocity.

$$\vec{v_L} = \vec{\omega_\oplus} \times \vec{r_{site}} \qquad or \qquad v_L = |\vec{\omega_\oplus} \times \vec{r_{site}}| = \omega_\oplus r_{site} \cos(\phi_{gc}) \tag{2.30}$$

Finding the best launch window to launch a satellite.

The launch window is the time period during which we can launch a satellite into a desired orbit and still retain the preferred parameters for the mission.

Finding the auxiliary angle, λ_u, is the next step to find the acceptable launch window.

$$\cos(\lambda_u) = \frac{\cos(\beta)}{\sin(i)} \tag{2.31}$$

When is the best time to launch a satellite?
When the angles are in exact position as intended, the Greenwich sidereal time, θ_{GMST} could be found.

$$\lambda_u = \theta_{LST} - \Omega$$

$$\theta_{GMST} = \Omega + \lambda_u - \lambda$$

The time of day (UT) for launch,

$$UT = \frac{\theta_{GMST} - \theta_{GMST_{0h}}}{\omega_{\oplus}} \tag{2.32}$$

Table 2.1 indicate the availability of some of launch service location. To avoid crowded areas, minimum and maximum azimuth values should have been determined. But sometimes launch services do not provide such data. And to determine the orbital inclination, Eq. 2.28 is needed.

2.4 Combined Maneuvers

Talking about coplanar and non-coplanar maneuver is necessary for completing or achieving orbit raising. But in a real-life mission, sometimes it is required to do an unexpected maneuver. While the plan is using The Hohmann transfer, we could not forget about the unpredictable things that would happen in outer space, and sometimes we need the non-coplanar maneuver to avoid any disaster that could occur. Like the changes of inclination only or the right ascension of the ascending node only or even both. Doing this kind of changes is required more time to achieve the final orbit. If the circumstances are possible, it's best to combine both maneuver into one specific maneuver.

Using a combined maneuver will show some benefits that produce during the process of maneuvering. The overall change in velocity requirements will be reduced, and because of two methods of maneuver is combined into one single maneuver, it will show that the number of separate burns is diminished, and the time to complete this operation will be decreased.

It is essential to keep the change in velocity low, and that is the reason why **minimum-inclination maneuvers** are required. Precisely the combination of inclination and altitude. Every velocity encompasses the Hohmann transfer, and the rotation of the velocity vector is the angle that indicates the desire each provider use to change the inclination.
To achieve the minimum combined plane maneuver,

$$\begin{cases} \textit{Iterate}: & \sin(s\Delta i) = \dfrac{\Delta v_a v_{final} v_{trans_b} \sin((1-s)\Delta i)}{\Delta v_b v_{initial} v_{trans_a}} \\ \textit{or} \quad \textit{Estimate}: & R = \dfrac{r_{final}}{r_{initial}} \qquad s \approx \dfrac{1}{\Delta i} \arctan\left[\dfrac{\sin(\Delta i)}{R^{3/2} + \cos(\Delta i)}\right] \end{cases} \tag{2.33}$$

$$\Delta i_{initial} = s\Delta i \qquad\qquad \Delta i_{final} = (1-s)\Delta i \tag{2.34}$$

$$\begin{aligned} \Delta v_{initial} &= \sqrt{v_{initial}^2 + v_{trans_a}^2 - 2 v_{initial} v_{trans_a} \cos(\Delta i_{initial})} \\ \Delta v_{final} &= \sqrt{v_{final}^2 + v_{trans_b}^2 - 2 v_{final} v_{trans_b} \cos(\Delta i_{final})} \end{aligned} \qquad \Delta v = \Delta v_{initial} + \Delta v_{final} \tag{2.35}$$

**note: s = a scaling term (need to be carefull with the units)*

In planning a satellite mission, the **Fixed-Δv** maneuvering is considered. A solid propellant in rockets is the caused in having a stable fixed-Δv capacity, and it couldn't be turn on and off. In this part, another parameter is added, which is a payload angle, γ. The payload angle is the angle between the initial velocity vector and the new velocity vector.

$$\cos(\gamma) = -\frac{v_{initial}^2 + \Delta v^2 - v_{final}^2}{2 v_{initial} \Delta v} \tag{2.36}$$

$$\begin{aligned} \Delta i &\rightarrow \textit{decreasing} \quad -180^\circ \leq \gamma \leq 0^\circ \\ \Delta i &\rightarrow \textit{increasing} \quad 0^\circ < \gamma < 180^\circ \end{aligned}$$

$$\cos(\Delta i_1) = \frac{v_{initial}^2 + v_{trans_a}^2 - \Delta v^2}{2 v_{initial} v_{trans_a}} \tag{2.37}$$

note: if the **fixed-Δv and the inclination, Δi, is not acceptable, another motor or another method is required.*

In consideration of circular orbit, to implement the use of combined maneuvers, two crucial points need to be look over. First, it is essential to know the inclination change for fixed-Δv. Then, input all the velocities changes and the total velocity change for the maneuver.

2.5 Lambert's Problem

Born on August 26th, 1728 (*Johann Heinrich Lambert*, 2020), Johann Heinrich Lambert was the mastermind behind ***Lambert's problem***. In the 18th century, Lambert proposed the idea that an orbit could be determined by only having two position vectors and the time of flight. After Lambert proposed this idea, the problem then solved by Joseph-Louis Lagrange whose born on January 25th, 1736 (*Joseph-Louis Lagrange*, 2020), with mathematical proof.

In the discussion of Lamberts problem, the time of flight is crucial because of its relationship with other variables. It is stated in (*Lambert's problem*, 2019) that:

> *The transfer time of a body passing between two points on a conic trajectory is only a function of the sum of the distances between the two points from the origins of the force, the horizontal distance between the points, and the conics semimajor axis.*

FIGURE 2.9: Johann Heinrich Lambert (*Johann Heinrich Lambert*, 2020)

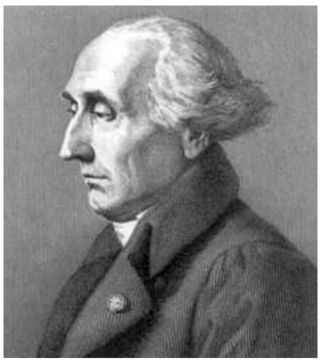


FIGURE 2.10: Joseph-Louis Lagrange (*Joseph-Louis Lagrange*, 2020)

2.5.1 Introduction to Lambert's Problem

Lambert's problem is the problem of the two position vectors and the time of flight between the both of them. Different from Hohmann's transfer, Lambert's problem doesn't have an initial orbit because the orbit itself is not yet fully known.

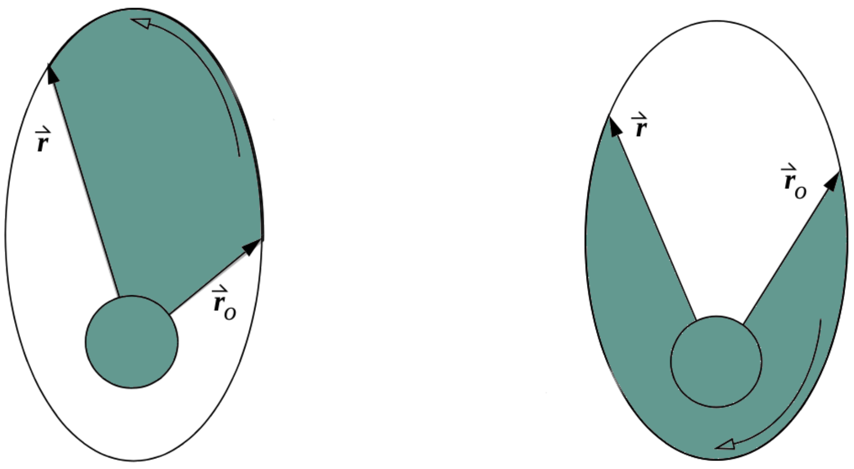
In Hohmann transfer, it focused on its apoapsis and periapsis, but in Lambert's problem, those two aspects are not necessary; in fact, it's unknown. Three variables impact in finding the solution of Lambert's problem, which are:

1. Two Position Vectors.
2. The Time of Flight.

and both of them are fully known from the beginning. But the orbit between the endpoints is unknown.

Lambert's problem have a transfer methods, t_m. Transfer method itself is a process to travel between two points. In Figure 2.11, which explain that there are two way to do a this method. Figure 2.11a shows a short way of transfer method with true anomaly is fall behind 180 degree and the transfer method is equal to +1. On the contrary, Figure 2.11b shows a long way of transfer method with true anomaly exceed 180 degree and the transfer method is equal to -1.

With the help of known two position vectors, this transfer method will help in finding an orbit. The orbital plane will be found when the two vectors are placed.



(A) A short way transfer method with ($\Delta\nu < 180°$ & $t_m = +1$)

(B) A long way transfer method with ($\Delta\nu > 180°$ & $t_m = -1$)

FIGURE 2.11: Transfer Methods in Relation with Lambert's Problem (Vallado, 2013)

To make it all more clearer, once the transfer method is defined, this produces only one specific solution to the problem.

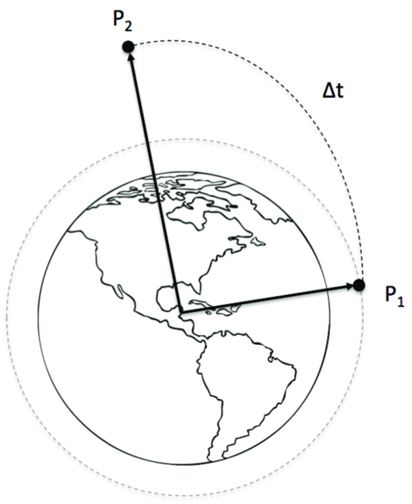


FIGURE 2.12: A Simple Definition of Lambert's Problem (*Figure 1. Schematic diagram of the classical Lambert problem involving...*, n.d.)

In Figure 2.12, it shows the two position vectors (rather than vector $\vec{r_0}$ and $\vec{r}$, we use point P_1 and P_2 respectively to make it easier) and a time of flight. With this information, there are two possibilities in which:

1. The two-position vectors are in the same plane of orbit but displaced over time.

2. The two-position vectors are placed in different planes or orbits in which a transfer orbit is going to took place.

The general idea to do a transfer orbit from point P_1 to P_2 is by doing a maneuver and finding a trajectory. This maneuver and trajectory are done randomly based on the necessity of the problem. There are many ways to reach the final point P_2, like doing a single-impulse maneuver, two-impulse maneuvers, or n-impulse maneuvers. The time of flight is also a factor in this transfer orbit, and how much time will be consumed during this transfer is solely depends on the organization that launched this process. If we want the transfer faster to reach the destination (final orbit), we will use more fuel in the process. But if we want to restrict the fuel consumption during the transfer orbit, then the transfer will take longer to achieve the final orbit.

By doing a transfer orbit, a transfer angle or the angle in the original orbit-determination problem between two position vector need to be determined. And assuming a direction is a must, which:

$$\begin{aligned}\cos(\Delta\nu) &= \frac{\vec{r_0}\cdot\vec{r}}{r_0 r}\\ \sin(\Delta\nu) &= t_m\sqrt{1-\cos^2(\Delta\nu)}\end{aligned} \tag{2.38}$$

According to (Vallado, 2013) and (Izzo, 2015) that besides Gauss's solver on the famous Lambert problem, there are others that try to solve it. It was mentioned multiple times that Gooding's solver is one of the fastest algorithms to solve Lambert's problem and the most accurate at that time.

The development of Lambert's problem is widely developed by various intelligent people. Lancaster and Blanchard (1969), Gooding (1990), Battin (1999), Thorne (2004), Izzo (2014) are among the others that try to develop their Lambert's problem solver.

2.5.2 Classical Algorithm

Lambert's problem is based on its problem geometry, and it contributes a solution in finding the minimum-energy transfer between the two known points from position vectors.

As mentioned in the previous passage, Lambert's problem transfer time depends on the semimajor axis. But besides the semimajor axis, the chord in its geometry is also essential. The chord is the length between two position vectors, c, which is described in Figure 2.13.

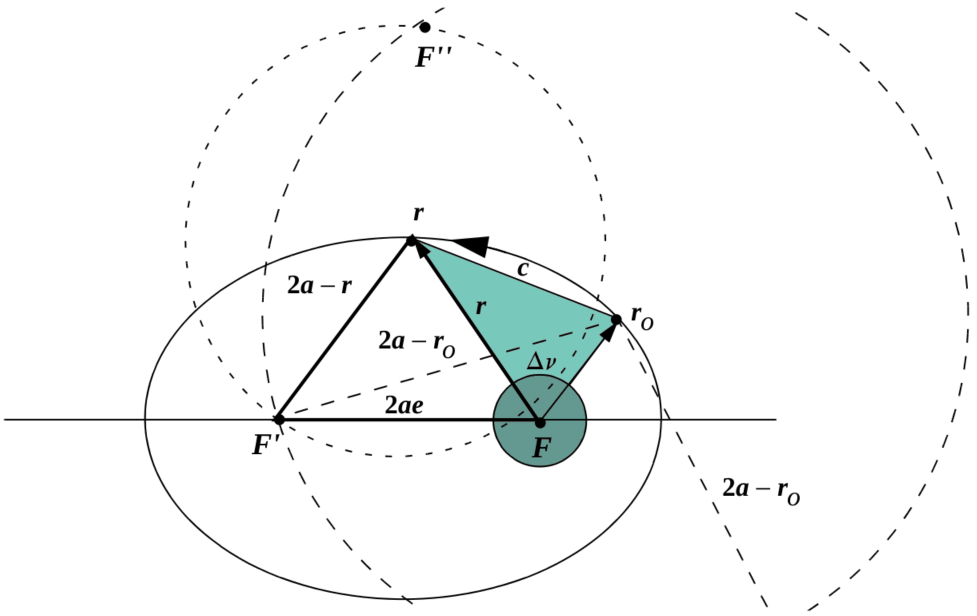


FIGURE 2.13: The Problem's Geometry of the Lambert Problem (Vallado, 2013)

In finding the chord line, it is necessary to use the cosine law, which is;

$$c = \sqrt{r_0^2 + r^2 - 2r_0 r\cos(\Delta\nu)} \tag{2.39}$$

Through this triangle as Figure 2.13 provided, it produces a ***Semiperimeter,*** s.

$$s = \frac{r_0 + r + c}{2} \tag{2.40}$$

In Figure 2.13, we could see there is a symbol F, and it is called the Foci. The distance from foci to any desired point during the process, r_0 or r, is equal twice the semimajor axis. The distance between the two focis is called the ***semiparameter*** which represented by $2ae$.

To conclude this part, Lambert provides the algorithm for minimum energy, which also called the classical algorithm.

$$
\begin{gathered}
\cos(\Delta\nu) = \frac{\vec{r_0} \cdot \vec{r}}{r_0 r} \\
c = \sqrt{r_0^2 + r^2 - 2r_0 r \cos(\Delta\nu)} \\
s = \frac{r_0 + r + c}{2} \\
a_{min} = \frac{s}{2} \quad p_{min} = \frac{r_0 r}{c}(1 - \cos(\Delta\nu)) \\
e_{min} = \sqrt{1 - \frac{2p_{min}}{s}} \\
\\
\alpha_e = \pi \quad \sin(\frac{\beta_e}{2}) = \sqrt{\frac{s-c}{s}} \\
t_{min(a_{min})} = \sqrt{\frac{a_{min}^3}{\mu}}[\alpha_e \mp (\beta_e - \sin(\beta_e))] \\
t_{min(absolute)} = \frac{1}{3}\sqrt{\frac{2}{\mu}}\{s^{3/2} - (s-c)^{3/2}\} \\
\vec{v_0} = \frac{\sqrt{\mu p_{min}}}{r_0 r \sin(\Delta\nu)}\left\{\vec{r} - \left[1 - \frac{r}{p_{min}}\{1 - \cos(\Delta\nu)\}\right]\vec{r_0}\right\}
\end{gathered}
\tag{2.41}
$$

**Notes:*

1. p_{min} is a semiparameter.

2. α_e and β_e is a new constant for elliptical orbits that Prussing and Conways introduced according to (Vallado, 2013).

3. The minus and plus ($\mp$) sign is dependent on the use of short way or long way transfer method.

2.5.3 The Variety of Lambert's Problem Solvers

Gauss's Solution for Lambert's Problem

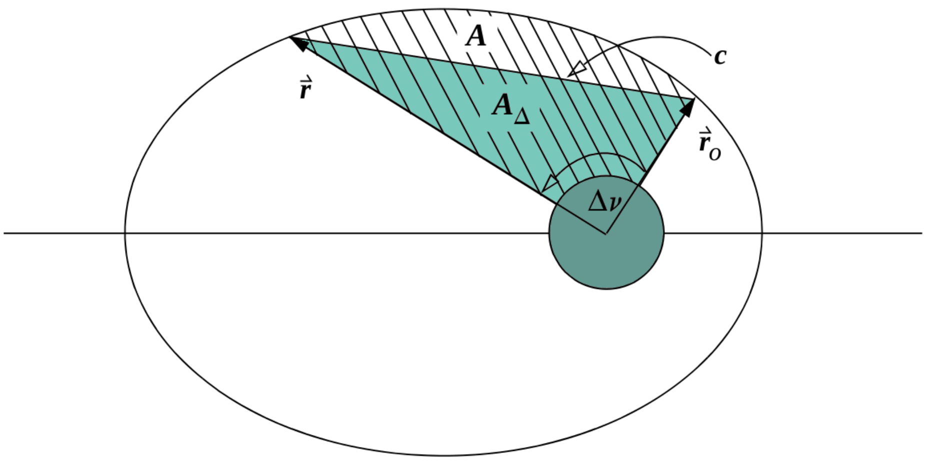


FIGURE 2.14: The Process of Gauss's Solution for Lambert's Problem (Vallado, 2013)

Originally, Gauss's method is the same as the classical algorithm that relies on its problem's geometry and only capable of elliptical transfer. But as the times goes by and idea float in, Gauss proposed an approach that involves the two position vectors that will create an area swept out during the transfer orbit.

Figure 2.14 present us with the general idea how Gauss solve the Lambert's problem. There is two zones that are useful during the process, which are; the shaded triangle zone, A_Δ, and the whole total area swept out during the process by the satellite, A.

Thorne's Solution for Lambert's Problem

In 2004, Thorne successfully developed a series of solutions in the hope of having a better solver to solve Lamberts problem. It is a massive factor in the development of finding the transfer time for either elliptical or hyperbolic. As already explained from the previous passages, the transfer time is equitable to a complex and challenging function of the observed orbits unknown semi-major axis. With Thornes solver needed to improve convergence properties, it is advisable to solve the problem using a series of inversion and reversion. Because Thornes solver could directly encompass the time of flight, this knowledge is how we could differentiate Thornes solution from others. And from this information, we could take the time derivatives from the series equation to minimize the Δv production during the orbit maneuvering.

Universal Variables to Solve Lambert's Problem

Bate, Mueller, and White develop universal variables in 1971. This solution is helpful when we talk about applying to many intercept and rendezvous problems.

There are three variables that dependent with each other in this method, which are: The ***The Universal variable***, χ, relates energy, ξ, and angular momentum, h.

$$\chi = \sqrt{a}\Delta E$$

Bate, Mueller, and White developed an idea of presenting a Newton-Raphson's iterative in finding the universal variables. This bisection will sacrifices the processing time a little to get a much more robust process for a broad diversity of orbits. It is important to note that using this universal variable is difficult because the iteration is not always behaved well.

Battin's Method in Solving Lambert's Problem

Battin developed this method in 1987. The technique he produced is robust and makes it easier to be used in 180° transfer, which mostly found in Lambert cases. His method is a very long derivation, and to many people that are not familiar with this subject will assume this is hard to implement.

Most of Battin's algorithm uses the initial development of Lambert and Gauss formulation. At the same time, Battin introduces a new variable called the ***parabolic mean point radius***, r_{op}. It explains the arithmetic mean between the fundamentals of ellipse's semimajor axis and the line section from the primary target to the point arithmetic mean chord position.

Gooding's Method in Solving Lambert's Problem

In 1990, Gooding developed an algorithm to solve the Lambert problem. His method is purely algebraic and iterates in low computational cost. Gooding's algorithm is considered to be the most accurate and efficient Lambert solver for many people.

2.5.4 Izzo's Lambert's Problem

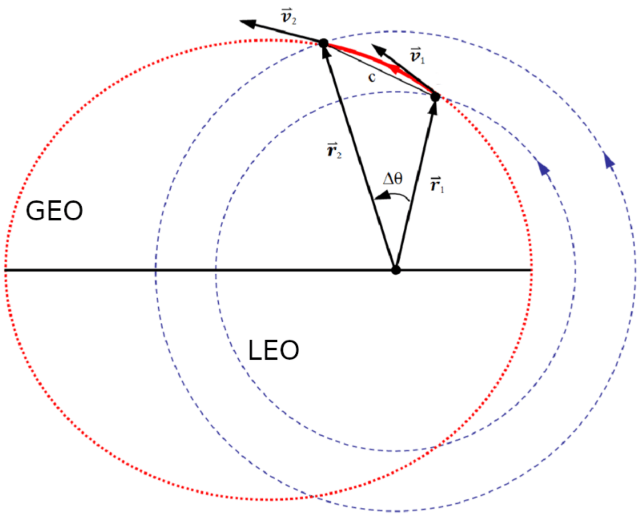


FIGURE 2.15: Geometry of Lambert's Problem (Conte, 2014)

The development of Izzo's method (2014) in solving Lambert's problem is inspired by Lancaster and Blanchard's solution. With this relation, Izzo re-derive some of their equation in hope to explain his idea. Parameter λ is being considered.

$$s\lambda = \sqrt{r_1 r_2}\cos(\frac{\theta}{2})$$

Because of the relation between the line chord and semiperimeter, we could re-write λ as:

$$\lambda^2 = \frac{s-c}{s}$$

with the parameter $\lambda \in [-1, 1]$ becomes positive when $\theta \in [0, \pi]$ and becomes negative while $\theta \in [\pi, 2\pi]$.(Izzo, 2015)

Later, Izzo introduce a new time of flight for a non dimensional, which is:

$$\begin{aligned} T &= \frac{1}{2}\sqrt{\frac{\mu}{a_m^3}}(t_2 - t_1) \\ &= \sqrt{2\frac{\mu}{s^3}}(t_2 - t_1) \end{aligned}$$

Izzo stated that there is one specific advantage by using λ and T which comes from the fact that T itself is a function of a/a_m and the only λ. All of this means, Izzo's goal is to make a simpler equation in solving Lambert's problem.

In Lancaster and Blanchard's equation, sometimes it is not a good step to alter the time of flight with a/a_m, thus Izzo introduce new quantities to avoid the problem.

$$x = \begin{cases} \cos\frac{\alpha}{2} \\ \cosh\frac{\alpha}{2} \end{cases}, \; y = \begin{cases} \cos\frac{\beta}{2} \\ \cosh\frac{\beta}{2} \end{cases} \tag{2.42}$$

which could also be:

$$\begin{cases} \sqrt{1-x^2} = \sin\frac{\alpha}{2} \\ \sqrt{x^2-1} = \sinh\frac{\alpha}{2} \end{cases}, \; \begin{cases} \lambda\sqrt{1-x^2} = \sin\frac{\beta}{2} \\ \lambda\sqrt{x^2-1} = \sinh\frac{\beta}{2} \end{cases} \tag{2.43}$$

and also $y = \sqrt{1 - \lambda^2(1-x^2)}$. From the equation above, the auxiliary angles φ and ψ with relation to x, we could get:

$$\begin{matrix} \cos\varphi = xy - \lambda(1-x^2) \\ \cosh\varphi = xy + \lambda(x^2-1) \end{matrix}, \quad \begin{matrix} \sin\varphi = (y + x\lambda)\sqrt{1-x^2} \\ \sinh\varphi = (y + x\lambda)\sqrt{x^2-1} \end{matrix} \tag{2.44}$$

$$\begin{matrix} \cos\psi = xy + \lambda(1-x^2) \\ \cosh\psi = xy - \lambda(x^2-1) \end{matrix}, \quad \begin{matrix} \sin\psi = (y - x\lambda)\sqrt{1-x^2} \\ \sinh\psi = (y - x\lambda)\sqrt{x^2-1} \end{matrix} \tag{2.45}$$

and from both of this equation, the time of flight equation is produced which is valid for parabolic, hyperbolic, and elliptic cases:

$$T = \frac{1}{1-x^2}\left(\frac{\psi + M\pi}{\sqrt{|1-x^2|}} - x + \lambda y\right) \tag{2.46}$$

Izzo specifically noted that M must equal to zero ($M = 0$) for hyperbolic and parabolic cases.

Izzo's equation time of flight which is Eq. 2.46, will encounter a loss of precision due to numerical cancellation in the single revolution case. It happened because $x \approx 0$ where the both of $1 - x^2$ and ψ will be zero. To overcome this problem, Izzo uses Battin's elegant result setting:

$$\begin{aligned} \eta &= y - \lambda x \\ S_1 &= \frac{1}{2}(1 - \lambda - x\eta) \\ Q &= \frac{4}{3}{}_1F_2(3,\ 1,\ \frac{5}{2},\ S_1) \\ 2T &= \eta^3 Q + 4\lambda\eta \end{aligned} \tag{2.47}$$

${}_1F_2(a,\ b,\ c,\ d)$ is the Gaussian function or it is the ordinary hyper-geometric function which could be found by straightly computing the hyper-geometric series. Then Izzo did a study with the parabolic case by subtituting $x = 1$ and $y = 1$, into Eq. 2.47 and obtain the new time of flight equation for single revolution:

$$T_{(x=1)} = T_1 = \frac{2}{3}(1 - \lambda^3) \tag{2.48}$$

Izzo proposed the idea to derives the formulas of the time of flight which will be available with all cases including single and multiple revolutions, elliptical and hyperbolic.

$$\begin{aligned}(1-x^2)\frac{dT}{dx} &= 3Tx - 2 + 2\lambda^3\frac{x}{y}\\(1-x^2)\frac{d^2T}{dx^2} &= 3T + 5x\frac{dT}{dx} + 2(1-\lambda^2)\frac{\lambda^3}{y^3}\\(1-x^2)\frac{d^3T}{dx^3} &= 7x\frac{d^2T}{dx^2} + 8\frac{dT}{dx} - 6(1-\lambda^2)\lambda^5\frac{x}{y^5}\end{aligned} \tag{2.49}$$

But this formula is not valid when $\lambda^2 = 1,\ x = 0$ and $x = 1$. By applying de I'H$\hat{o}$pital's rule, this could overcome the problem and have a new formula for the parabolic case:

$$\left.\frac{dT}{dx}\right|_{x=1} = \frac{2}{5}(\lambda^5 - 1) \tag{2.50}$$

This time of flight equation brings a great advantage which is operates in the low computational cost of computing T. Because of Izzo's solver emphasizes on the Lancaster-Blanchard variable x using a Hauseholder iteration scheme feeded by a simple initial guess, it makes Izzo's solver simpler to implement.

CHAPTER 3
RESEARCH METHODOLOGY

Research methodology in this study is divided by several parts that consist of understanding the definition of trajectory designs and how to implement them, Hohmann and bi-elliptical Hohmann transfer, both Lambert's problem and Izzo's Lambert problem, and phyton.

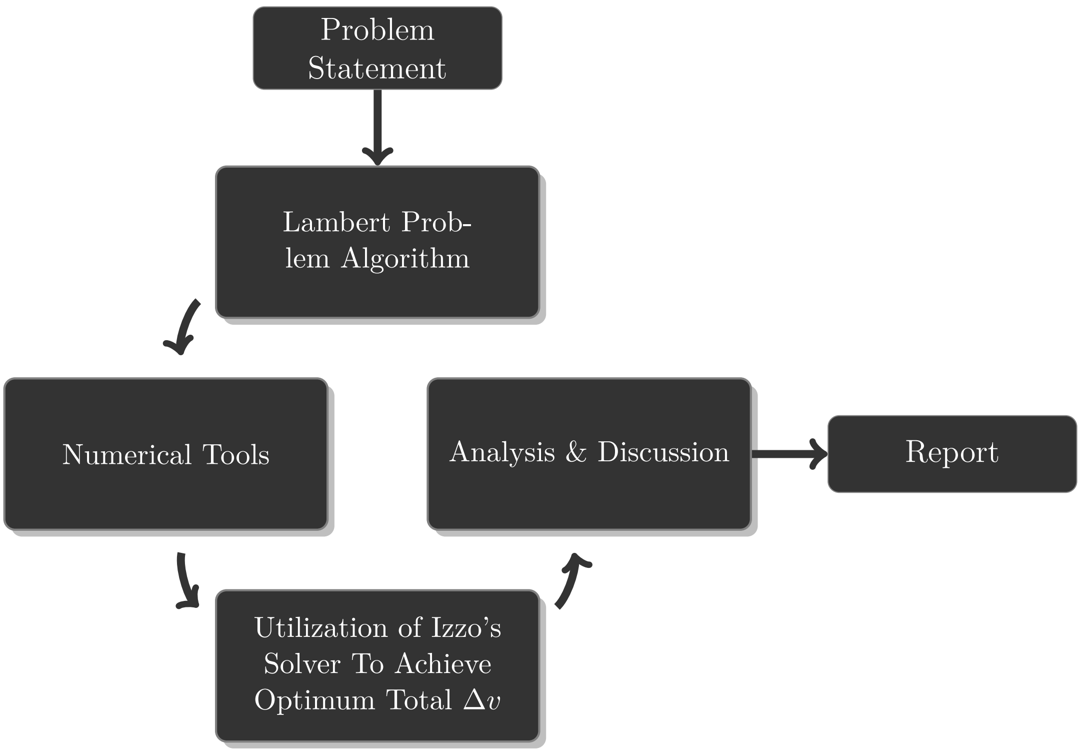

Problem Statement
Lambert Problem Algorithm
Numerical Tools
Analysis & Discussion
Report
Utilization of Izzo's Solver To Achieve Optimum Total Δv

FIGURE 3.1: Steps taken during research study

Before jumping to the main focus of this study, which is Lambert's problem, understanding the fundamental of the theoretical aspect is a must to do. Start from reading the two-body problem (which consist of the equation of relation motion and the theory of eccentricity that related to the shape of orbits), finding the importance of studying the orbital elements and state vector, study how to use python to implement various of the algorithm until reaching Lambert's problem itself. Figure 3.1 shows a chart or diagram which explains the overview steps that are taken during the research study.

3.1 Problem Statement

There are several ways to launch a satellite. Some have to be placed in LEO (Low Earth Orbit), and several launch services needed their satellites to be placed in GEO (Geosynchronous Equatorial Orbit). In general, when a satellite launched into space, usually, it is placed in LEO by the rocket launcher service before it is then maneuvered into GEO. And this statement is the target of this research study.

There are several techniques that can be utilized to design the orbit transfer from LEO to GEO. Hohmann transfer and bi-elliptical Hohmann transfer are some of the examples. But, during this research study, one of the techniques that will be the main focus is by using Lambert's Problem.

The main focus in this research project is finding the amount of Δv both at the initial and the final positions required to achieve the transfer trajectory, which may vary with the variation of initial orbital elements. And in the end, Total Δv could be found.

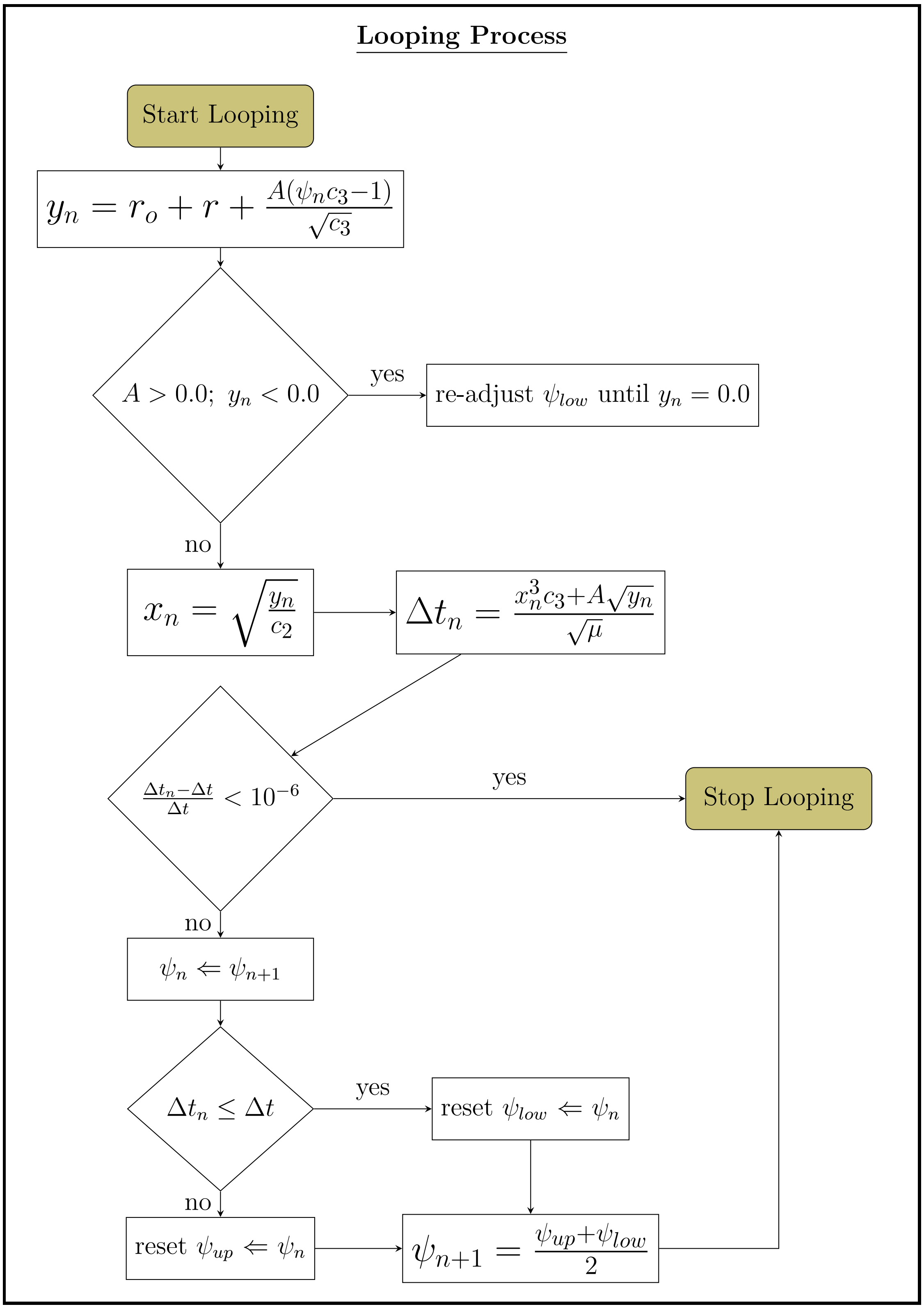
3.2 Lambert Problem Algorithm

In this research study, it has two main focuses on the development of Lambert's problem. There are universal variables and the Izzo's Lambert problem. Each development has a different theoretical approach, but in the end, they have the same goals.

The purpose of this section is to show a summary algorithm of the equation presented by each developer. The first one is the algorithm for universal variables by Bate, Mueller, and White.

Algorithm 1 *Universal Variables* (Vallado, 2013)

```
$\cos(\Delta\nu) = \frac{\vec{r_0}\cdot\vec{r}}{r_0 r}$
$\sin(\Delta v) = t_m\sqrt{1 - cos^2(\Delta v)}$
$\boxed{A = t_m\sqrt{rr_0(1 + \cos(\Delta\nu))}} \xrightarrow{\text{if A = 0.0}}$ Could not compute orbit
If A $\neq$ 0.0, then continue;
$\psi_n = 0.0 \quad ; \quad c_2 = \frac{1}{2} \quad ; \quad c_3 = \frac{1}{6}$
$\psi_{up} = 4\pi^2 \quad ; \quad \psi_{low} = -4\pi$
```


Looping Process
Start Looping
$y_n = r_o + r + \frac{A(\psi_n c_3 - 1)}{\sqrt{c_3}}$
$A > 0.0;\ y_n < 0.0$
yes
re-adjust ψ_{low} until $y_n = 0.0$
no
$x_n = \sqrt{\frac{y_n}{c_2}}$
$\Delta t_n = \frac{x_n^3 c_3 + A\sqrt{y_n}}{\sqrt{\mu}}$
$\frac{\Delta t_n - \Delta t}{\Delta t} < 10^{-6}$
yes
Stop Looping
no
$\psi_n \Leftarrow \psi_{n+1}$
$\Delta t_n \leq \Delta t$
yes
reset $\psi_{low} \Leftarrow \psi_n$
no
reset $\psi_{up} \Leftarrow \psi_n$
$\psi_{n+1} = \frac{\psi_{up} + \psi_{low}}{2}$

It is important to note that we need to configure the value of c_2 and c_3 when ψ_n change to ψ_{n+1}. then:

$$\begin{tikzcd}
\boxed{f = 1 - \frac{y_n}{r_0}} \arrow[dr] & \boxed{\dot{g} = 1 - \frac{y_n}{r}} \arrow[d] & \boxed{g = A\sqrt{\frac{y_n}{\mu}}} \arrow[dl] \\
& \boxed{\vec{v_0} = \frac{\vec{r} - f\vec{r_0}}{g} \qquad \vec{v} = \frac{\dot{g}\vec{r} - \vec{r_0}}{g}} &
\end{tikzcd}$$

The end of the algorithm

The second algorithm, which is the main target of this study research is about Izzo's Lambert Problem.

Algorithm 2 *Izzo's Lambert Problem* (Izzo, 2015)

```
Parameters: $\boldsymbol{r_1} = [r_{11}, r_{12}, r_{13}]$, $\boldsymbol{r_2} = [r_{21}, r_{22}, r_{23}]$, $t$, and $\mu$

Require: $t > 0,\ \mu > 0$
$\boldsymbol{c} = \boldsymbol{r_2} - \boldsymbol{r_1}$
$c = |\boldsymbol{c}|, \quad r_1 = |\boldsymbol{c_1}|, \quad r_2 = |\boldsymbol{c_2}|$
$s = \frac{1}{2}(r_1 + r_2 + c)$
$\hat{\boldsymbol{i}}_{r,1} = \frac{\boldsymbol{r_1}}{r_1}, \ \hat{\boldsymbol{i}}_{r,2} = \frac{\boldsymbol{r_2}}{r_2}$
$\hat{\boldsymbol{i}}_{h,initial} = \hat{\boldsymbol{i}}_{r,1} \times \hat{\boldsymbol{i}}_{r,2}$
$\hat{\boldsymbol{i}}_h = \frac{\hat{\boldsymbol{i}}_{h,initial}}{|\hat{\boldsymbol{i}}_{h,initial}|}$
$\lambda^2 = 1 - c/s, \quad \lambda = \sqrt{\lambda^2}$
IF $\rightarrow$ $\boxed{r_{11}r_{22} - r_{12}r_{21} < 0}$ $\xrightarrow{\text{then}}$ $\boxed{\lambda = -\lambda \quad ; \quad \hat{\boldsymbol{i}}_h = -\hat{\boldsymbol{i}}_h}$
$\hat{\boldsymbol{i}}_{t,1} = \hat{\boldsymbol{i}}_h \times \hat{\boldsymbol{i}}_{r,1} \qquad \hat{\boldsymbol{i}}_{t,2} = \hat{\boldsymbol{i}}_h \times \hat{\boldsymbol{i}}_{r,2}$
$T = \sqrt{\frac{2\mu}{s^3}}t$
$x_{list},\ y_{list} = findxy(\lambda, T)$
```

Cont...

findxy(λ, T): by computing all $x,\ y$ for single revolution solution

Requirement: $|\lambda| < 1,\ T > 0$

$M_{max} = floor(T/\pi)$

$T_{00} = \arccos\lambda + \lambda\sqrt{1-\lambda^2}$

$$\begin{tikzcd}
\textbf{IF} \arrow[r] & \boxed{T < T_{00} + M_{max}\pi \ and \ M_{max} > 0} \arrow[d, "\textbf{then}"'] \\
& \boxed{\text{start Halley iterations from } x = 0,\ T = T_0 \text{ and find } T_{min}(M_{max})}
\end{tikzcd}$$

$$\begin{tikzcd}
\textbf{IF} \arrow[r] & \boxed{T_{min} > T} \arrow[d, "\textbf{then}"'] & \\
& \boxed{M_{max} = M_{max} - 1} \arrow[r] & \textbf{end if}
\end{tikzcd}$$

end if

Single Revolution

$T_0 = \arccos\lambda + \lambda\sqrt{1-\lambda^2} + M\pi$

$T_1 = \dfrac{2}{3}(1-\lambda^3)$

compute x_0, *where* :

$$x_0 = \left(\frac{T_0}{T}\right)^{\frac{2}{3}} - 1, \qquad T \geq T_0$$

$$x_0 = \frac{5}{2}\frac{T_1(T_1 - T)}{T(1-\lambda^5)} + 1, \qquad T < T_1$$

$$x_0 = \left(\frac{T_0}{T}\right)^{log_2\left(\frac{T_1}{T_0}\right)} - 1, \qquad T_1 < T$$

$$\gamma = \sqrt{\frac{\mu s}{2}} \quad , \quad \rho = \frac{r_1 - r_2}{c} \quad , \quad \sigma = \sqrt{(1-\rho^2)}$$

Cont...

```
for each $x,\ y$ in $x_{list},\ y_{list}$
    do
        $V_{r,1} = \gamma[(\lambda y - x) - \rho(\lambda y + x)]/r_1$
        $V_{r,2} = -\gamma[(\lambda y - x) + \rho(\lambda y + x)]/r_2$
        $V_{t,1} = \gamma\sigma(y + \lambda x)/r_1$
        $V_{t,2} = \gamma\sigma(y + \lambda x)/r_2$
        $\mathbf{v}_1 = V_{r,1}\hat{\boldsymbol{i}}_{r,1} + V_{t,1}\hat{\boldsymbol{i}}_{t,1}$
        $\mathbf{v}_2 = V_{r,2}\hat{\boldsymbol{i}}_{r,2} + V_{t,2}\hat{\boldsymbol{i}}_{t,2}$
end for
```

3.3 Numerical Tools

The next step to be taken is to select a numerical tool. There are plenty of tools that could be used to design such algorithms. It is just a matter of preference. During this research study, choosing to use Atom or Spyder IDE (Anaconda Python) is the best option.

Using these tools makes it possible to create a self-generated simulation with the correct algorithm and data. In this research study, I try to test several topics related to Lambert's problem solver and familiarize myself with designing a proper algorithm.

Below are two of the examples of the topics that I've been working on this past research study. These two examples explain the overview look of how to design a proper algorithm.

3.3.1 Hohmann & Bi-elliptical Transfer

```python
import numpy as np
# Earth Properties (JGM 2)
ER = 6378.1363  # Earth's radius (km)
MIU_E = 3.986004415e5  # km^3/s^2 (earth)
```

```python
DU = ER  # Distance Unit in km
TU = np.sqrt(DU**3 / MIU_E)  # Time Unit in second
MIU = 1  # Normlaized GM
def hohmann(ri, rf):
    """
    Performs Hohmann transfer between two circular orbits.
    Keyword Arguments:
    ri -- radius of initial orbit
    rf -- radius of final orbit
    """
    at = (ri + rf) / 2

    vi = np.sqrt(MIU / ri)
    vf = np.sqrt(MIU / rf)
    vt_a = np.sqrt(2 * MIU / ri - MIU / at)
    vt_b = np.sqrt(2 * MIU / rf - MIU / at)

    delta_v_a = vt_a - vi
    delta_v_b = vf - vt_b
    delta_v = (np.abs(delta_v_a) + np.abs(delta_v_b)) * (ER / TU)  # km/s

    tau_t = (np.pi * np.sqrt(at ** 3 / MIU)) * (TU / 3600)  # hours
    return delta_v, tau_t

def bielliptic(ri, rb, rf):
    """
    Perform bielliptic transfer.
    Keyword Arguments:
    ri -- radius of initial orbit
    rf -- radius of final orbit
    rb -- radius of transfer orbit
    """
    at1 = (ri + rb) / 2
    at2 = (rb + rf) / 2

    vi = np.sqrt(MIU / ri)
    vt1_a = np.sqrt(2 * MIU / ri - MIU / at1)
    vt1_b = np.sqrt(2 * MIU / rb - MIU / at1)
    vt2_b = np.sqrt(2 * MIU / rb - MIU / at2)
    vt2_c = np.sqrt(2 * MIU / rf - MIU / at2)
    vf = np.sqrt(MIU / rf)

    delta_v_a = vt1_a - vi
    delta_v_b = vt2_b - vt1_b
    delta_v_c = vf - vt2_c

    delta_v = (np.abs(delta_v_a) + np.abs(delta_v_b) + np.abs(delta_v_c))
            * (ER / TU)  # km/s

    tau_t = (np.pi * np.sqrt((at1 ** 3) / MIU) +
             np.pi * np.sqrt((at2 ** 3) / MIU)) * (TU / 3600)  # hours
    return delta_v, tau_t
```

This algorithm is based on all of the equations in Ch. 2.2.1 and Ch. 2.2.2. Except, in Ch. 2.2.2, there are equations for Ratio comparison, which will be explained after.

The algorithm present above shows that both Hohmann transfer and bi-elliptical Hohmann transfer have the same goals, which are finding Δv and time of flight, τ.

3.3.2 Hohmann Transfer vs Bielliptical Transfer

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rc
rc('text', usetex=True)
plt.style.use('ggplot')
R = np.linspace(2, 75, num=10000)
```

```
RS = [15.58, 40, 60, 100, 200]
def hohmann_ratio(R):
    """
    Calculate the ratio of delta v for Hohmann Transfer
    Keyword Arguments:
    R -- radius ratio, rf / ri
    """
    ratio = (((1 - (1 / R)) * np.sqrt(2 * R / (1 + R))) +
                                (np.sqrt(1 / R)) - 1)
    return ratio

def bielliptical_ratio(RS):
    """
    Calculate the ratio of delta v for Hohmann Transfer
    Keyword Arguments:
    R  -- radius ratio, rf / ri
    Rs -- new radius defination, rb / ri
    """
    x = np.sqrt(2 * RS / (1 + RS)) - 1
    y = np.sqrt(2 / RS) * (np.sqrt(1 / (1 + RS / R)) -
                                np.sqrt(1 / (1 + RS)))
    z = np.sqrt(1 / R) * (np.sqrt(2 * RS / (R + RS)) - 1)

    ratio = np.abs(x) + np.abs(y) + np.abs(z)
    return ratio

fig = plt.figure(figsize=(10, 7))

# special points
xcoords = [11.94, 15.58]
ycoords = [0.6, 0.68]

for (xc, yc) in zip(xcoords, ycoords):
    plt.axvline(x=xc, ls='--', alpha=0.4)
    plt.text(xc - 0.5, yc, str(xc))

plt.plot(R, hohmann_ratio(R), 'b', label='Hohmann')
for ratio in RS:
    plt.plot(R, biellipticaI_ratio(ratio), label='R*=' +
                                                str(ratio))

    plt.xlabel(r"$\frac{r_f}{r_i}$")
plt.ylabel(r"$\frac{\Delta v}{v_i}$")
plt.legend(loc=0)
plt.grid(True)
plt.show()
plt.savefig("HohmanVsBieliptic.pdf", dpi=600)
```

This algorithm shows how Eq. 2.17 interacts with Eq. 2.18 and Eq. 2.19, and creates a comparison between the two transfers. Figure 3.2 shows the result of the algorithm above.

And as I mentioned in the previous chapter (Ch. 2.2.2), getting the result of this comparison is essential to see which transfer is the most useful. But each of these transfers has its advantage, and to choose one of them is solely based on the individual needs of every organization involved in exploring the orbital maneuvering.

3.4 Calculation

In this research study, the main goal is to find Δv for all the possible inputs. Those inputs could be based on the initial or final radius ($\Delta v = f(r)$), the time of flight ($\Delta v = f(\Delta t)$ or $\Delta v = f(\tau)$), or the launch site from various available launcher ($\Delta v = f(launchsite)$).

In 3.3.1, explain the algorithm which shows the design algorithm of Hohmann and bi-elliptical transfer. To solve this algorithm, the neccessary input outside this algorithm are initial and final radius. But in bi-elliptical transfer, the algorithm needs another input which is radius in point b (see Fig. 2.6).

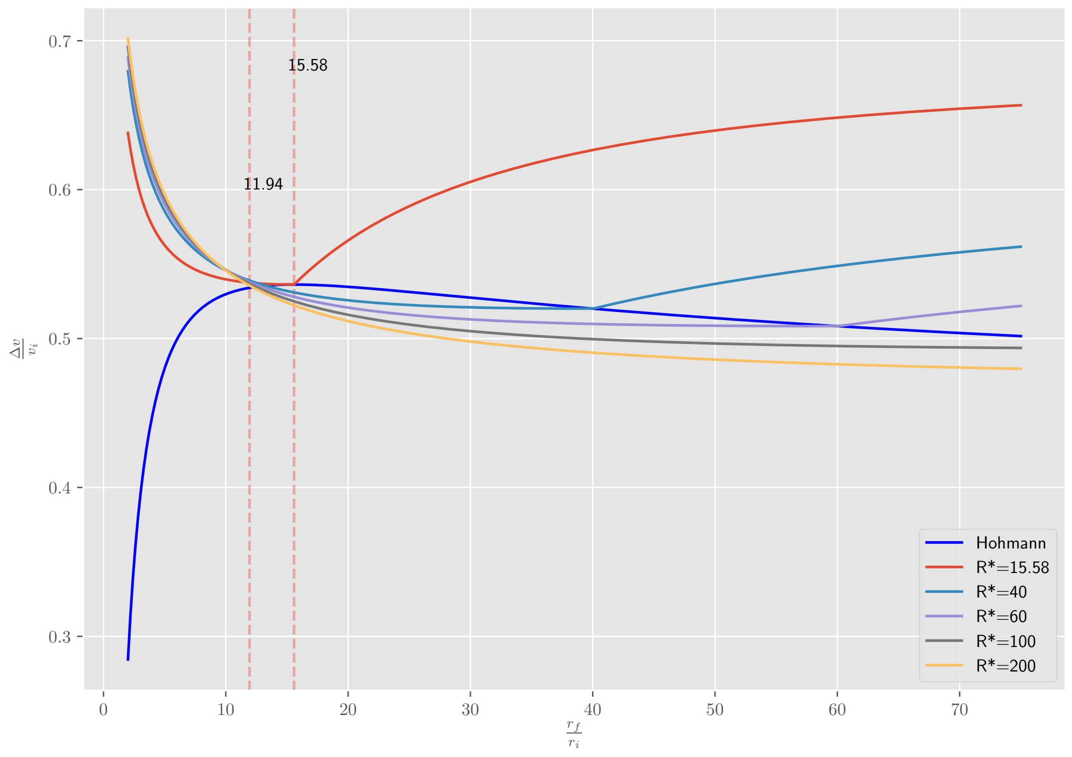


FIGURE 3.2: Comparison between Hohmann Transfer and Bi-elliptical Transfer

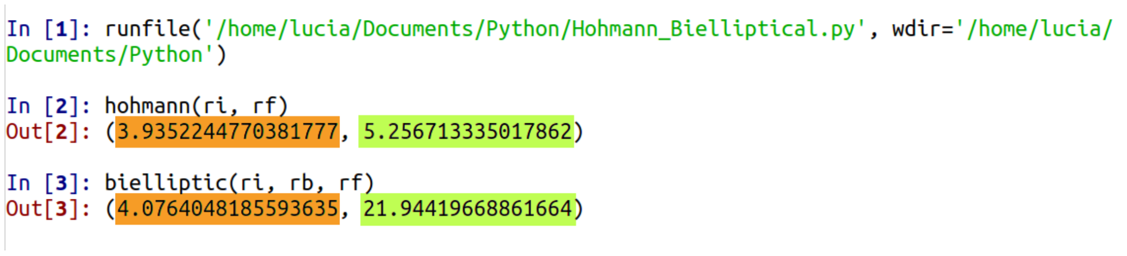
Below is an example of the inputs for algorithm in 3.3.1:

```
#Example
io = 191.34411                    #Initial Orbit altitude (km)
bo = 47836.00                    #Bi-elliptical transfer altitude (km)
fo = 35781.35                    #Final Orbit altitude (km)

ri = (io + ER) / ER         #Initial orbit radius in Canonical unit
rb = (bo + ER) / ER         #Bi-elliptical radius in Canonical unit
rf = (fo + ER) / ER         #Finalorbit radius in Canonical unit
```

Once these parameters entered into the algorithm, the numerical tools (atom or spyder IDE) will run the programming and quickly shows the result for the total change in velocity, Δv, and time of flight, τ.

If the complete algorithm above runs correctly, the result of Hohmann and bi-elliptical transfer will be shown in Fig. 3.3. The ***orange*** highlight indicate the velocity change, Δv, while the ***green*** highlight indicates the time of flight, τ.



```
In [1]: runfile('/home/lucia/Documents/Python/Hohmann_Bielliptical.py', wdir='/home/lucia/
Documents/Python')

In [2]: hohmann(ri, rf)
Out[2]: (3.9352244770381777, 5.256713335017862)

In [3]: bielliptic(ri, rb, rf)
Out[3]: (4.0764048185593635, 21.94419668861664)
```

FIGURE 3.3: Result algorithm of Hohmann Transfer and Bi-elliptical Transfer

*It is important to note that the unit for the velocity change, Δv, is in km/s and for the time of flight, τ, is in *hours*.

3.5 Analysis & Discussion

This section will discuss how the result of the research goes. It could be a success, or it could be a disaster. The analysis may go through the process of comparing the result with the theoretical aspects. Is the result has the same or similar or utterly different answer will be shown in the result.

This analysis may also discuss the possibility of how the Δv is produced. The result of the Δv might be the optimum value, or it might not, and this assumption will be proven in this section. The discussion of how Δv is interacting with the time of flight will also be written.

3.6 Report

Writing this dissertation is the final step of this research study. This dissertation is a summary of what I have been doing during the research study. Everything that I learned from a scratch into the final product is written in here and well documented.

This dissertation is written with a clear purpose. It will be submitted and presented to a particular audience though not all of the theoretical aspects of this report will be presented during the defence. The information in this report will make the audience who are not familiar with this subject understand the purpose behind this research study.

This report also addresses the problems or issues behind the research subject. There is some specific requirement to complete this dissertation provided by my University that must be followed.

CHAPTER 4
RESULTS AND DISCUSSIONS

4.1 Proof: Izzo's Lambert Problem algorithm are The Same as Universal Variables algorithm

By conducting several examples between Izzos Lambert problem algorithm and Universal variables algorithm, the result between the two certainly is the same. Table 4.1 shows the proof of it (see Appendix B for both Izzo's Lambert problem and universal variables algorithms).

Comparison Result between Izzo's solver and Universal Variables					
Initial Parameters		Izzo's Solver		Universal Variable	
r_0 r Δt	$220i + 0j + 0k$ $1000i + 2255j + 0k$ 4560 *seconds*	v_0 v	$53.31i + 26.74j + 0k$ $-8.61i - 13.52j + 0k$	v_0 v	$53.31i + 26.74j + 0k$ $-8.61i - 13.52j + 0k$
r_0 r Δt	$150i + 50j + 0k$ $500i + 1500j + 0k$ 4560 *seconds*	v_0 v	$54.60i + 44.64j + 0k$ $-8.94i - 18.91j + 0k$	v_0 v	$54.60i + 44.64j + 0k$ $-8.94i - 18.91j + 0k$
r_0 r Δt	$1000i + 225j + 0k$ $2545i + 1500j + 0k$ 4560 *seconds*	v_0 v	$25.22i + 8.697j + 0k$ $-12.80i - 6.357j + 0k$	v_0 v	$25.22i + 8.697j + 0k$ $-12.80i - 6.357j + 0k$

TABLE 4.1: Comparison Result between Izzo's Solver and Universal Variable

Following the above information (Table 4.1), it is evident that Izzos Lambert problem algorithm and universal variable algorithm produce the same result. But between one of them have the one advantage that people will take into consideration when using the algorithm, which is the speed. In theory, Izzos algorithm is, in fact, the fastest one. And during my dissertation, by running a command, the argument could be proven.

By running it using Ubuntu 20.04.1 LTS with 7.6 GiB memory and Intel® Core™ i7-6500U CPU @ 2.50GHz × 4 processor, the next step is to time it statistically. In this example, the example number two from Table 4.1 is used.

- **Izzo's Solver (In terminal, type: %timeit -n 10000 %run v0_v.py)**
  Result: *0.609 ms ± 2.68 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)*
- **Universal Variable (In terminal, type: %timeit -n 10000 %run Universal.py)**
  Result: *1.22 ms ± 4.77 µs per loop (mean ± std. dev. of 7 runs, 10000 loops each)*

The -n 10000 represents 10000 loops, while it is set by default to do 7 runs each time %timeit -n 10000 %run (type your file.py) being run. From the result, we could se the average time of the running process takes. And it is safe to say that Izzo's solver is roughly 2 times faster than universal variables, in which the theory has mentioned that Izzo's is the faster among others.

4.2 Relation of Δv between Lambert's Problem and Hohmann Transfer

During my thesis project, I found out throughout the process of finding the Δv using Izzo's solver that we could also find the result for Δv of Hohmann transfer. Figure 4.1 shows that the Δv_{min} from Izzo's solver plot is, in fact, the Δv of Hohmann Transfer.

Figure 4.1 is a plot produced using Izzo's solver. The minimum Δv (Δv_{min}) could also be said as the Δv of Hohmann Transfer. By definition, Hohmann transfer requires a small amount of fuel consumption because Hohmann transfer is the most energy-efficient transfer. The Δv of Hohmann transfer is relatively smaller than others solvers. But it requires a longer time of flight to achieve the final destination. That is why we could draw this assumption by looking at Figure 4.1.

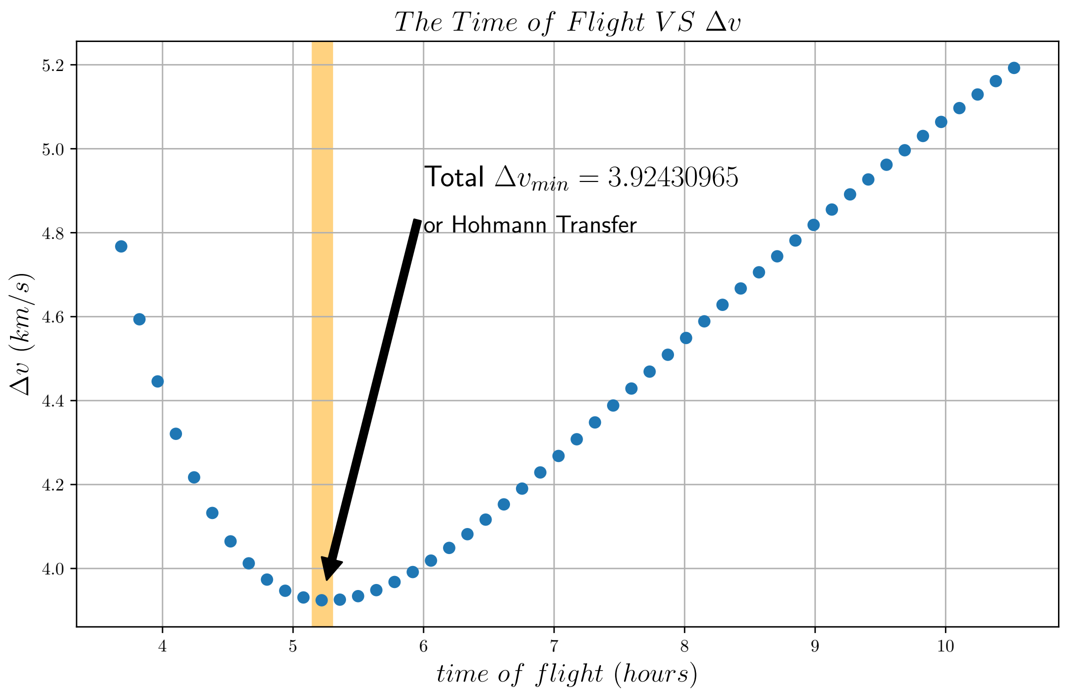
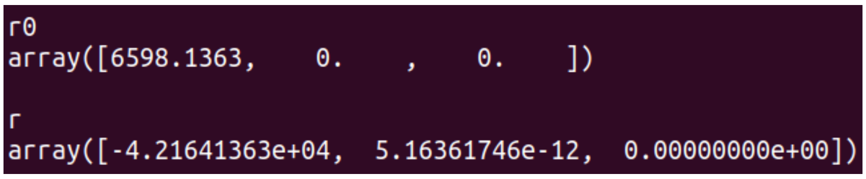


FIGURE 4.1: Relation Between Izzo's Lambert Problem and Hohmann Transfer. (See Appendix B for the algorithm)

In Chapter 3, especially in 3.3.1, shows the algorithm for Hohmann transfer. And by running the algorithm of Izzo's Lambert problem, the r_0 *and* r could be found (Figure 4.2), then transfer it to the Hohmann transfer algorithm.

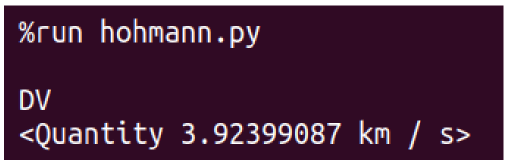


```
r0
array([6598.1363,    0.    ,    0.    ])

r
array([-4.21641363e+04,  5.16361746e-12,  0.00000000e+00])
```

FIGURE 4.2: r_0 & r value from Izzo's Solver Result

By running the Hohmann transfer algorithm using the parameters above, the result (Figure 4.3) will be the same as the minimum Δv (Figure 4.1) from Izzo's Lambert problem solver.



```
%run hohmann.py

DV
<Quantity 3.92399087 km / s>
```

FIGURE 4.3: Initial DV as the Result for Δv of Hohmann Transfer

4.3 Result of Izzo's Solver

Figure 4.4 is one of the example plot taken during this period of time which initial parameters $i_0 = 0°$. With the help of classical orbital elements (COE) as the initial and final orbit parameters, we could find the r_0 and r that leads in finding the total Δv. In setting both the initial and final orbit, eccentricity (e), altitude (h in km), semi-major axis (a in km), semiparameter (p in km), inclination (i in °), right ascension of the ascending node (Ω in °), argument of perigee (ω in °), and true anomaly (ν in °) are needed.

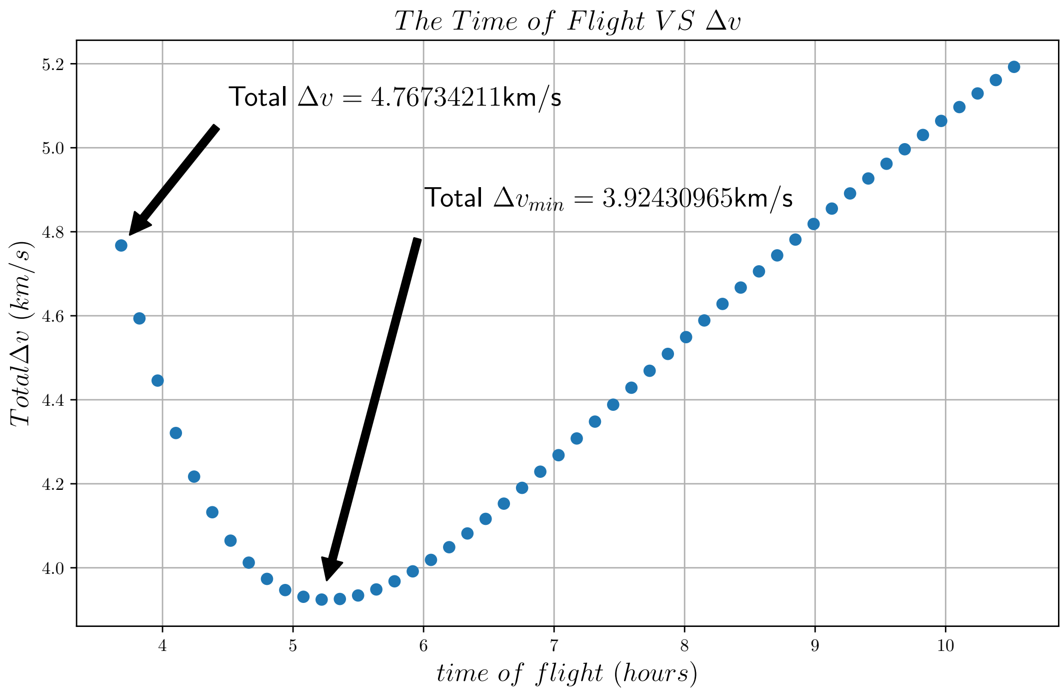


FIGURE 4.4: Izzo's Solver with $i_0 = 0°$

Throughout this process, all simulation in the final orbit (GEO) is using the same parameters with altitude is 35,786 km, true anomaly is set for 180°, and all other parameters are zero.

From Figure 4.4, it shows that in the first 5 hours, the total Δv is descending from the first tipping point. The first point as mentioned in the plot is equal to 4.76734211 km/s. Then, it starts to rise gradually with the highest total Δv is around 5.2 km/s.

4.3.1 The Variation of Result Using Izzo's Solver

In this section, it will show how the variation of initial orbital elements could affect the result of Total Delta-V.

Change of Inclination, i_0

The first change of initial orbital elements is the inclination. The inclination is vary start from 0° to 180° into one plot.

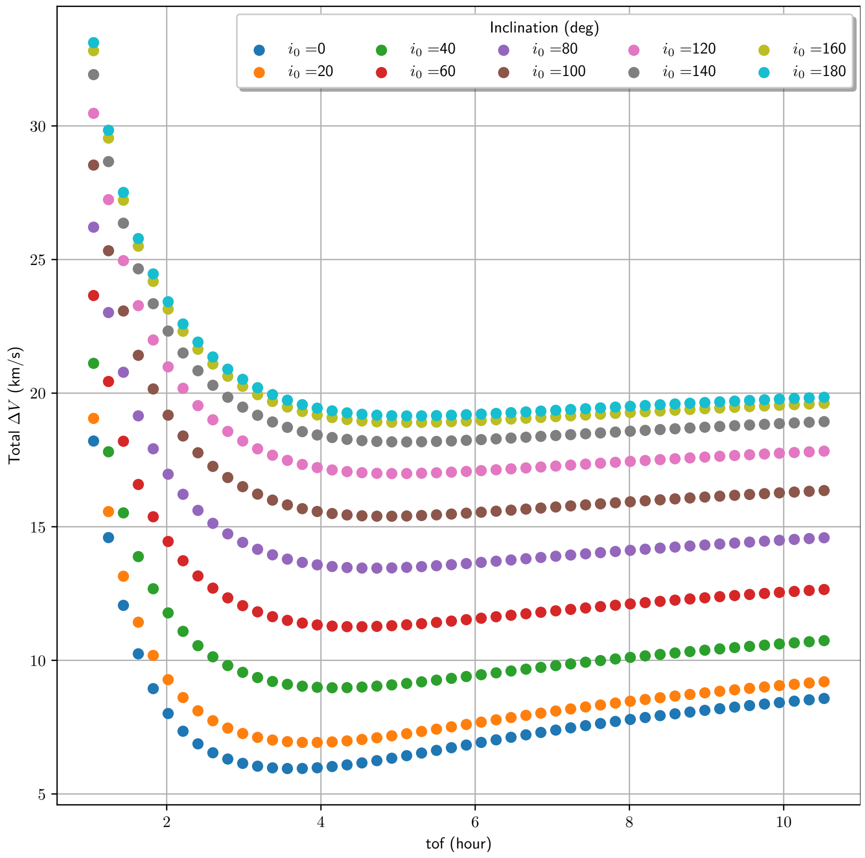


FIGURE 4.5: Inclination change vary from $i_0 = 0°$ to $i_0 = 180°$

From the information gathered in Figure 4.5, it is possible to state that when inclination change into a wider angle, the Total Δv gets higher.

$$i_0 \uparrow, \quad \Delta v_{total} \uparrow$$

Having $i_0 = 0°$ could save a lot of Total Δv and it is wisable to choose it rather then the other angles. The Δv_{min} is approximately around 6 km/s with 3.5 hours time-of-flight, in which the lowest among others. The highest is approximately around 19 km/s of Total Δv_{min} and 4.5 hours of time-of-flight with $i_0 = 180°$.

Change of Right Ascension of The Ascending Node, Ω_0

The second simulation is by changing the initial orbital elements of Right Ascension of The Ascending Node. The change is vary from 0° to 180° and produced it into one plot.

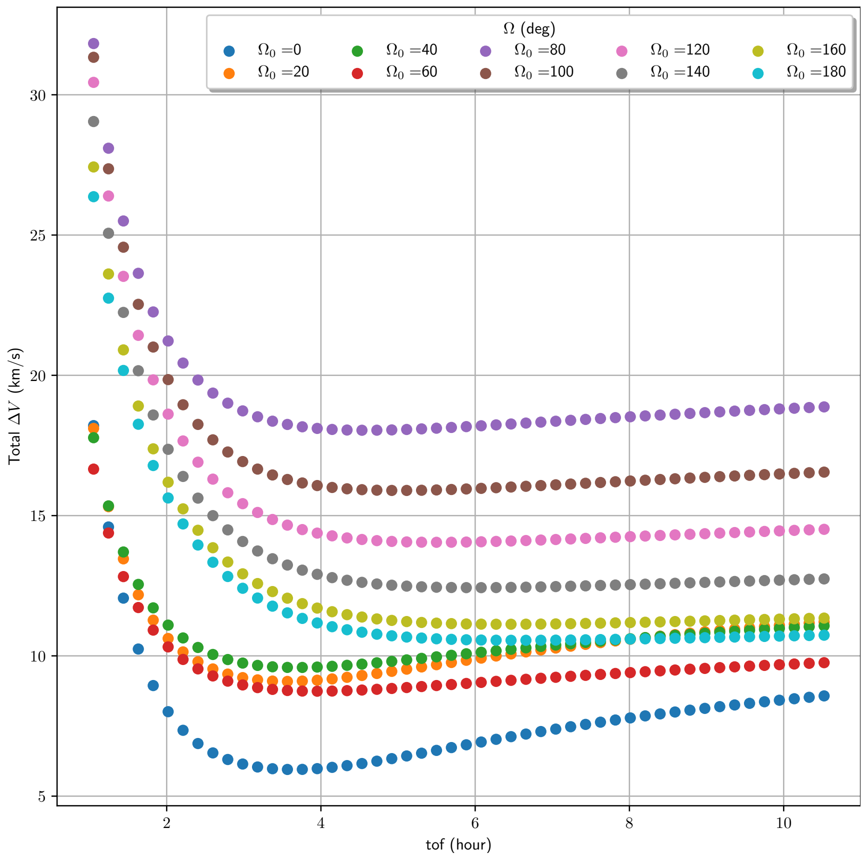


FIGURE 4.6: Right Ascension of The Ascending node change vary from $\Omega_0 = 0°$ to $\Omega_0 = 180°$

Different from inclination change, Figure 4.6 shows the changes in Ω_0 is not steadily rising. When Ω_0 change from $\Omega_0 = 0°$ to $\Omega_0 = 60°$, the changes is up and down. But when it reach 80° the highest Total Delta-V, it start decreasing gradually.

The value of $\Omega_0 = 0°$ is the same as $i_0 = 0°$ because there is no change in intital orbit. But if we look at the second lowest Total Δv_{min}, it is not 20° angle like in inclination variation plot, but rather it's 60° angle. The Total Δv_{min} it self is higher, approximately reaching 9 km/s, while for $i_0 = 20°$, the Total Δv_{min} is only around 7.5 km/s.

Change of Argument of Perigee, ω_0

The third simulation is by changing the initial orbital elements of Argument of Perigee, ω_0. The change is vary from 0° to 180° and produced it into one plot.

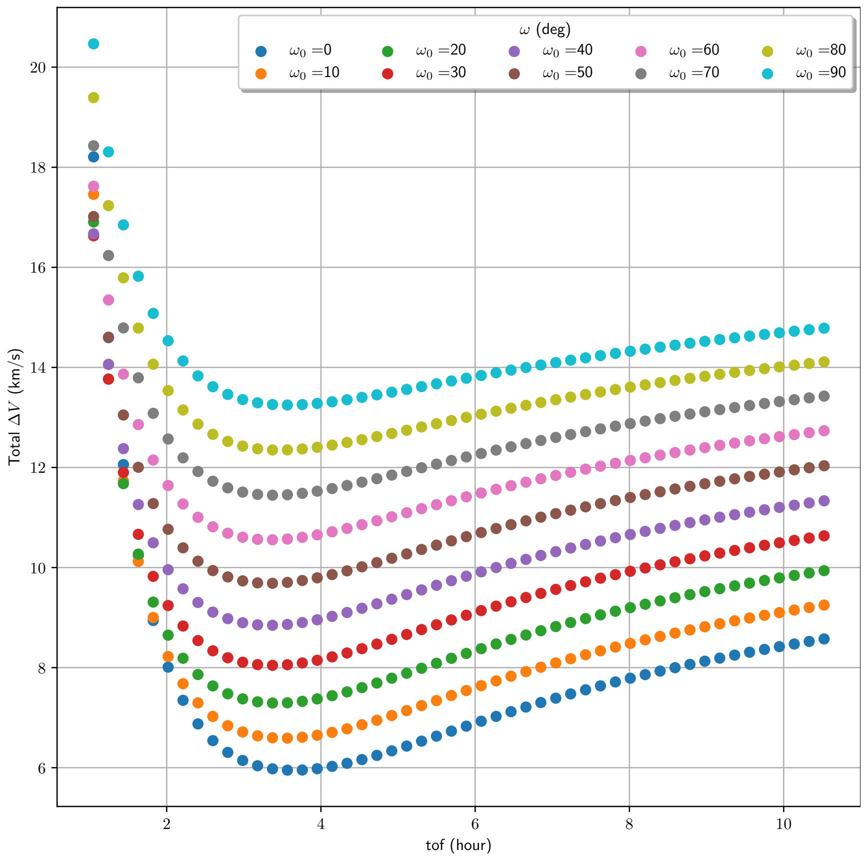


FIGURE 4.7: Argument of Perigee change vary from $\omega_0 = 0°$ to $\omega_0 = 180°$

Same as inclination, the change of ω_0 makes the Total Delta-V higher when the angle become wider.

$$\omega_0 \uparrow, \quad \Delta v_{total} \uparrow$$

As mention in previous simulation (change in Ω_0), the value of Total Δv_{min} at 0° is the same that approximately around 6 km/s and 3.5 hours of time-of-flight. But the second lowest Total Δv_{min} is at 20° angle that approximately almost reaching 7 km/s. The unique difference from the first two simulation is that the time-of-flight don't look any different when the angle become wider. It stay at around 3.5 hours of time-of-flight when it reach the Total Δv_{min}.

Change of Altitude, h_0

The fourth and last simulation is by changing the initial orbital elements of altitude, h_0. The change is vary from 220 km to 400 km and produced it into one plot.

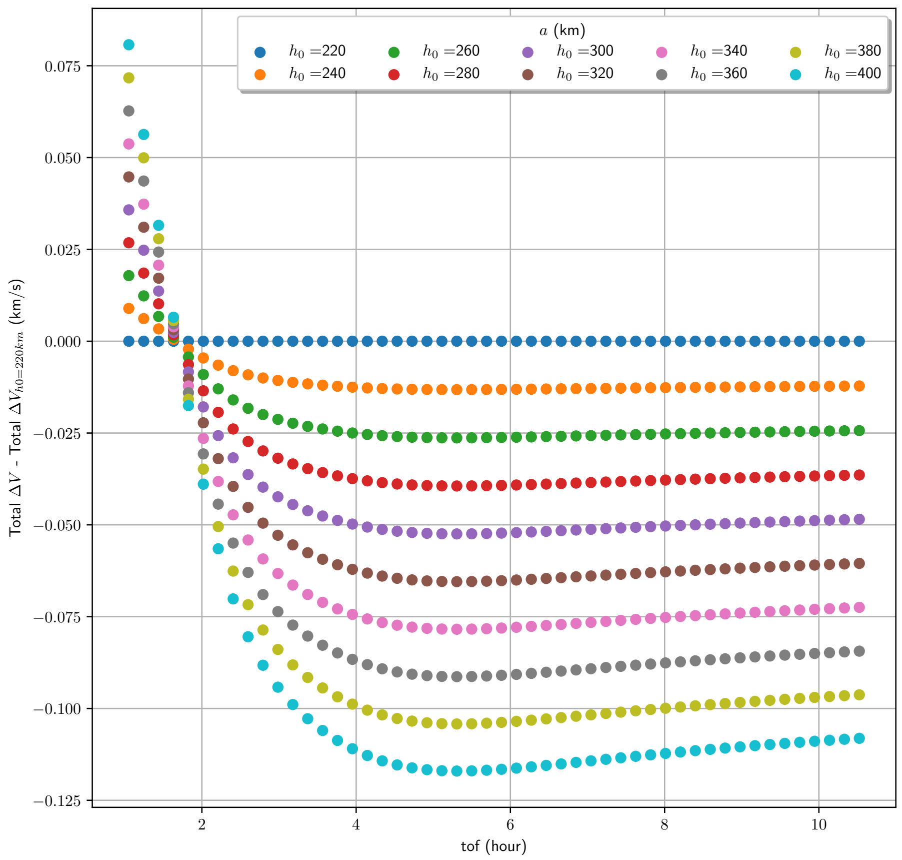


FIGURE 4.8: Altitude change vary from $h_0 = 220$km to $h_0 = 440$km

Figure 4.8 shows the variation of Total Δv with respect to Total Δv at 220 km altitude (h_0). It's also means that $h_0 = 220\ km$ acts as the benchmark to others. From the plot, it could give a conclusion that placing the initial altitude, h_0, has a big impact on how to save Total Δv. Because,

$$h_0 \uparrow, \quad \Delta v_{total} \downarrow$$

Placing at $h_0 = 400\ km$ that in this simulation is farthest altitude, could save roughly around 0.1 km/s or 100 m/s of Total Δv.

4.4 Simulation Parameters

Simulation parameters used in the change of $i_0,\ \Omega_0,\ \omega_0,\ and\ h_0$.

Initial Orbit LEO (Change in i_0)								
	e	$h\ (km)$	$a\ (km)$	$p\ (km)$	i	Ω	ω	ν
Simulation 1	0	220	$h+ER$	$a(1-e^2)$	0°	0°	0°	0°
Simulation 2	0	220	$h+ER$	$a(1-e^2)$	20°	0°	0°	0°
Simulation 3	0	220	$h+ER$	$a(1-e^2)$	40°	0°	0°	0°
Simulation 4	0	220	$h+ER$	$a(1-e^2)$	60°	0°	0°	0°
Simulation 5	0	220	$h+ER$	$a(1-e^2)$	80°	0°	0°	0°
Simulation 6	0	220	$h+ER$	$a(1-e^2)$	100°	0°	0°	0°
Simulation 7	0	220	$h+ER$	$a(1-e^2)$	120°	0°	0°	0°
Simulation 8	0	220	$h+ER$	$a(1-e^2)$	140°	0°	0°	0°
Simulation 9	0	220	$h+ER$	$a(1-e^2)$	160°	0°	0°	0°
Simulation 10	0	220	$h+ER$	$a(1-e^2)$	180°	0°	0°	0°

TABLE 4.2: Initial Orbit Parameters LEO,
change in i_0 with **ER = 6378 *km***

Initial Orbit LEO (Change in Ω_0)								
	e	$h\ (km)$	$a\ (km)$	$p\ (km)$	i	Ω	ω	ν
Simulation 1	0	220	$h+ER$	$a(1-e^2)$	$0°$	$0°$	$0°$	$0°$
Simulation 2	0	220	$h+ER$	$a(1-e^2)$	$0°$	$20°$	$0°$	$0°$
Simulation 3	0	220	$h+ER$	$a(1-e^2)$	$0°$	$40°$	$0°$	$0°$
Simulation 4	0	220	$h+ER$	$a(1-e^2)$	$0°$	$60°$	$0°$	$0°$
Simulation 5	0	220	$h+ER$	$a(1-e^2)$	$0°$	$80°$	$0°$	$0°$
Simulation 6	0	220	$h+ER$	$a(1-e^2)$	$0°$	$100°$	$0°$	$0°$
Simulation 7	0	220	$h+ER$	$a(1-e^2)$	$0°$	$120°$	$0°$	$0°$
Simulation 8	0	220	$h+ER$	$a(1-e^2)$	$0°$	$140°$	$0°$	$0°$
Simulation 9	0	220	$h+ER$	$a(1-e^2)$	$0°$	$160°$	$0°$	$0°$
Simulation 10	0	220	$h+ER$	$a(1-e^2)$	$0°$	$180°$	$0°$	$0°$

TABLE 4.3: Initial Orbit Parameters LEO,
change in Ω_0 with **ER = 6378** $\boldsymbol{km}$

Initial Orbit LEO (Change in ω_0)								
	e	$h\ (km)$	$a\ (km)$	$p\ (km)$	i	Ω	ω	ν
Simulation 1	0	220	$h+ER$	$a(1-e^2)$	$0°$	$0°$	$0°$	$0°$
Simulation 2	0	220	$h+ER$	$a(1-e^2)$	$0°$	$0°$	$10°$	$0°$
Simulation 3	0	220	$h+ER$	$a(1-e^2)$	$0°$	$0°$	$20°$	$0°$
Simulation 4	0	220	$h+ER$	$a(1-e^2)$	$0°$	$0°$	$30°$	$0°$
Simulation 5	0	220	$h+ER$	$a(1-e^2)$	$0°$	$0°$	$40°$	$0°$
Simulation 6	0	220	$h+ER$	$a(1-e^2)$	$0°$	$0°$	$50°$	$0°$
Simulation 7	0	220	$h+ER$	$a(1-e^2)$	$0°$	$0°$	$60°$	$0°$
Simulation 8	0	220	$h+ER$	$a(1-e^2)$	$0°$	$0°$	$70°$	$0°$
Simulation 9	0	220	$h+ER$	$a(1-e^2)$	$0°$	$0°$	$80°$	$0°$
Simulation 10	0	220	$h+ER$	$a(1-e^2)$	$0°$	$0°$	$90°$	$0°$

TABLE 4.4: Initial Orbit Parameters LEO,
change in ω_0 with **ER = 6378** $\boldsymbol{km}$

Initial Orbit LEO (Change in h_0)								
	e	h (km)	a (km)	p (km)	i	Ω	ω	ν
Simulation 1	0	220	$h+ER$	$a(1-e^2)$	0°	0°	0°	0°
Simulation 2	0	240	$h+ER$	$a(1-e^2)$	0°	0°	0°	0°
Simulation 3	0	260	$h+ER$	$a(1-e^2)$	0°	0°	0°	0°
Simulation 4	0	280	$h+ER$	$a(1-e^2)$	0°	0°	0°	0°
Simulation 5	0	300	$h+ER$	$a(1-e^2)$	0°	0°	0°	0°
Simulation 6	0	320	$h+ER$	$a(1-e^2)$	0°	0°	0°	0°
Simulation 7	0	340	$h+ER$	$a(1-e^2)$	0°	0°	0°	0°
Simulation 8	0	360	$h+ER$	$a(1-e^2)$	0°	0°	0°	0°
Simulation 9	0	380	$h+ER$	$a(1-e^2)$	0°	0°	0°	0°
Simulation 10	0	400	$h+ER$	$a(1-e^2)$	0°	0°	0°	0°

TABLE 4.5: Initial Orbit Parameters LEO,
change in h_0 with **ER = 6378 *km***

Final Orbit LEO								
	e	h (km)	a (km)	p (km)	i	Ω	ω	ν
All Simulation	0	35,786	$h+ER$	$a(1-e^2)$	0°	0°	0°	120°

TABLE 4.6: Final Orbit Parameters GEO with **ER = 6378 *km***

4.5 Remarks on How Total Δv Reduction Affect The Launch Cost

Using the Tsiolkovsky's rocket equation, it is possible to calculate how much payload or mass per-satellite could be save with Total Δv reduction when travelling from LEO to GEO.
Tsiolkovsky's equation:

$$\Delta v = v_e ln \frac{m_0}{m_f} \tag{4.1}$$

Where,

Δv	The maximum change of velocity reduction
v_e	$I_{sp}g_0$
	I_{sp} Specific impulse
	g_0 Standard gravity (9.80665 m^2/s)
m_0	The initial mass
m_f	The final mass (mass after the Total Δv reduction)

TABLE 4.7: Tsiolkovsky's equation Parameters Explaination

The result from the variation of Total Δv from LEO to GEO due to altitude change (h_0) is the one taken for this calculation in which the Total Δv reduction value is around 0.1 km/s.

Choosing the parameter of $\Delta v = 0.1$ km/s, with the help of initial BRIsat (Page, 2020) mass and the I_{sp} of apogee kick motor ($I_sp = 282.9s$), found:

	Initial	Final
mass	3540 kg	3414.684 kg
Launch Cost ($ 30,000 per kg)	$ 106,200,000	$ 102,440,520

TABLE 4.8: New Payload / Mass & Cost Travel From LEO to GEO

Table 4.8 gives the number that needed to analyze how much payload or mass and cost could be save with only 0.1 km/s of Total Δv reduction.

Payload Savings:	125.316 kg
Launch Cost Savings:	$ 3,759,480

TABLE 4.9: Payload and Cost savings with only 0.1 km/s of Total Δv reduction

CHAPTER 5
SUMMARY, CONCLUSION, RECOMMENDATION

5.1 Summary

To do a transfer orbit from LEO to GEO is beneficial since, in GEO, it moves in the same direction as Earth's rotation under the same period. But placing it in GEO is very expensive in terms of the launch cost. By reducing the total Δv, it will give a lower launch cost or the satellite could bring more payload. While considering it, still, the main focus of this thesis was to do a transfer orbit, in which Lambert's problem is the one used. Lambert's problem uses only two impulses to complete the transfer and can be considered in a single revolution or multiple revolutions. There are many techniques to solve Lambert's problem. One of which is used in this thesis as the main focus was Izzo's solver. Based on numerical experiments in this thesis, Izzo's algorithm was the fastest algorithm with only two iterations compared to the universal variables technique, which requires more than two iterations. With the correct time of flight and varying the initial orbital elements, it was possible to know how the total Δv affected while the final orbit remained constant.

5.2 Conclusion

In conclusion, it has been a success in terms of implementing Lambert's problem. Izzo's algorithm and Lambert's universal variable have been compared and the results were satisfying. Izzo's algorithm produced a shorter time, which is based on the simulation taken (Chapter 4) during this thesis project that Izzo's algorithm is roughly 2 times faster (computational time) than Universal Variables.

Izzo's solver has also been used where one of the initial orbital elements varies. Throughout the simulation, the results were rewarding. The variation of the initial orbital elements like the inclination (i_0), right ascension of the ascending node (Ω_0), argument of perigee (ω_0), and altitude (h_0) could change the result of Total Δv. Choosing to vary the initial altitude (h_0) was a good example to show that by reducing the total Δv, it is possible to get a lower launch cost. Placing it at 400 km (the farthest altitude in this thesis) could save roughly around 0.1 km/s or 100 m/s of Total Δv. It may look small, but with only 0.1 km/s reduction, it could reduce the total launch cost up to 3,759,480 USD to travel from LEO to GEO.

5.3 Recommendation

For the future project, it would be nice to continue this project with the focus on multiple revolutions following the one done in this thesis: a single revolution. While multiple revolutions are an idea, students could also focus on how multiple impulses ($n > 2$) affect the Δv. Another idea that could be available is by varying not only one initial orbital elements like taken on this thesis, but with all initial orbital elements.

References

Advantages of Satellites | Telesat. (n.d.). Retrieved 2020-02-07, from https://www.telesat.com/about-us/why-satellite/advantages-satellites

antaranews.com. (2020, January). *LAPAN tingkatkan kualitas LAPAN Fire Hotspot pantau titik panas.* Retrieved 2020-02-11, from https://www.antaranews.com/berita/1269897/lapan-tingkatkan-kualitas-lapan-fire-hotspot-pantau-titik-panas

Conte, D. (2014). *(PDF) Determination of Optimal Earth-Mars Trajectories to Target the Moons of Mars.* Retrieved 2020-09-15, from https://www.researchgate.net/publication/323256808_Determination_of_Optimal_Earth-Mars_Trajectories_to_Target_the_Moons_of_Mars doi: 10.13140/RG.2.2.22494.74563

Curtis, H. D. (2014). *Orbital mechanics for engineering students* (Third edition ed.). Amsterdam ; Boston: Elsevier, BH, Butterworth-Heinemann is an imprint of Elsevier. (OCLC: ocn852806044)

Delta-v budget. (2019, November). Retrieved 2020-02-11, from https://en.wikipedia.org/w/index.php?title=Delta-v_budget&oldid=927898166 (Page Version ID: 927898166)

evn. (n.d.). *LAPAN: Indonesia Kembangkan Satelit, Bukan Misi ke Bulan.* Retrieved 2020-02-11, from https://www.cnnindonesia.com/teknologi/20190722160923-199-414412/lapan-indonesia-kembangkan-satelit-bukan-misi-ke-bulan

Figure 1. Schematic diagram of the classical Lambert problem involving... (n.d.). Retrieved 2020-03-26, from https://www.researchgate.net/figure/Schematic-diagram-of-the-classical-Lambert-problem-involving-the-orbital-transfer-from-an_fig1_323682490

Izzo, D. (2015, January). Revisiting Lamberts problem. *Celestial Mechanics and Dynamical Astronomy*, *121*(1), 1–15. Retrieved 2020-03-30, from http://link.springer.com/10.1007/s10569-014-9587-y doi: 10.1007/s10569-014-9587-y

Johann Heinrich Lambert. (2020, March). Retrieved 2020-03-26, from https://en.wikipedia.org/w/index.php?title=Johann_Heinrich_Lambert&oldid=943819324 (Page Version ID: 943819324)

Joseph-Louis Lagrange. (2020, March). Retrieved 2020-03-26, from https://en.wikipedia.org/w/index.php?title=Joseph-Louis_Lagrange&oldid=947091541 (Page Version ID: 947091541)

Lambert's problem. (2019, July). Retrieved 2020-03-26, from https://en.wikipedia.org/w/index.php?title=Lambert%27s_problem&oldid=906323856 (Page Version ID: 906323856)

Launching Satellites | ESOA. (n.d.). Retrieved 2020-02-11, from https://www.esoa.net/technology/launching-satellites.asp

Orbital maneuver. (2019, December). Retrieved 2020-02-12, from https://en.wikipedia.org/w/index.php?title=Orbital_maneuver&oldid=931346564 (Page Version ID: 931346564)

Otani, Y., Ohkami, Y., Naohiko KOHTAKE, & Sakurai, T. (2012). Dual-Use Concept on Civil and Defense Uses of Outer Space. *TRANSACTIONS OF THE JAPAN SOCIETY FOR AERONAUTICAL AND SPACE SCIENCES, AEROSPACE TECHNOLOGY JAPAN*, *10*(ists28), Tv_1–Tv_5. Retrieved 2020-02-10, from http://japanlinkcenter.org/DN/JST.JSTAGE/tastj/10.Tv_1?lang=en&from=CrossRef&type=abstract doi: 10.2322/tastj.10.Tv_1

Page, G. S. (2020). *BRIsat.* Retrieved 2020-08-27, from https://space.skyrocket.de/doc_sdat/brisat-1.htm

PUSAT KAJIAN KEBIJAKAN PENERBANGAN DAN ANTARIKSA. (n.d.). Retrieved 2020-02-10, from https://puskkpa.lapan.go.id/index.php/subblog/read/2019/142/Forum-Group-Discussion-FGD-tentang-GSO

Rakornas Inderaja 2020 : Satelit adalah Jawaban Masa Depan. (n.d.). Retrieved 2020-02-11, from https://lapan.go.id/post/6011/menristek-lapan-kembangkan-satelit-untuk-pertahanan-dan-komersial

- REMOTE SENSING DATA: APPLICATIONS AND BENEFITS. (n.d.). Retrieved 2020-02-10, from https://www.govinfo.gov/content/pkg/CHRG

-110hhrg41573/html/CHRG-110hhrg41573.htm

Satellite Geostationary Satellite Orbit, GEO ż Electronics Notes. (n.d.). Retrieved 2020-02-10, from https://www.electronics-notes.com/articles/satellites/basic-concepts/geostationary-orbit-geo.php

Satellite Orbits | ESOA. (n.d.). Retrieved 2020-09-03, from https://www.esoa.net/technology/satellite-orbits.asp

Simple control laws for low-thrust orbit transfer. (n.d.). Retrieved 2020-02-12, from https://trs.jpl.nasa.gov/bitstream/handle/2014/38468/03-2057.pdf?sequence=1&isAllowed=y

Single impulse maneuvers in circular orbit. (n.d.). Retrieved 2020-02-12, from https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&cad=rja&uact=8&ved=2ahUKEwiI4Li8j_vnAhUXfHOKHY2wCIAQFjAAegQIAhAB&url=http%3A%2F%2Fwk.ixueshu.com%2Ffile%2Fbc9011aede2f31ea318947a18e7f9386.html&usg=AOvVaw3FxaQheVOH_Dn-iiIGvK6y

SPACE FREIGHTER | Cannae. (n.d.). Retrieved 2020-08-27, from http://cannae.com/space-freighter/

Vallado, D. A. (2013). *Fundamental of astrodynamics and applications.* Microcosm Press and Springer.

Appendices

Appendix A: Izzo's Solver Algorithm

In order to produce all the figures in Chapter 4, creating the python code below is important. Each python code (per number) need to be **created in the same folder with seperate python files** (.py).
The main Izzo's solver code are:

1. **Jit file (jit.py)**;

```python
import inspect
import warnings

def ijit(first=None, *args, **kwargs):
    """
    Identity JIT, returns unchanged function.
    """
    def jit_(f):
        return f
    if inspect.isfunction(first):
        return first
    else:
        return jit_
try:
    import numba

    jit = numba.njit
except ImportError:
    warnings.warn(
       "Could not import numba package"
       "Functions will work properly but the CPU intensive"
```

```
        "algorithms will be slow. Consider installing numba to"
        "boost performance."
    )
    jit = ijit
```

2. **Hypergeometric function $_2F_1(3, 1, 5/2, x)$ file (hyperf.py)**;

```
import numpy as np
from numba import jit
# from .jit import jit

@jit
def hyp2f1b(x):
    """
    Hypergeometric function 2F1(3, 1, 5/2, x)
    """
    if x >= 1.0:
        return np.inf
    else:
        res = 1.0
        term = 1.0
        ii = 0
        while True:
            term = term * (3 + ii) * (1 + ii) /
             (5 / 2 + ii) * x / (ii + 1)
            res_old = res
            res += term
            if res_old == res:
                return res
            ii += 1
```

3. **Izzo's solver file (solver.py)**;

```
from numba import jit
import numpy as np
```

```python
from hyperf import hyp2f1b
import warnings
warnings.filterwarnings('ignore')
@jit
def izzo(miu, r1, r2, t, M, numiter, rtol):
    """
    Applie izzo algorithm to solve Lambert's problem.

    Parameter:

        miu     :   Gravitational constant
        r1      :   Initial position vector
        r2      :   Final position vector
        t       :   Time of flight between both positions
        M       :   Number of revolutions
        numiter :   Number of iterations
        rtol    :   Error tolerance

    Goals:

        v1      :   Initial velocity vector
        v2      :   Final velocity vector
    """
    #   Precondition
    assert t > 0
    assert miu > 0
    # Check collinearity of r1 and r2
    if np.all(np.cross(r1, r2) == 0):
        raise ValueError("Lambert solution cannot be computed\
                             for collinear vectors")
    # chord
    c = r2 - r1
    c_norm, r1_norm, r2_norm = np.linalg.norm(c),
                                       np.linalg.norm(r1),
                                       np.linalg.norm(r2)

    # semiperimeter
    s = (r1_norm + r2_norm + c_norm) * 0.5

    i_r1, i_r2 = r1 / r1_norm, r2 / r2_norm
    i_h = np.cross(i_r1, i_r2)
    i_h = i_h / np.linalg.norm(i_h)

    # ll is lambda
    ll = np.sqrt(1 - min(1.0, c_norm / s))

    if i_h[2] < 0:
        ll = -ll
        i_h = - i_h

    # from Poliastro
    i_t1, i_t2 = np.cross(i_h, i_r1), np.cross(i_h, i_r2)

    # Non dimensional time of flight
    T = np.sqrt(2 * miu / s ** 3) * t

    # Find x & y list
    xy = find_xy(ll, T, M, numiter, rtol)

    # Reconstruct
    gamma = np.sqrt(miu * s / 2)
    rho = (r1_norm - r2_norm) / c_norm
    sigma = np.sqrt(1 - rho ** 2)

    for x, y in xy:
        V_r1, V_r2, V_t1, V_t2 = reconstruct(
                x, y, r1_norm, r2_norm, ll, gamma, rho, sigma
            )
            v1 = V_r1 * i_r1 + V_t1 * i_t1
            v2 = V_r2 * i_r2 + V_t2 * i_t2
            yield v1, v2
@jit
def reconstruct(x, y, r1, r2, ll, gamma, rho, sigma):
    """
    Reconstruct solution velocity vectors.
    """
    V_r1 = gamma * ((ll * y - x) - rho * (ll * y + x)) / r1
    V_r2 = -gamma * ((ll * y - x) + rho * (ll * y + x)) / r2
    V_t1 = gamma * sigma * (y + ll * x) / r1
    V_t2 = gamma * sigma * (y + ll * x) / r2
    return [V_r1, V_r2, V_t1, V_t2]
@jit
def find_xy(ll, T, M, numiter, rtol):
    """
    Computes all x, y for given number of revolutions.
    """
    assert np.abs(ll) < 1
    assert T > 0

    M_max = np.floor(T / np.pi)
    T_00 = np.arccos(ll) + ll * np.sqrt(1 - ll ** 2)

    if T < T_00 + M_max * np.pi and M_max > 0:
        _, T_min = compute_T_min(ll, M_max, numiter, rtol)
        if T < T_min:
            M_max -= 1
    if M > M_max:
        raise ValueError("No feasible solution, try lower M")
    # Initial Guess (Single Revolution)
    for x_0 in initial_guess(T, ll, M):
        # Start Householder iteration from x_0 and find x, y
        x = householder(x_0, T, ll, M, rtol, numiter)
        y = compute_y(x, ll)

        yield x, y
@jit
def compute_y(x, ll):
    """
    Computes y.
    """
    return np.sqrt(1 - ll ** 2 * (1 - x ** 2))
@jit
def compute_psi(x, y, ll):
    """
    Computes psi (the auxiliary angle).
    """
    if -1 <= x < 1:
        # Elliptic motion
        # by using arc cosine, we could avoid numerical errors
        return np.arccos(x * y + ll * (1 - x ** 2))
    elif x > 1:
        # Hyperbolic motion
        # Bijective
        return np.arcsinh((y - x * ll) * np.sqrt(x ** 2 - 1))
    else:
        # Parabolic motion
        return 0.0
@jit
def t_equation(x, T0, ll, M):
    """
    Time of Flight equation
    """
    return t_equation_y(x, compute_y(x, ll), T0, ll, M)
@jit
def t_equation_y(x, y, T0, ll, M):
    """
    Time of Flight equation with computed y.
    """
    if M == 0 and np.sqrt(0.6) < x < np.sqrt(1.4):
        eta = y - ll * x
        S_1 = (1 - ll - x * eta) * 0.5
        Q = 4 / 3 * hyp2f1b(S_1)
        T_ = (eta ** 3 * Q + 4 * ll * eta) * 0.5
    else:
        psi = compute_psi(x, y, ll)
        T_ = np.divide(
            np.divide(psi + M * np.pi,
            np.sqrt(np.abs(1 - x ** 2))) - x + ll * y,
            (1 - x ** 2),
        )
    return T_ - T0
@jit
def t_equation_p(x, y, T, ll):
    return (3 * T * x - 2 + 2 * ll ** 3 * x / y) / (1 - x ** 2)
@jit
def t_equation_p2(x, y, T, dT, ll):
    return (3 * T + 5 * x * dT + 2 * (1 - ll ** 2) * ll ** 3
            / y ** 3) / (1 - x ** 2)
@jit
def t_equation_p3(x, y, _, dT, ddT, ll):
    return (7 * x * ddT + 8 * dT - 6 * (1 - ll ** 2) *
                    ll ** 5 * x / y ** 5) / (1 - x ** 2)
@jit
def compute_T_min(ll, M, numiter, rtol):
    """
    Compute minimum T.
    """
    if ll == 1:
        x_T_min = 0.0
        T_min = t_equation(x_T_min, 0.0, ll, M)
    else:
        if M == 0:
            x_T_min = np.inf
            T_min = 0.0
        else:
            # Set x_i > 0 to avoid problems at ll = -1
            x_i = 0.1
            T_i = t_equation(x_i, 0.0, ll, M)
            x_T_min = halley(x_i, T_i, ll, rtol, numiter)
            T_min = t_equation(x_T_min, 0.0, ll, M)

    return [x_T_min, T_min]
@jit
def initial_guess(T, ll, M):
    """
    Initial guess.
    """
    if M == 0:
        # Single revolution
        T_0 = np.arccos(ll) + ll * np.sqrt(1 - ll ** 2) +
                M * np.pi
        T_1 = 2 * (1 - ll ** 3) / 3
        if T >= T_0:
            x_0 = (T_0 / T) ** (2 / 3) - 1
        elif T < T_1:
            x_0 = 5 / 2 * T_1 / T * (T_1 - T) / (1 - ll ** 5) + 1
        else:
            x_0 = (T_0 / T) ** (np.log2(T_1 / T_0)) - 1

            return [x_0]
        else:
            # Multiple revolution
            x_0l = (((M * np.pi + np.pi) / (8 * T)) ** (2 / 3) - 1) /
                    (((M * np.pi + np.pi) / (8 * T)) ** (2 / 3) + 1)
            x_0r = (((8 * T) / (M * np.pi)) ** (2 / 3) - 1) / (
                ((8 * T) / (M * np.pi)) ** (2 / 3) + 1)

            return [x_0l, x_0r]
@jit
def halley(p0, T0, ll, tol, maxiter):
    """
    Find a minimum of time of flight equation using the Halley
    method.
    """
    for ii in range(maxiter):
        y = compute_y(p0, ll)
        fder = t_equation_p(p0, y, T0, ll)
        fder2 = t_equation_p2(p0, y, T0, fder, ll)
        if fder2 == 0:
            raise RuntimeError("Derivative was zero")
        fder3 = t_equation_p3(p0, y, T0, fder, fder2, ll)

        # Halley step (cubic)
        p = p0 - 2 * fder * fder2 / (2 * fder2 ** 2 -
                                        fder * fder3)

        if abs(p - p0) < tol:
            return p
        p0 = p

    raise RuntimeError("Failed to converge")
```

```python
@jit
def householder(p0, T0, ll, M, tol, maxiter):
    """
    Find a zero of time of flight equation using the Householder
    method.
    """
    for ii in range(maxiter):
        y = compute_y(p0, ll)
        fval = t_equation_y(p0, y, T0, ll, M)
        T = fval + T0
        fder = t_equation_p(p0, y, T, ll)
        fder2 = t_equation_p2(p0, y, T, fder, ll)
        fder3 = t_equation_p3(p0, y, T, fder, fder2, ll)

        # Householder step (quartic)
        p = p0 - fval * (
            (fder ** 2 - fval * fder2 / 2)
            / (fder * (fder ** 2 - fval * fder2) + fder3 *
            fval ** 2 / 6)
        )
        if abs(p - p0) < tol:
            return p
        p0 = p
    raise RuntimeError("Failed to converge")
```

4. **Hohmann Transfer file for finding the time of flight (hohmann.py);**

```python
import numpy as np
ER = 6378  # Earth's radius (km)
MIU_E = 398600  # km^3/s^2 (earth)
DU = ER  # Distance Unit in km
TU = np.sqrt(DU ** 3 / MIU_E)  # Time Unit in second
MIU = 1  # Normlaized GM

```

```python

def hohmann(ri, rf):
    """
    Performs Hohmann transfer between two circular orbits.
    Keyword Arguments:
    ri -- radius of initial orbit
    rf -- radius of final orbit
    """
    ri = ri
    rf = rf
    at = (ri + rf) / 2

    T2 = (2 * np.pi / np.sqrt(MIU_E)) * (at ** (3 / 2))

    DT = T2 / 2

    return DT #time of flight
```

5. **Classical orbital elements to state vector for finding the initial and final orbit (coe2rv.py);**

```python
import numpy as np
miu = 398600  # km^3 / s^2
ER = 6378
def coe2rv(p, e, i, RA, AP, TA):
    """
    In order to find the State Vector r0 & r
    Parameter:
        a = Semimajor axis
        p = semiparameter
        e = Eccentricity
        i = Inclination
        RA = Right Ascension of the ascending node
        AP = Argument of Pergiee
        TA = True Anomaly
    """
    i = np.radians(i)
    RA = np.radians(RA)
    AP = np.radians(AP)
    TA = np.radians(TA)

    par1 = np.array([np.cos(TA), np.sin(TA), 0])
    r_pqw = (p / (1 + e * np.cos(TA))) * par1

    par2 = np.array([-np.sin(TA), e + np.cos(TA), 0])
    v_pqw = np.sqrt(miu / p) * par2

    c_RA = np.cos(RA)
    c_AP = np.cos(AP)
    c_i = np.cos(i)
    s_RA = np.sin(RA)
    s_AP = np.sin(AP)
    s_i = np.sin(i)

    ijk_pqw = np.array([[c_RA * c_AP - s_RA * s_AP * c_i, - c_RA *
                           s_AP - s_RA * c_AP * c_i, s_RA * s_i],
                        [s_RA * c_AP + c_RA * s_AP * c_i, - s_RA *
                          s_AP + c_RA * c_AP * c_i, - c_RA * s_i],
                        [s_AP * s_i, c_AP * s_i, c_i]])

    r_ijk = np.dot(ijk_pqw, r_pqw)
    v_ijk = np.dot(ijk_pqw, v_pqw)

    return r_ijk, v_ijk
```

Appendix B: Result Algorithm

After creating python codes above that placed in the same folder, finding the plot result in Chapter 4 is the next step. Readers could personalize the name of the file (no specific name needed), unlike the codes presented in Appendix A.

1. **Finding v_0 & v from Izzo's solver (Table 4.1)**;

```
import numpy as np
from solver import izzo

r0 = np.array([150, 50, 0])  #insert initial position (km)
r = np.array([500, 1500, 0]) #insert final position (km)
t = 76 * 60  # time of flight (second)
miu = 398600  # km^3 / s^2
ER = 6378

def velocity(miu, r0, r, t, M=0, numiter=35, rtol=1e-8):

    sols = izzo(miu, r0, r, t, M, numiter, rtol)
    for v1, v2 in sols:
        yield v1, v2

(v1, v2), = velocity(miu, r0, r, t)
```

2. **Finding v_0 & v from Lambert's Universal Variables (Table 4.1)**;

```
import numpy as np
r0 = np.array([150, 50, 0])  #insert initial position (km)
r = np.array([500, 1500, 0]) #insert final position (km)
t_m = +1  # short way
```

```python
miu = 398600  # km^3 / s^2
ER = 6378.1363
delta_t = 76 * 60

def c2(psi):
    """
    Second Stumpff function (c2)
    """
    val = 0
    if psi > val:
        c = (1 - np.cos(np.sqrt(psi))) / psi
    elif psi < val:
        c = (np.cosh(np.sqrt(-psi)) - 1) / (-psi)
    else:
        c = 1.0 / 2.0
    return c

def c3(psi):
    """
    Third Stumpff function (c3)
    """
    val = 0
    if psi > val:
        s = (np.sqrt(psi) - np.sin(np.sqrt(psi))) /
                (psi * np.sqrt(psi))
    elif psi < val:
        s = (np.sinh(np.sqrt(-psi)) - np.sqrt(-psi)) /
                (-psi * np.sqrt(-psi))
    else:
        s = 1.0 / 6.0
    return s

def universal_variables(r0, r, delta_t, t_m):
    """
    Lambert's Problem Solution using Universal Variables.
    where:
        - n_ro & n_r     = Magnitude of both r0 & r
        - TA             = True Anomaly
    In the process of Looping, the number 35 could be replaced.
    It depends on the number of iteration available.
    """
    n_r0 = np.dot(r0, r0)**0.5
    n_r = np.dot(r, r)**0.5

    cos_TA = np.dot(r0, r) / (n_r0 * n_r)
    # sin_TA = t_m * np.sqrt(1 - cos_TA ** 2)
    # TA = np.degrees(np.arcsin(sin_TA))

    A = t_m * (n_r * n_r0 * (1 + cos_TA))**.5
    if A == 0.0:
        raise RuntimeError("A must be positive.\
                                    Cannot compute orbit")
    psi_n = 0.0
    psi_up = 4 * np.pi**2
    psi_low = -4 * np.pi

    count = 0
    while count < 35:
        y_n = n_r0 + n_r + (A * (psi_n * c3(psi_n) - 1) /
                                    c2(psi_n) ** 0.5)

        if A > 0.0 and y_n < 0.0:
            # then readjust psi_low until y_n > 0.0
            pass

        x_n = np.sqrt(y_n / c2(psi_n))
        delta_t_n = (x_n ** 3 * c3(psi_n) + A * np.sqrt(y_n)) /
                                                np.sqrt(miu)
        if np.abs((delta_t_n - delta_t) / delta_t) < 1e-6:
            break
        else:
            count += 1
            if delta_t_n <= delta_t:
                psi_low = psi_n
            else:
                psi_up = psi_n

            psi_n = (psi_up + psi_low) / 2
    else:
        print('FALSE')

    f = 1 - y_n / n_r0
    g = A * np.sqrt(y_n / miu)
    g_dot = 1 - (y_n / n_r)

    v0 = (r - f * r0) / g
    v = (g_dot * r - r0) / g
    return v0, v
v0, v = universal_variables(r0, r, delta_t, t_m)
```

→ **Algorithm below is used as the main code for the remaining of plot result.**

3. **The main code in finding the Total** Δv;

```
from solver import izzo
import numpy as np
from hohmann import hohmann
from coe2rv import coe2rv
from numpy.linalg import norm

```

```python
import matplotlib
import matplotlib.pyplot as plt
matplotlib.rcParams["text.usetex"] = True

miu = 398600  # km^3 / s^2
ER = 6378
ER = 6378.1363  # Earth's radius (km)
MIU_E = 3.986004415e5  # km^3/s^2 (earth)
DU = ER  # Distance Unit in km
TU = np.sqrt(DU ** 3 / MIU_E)  # Time Unit in second
MIU = 1  # Normlaized GM

def semi_p(a, e):
    p = a * (1 - e ** 2)
    return p
"""
Where:
    e = Eccentricity, h = Altitude, a = Semimajor axis,
    p = semiparameter, inc = Inclination,
    Omega = Right ascension of the ascending node,
    omega = Argument of perigee, nu = True anomaly.
"""
# Initial orbit
e0 = 0
h0 = 220  # altitude (km)
a0 = (h0 + ER)
p0 = semi_p(a0, e0)
inc0 = 0
Omega0 = 0
omega0 = 0
nu0 = 0
r0, v0 = coe2rv(p0, e0, inc0, Omega0, omega0, nu0)

```

```python
# Final orbit
e1 = 0
h1 = 35786  # altitude (km)
a1 = (ER + h1)
p1 = semi_p(a1, e1)
inc1 = 0
Omega1 = 0
omega1 = 0
nu1 = 120
r, v = coe2rv(p1, e1, inc1, Omega1, omega1, nu1)

r1_ = norm(r0)
r2_ = norm(r)

DT = hohmann(r1_, r2_)
ts = np.linspace(0.2 * DT, 2 * DT, 50)  # second

def izzo_vary_tof(r0, r, ts, M=0):
    """
    Delta-V variation when time of flight vary
    Keyword Arguments:
    r0   -- initial position (RE)
    r    -- final position (RE)
    ts   -- array of time of flight (TUs)
    M    -- (default 0)
    """
    delta_vs = np.array([])
    for t in ts:
        ((v0_izzo, v_izzo),) = izzo(miu, r0, r, t=t, M=0,
                                    numiter=35, rtol=1e-8)
        Dv0 = norm(v0_izzo - v0)
        Dv1 = norm(v_izzo - v)
        Dv = Dv0 + Dv1
```

```python
            delta_vs = np.append(delta_vs, Dv)
        return delta_vs
# Enter variation of the simulation
```

There are several simulation taken in this thesis, starts from just a simple plot result (Figure 4.1) to the variation of initial altitudes. By putting the next list of codes into the code above (after the command of "# Enter variation of the simulation"), finding the remainder of the plot result is achievable. For each variation, readers could create a new file or just change the variation of the simulation in the same file.

- **Finding the plot result in Figure 4.1**;

```python
delta_vs = izzo_vary_tof(r0, r, ts, M=0)
fig, ax = plt.subplots(figsize=(10, 6))
ax.plot(ts / 3600, delta_vs, "o")
ax.annotate(r'Total $\Delta v_{min}=5.95136076$',
            xy=(3.56534567, 5.95136076), xytext=(5, 17),
            arrowprops=dict(facecolor='black', shrink=0.05), fontsize=17)
ax.annotate('or Hohmann Transfer',
            xy=(3.56534567, 5.95136076), xytext=(5, 16), fontsize=14)
ax.axvspan(3.5, 3.65, color='orange', alpha=0.5)
ax.set_xlabel("time of flight (hour)", fontsize=14)
ax.set_ylabel(r"Total $\Delta V$ (km/s)", fontsize=14)
ax.set_title(r"The Time of Flight VS $\Delta v$", fontsize=16)
ax.grid("both")
plt.savefig("delta_v_hoh.pdf", dpi=600)
plt.show()
```

- **Finding the plot result in Figure 4.5 (Initial Inclination variation)**;

```python
# Vary inclination only
fig, ax = plt.subplots(figsize=(9, 9))
inc0s = np.linspace(0, 180, 10)
for inc0 in inc0s:
    r0, v0 = coe2rv(p0, e0, inc0, Omega0, omega0, nu0)
    delta_vs = izzo_vary_tof(r0, r, ts, M=0)
    label = r"$i_0=$" + str(int(inc0))
    ax.plot(ts / 3600, delta_vs, "o", label=label)
ax.set_xlabel("tof (hour)")
ax.set_ylabel(r"Total $\Delta V$ (km/s)")
plt.grid("both")
plt.legend(loc="best", ncol=5, shadow=True, fancybox=True,
                                title=r"Inclination (deg)")
plt.savefig("izzo_inc0_vary.pdf", dpi=600)
plt.show()
```

- **Finding the plot result in Figure 4.6 (Initial RAAN variation);**

```
# Vary Omega only
fig, ax = plt.subplots(figsize=(9, 9))
Omega0s = np.linspace(0, 180, 10)
for Omega0 in Omega0s:
    r0, v0 = coe2rv(p0, e0, inc0, Omega0, omega0, nu0)
    delta_vs = izzo_vary_tof(r0, r, ts, M=0)
    label = r"$\Omega_0=$" + str(int(Omega0))
    ax.plot(ts / 3600, delta_vs, "o", label=label)
ax.set_xlabel("tof (hour)")
ax.set_ylabel(r"Total $\Delta V$ (km/s)")
plt.grid("both")
plt.legend(loc="best", ncol=5, shadow=True, fancybox=True,
                                title=r"$\Omega$ (deg)")
plt.savefig("izzo_Omeg0_vary.pdf", dpi=600)
plt.show()
```

- **Finding the plot result in Figure 4.7 (Initial Argument of Perigee variation);**

```
# Vary omega only
fig, ax = plt.subplots(figsize=(9, 9))
omega0s = np.linspace(0, 90, 10)
```

```python
for omega0 in omega0s:
    r0, v0 = coe2rv(p0, e0, inc0, Omega0, omega0, nu0)
    delta_vs = izzo_vary_tof(r0, r, ts, M=0)
    label = r"$\omega_0=$" + str(int(omega0))
    ax.plot(ts / 3600, delta_vs, "o", label=label)
ax.set_xlabel("tof (hour)")
ax.set_ylabel(r"Total $\Delta V$ (km/s)")
plt.grid("both")
plt.legend(loc="best", ncol=5, shadow=True, fancybox=True,
                                title=r"$\omega$ (deg)")
plt.savefig("izzo_omeg0_vary.pdf", dpi=600)
plt.show()
```

- **Finding the plot result in Figure 4.8 (Initial altitude variation)**;

```python
# Vary h only
Dvs220 = np.array(
    [
        18.2073288, 14.59455162, 12.05787424, 10.24392655,
        8.94191278, 8.01047726,  7.34692165, 6.87536006,
        6.54116087,  6.30630289, 6.14489581, 6.03938881,
        5.9777264,  5.95136076,  5.95391769, 5.9803353,
        6.02634382,  6.08818925,  6.16251882, 6.24635666,
        6.33711321,  6.43259262,  6.53098262, 6.6308257,
        6.73097761, 6.83056024,  6.9289149, 7.02555964,
        7.12015226,  7.21245948, 7.30233191, 7.38968411,
        7.47447885,  7.55671487,  7.63641742, 7.71363095,
        7.78841352, 7.86083254,  7.93096153, 7.99887763,
        8.0646598,  8.12838741,  8.19013927, 8.24999281,
        8.30802365, 8.36430515,  8.41890819,  8.47190107,
        8.52334935,  8.57331589,
    ])
fig, ax = plt.subplots(figsize=(9, 9))
h0s = np.linspace(220, 400, 10)
```

```python
for h0 in h0s:
    a0 = (ER + h0)
    p0 = semi_p(a0, e0)
    r0, v0 = coe2rv(p0, e0, inc0, Omega0, omega0, nu0)
    delta_vs = izzo_vary_tof(r0, r, ts, M=0)
    label = r"$h_0=$" + str(int(h0))
    total_delta_vs = delta_vs - Dvs220
    ax.plot(ts / 3600, total_delta_vs, "o", label=label)
ax.set_xlabel("tof (hour)")
ax.set_ylabel(r"Total $\Delta V$ - Total $\Delta V_{h0=220km}$\
                                                               (km/s)")
plt.grid("both")
plt.legend(loc="best", ncol=5, shadow=True, fancybox=True,
                                            title=r"$a$ (km)")
plt.savefig("izzo_h0_vary.pdf", dpi=600)
plt.show()
```

REMEMBER: To put all the code files both of Appendix A and B in the **same folder** for all the code in Appendix B could be run.

Curriculum Vitae

	Basic Information
Name	Maria Lucia
Place of Birth	Jakarta
Date of Birth	13 June 1997
Address	Gardenia Loka E-9 no. 11, Tangerang Selatan, 15324
Year	**Education**
2015 - present	International University Liaison Indonesia (IULI)
2012 - 2015	Candle Tree Senior High School
2009 - 2012	Ricci 2 Junior High School
2003 - 2009	Ricci 2 Elementary School
Year	**Seminars & Workshops**
2015	Modern Metro Systems for Environmental Friendly Megacities
2016	Tomorrow's Enterpreneurs Today
2017	Elevator & Angle of Attack (AOA)
Year	**Work Experiences**
2017	Internee in the Operation Control Centre at PT. citilink Indonesia
2019	Internee in the Research and Development at PT. Aerozeta Engineering