INTERNATIONAL UNIVERSITY LIAISON INDONESIA (IULI)

Preliminary Orbit Transfer Design from Low Earth Orbit to Geostationary Earth Orbit

Presented to the Faculty of
Engineering

In Partial Fulfilment
Of the Requirements for the Degree
Sarjana Teknik
In
Aviation Engineering

By
Jason NATHANAEL

July 23, 2020

“*Nothing takes place in the world whose meaning is not that of some maximum or minimum.*”

Leonhard Euler

INTERNATIONAL UNIVERSITY LIAISON INDONESIA (IULI)

Abstract

Faculty of Engineering
Department of Aviation Engineering

Sarjana Teknik

Preliminary Orbit Transfer Design from Low Earth Orbit to Geostationary Earth Orbit

by Jason NATHANAEL

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

STATEMENT BY THE AUTHOR

I hereby declare that this submission is my own work and to the best of my knowledge, it contains no material previously published or written by another person, nor material which to a substantial extent has been accepted for the award of any other degree or diploma at any educational institution, except where due acknowledgement is made in the thesis.

Jason Nathanael_____	______________________
Student	Date

APPROVAL PAGE

Preliminary Orbit Transfer Design from Low Earth Orbit to Geostationary Earth Orbit

Jason Nathanael

July 23, 2020

Faculty of Engineering

0 Triwanto Simanjuntak, PhD ____________
Advisor I, Department of Aviation Engineering Date

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

Acknowledgements

This thesis

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

CONTENTS

Abstract ii

STATEMENT BY THE AUTHOR iii

APPROVAL PAGE iv

Acknowledgements v

1 INTRODUCTION 1

- 1.1 General Statement of Problem Area 1
- 1.2 Research Purpose 1
- 1.3 Significance of the Study 2
- 1.4 Theoretical Perspective 2
- 1.5 Research Questions and Hypothesis 3
  - 1.5.1 Questions 3
  - 1.5.2 Hypothesis 3
- 1.6 Methodology 4
- 1.7 Design and Instrumentation 4
- 1.8 Data Analysis 4

2 LITERATURE REVIEW 6

- 2.1 Two Body Problem 6
  - 2.1.1 Introduction to Two Body Problem 6
  - 2.1.2 Newton's Law of Universal Gravitation 6
  - 2.1.3 Kepler's Laws 8
  - 2.1.4 Inertial Frame of Reference 9
  - 2.1.5 The Two Body Equation 10
- 2.2 Classical Orbital Elements 12
  - 2.2.1 Introduction to Parameterization and COE 12
  - 2.2.2 Semimajor Axis (a) 12
  - 2.2.3 Eccentricity (e) 13

2.2.4 Inclination (i) . . . 14
2.2.5 Longitude of Ascending Node (Ω) . . . 15
2.2.6 Argument of Periapsis (ω) . . . 15
2.2.7 True Anomaly (v) . . . 15
2.3 Conversion Algorithm Between Position and Velocity Vectors to Classical Orbital Elements . . . 15
2.3.1 From RV to COE . . . 15
2.3.2 From COE to RV . . . 17
2.4 Orbit Classification . . . 17
2.4.1 Low Earth Orbit . . . 18
2.4.2 Medium Earth Orbit . . . 19
2.4.3 Geosynchronous Orbit and Geostationary Orbit . . . 19
2.4.4 High Earth Orbit . . . 20
2.5 Orbital Maneuvering . . . 21
2.5.1 Introduction to Orbital Maneuvering . . . 21
2.5.2 Rocket Equation and Delta V . . . 22
2.5.3 Hohmaan Transfer . . . 22
2.5.4 Bi-elliptic Transfer . . . 23
2.5.5 Non-co-planar Maneuvers . . . 23
2.6 Satellite Orbit Definition and Two-Line Element (TLE) . . . 24
2.7 Perturbation to a Satellite's Orbit . . . 24
2.7.1 Atmospheric Drag Perturbation . . . 25
2.7.2 Oblateness Perturbation . . . 26
2.7.3 Third-Body Perturbation . . . 27
2.8 Lambert's Problem . . . 27
2.9 Developments of Spacecraft Trajectory Optimization . . . 31
2.9.1 Missions around Earth's Sphere of Influence . . . 32
Optimal LEO-GEO Intermediate Acceleration Orbit Transfer Using Nuclear Propulsion . . . 32
Minimum Fuel LEO Aeroassisted Orbit Transfer of Small Spacecraft with Inclination Change . . . 32
2.9.2 Deep Space Exploration . . . 32
Hayabusa-2 . . . 32
Juno . . . 33
Earth to Mars using Primer Vector . . . 35

3 RESEARCH METHODOLOGY **36**
3.1 Research Outline 36
3.2 Scientific Computing 37
3.2.1 Python Programming Language 37
3.2.2 Anaconda Platform 37
3.2.3 NumPy Package 37
3.2.4 SciPy Ecosystem 38
3.2.5 MatPlotLib Plotter 38
3.3 Building Math Toolbox - Converting from TLE into Orbital Elements 38
3.3.1 Introduction to LAPAN A2 38
3.3.2 Obtaining LAPAN A2 data from Celestrak 39
3.3.3 Conversion Regime 39
3.4 Building Math Toolbox - Conversion Between State Vectors and Orbital Elements 41
3.4.1 From Orbital Elements into State Vectors 42
3.4.2 From State Vectors into Orbital Elements 42
3.5 Building Math Toolbox - Orbit Propagation 44
3.5.1 Orbital Propagation using ODE Solver 44
3.5.2 Orbital Propagation in Consideration for J2 Perturbation 45
3.6 Building Math Toolbox - Orbit Maneuvering 45
3.6.1 Calculation of Orbit Change Using Hohmann Transfer 46
3.6.2 Calculation of Orbit Change Using Bi-elliptic Transfer 47
3.6.3 Non-co-planar Maneuver - Inclination Change Only 49
3.6.4 Non-co-planar Maneuver - Right Ascension of Ascending Node Change Only 49
3.6.5 Non-co-planar Maneuver - Both Changes 49
3.7 Building Math Toolbox - Lambert's Problem Solver 50
3.8 Mission Design - Selection of Suitable Launch Vehicle and Launch Site 53
3.9 Mission Design - Scenario One 53
3.9.1 Calculation of Maneuver from Initial Orbit (LEO) to GSO via Hohmann and Bi-Elliptic 53
3.9.2 Calculation of Maneuver from GSO to GEO via Non-Co-Planar Maneuver 54
3.10 Mission Design - Scenario Two 54
3.10.1 Calculation from Inclined Plane to Equatorial Plane 54
3.10.2 Calculation of Orbit Enlargement using Hohmann and Bi-elliptic Transfer 54
3.11 Mission Design - Scenario Three (via Lambert Solver) 55
3.12 Mission Analysis - Total Delta V Comparison 55
3.12.1 Propagation of Orbit With and Without J2 Perturbation 55

3.13 Mission Analysis - Possible Payload Calculation 55

4 RESULTS AND DISCUSSION **56**

4.1 Mathematical Tools Confirmation 56

4.1.1 Checking of the Conversion Regime 56

4.1.2 Propagation of LAPAN A2 Orbit With and Without J2 57

4.2 Mission Design - Launch Vehicle 61

4.2.1 Introduction to Epsilon Rocket 61

4.2.2 Epsilon Rocket Specification 63

4.3 Initial and Final Orbit Definition 65

4.3.1 Initial LEO (Parking Orbit) Definition 65

4.3.2 Final Geostationary Orbit Definition 65

4.4 Scenario One Result 66

4.5 Scenario Two Result 66

4.6 Scenario Three (Lambert Solver) Result 66

4.7 Comparison of Each Scenario 66

4.8 Payload Calculation 66

4.9 Final Discussion 66

5 CONCLUSIONS AND FUTURE WORKS **67**

5.1 Conclusion 67

5.2 Future Work 67

Bibliography **68**

LIST OF FIGURES

2.1 Visualization of the law of universal gravitation. 7
2.2 Visualization of Kepler's First Law. 8
2.3 Perihelion and Aphelion. Image credit: (Chris55, 2015) 9
2.4 A perfectly inertial, fixed-mass system, and the forces acting on the satellite. Image credit: (Vallado and McClain, 2013) 10
2.5 A perfectly inertial, two-body system. Image credit: (Vallado and McClain, 2013) 10
2.6 Cartesian Vectors of Position and Velocity. Image credit: (0.39, 2004) 13
2.7 Visualization of the six Classical Orbital Elements. Image credit: Lasunncty, 2007 14
2.8 Low Earth Orbit. Image credit: *Low Earth Orbit* 18
2.9 Galileo Satellite Constellation. Image credit: European Space Agency, 2020b . . 19
2.10 Geostationary Earth Orbit. Image credit: European Space Agency, 2020a 20
2.11 Earth's Lagrange Points. Image credit: *Catalog of Earth Satellite Orbits* 21
2.12 Hohmann Transfer, a maneuver using two burns. Image credit: Vallado and McClain, 2013 23
2.13 Bi-elliptic Transfer, two Hohmann transfer in succession. Image credit: Vallado and McClain, 2013 23
2.14 The Density of the Atmosphere Up to 1000 km According To US Standard Atmosphere. Image credit: Curtis, 2014 26
2.15 Effect of the Moon on a spacecraft as a third-body perturbation. Image credit: Curtis, 2014 27
2.16 Visualization of Lambert's Problem on a single orbit as a way for orbit determination. Image credit: Curtis, 2014 28
2.17 When finding the trajectory between two point on an orbit, there's the possible long way and the possible short way. Image credit: Vallado and McClain, 2013 28
2.18 Hayabusa-2 Trajectory to Ryugu. Image credit: Tsuda et al., 2013 33
2.19 Hayabusa-2 Spacecraft. Image credit: Tsuda et al., 2013 34
2.20 Juno's Trajectory to Jupiter. Image credit: Matousek, 2007 35
2.21 Juno Spacecraft, showing the extended solar cell array. Image credit: Matousek, 2007 35

3.1 3D Model of LAPAN A2. Image credit: Hardhienata, Triharjanto, and Mukhayadi, 2011 39

4.1 3D Orbit Visualization of LAPAN A2 (16th June 2020), With and Without J2 Perturbation - On A Single Graph 57
4.2 3D Orbit Visualization of LAPAN A2 (16th June 2020), With and Without J2 Perturbation - Side-by-Side Comparison 58
4.3 Classical Orbital Elements of LAPAN A2 (16th June 2020) - Without J2 Perturbation 58
4.4 Classical Orbital Elements of LAPAN A2 (16th June 2020) - With J2 Perturbation 59
4.5 Comparison of Right Ascension of Ascending Node and Argument of Perigee of LAPAN A2 (16th June 2020), With and Without J2 Perturbation 59
4.6 Top and Side View of LAPAN A2 Orbit (16th June 2020), With J2 Perturbation . 60
4.7 Diagram of Epsilon Launch Vehicle.) Image credit: Agency, 2018 62
4.8 Uchinoura Space Center, the Launch Site for Epsilon Launch Vehicle. Image credit: Agency, 2018 62
4.9 Launch of Epsilon Launch Vehicle from Uchinoura Space Center. Image credit: *Uchinoura Space Center* 63
4.10 Specification of Epsilon Launch Vehicle. Image credit: Agency, 2018 64
4.11 Accuracy of the Epsilon Launch Vehicle. Image credit: Agency, 2018 64

LIST OF TABLES

2.1 Elements inside a Two-Element Set data. 25

3.1 Two Line Element of LAPAN A2. 40

LIST OF ABBREVIATIONS

UAV	**U**nmanned **A**erial **V**ehicle
COG	**C**enter **O**f **G**ravity
CAD	**C**omputer **A**ided **D**esign
RPM	**R**otation **P**er **M**inute
ESC	**E**lectronic **S**peed **C**ontrol

Dedicated to my parents

CHAPTER 1

INTRODUCTION

1.1 General Statement of Problem Area

The Indonesian space agency, Lembaga Penerbangan dan Antariksa Nasional (LAPAN), has been focusing on building and deploying microsatellites for quite some time. The microsatellites were deployed to Low Earth Orbit (LEO), which is an orbit which has an altitude of fewer than 2000 kilometers from the Earth's surface. The choice of using microsatellites and launching them into Low Earth Orbit is driven primarily due to lower cost and lower development resources needed. Examples of such microsatellites are LAPAN-TUBSAT and LAPAN-A2. These microsatellites serve very important purposes in monitoring Indonesia's vast territory, from disaster mitigation to ship identification.

In the future, Indonesia has also expressed its intention to build its own satellites for communication services and deploy the satellites to Geostationary Earth Orbit (GEO). Geostationary Earth Orbit is a type of orbit where the satellite moves in accordance with earth rotation, which makes the satellite looks fixed in the sky when seen by a viewer on the ground. The altitude of Geostationary Earth Orbit is 35 786 km from Earth's surface, which is much higher than LEO.

Thus, it is planned to first deploy the satellites in LEO and then transferred to GEO through orbital maneuvers. In order to get to GEO via LEO using orbital maneuvers, the most optimal transfer trajectories are desired. The most optimal trajectory will save Delta V, which means how much change of velocity in the orbit can the satellite/spacecraft move by using its thrusters. As Delta V comes from the propellant that is brought by the satellite/spacecraft, by reducing the Delta V required, there will be saving in cost due to less propellant needed and more of the satellite mass is dedicated to actual useful payload instead of propellant.

1.2 Research Purpose

The objectives of this research are to investigate:

- Finding various scenarios of trajectory from Low Earth Orbit to Geostationary Earth Orbit
- Comparing the propellant cost of such trajectories
- Calculating the estimated mass that can be brought with a type of launch vehicle

1.3 Significance of the Study

The results of this research are expected:

- Can be used by LAPAN or other entity as the nominal trajectory for a future real mission to establish GEO satellite.
- Comparisons of various techniques to achieve Geostationary Orbit.
- Can be used to estimate the fuel budget, hence the total mass of the satellite.

1.4 Theoretical Perspective

Trajectory optimization is an act of finding the best way from one place to another, in this case, from Low Earth Orbit to Geostationary Earth Orbit. As the term "best" is subjective, the most common objective of trajectory optimization is to minimize the propellant required for the trajectory. Besides the minimal propellant required, the optimization also needs to consider the time required for the trajectory, as without a limit on time, the calculation simply trade time required with propellant required. A trajectory optimization problem is also a continuous problem, as the system is nonlinear and various events can occur during the trajectory, such as instantaneous velocity changes from the rocket engine, perturbation from other celestial objects, or various conditions that are not known explicitly during the onset of the launch.

A type of solution for the trajectory optimization problem is called Primer Vector Theory. Primer Vector Theory has its origins back in 1925, when Walter Hohmann, a German scientist, wrote a book describing a way to move from one circular orbit to another circular orbit. This transfer, called the Hohmann Transfer, is found to be one of the most fuel-efficient and optimal transfers from one circular orbit to another circular orbit in the simplest of cases.

Based on Hohmann's conjecture, using the calculus of variation, the Hohmann can be expanded further. A generalization of this calculus of variation is called the Optimal Control. In Optimal Control, it has a set of Necessary Conditions (NC) which must be fulfilled, and Sufficient Conditions (SC) if they are available. When the NC of a case is fulfilled, it will become the optimal solution.

By considering the spacecraft position vectors, velocity vectors, thrust vectors, and the gravitational vector, it can be derived using the calculus of variations a vector that shows the direction where the spacecraft should go in order to optimally transfer its orbit. This vector is called the primer vector, and the optimal thrust unit vector is in the direction of the primer vector.

As mentioned above, there are several cases of primer vector theory. The first case is for a constant specific impulse engine with a constant thrust, the second is for a constant specific impulse engine with an impulsive thrust, and the last one is for variable specific impulse engine. For the purpose of this research, this study focuses only on the CSI type of engine with impulsive thrust, meaning that the duration of thrust (engine burn) is very small compared with the time between burns, so the duration of burn can be considered as zero.

1.5 Research Questions and Hypothesis

1.5.1 Questions

1. Can primer vector theory be used to find the trajectory from Low Earth-Orbit to Geostationary Earth Orbit?

2. Is the trajectory solved according to primer vector theory more optimized than the trajectory found using typical Hohmann transfer or any other method?

3. How influential is the initial orbit parking provided by various launch services in regards to Delta V available?

4. For some common small orbital launcher systems (such as Electron by Rocket Lab), how much payload can be brought to Geostationary Earth Orbit using this method?

1.5.2 Hypothesis

1. By utilizing primer vector theory, an optimal transfer orbit from Low Earth Orbit to Geostationary Earth Orbit can be found.

2. Using Primer Vector Theory would result in a trajectory that requires less Delta V than Hohmann transfer.

3. The initial orbit parking will have an impact on conserving Delta V for orbital maneuvers.

4. The weight of usable payload to be brought to Geostationary Earth Orbit small orbital launcher services. E.g. Electron by Rocket Lab

1.6 Methodology

Steps that will be taken for this thesis are:

1. Determine the initial Low Earth Orbit parameters which the satellite initially orbits.
2. Determine the target Geostationary Earth Orbit parameters that the satellite wanted to transfer to.
3. Derive the equations for Primer Vector Theory.
4. Development of mathematical tools to solve the problem using Python programming language.
5. Validation of the results by comparing the solved result to results obtained using Hofmann transfer method, whether it is more optimized or not.
6. Determination of various candidate launch sites and their initial orbit parking. Calculating the amount of Delta V available for a base-line small orbital launch vehicle in order to reach the target Low Earth Orbit for each launch site.
7. Determining the amount of usable payload available for a small Orbital launch Vehicle (Electron by Orbital Lab), using the most optimal transfer orbit and the most optimal launch site gathered from previous steps. The amount is determined by calculating the Delta V left after final orbital insertion.

1.7 Design and Instrumentation

This research will focus on the trajectory design of a communication satellite which is initially in Low Earth Orbit (LEO) and wanted to move to the Geostationary Earth Orbit (GEO).

The research will be conducted purely by a numerical method which utilize scientific computational tools, specifically the Python programming language. A computer program made with Python will be used to calculate and solve the problem after the equations have been derived.

The research will also identify and analyze publicly available information for the determination of launch sites and specifications of the rocket.

1.8 Data Analysis

The data analysis for this research will be conducted numerically and also will be consulted with the publicly available research literature. The calculation process will be conducted using a program made with Python programming language at its related scientific libraries. The program written in Python programming language will be used for data analysis for the following parameters, but not limited to:

- Orbit propagator
- The transformation from state vectors to classical orbital elements and vice-versa
- Computations related to Primer Vector Theory
- Delta V for numerous orbit raising scenarios

CHAPTER 2

LITERATURE REVIEW

2.1 Two Body Problem

2.1.1 Introduction to Two Body Problem

One of the laying foundation in Astrodynamics is the two-body problem. The two-body problem is interested in the motion of two masses due to gravitational forces acted by each other. In a two-body problem, any other forces acting on the two-mass system, such is the presence of another body ("third-body") are neglected, and so the two bodies are also considered to be not colliding because any frictional or impact forces are not considered.

The basis of two-body problem is classical mechanics, and thus, the Newton's Law of Motion. Another law that forms the basis of two-body problem is Newton's Law of Universal Gravitation and can be further expanded using Kepler's Law.

2.1.2 Newton's Law of Universal Gravitation

Originally written by Newton in his book "Philosophiæ Naturalis Principia Mathematica" , the Newton's Law of Universal Gravitation states that:

Definition 1 *Every object with mass shall attract another mass, with a force that acts along a line that intersects with the two objects. The magnitude of the force is proportional to the product of the two objects, and shall be inversely proportional to their distance between them squared.*

In mathematical terms,

$$F = G\frac{m_1 m_2}{r^2} \tag{2.1}$$

where

- F is for the force between the masses. There are two forces, $F1$, which is the force from m_1 to m_2, and $F2$, which is the force from m_2 to m_1. Both have the same magnitude.

- G is the gravitational constant. The measured value of G in SI units, obtained through experiments, is $6.674 * 10^{-11} m^3 \cdot kg^{-1} \cdot s^{-2}$.
- m_1 is the first mass.
- m_2 is the second mass.
- r is the distance between the centers of first mass and second mass.

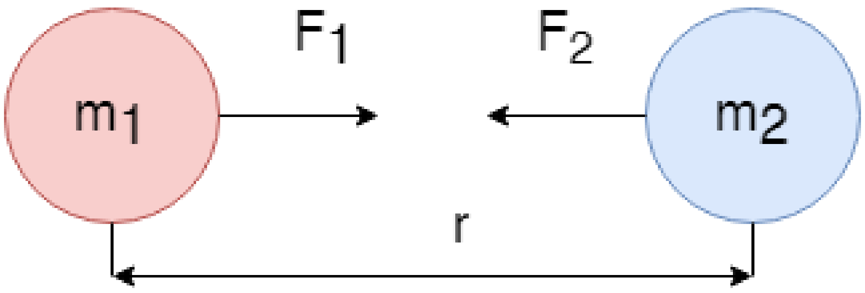


FIGURE 2.1: Visualization of the law of universal gravitation.

As shown the diagram, there exists two forces with opposing direction yet with the same magnitude. To account for this, the equation can be rewritten as a vector equation as the following:

$$\mathbf{F_{21}} = -G\frac{m_1 m_2}{|\mathbf{r_{12}}|^2}\mathbf{\hat{r}_{12}} \tag{2.2}$$

where

- F_{12} is the force acting on m_2 due to m_1
- G is the gravitational constant
- m_1 and m_2 is the mass of the two objects
- $|r_{12}|$ is the distance between the two masses
- $\hat{r}_{12}$ is the unit vector from m_1 to m_2

for the force acting on the other direction,

$$F_{12} = F_{21} \tag{2.3}$$

2.1.3 Kepler's Laws

Written by Johannes Kepler between 1609 and 1619 based on his observations of orbits of the planets, the Kepler's laws is a set of three laws that govern the motion of heavenly bodies, in this case the planets, around its center (the Sun). It is also found that the Kepler's Laws apply not only on planets, but may also apply to any orbiting body, such as a satellite orbiting the Earth. The Kepler's Laws are as follow:

Definition 2 ***Kepler's First Law:*** *Each planet orbits in an elliptical path, with the Sun in one of the foci.*

Expanded further, the elliptical path taken by an object can be any variation of a conic section, such as an ellipse, a hyperbola, a parabola, or a circle. For cases of hyperbola and parabola, because they are divergent, thus the satellites do not repeat their position and these orbits are called open orbits. For cases of circle and elliptical orbit, because the satellite repeat their position over time, are called closed orbits.

Through Newton's law of universal gravitation, it is known that the masses of the object affects the force of gravity between them. Taking this into account, the law can be expanded further to: the center of mass of the two object, called the barycenter, will be one of the foci. In case of a very large difference between masses of the objects, such as the Sun and its planes, or the Earth and an orbiting man-made satellite, the barycenter is located inside the heaviest object (such as the Sun).

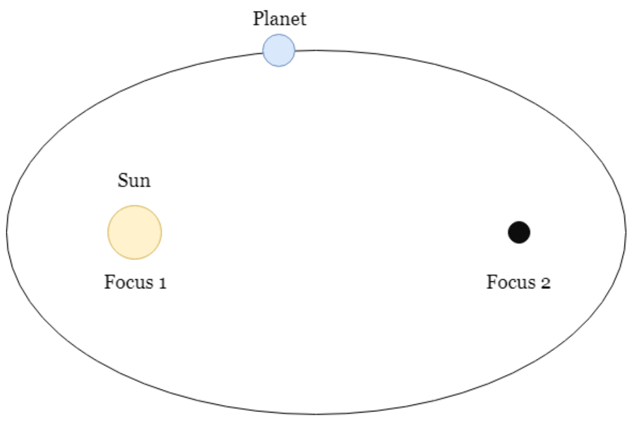


FIGURE 2.2: Visualization of Kepler's First Law.

Definition 3 ***Kepler's Second Law:*** *A straight line between the Sun and a planet shall sweep equal amount of area with equal amount of time.*

As an orbit is elliptical, there will be a point where the orbiter will be closest to the primary body/barycenter, and there wlll be point where the planet will be farthest. Such points are called $apoapsis$ for the farthest point and $periapsis$ for the closest point. If the primary body is the Sun, the points are called $aphelion$ and $perihelion$, while for orbits where the Earth is the primary body, these are called $apogee$ and $perigee$, respectively.

Due to the Kepler's Law, this means that the speed of an orbiter is not constant. It will be the slowest when it is in apoapsis, and will be the fastest when it is in periapsis.

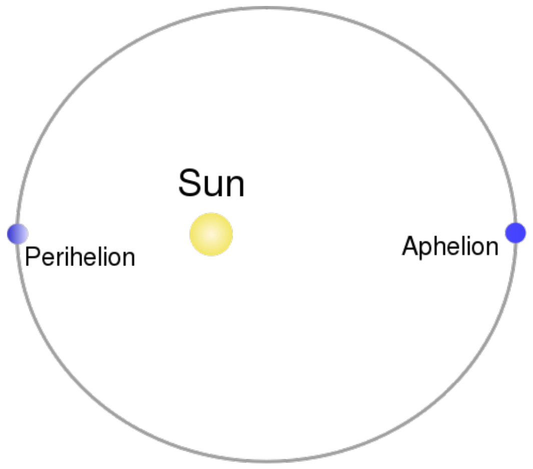


FIGURE 2.3: Perihelion and Aphelion. Image credit: (Chris55, 2015)

Definition 4 ***Kepler's Third Law:*** *The square of orbital period of a planet shall be proportional to the cube of its semimajor axis.*

Written mathematically,

$$T^2 = \frac{4\pi^2}{G(M_1 + M_2)}a^3 \tag{2.4}$$

Where

- T is the orbital period
- G is the gravitational constant
- M_1 and M_2 are masses of the body
- a is the semimajor axis

2.1.4 Inertial Frame of Reference

For analysis of a two-body problem, a coordinate system must first be defined. The definition is as follows: it is a perfectly inertial system, so it doesn't rotate nor accelerate. The bodies are considered as point masses. For a fixed-mass system, where we only consider the gravity from the point central body, the initial point of the coordinate system is located at the center of mass of the system.

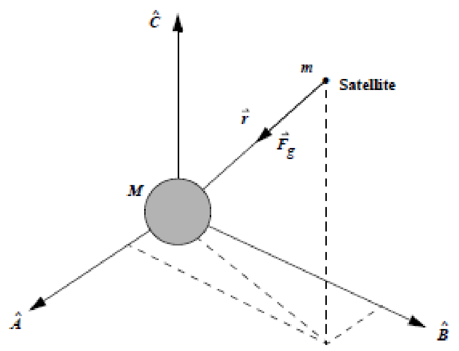


FIGURE 2.4: A perfectly inertial, fixed-mass system, and the forces acting on the satellite. Image credit: (Vallado and McClain, 2013)

For a two-body system on an inertial reference plane, taking into account the forces between the primary body and the orbiting body, the coordinate system can be made as follows:

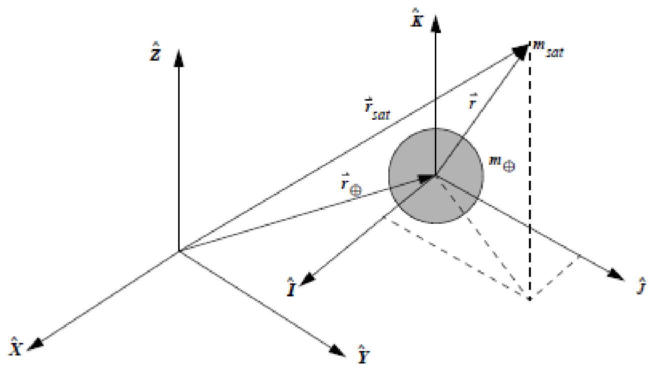


FIGURE 2.5: A perfectly inertial, two-body system. Image credit: (Vallado and McClain, 2013)

2.1.5 The Two Body Equation

Based upon the geometry in Fig. 2.5, we can derive the equation for a two-body system. The mass of the satellite is labeled as m_{sat} while the mass of the primary body is labeled as $m_{\oplus}$. At the same time, the position vector is labeled as $\hat{r}_{\oplus}$ and $\hat{r}_{sat}$. To analyze the forces on the system, Newton's law of universal gravitation is written as:

$$\hat{F}_g = -\frac{Gm_{\oplus}m_{sat}}{r^2}\frac{\hat{r}}{r} \tag{2.5}$$

and a vector from the primary body to the satellite is:

$$\hat{r}_{\oplus sat} = \hat{r}_{sat} - \hat{r}_{\oplus} \tag{2.6}$$

As we use a inertial coordinate system, by simply differentiating each of the vector component to its second derivatives we can found the acceleration of the satellite relative to the main body:

$$\ddot{\hat{r}}_{\oplus sat} = \ddot{\hat{r}}_{sat} - \ddot{\hat{r}}_{\oplus} \tag{2.7}$$

The Second Law of Newton states that, for a fixed-mass system:

$$\Sigma\hat{F} = \frac{d(m\hat{v})}{dt} = m\hat{a} \tag{2.8}$$

combined with the aforementioned law of universal gravitation, the force on the satellite can be written as:

$$\hat{F}_{g_{sat}} = m_{sat}\ddot{\hat{r}}_{sat} = -\frac{Gm_{\oplus}m_{sat}}{r^2}\frac{\hat{r}}{r} \tag{2.9}$$

and the force on the primary body can be written as:

$$\hat{F}_{g\oplus} = m_{\oplus}\ddot{\hat{r}}_{\oplus} = \frac{Gm_{\oplus}m_{sat}}{r^2}\frac{\hat{r}}{r} \tag{2.10}$$

As evident, the magnitude of the forces are the same but they are pointing in the opposite direction, as denoted by the negative sign on satellite's force and positive sign on the primary body's force.

Grouping the equation together and solving them together for $\ddot{\hat{r}}$ we can get:

$$\ddot{\hat{r}} = -\frac{Gm_{\oplus}}{r^2}\frac{\dot{r}}{r} - \frac{Gm_{sat}}{r^2}\frac{\dot{r}}{r} \tag{2.11}$$

or

$$\ddot{\hat{r}} = -\frac{G(m_{\oplus} + m_{sat})}{r^2}\frac{\hat{r}}{r} \tag{2.12}$$

This is the two-body equation.

In most cases, the difference of mass between the primary body and the orbiting body is very significant, with the orbiting body smaller by a few orders of magnitude. We can utilize the standard gravitational parameter μ, defined as the product of gravitational constant and mass, or:

$$\mu = G(m_1 m_2) \tag{2.13}$$

Because the mass of the primary body is many order of magnitude bigger than the orbiting body, we can write it as

$$\mu = G m_{\oplus} \tag{2.14}$$

As such, the two body-equation can be rewritten by ignoring the orbiting body's mass and replacing with μ :

$$\ddot{\hat{r}} = -\frac{\mu}{r^2}\frac{\hat{r}}{r} \tag{2.15}$$

Vallado and McClain, 2013

2.2 Classical Orbital Elements

2.2.1 Introduction to Parameterization and COE

A trajectory of orbiting body can be described in a Cartesian coordinate system using 2 vectors. The vectors are $position(\mathbf{r})$ and $velocity(\mathbf{v})$. Using the aforementioned inertial-frame, we can define that the origin of the coordinate system is the primary body as a center mass, and the frame of reference is not rotating. The direction of X axis is the vernal(March) equinox, the XY plane is the primary body's equatorial plane, while the Z axis coincides with the primary body's axis of rotation and pointing upwards. (Curtis, 2014)

In this frame of reference, in component form, the vector is given by:

$$\mathbf{r} = X\hat{\mathbf{I}} + Y\hat{\mathbf{J}} + Z\hat{\mathbf{K}}$$

$$\mathbf{v} = v_x\hat{\mathbf{I}} + v_y\hat{\mathbf{J}} + v_z\hat{\mathbf{K}}$$

While these two vectors can define an orbit, a more understandable orbit definition can be made. These two vectors can be parameterized into six classical orbital elements (COE). These six COEs can define an orbit in a more approachable way. The process of converting from $\mathbf{r}$ and $\mathbf{v}$ vector into orbital elements is called parameterization.

2.2.2 Semimajor Axis (a)

The semimajor axis defines how big the orbit is. It is the sum of r_a (radius of apoapsis) and r_p (radius of periapsis) divided by two. This is the average distance between the center of the

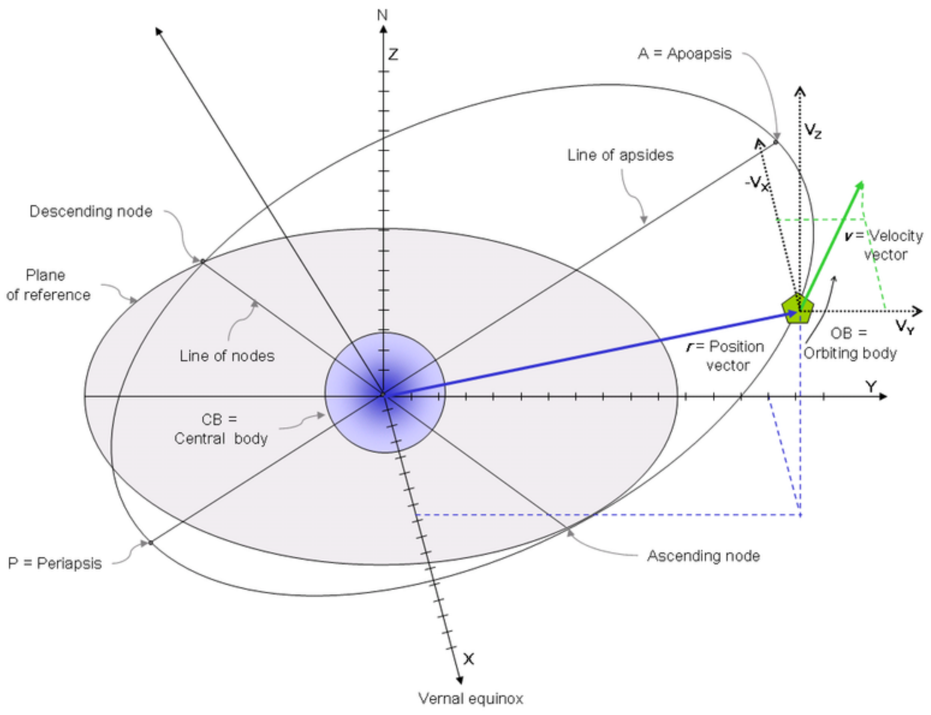


FIGURE 2.6: Cartesian Vectors of Position and Velocity. Image credit: (0.39, 2004)

bodies. (Federal Aviation Administration, 2018)

$$a = \frac{r_a + r_v}{2} \tag{2.16}$$

The semimajor axis can also be computed using the Vis-Viva equation, which describes the interaction between the two bodies by considering energy. Computing the semimajor axis using Vis-Visa equation is as follows (Vallado and McClain, 2013)

$$a = (\frac{2}{r} - \frac{v^2}{\mu})^{-1} \tag{2.17}$$

2.2.3 Eccentricity (e)

Eccentricity defines the shape of the orbit. It ranges from zero to one, higher value means the orbit is more elongated, i.e., a perfectly circular orbit would have an eccentricity of zero, while a parabolic orbit would have a value of eccentricity approaching infinity. Eccentricity can be computed as the ratio of the distance between the two foci to the length of major axis (twice the semimajor axis). However, this only applies to elliptical orbits (Federal Aviation

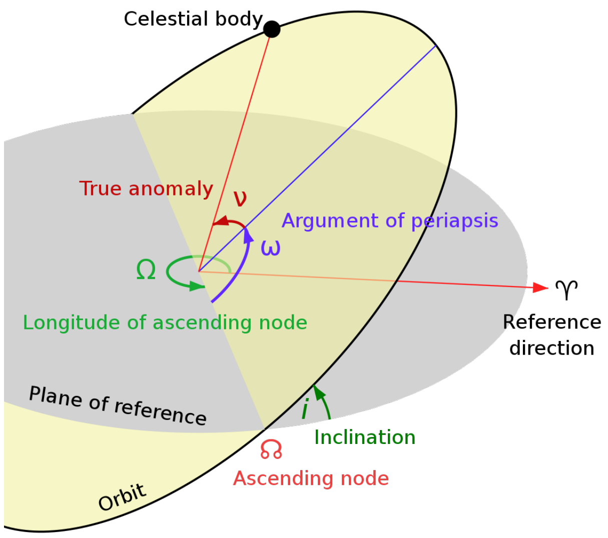


FIGURE 2.7: Visualization of the six Classical Orbital Elements. Image credit: Lasunncty, 2007

Administration, 2018) :

$$e = \frac{2c}{2a} \tag{2.18}$$

To consider for all conic section, it can be defined as (Vallado and McClain, 2013):

$$a = \frac{a - r_p}{a} = \frac{r_a - a}{a} \tag{2.19}$$

2.2.4 Inclination (i)

Inclination defines the tilt of the orbit. The angle is measured from the equatorial plane of the primary body. (**federal_aviation_administation_describing_2018**) Mathematically, it is the angle between the unit vector $\hat{K}$ to the angular momentum $\hat{h}$ The range is from $0\,\text{deg}$ to $180deg$ - orbit that are inclined at $0deg$ and $180deg$ are called equatorial orbits, while other orbits are called inclined orbits. Because of the measurement of the angle, Inclination also defines the direction the satellite is rotating. Angles from $0deg$ to $90deg$ rotates with the primary body - these are called prograde orbits. Angles from $90deg$ to $180deg$ rotates opposing the primary body - called retrograde orbits. (Vallado and McClain, 2013). An orbit at exactly $90deg$ are called polar orbits, because they pass through the poles. Inclination can be written mathematically as:

$$COS(i) = \frac{\hat{K} \cdot \hat{h}}{|\hat{K}| \cdot |\hat{h}|} \tag{2.20}$$

2.2.5 Longitude of Ascending Node (Ω)

Longitude of Ascending Node describes the orientation of the orbit to a reference direction. The reference is the $\hat{I}$ unit vector, and the angle is measured eastward from the unit vector. When the satellite passes through the equatorial plane from the south to north, the point is called ascending node. Likewise, when the satellite passes through from north to south, the point is called descending node. For equatorial orbits, the Longitude of Ascending Node is undefined. The range of Ω is $0deg$ to $180deg$.

2.2.6 Argument of Periapsis (ω)

Argument of Periapsis describes the location of the periapsis. It is the angle measured from the ascending node, in direction with the sateliite's motion, to the periapsis. Perfectly circular or equatorial orbit do not have argument of periapsis because they don't have periapsis or ascending node. The range of ω is $0deg$ to $180deg$.

2.2.7 True Anomaly (v)

True anomaly describes the location of the satellite at that instant. It is measured from the periapsis. As perfectly circular orbits do not have periapsis, it is undefined for perfectly circular orbit.

2.3 Conversion Algorithm Between Position and Velocity Vectors to Classical Orbital Elements

Defining an orbit using state vectors or classical orbital elements have their own benefits. An orbit defined using state vectors are easier to be analyzed numerically, however it is not really visualizable. Likewise, an orbit defined using classical orbital elements are much more easier to be imagined. As such, conversion between these two are common. These conversion between the two are as follows:

2.3.1 From RV to COE

To convert from state vectors to classical orbital elements, we need some intermediate vectors. The first one is the angular-momentum vector, defined as:

$$\hat{h} = \hat{r} \times \hat{v} \tag{2.21}$$

The second intermediate vector is the node vector. This is the vector pointing to the node. A magnitude of zero indicates that the orbit is equatorial.

$$\hat{n} = \hat{K} \times \hat{h} \tag{2.22}$$

Then, we can start computing the classical orbital elements. The first item to be determined is the eccentricity. Equation for eccentricity based upon vectors is:

$$\hat{e} = \frac{(v^2 - \frac{\mu}{r})\hat{r} - (\hat{r} \cdot \hat{v})\hat{v}}{\mu} \tag{2.23}$$

Then, compute the specific mechanical energy. This equation is valid for any orbit:

$$\xi = \frac{v^2}{2} - \frac{\mu}{r} \tag{2.24}$$

We then have two orbit possibilities. The first one is parabolic, with eccentricity of 1.0. The second one is non-parabolic orbit, with the value of eccentricity anything but 1.0.

For non-parabolic orbit, we can determine the semimajor axis as:

$$a = -\frac{\mu}{2\xi} \tag{2.25}$$

and the semiparameter is defined as:

$$p = a(1 - e^2) \tag{2.26}$$

For parabolic orbits (e = 1.0), the semimajor axis is infinite (∞) and the semiparameter is:

$$p = \frac{h^2}{\mu} \tag{2.27}$$

We then compute the angles, starting from the inclination (i), given by

$$\cos(i) = \frac{h_K}{|\hat{h}|} \tag{2.28}$$

For longitude of ascending node, (Ω), given by:

$$\cos(\Omega) = \frac{n_I}{|\hat{n}|} \qquad IF(n_J < 0)\ THEN \Omega = 360deg - \Omega \tag{2.29}$$

For argument of periapsis, (ω), given by:

$$\cos(\omega) = \frac{\hat{n} \cdot \hat{e}}{|\hat{n}| \cdot |\hat{e}|} \qquad IF(e_K) < 0\ THEN \omega = 360deg - \omega \tag{2.30}$$

For true anomaly, (v), given by

$$\cos(v) = \frac{\hat{r} \cdot \hat{r}}{|\hat{e}||\hat{r}|} \qquad IF(\hat{r} \cdot \hat{v} < 0)\ THENv = 360deg - v \tag{2.31}$$

2.3.2 From COE to RV

This algorithm is the reverse of the previous one - it transforms from classical orbital elements into state vectors. First of all, to account for all types of orbits, semiparameter is used instead of semimajor axis. The formula for semiparameter is as follows:

$$p = a(1 - e^2) \tag{2.32}$$

We then consider a new coordinate system. The coordinate system is centered in the orbit's focus, and the orbital plane lay flat in the coordinate system. This coordinate system is called as Perifocal coordinate system. To find the r in perifocal coordinate system, it is defined as:

$$\hat{r}_{PQW} = \begin{bmatrix} \frac{p\cos(v)}{1+e\cos(v)} \\ \frac{p\sin(v)}{1+e\cos(v)} \\ 0 \end{bmatrix} \tag{2.33}$$

Then the velocity vector in perifocal coordinate system is defined as:

$$\hat{v}_{PQW} = \begin{bmatrix} -\sqrt{\frac{\mu}{p}}\sin(v) \\ \sqrt{\frac{\mu}{p}}(e + \cos(v)) \\ 0 \end{bmatrix} \tag{2.34}$$

The last step is to convert from perifocal coordinate system to geocentric equatorial system, by utilizing rotation matrices:

$$\hat{r}_IJK = [ROT3(-\Omega)][ROT1(-i)][ROT3(-\omega)]\hat{r}_PQW \tag{2.35}$$

$$\hat{v}_IJK = [ROT3(-\Omega)][ROT1(-i)][ROT3(-\omega)]\hat{v}_PQW \tag{2.36}$$

2.4 Orbit Classification

All satellites orbiting the Earth are not located nor using the exact same orbit. Each satellite has their own trajectory and orbit, which alters their behavior around the Earth. As evident from the Kepler's Law, a lower altitude satellite will go around the Earth way faster than a satellite orbiting higher, for example. Another example is a satellite orbiting with high inclination will cover more ground area than a satellite with low inclination. These various types of orbits are used accordingly to each individual satellite's mission objectives. (*Catalog of Earth Satellite Orbits*)

To aid in identification of orbit types, orbits can be classified according to several parameters: their altitude, the body they are orbiting, inclination, and their eccentricity. For the purposes of this research, which focuses on Earth-orbiting satellites, the classification of interest is orbit classification through their altitude.

2.4.1 Low Earth Orbit

Low Earth Orbits are orbits located starting from the boundary of Earth's atmosphere to space (100 km, the Kármán line) until 1000 km from the planet's surface. It's versatility and ease of access (due to its lower altitude) makes LEO one of the most commonly used orbit. Due to it's close proximity, it is used for imaging satellites to create high resolution images. Space stations, such as the International Space Station, is also located at LEO because its easier for astronauts to travel back and forth. As the satellite in LEO orbits faster than in the higher orbit (a LEO satellite takes around 90 minutes to orbit the Earth), this orbit is also useful for missions that require fast Earth coverage. Lesser constrains also makes the Low Earth Orbit as the one of the most preferred orbit for Earth monitoring satellites. (*Low Earth Orbit*)

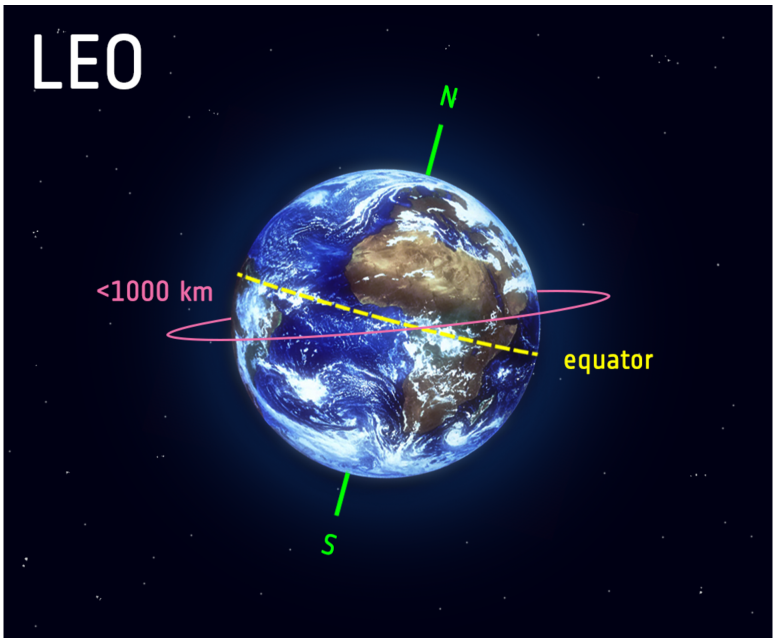


FIGURE 2.8: Low Earth Orbit. Image credit: *Low Earth Orbit*

However, Low Earth Orbit might not be as useful for communication satellites. Their high velocity makes ground tracking harder, and also the make the satellite appear for a briefer time according to a ground observer. The allure for communication satellite in Low Earth Orbit is not lost, however, as their lower altitude allows for shorter latency. A way to circumvent this weakness is by creating a constellation of hundreds or even thousands of satellites, which is a method currently used by Starlink, a global broadband network currently being built by US-based space start-up SpaceX. (Caleb Henry, 2020)

2.4.2 Medium Earth Orbit

Medium Earth Orbit are classification of orbits located between Low Earth Orbit (1000 km) until below Geosynchronous Orbit (35 786 km). It is used for various purposes, but the most prominent one are for navigation satellites (Global Navigation Satellite System). 3 major GNSS are located entirely within Medium Earth Orbit: Global Positioning System (US), Galileo (EU), and GLONASS (Russia). These navigation satellites provide services that blankets the entire Earth. European Space Agency, 2020b

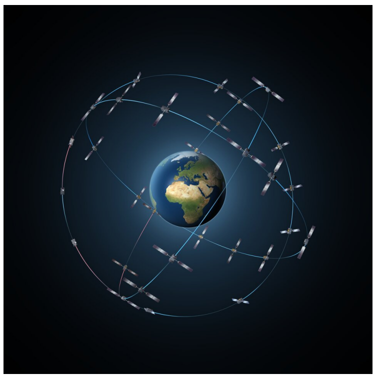


FIGURE 2.9: Galileo Satellite Constellation. Image credit: European Space Agency, 2020b

2.4.3 Geosynchronous Orbit and Geostationary Orbit

As consequence of Kepler's Law, there exist an altitude where the speed of the orbiting satellite matches the rotational speed of Earth. Therefore, according to a ground observer, the satellite appear fixed to the sky. This orbit is called Geosynchronous Orbit and its location is exactly 35 786 km above sea level. The satellite may drift to the North nor the South, but it will always appear at the same longitude. (*Catalog of Earth Satellite Orbits*)

If the orbit is circular and located directly above the equator, it will stay exactly at the same place, never drifting to the south nor north. It will always stay over that single location. This case is called a Geostationary Orbit. This ”fixed” location consequently enable various mission profiles. As it is always fixed, it is extremely useful for weather satellites, which requires constant monitoring of the same ground area to monitor cloud, water vapor, and wind movement. They are also useful for home satellite television, which enables the antenna to stay fixed and no need for active tracking. Some communication satellite also orbits in Geostationary Orbit. One last advantage is that because Geostationary orbit is quite high, satellite on this orbit can cover much more surface area than a satellite in lower orbit, lowering the number of satellite needed for a constellation with huge coverage. European Space Agency, 2020b

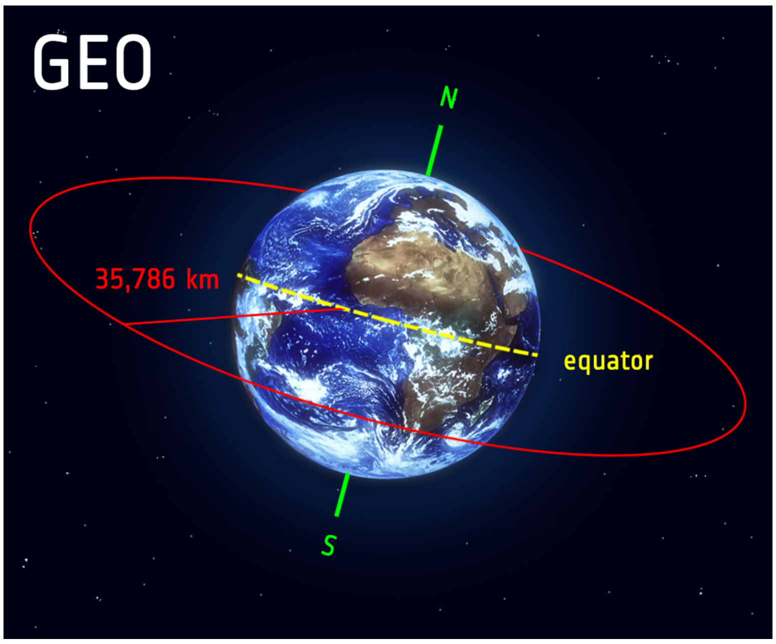


FIGURE 2.10: Geostationary Earth Orbit. Image credit: European Space Agency, 2020a

2.4.4 High Earth Orbit

High Earth Orbit revers to any orbit higher than 35 768 km that is still within the Earth’s sphere of influence. These orbits are very niche and rarely used, not to mention launching satellites to this altitude is quite difficult. A special type of orbit however, exist in high earth orbits. As the Sun and Earth exert force to each other, there exist points where the forces cancel out. These points are called Lagrange Points, and satellites placed in this orbit will move together with the Earth around the sun, fixed in place in Sun-Earth reference system. 5 Lagrange points exist, however only two are stable, called L4 and L5, while the rest requires constant maintenance. Example of satellites in Lagrange points are WMAP (a background microwave radiation mapper) and the future James Webb Space Telescope, an infrared telescope. (*Catalog of Earth Satellite Orbits*)

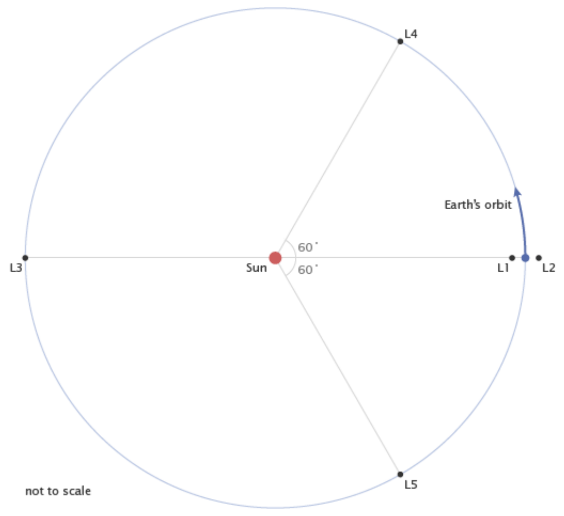


FIGURE 2.11: Earth's Lagrange Points. Image credit: *Catalog of Earth Satellite Orbits*

2.5 Orbital Maneuvering

2.5.1 Introduction to Orbital Maneuvering

To be useful, a satellite needs to be put on an exact orbit that facilitates its mission objectives. This orbit may not be achievable during launch, the satellite needs additional tuning, or even the desired location, such as orbiting other heavenly bodies, are straight up not achievable in a single orbit. Therefore, the satellite needs to move around after orbiting. This movement from one orbit to another orbit is called orbital maneuvering. Orbital maneuvering covers all changes done to a spacecraft's orbit after it has gone into an orbit. (Vallado and McClain, 2013)

In general, orbital maneuvering can be classified into 3 groups: planar, co-planar, and fixed delta-V maneuver. These maneuvers are achieved by doing burns, adding or subtracting a spacecraft's velocity using its propulsion. These burns can be considered instantaneous or continuous, for example in low-thrust burns. Force applied in the orbit plane can change eccentricity, semimajor axis, and argument of periapsis, while burns normal to the plane can change inclination and right ascension of the ascending node (RAAN). The most common goal of orbital maneuvering is to minimize fuel required (delta-V), but sometimes time is also critical. (Vallado and McClain, 2013)

2.5.2 Rocket Equation and Delta V

To move around in vacuum, almost all propulsion system rely on the concept of momentum. Mass, typically gas, is spewed out of a nozzle and then due to Newton's 3rd Law's, the system got pushed in the opposite direction. This concept can be applied in various manners: in a chemical rocket, fuel is mixed with an oxidizer and then ignited. This is the origin of the word "burn", describing the act of igniting this mixture to achieve change in velocity. The equation governing this is called the Rocket Equation, also called Tsiolkovsky's Equation, after Konstantin Tsiolkovsky, a Russian scientist who first wrote this equation in 1903. (Pettit, 2012)

The rocket equation is as follows:(*Ideal Rocket Equation*)

$$\Delta V = I_{sp} g_0 ln \frac{m_0}{m_f} \tag{2.37}$$

where

- ΔV is the change of velocity of the vehicle
- I_{sp} is specific impulse
- g_0 is gravity acceleration
- m_0 is the initial mass with propellant (wet mass)
- m_f is the final mass after burn (dry mass)

Delta-V can also be described as a form of energy. When denoting where the spacecraft wants to go, the energy required to go there is denoted in form of "Delta-V required". This number is then compared to the number of Delta-V available to the spacecraft. If the number of delta-v available is higher than the number of delta-v required, the spacecraft can accomplish the maneuver, but if it isn't, then the maneuver is not possible. Consequently, mission planner try to reduce the amount of delta-v required through various means. (Pettit, 2012)

2.5.3 Hohmaan Transfer

Hohmann transfer is a type of co-planar maneuver, meaning that the initial orbit and the final orbit lies in the final plane, so changes can only be observed on the orbit's size, shape, and argument of perigee. Hohmann transfer is named after its inventor, Walter Hohmann. In this transfer, there are two tangential burns, and the flight path angle must be exactly zero. The initial or the final orbit can be elliptical or circular, and the transfer orbit between those two are always elliptical or circular, never parabolic or hyperbolic, as a requirement for the second tangential burn. (Vallado and McClain, 2013)

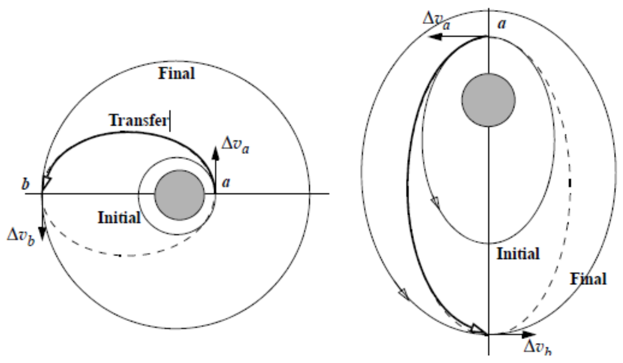


FIGURE 2.12: Hohmann Transfer, a maneuver using two burns. Image credit: Vallado and McClain, 2013

2.5.4 Bi-elliptic Transfer

A bi-elliptical transfer is a variant of Hohmann transfer, where two Hohmann transfer are performed in succession. The spacecraft is first put into an initial transfer orbit, then after a tangential burn, put into a second transfer orbit, and then put into the final orbit. As with Hohmann transfer, the flight path angle during the burn must also be exactly zero. (Vallado and McClain, 2013)

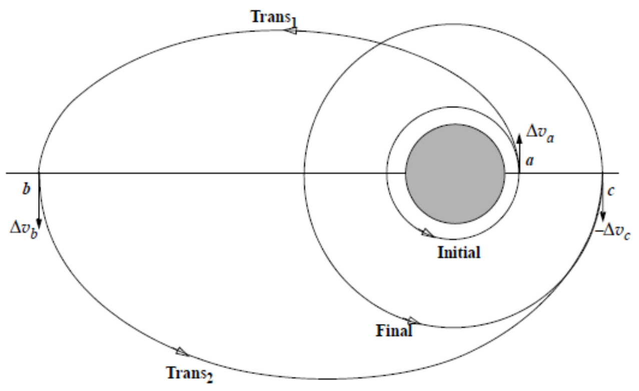


FIGURE 2.13: Bi-elliptic Transfer, two Hohmann transfer in succession. Image credit: Vallado and McClain, 2013

2.5.5 Non-co-planar Maneuvers

To change an orbit inclination and/or right ascension of the ascending node, we will need non-coplanar maneuvers. A non-co-planar maneuver is a burn that is applied out of orbital plane. There are 3 possibilities of a non-co-planer burn: a change in inclination, a change in right ascension of the ascending node, or a change of both. A non-co-planar maneuver is necessary in real life because the location of the launch site dictates where an orbit will be. Considering satellites need to be launched from a dedicated facility, the location of that facility will most likely put the satellite in an undesirable location. Therefore, a non-co-planar burn is necessary to fix this issue. Another issue why non-co-planar burn is needed is when there are timing constrains regarding the satellite's launch. (Vallado and McClain, 2013)

2.6 Satellite Orbit Definition and Two-Line Element (TLE)

Once a satellite is in orbit, it is important to keep track of its location and its current orbit. Without the ability to track it or the ability to correctly describe its orbit, a satellite capability is much less reduced. To describe a satellite's orbit, the classical orbital elements is one of methods available. Still, a physical tracking of the object must first be done.

One of the most common way to know a satellite's orbit and location is using NORAD's Two Line Element sets. The Joint Space Operations Center (JSPOC), operated by the United States' Air Force Space Command, observe and track each individual object currently orbiting the Earth. After some processing, the data is then distributed to the public for free in two-line format - hence the name. (Vallado and Cefola, 2012)

Based upon data from the TLE, the satellite's exact orbit and location can be determined. (Transilvania University of Braşov, Braşov, Romania et al., 2016) An example TLE data is as follows:

```
ISS (ZARYA)
1 25544U 98067A   20182.51943373  .00000997  00000-0  25859-4 0  9996
2 25544  51.6454 282.4729 0002513 101.4450   8.1574 15.49473510234088
```

2.7 Perturbation to a Satellite's Orbit

The initial two-body equation is written assuming that the interaction between the body is only due to the gravitational forces between each other. However, in the real world this is not the case. When there is any additional forces that disturb this equation exist, the deviation is called a perturbation. Accounting for perturbation, the two body equation becomes: (Curtis, 2014)

$$\ddot{\hat{r}} = -\frac{G(m_{\oplus} + m_{sat})}{r^2}\frac{\hat{r}}{r} + p \tag{2.38}$$

where P is a vector that accounts for all perturbation beside the gravitational force between two body. Three particular perturbation of interest are shown below.

TABLE 2.1: Elements inside a Two-Element Set data.

Line 0		
Column	Example	Description
1-24	ISS (ZARYA)	Name of the tracked object
Line 1		
Column	Example	Description
1	1	Line Number
3-7	25544	Satellite Catalog Number
8	U	Satellite Classification, U for Unclassifed
10-17	98067A	International Designator
19-32	20182.51943373	Element Set Epoch (UTC)
34-43	.00000997	1st Derivative of Mean Motion with Respect to Time
45-52	00000-0	2nd Derivative of Mean Motion with Respect to Time, Decimal Point Assumed
54-61	25859-4	B* Drag Term
63	0	Element Set Type
65-68	999	Element Number
69	6	Checksum
Line 2		
Column	Example	Description
1	2	Line Number
3-7	25544	Satellite Catalog Number
9-16	51.6454	Orbit Inclination (degrees)
18-25	282.4729	Right Ascension of Ascending Node (degress)
27-33	0002513	Eccentricity (decimal point assumed)
35-42	101.4450	Argument of Perigee (degrees)
44-51	8.1574	Mean Anomaly (degrees)
53-63	15.49473510	Mean motion (revolutions/day)
64-68	23408	Revolution Number at Epoch
69	8	Checksum

2.7.1 Atmospheric Drag Perturbation

Atmosphere Drag perturbation is one of the most evident perturbation on a spacecraft trajectory. Despite that 99 percent of all atmosphere is below 100 km (called the Karman Line, the recognized boundary between Earth and space), however, due to the high speed of the spacecraft, even minuscule amount of air can be significant. The drag will slow down the spacecraft, eventually lowering down the spacecraft and make it de-orbit. (Curtis, 2014)

As atmosphere decreases with altitude, this effect is most pronounced in Low Earth Orbit. To combat the decreasing altitude due to drag, a spacecraft must be equipped with a sort of thruster to combat its altitude loss. This is called station keeping. Without station keeping, a satellite in Low Earth Orbit will eventually de-orbit. An example of this is the Hubble Space

Telescope, which was last boosted to its orbit in 2009, and will eventually de-orbit in 2025. Curtis, 2014

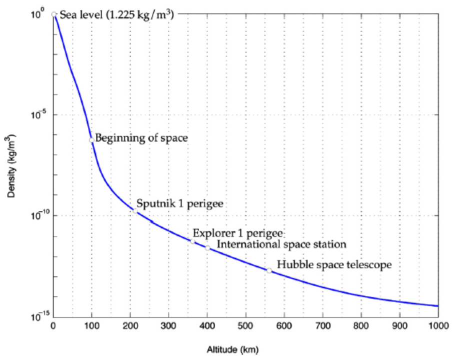


FIGURE 2.14: The Density of the Atmosphere Up to 1000 km According To US Standard Atmosphere. Image credit: Curtis, 2014

Based upon the drag equation:

$$D = \frac{1}{2}\rho {v^2}_{rel} C_D A \tag{2.39}$$

the acceleration p for the calculation can be written as

$$\mathbf{p} = -\frac{1}{2}\rho v_{rel}(\frac{C_D A}{m})v_{rel} \tag{2.40}$$

2.7.2 Oblateness Perturbation

While in many cases the Earth is considered a sphere, it is, however, not a truly spherical. It is actually an oblate spheroid - a sphere with a slight bump along the equator. Due to this, the gravity of Earth is not exactly uniform, but rather changes with latitude and radius. The Earth mass density is also not uniform, which makes its gravity varies.

This oblateness have an effect on the satellite orbit. When written, the perturbation Φ is given by invite series:

$$\Phi(r, \phi) = \frac{\mu}{r}\sum_{k=1}^{\infty} J_k(\frac{R}{r})_k P_k(\cos\phi) \tag{2.41}$$

J_k in the equation is the observed zonal harmonics - a dimensionaless number that is unique to each planetary body. As it is a infinite series, the J number extends to infinity, but the most significant one is J_2, where the value is $J_2 = 0.00108263$. (Curtis, 2014)

Perturbation acceleration p due to J2 perturbation can be written as:

$$\mathbf{p} = \frac{3}{2}\frac{J_2\mu R^2}{r^4}[\frac{x}{r}(5\frac{z^2}{r^2} - 1)\hat{i} + \frac{y}{r}(5\frac{z^2}{r^2} - 1)\hat{j} + \frac{z}{r}(5\frac{z^2}{r^2} - 3)\hat{k}] \tag{2.42}$$

2.7.3 Third-Body Perturbation

In space, there are various heavenly bodies which can affect the trajectory in various ways. When the effect of a third body's gravity is considered, is it called a three-body problem. For a satellite within the Earth's sphere of influence, one heavenly body with a noticeable effect is the Earth's moon. The Sun can also be considered as another third-body perturbation.

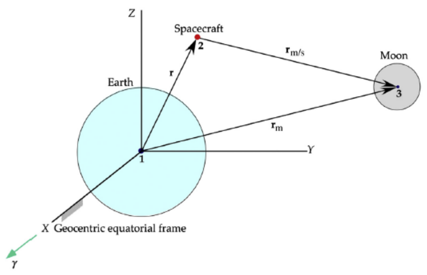


FIGURE 2.15: Effect of the Moon on a spacecraft as a third-body perturbation. Image credit: Curtis, 2014

2.8 Lambert's Problem

In the 18th Century, J. H. Lambert, an astronomer, had a problem determining an orbit. If two position vector and the time of flight of them is known, how would the trajectory would be? According to him, the time of flight between point $P1$ and $P2$ should be independent of the eccentricity of the orbit. Rather, it depends on the following factors: the summation of the position vectors ($r1$,$r2$), the length of the semi-major axis a and the distance of c, the straight line between point $P1$ and $P2$. (Curtis, 2014)

This problem of determining the trajectory between these two point is therefore known as Lambert's Problem. It is also known as an orbital boundary value problem. Due to its importance in the world of astrodynamics, as orbit determination is crucial, it is frequently researched. (Izzo, 2015)

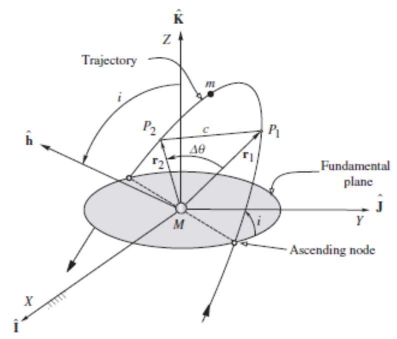


FIGURE 2.16: Visualization of Lambert's Problem on a single orbit as a way for orbit determination. Image credit: Curtis, 2014

Although devised to find a single orbit between two point, (orbit determination), the Lambert Problem can also be considered as a transfer problem between two orbits. This is due to the constrains on the Lambert problem, which are the position vectors, semimajor axis, and chord distance. Thus, by solving Lambert's problem, it can also be used to find a way (trajectory) to move from one orbit to other orbit. (Vallado and McClain, 2013)

In case of finding an orbit between two location, there are two possible direction to reach the same point: the short way ($\Delta v < 180deg$)) or the long way ($\Delta v > 180deg$). If the orbit is considered as taking the shorter way, the orbit has a value of transfer method of $t_m = +1$, while moving in the opposite direction has a value of transfer method $t_m = -1$. If the transfer method is known, the Lambert's Problem has one unique solution. (Vallado and McClain, 2013)

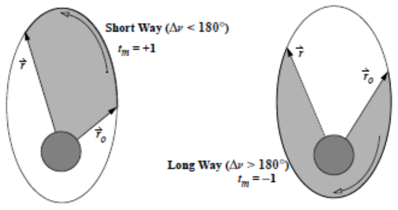


FIGURE 2.17: When finding the trajectory between two point on an orbit, there's the possible long way and the possible short way. Image credit: Vallado and McClain, 2013

As Lambert's Problem is frequently researched, there have been multiple ways to solve it. Vallado (Vallado and McClain, 2013) listed multiple possible algorithms to solve Lambert's problem, which are:

- **Lambert-Minimum Energy**: The Lambert-Minimum Energy algorithm is based upon geometrical analysis of the problem and solved using geometric principles. This solution by Lambert is applied best for orbit determination and also only account for a single revolution between point $P1$ and $P2$. This Lambert-Minimum Energy algorithm also only applies on elliptical transfer.

- **Lambert-Gauss** : Gauss' proposal to solve Lambert's problem utilize the area of triangles formed as the satellite sweeps across its orbit. This method works best only when the position of the vectors are close apart, when the location of the vectors are far enough, the Lambert-Gauss solution won't converge into an answer.
- **Lambert-Thorne** :This is one of the most modern solution to Lambert's problem. It is applicable for any time difference between point and besides working on elliptical orbit, also works on hyperbolic orbit. Lambert-Thorne solution focuses on the transfer time between the two orbit. As it is applicable to a wide array of orbits, the Lambert-Thorne solution has multiple algorithms, accounting for the shape of the orbit and whether the satellite is taking the short way or the long way. The Lambert-Thorne solution is also one of the most complicated solution to Lambert's Problem.
- **Lambert-Battin** : Another modern algorithm to solve Lambert's Problem, this method is based on Gauss' solution but made more robust. It is also can account for the possibility of 180 degrees difference between the two points, which, on other solution, is difficult as now the distance of the short way and the long way is exactly the same. As it is a result of development of Gauss' solution, it uses geometric analysis to solve the problem but rather than using triangles, it uses a *parabolic mean point radius*, a new term coined by Battin to explain his new calculation.
- **Lambert-Universal Variables**: This is a universal solution that can be applied to any type of transfer orbit, making it extremely useful to various problems. It's weakness is that is it not as robust as the other techniques, meaning that sometimes it fails to give a solution.

For the purpose of this literature review, the Lambert-Universal Variable algorithm to solve the Lambert's Problem is shown below, based upon Curtis' book. (Curtis, 2014)

1. Determine exactly the vectors $\hat{r}_1$ and $\hat{r}_2$, and also the transfer time Δt.
2. Calculate r_1 and r_2 using the following equation:

$$r_1 = \sqrt{\hat{r}_1 \cdot \hat{r}_1} \tag{2.43}$$

$$r_2 = \sqrt{\hat{r}_2 \cdot \hat{r}_2} \tag{2.44}$$

3. Select whether the direction taken is the short way or the long way. The short way is also called ”prograde trajectory” and the long way is called ”retrograde trajectory”.

4. Calculate the $\Delta\theta$ depending on the context. For prograde trajectory:

$$\Delta\theta = cos^{-1}(\frac{\hat{r}_1\hat{r}_2}{r_1r_2}) \quad if(\hat{r}_1 \times \hat{r}_2)_z \geq 0 \tag{2.45}$$

$$\Delta\theta = 360deg - cos^{-1}(\frac{\hat{r}_1\hat{r}_2}{r_1r_2}) \quad if(\hat{r}_1 \times \hat{r}_2)_z < 0 \tag{2.46}$$

For retrograde trajectory:

$$\Delta\theta = cos^{-1}(\frac{\hat{r}_1\hat{r}_2}{r_1r_2}) \quad if(\hat{r}_1 \times \hat{r}_2)_z < 0 \tag{2.47}$$

$$\Delta\theta = 360deg - cos^{-1}(\frac{\hat{r}_1\hat{r}_2}{r_1r_2}) \quad if(\hat{r}_1 \times \hat{r}_2)_z \geq 0 \tag{2.48}$$

5. Find a variable A, with the following equation

$$A = sin\Delta\theta\sqrt{\frac{r_1r_2}{1 - cos\Delta\theta}} \tag{2.49}$$

6. Calculate the value of z, which will tell us the shape of the orbit. A negative z value means hyperbolic orbit, a zero z means parabolic orbit, and a positive z means elliptical orbit. The equation of z is as follows:

$$\sqrt{\mu}\Delta t = [\frac{y(z)}{C(z)}]^{\frac{3}{2}}S(z) + A\sqrt{y(z)} \tag{2.50}$$

Solving for z requires using iterative methods which are shown below:

$$F(z) = [\frac{y(z)}{C(z)}]^{\frac{3}{2}}S(z) + A\sqrt{y(z)} - \sqrt{\mu}\Delta t \tag{2.51}$$

$$F'(z) = [\frac{y(z)}{C(z)}]^{\frac{3}{2}}\{\frac{1}{2z}[C(z) - \frac{3}{2}\frac{S(z)}{C(z)}] + \frac{3}{4}\frac{S(z)^2}{C(z)}\} + \frac{A}{8}[3\frac{S(z)}{C(z)}\sqrt{y(z)} + A\sqrt{\frac{C(z)}{y(z)}}] \quad (z \neq 0) \tag{2.52}$$

$$\frac{\sqrt{2}}{40}y(0)^{\frac{3}{2}} + \frac{A}{8}[\sqrt{y(0)} + A\sqrt{\frac{1}{2y(0)}} \quad (z = 0) \tag{2.53}$$

$$z_{i+1} = z_i\frac{F(z_i)}{F'(z_i)} \tag{2.54}$$

7. Find y by:

$$y(z) = r_1 + r_2 + A\frac{zS(z) - 1}{\sqrt{C(z)}} \tag{2.55}$$

8. Calculate Lagrange f, g, and $\dot{g}$ through:

$$f = 1 - \frac{[\sqrt{\frac{y(s)}{c(z)}}]^2}{r_1}C(z) = 1 - \frac{y(z)}{r_1} \tag{2.56}$$

$$g = \frac{1}{\sqrt{\mu}}\{[\frac{y(z)}{C(z)}]^{\frac{3}{2}} + A\sqrt{y(z)}\} - \frac{1}{\sqrt{\mu}}[\frac{y(s)}{C(s)}]^{\frac{3}{2}}S(z) = A\frac{y(z)}{\mu} \tag{2.57}$$

$$\dot{f} = \frac{\sqrt{\mu}}{r_1 r_2}\sqrt{\frac{y(z)}{C(z)}}[zS(z) - 1] \tag{2.58}$$

$$\dot{g} = 1 - \frac{[\sqrt{\frac{y(s)}{c(z)}}]^2}{r_2}C(z) = 1 - \frac{y(z)}{r_1} \tag{2.59}$$

9. After obtaining the Lagrange points, plug it in to obtain the value of velocity vector $\hat{v}_1$ and $\hat{v}_2$:

$$\hat{v}_1 = \frac{1}{g}(\hat{r}_2 - f\hat{r}_2) \tag{2.60}$$

$$\hat{v}_2 = \frac{1}{g}(\dot{g}\hat{r}_2 - \hat{r}_1) \tag{2.61}$$

10. As the value of the position vector and the velocity vector is now known, it can be inserted into a converter to convert it into its classical orbital elements.

Besides the regime shown above, a new approach by Izzo (Izzo, 2015) is also presented. This new approach of Izzo is further optimized - on most cases it only requires two iteration for orbit with one revolution considered, and much simpler in complexity, designed in modern times for modern computing. This algorithm will be further detailed in the next chapter and will be used as the Lambert solver for this research.

2.9 Developments of Spacecraft Trajectory Optimization

Advancement in space propulsion technologies has pushed the development of spacecraft trajectory optimization as well. In the past, the primary method of propulsion is the chemical rocket, which has an instantaneous (relative to the mission duration) change of velocity. Today, however, there are various method of propulsion with high efficiency, but has very low thrust, such as ion engines. These engines burn for a very long time, or even continuously during the mission. (Conway, 2010)

In instantaneous cases, the trajectory between burn can be considered Keplerian orbits, and for interplanetary travel, planetary flyby is also considered. But for low-thrust engines, the burn is continuous - it can be even burn during the entire trajectory, altering it as it goes. This significantly complicates the control problem. (Conway, 2010)

For recent developments in mission trajectory design, this will be divided into two main parts. The first part is for near-Earth satellite, and the second one is for deep space exploration (above 2 000 000 km or outside of Earth's sphere of influence). For deep space exploration, additional consideration must be taken into account compared to near-Earth mission, such as perturbation from other planetary bodies or planetary fly-by.

2.9.1 Missions around Earth's Sphere of Influence

Optimal LEO-GEO Intermediate Acceleration Orbit Transfer Using Nuclear Propulsion

This research by J. Albert studied a usage of a low-thrust nuclear propulsion with a constant acceleration of around 10^-2g. For this research, the optimal goal is not the least amount of propellant but rather the least amount of time. The magnitude of thrust was held constant but its acceleration vector was optimized. The study shown that the amount of time needed to go from LEO to GEO in this setting are sensitive to departure and arrival points. (Kechichian, 1997)

Minimum Fuel LEO Aeroassisted Orbit Transfer of Small Spacecraft with Inclination Change

Research by L. Darby and V. Rao focuses on the usage of small spacecraft, which can be deployed rapidly, to move within the atmosphere to increase its effectiveness. This orbit transfer is known as aeroassist. The initial orbit is a circular orbit. Solving for the nonlinear optimal control problem found that the spacecraft has the least amount of burn necessary (impulsive in this case) if the spacecraft enters the atmosphere exactly twice. The study also take into account the heating amount the spacecraft received when it went through the atmosphere. (Darby and Rao, 2011)

2.9.2 Deep Space Exploration

Hayabusa-2

Hayabusa-2 is an asteroid sample return mission by Japan Aerospace Exploration Agency (JAXA). Hayabusa-2, as the name implies, is the second mission to return a sample from an asteroid as a follow-up to earlier Hayabusa mission. The target is 1993 JU3 asteroid (now called 'Ryugu'), an asteroid believed to contain various organic matter and hydrated minerals. (Tsuda et al., 2013)

One of the technological cornerstone of Hayabusa-2 is its 4 ion engines, providing 10 mN thrust each and has a total delta-V of 2 km/s. This availability of the ion engine is actually the primary constrain during target selection. 1999 JU3 was selected as the target as it is within the capabilities of the ion engines and also has plenty of potential scientific discoveries. Thus, it can be said that propulsion technology affects the spacecraft's trajectory.

In an ion engine, electric power is the primary factor behind the amount of thrust generated. Thus, the trajectory must also consider the amount of sunlight received by the spacecraft. A method called Nonlinear Sequential Quadratic Programming (NLSQP) was used to calculate the low-thrust trajectory. The trajectory must also account for the spacecraft's return to Earth. Another constrain that must be considered are the margin for the ion engines and margin for launch windows.

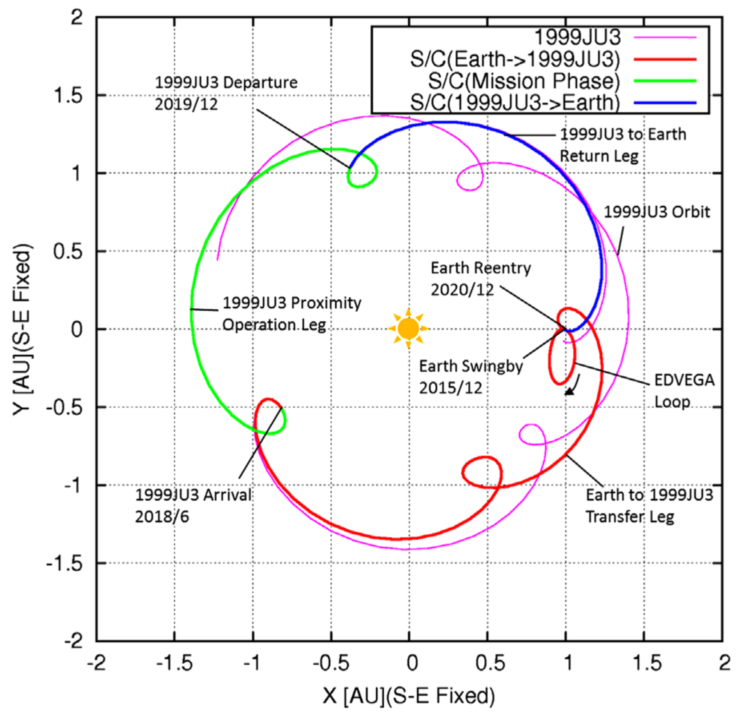


FIGURE 2.18: Hayabusa-2 Trajectory to Ryugu. Image credit: Tsuda et al., 2013

Juno

Juno is the latest mission from National Aeronautics and Space Administration (NASA / USA) to gas giant planet Jupiter. Juno's primary objectives are to find out the origin and the way Jupiter

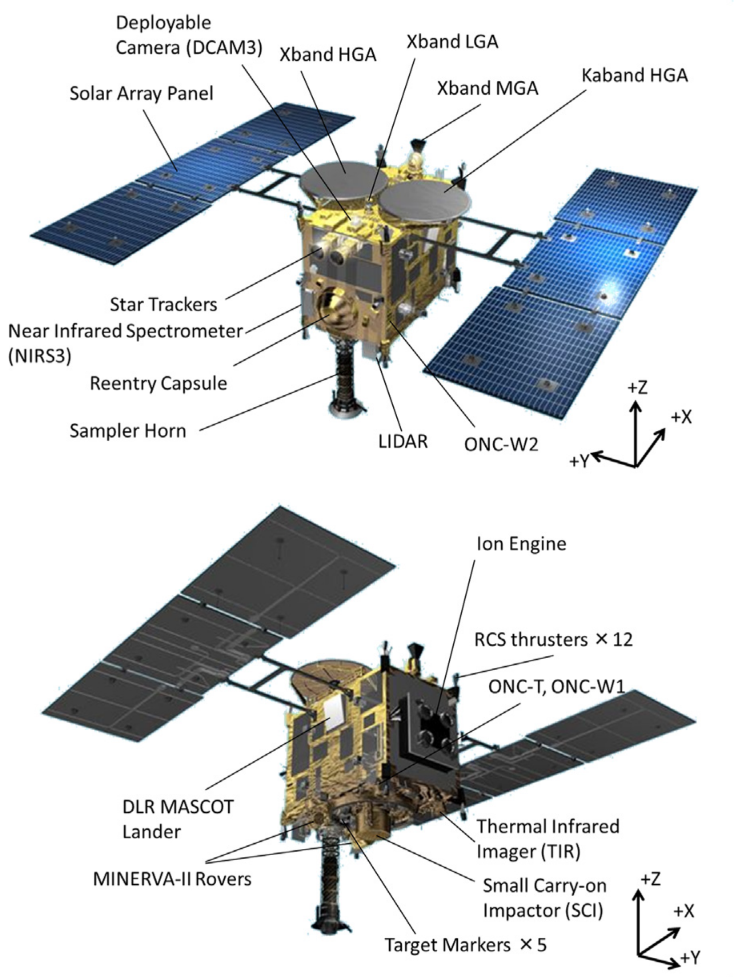


FIGURE 2.19: Hayabusa-2 Spacecraft. Image credit: Tsuda et al., 2013

develops over time through measurement of Jupiter's solid core, tracing heavy elements at the atmosphere, mapping the magnetic and gravity field and also exploring the polar regions of Jupiter. (Matousek, 2007)

For Juno mission, the spacecraft is spun like a spinning top, with the solar panel extended. This spinning motion act as the spacecraft's attitude control and also the extension of the solar panel both act as a power plant and also a thermal control.

In order to put Juno into Jupiter's orbit, Juno was first put into a deep space orbit. Then, 1 year after launch, a burn was initiated to make Juno encounter the Earth (Earth fly-by), 2 years after launch. This fly-by acted as gravity assist that slingshot-ed Juno to arrive at Jupiter 5.2 years after launch. The trajectory during Jupiter Orbit Insertion (JOI) was made to ensure that Juno follows a polar, 11 day orbit, while at the same time has high eccentricity, with periapsis of only 1.06 Jupiter's radius and apoapsis of 39 Jupiter's radius. This trajectory was chosen to ensure a global coverage of Jupiter while at the same time shields Juno from Jupiter's magnetosphere.

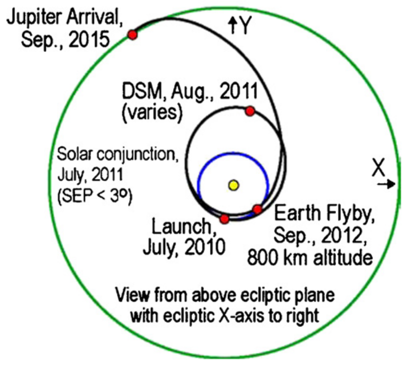


FIGURE 2.20: Juno's Trajectory to Jupiter. Image credit: Matousek, 2007

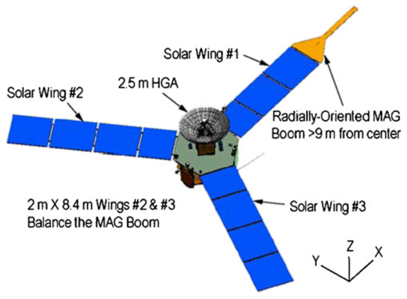


FIGURE 2.21: Juno Spacecraft, showing the extended solar cell array. Image credit: Matousek, 2007

Earth to Mars using Primer Vector

E. Fitrianingsih and R. Armellin from University of Surrey has calculated a mock spacecraft trajectory from Earth to Mars. The purpose of the study is to find an optimal trajectory that has a lowest delta-V. To optimize a trajectory there exist 2 methods, the first one is by creating a list of candidate orbit, directly calculate each one of them, and then select the one with the lowest required delta-v. Another method is by using analysis of the thrust vector, using a method called primer vector. For this research, the authors opted to use primer vector method to verify instead whether a Earth-Mars trajectory is already optimal or can be optimized further with additional burns (correction maneuvers.) The authors created a single orbit, and then tested the orbit upon the different departure date and transfer duration. The research concluded that primer vector can be used to determine whether an orbit, based upon its departure time and its travel time from Earth, can be optimized further or is already at its peak potential. (Fitrianingsih and Armellin, 2018)

CHAPTER 3

RESEARCH METHODOLOGY

3.1 Research Outline

To conduct this research, the author has come to this approach:

1. **Problem Formulation**. The Author formulated the initial problem based upon recent development in spacecraft trajectory. In this case, the author chose to design various methods for transferring a satellite from Low Earth Orbit to Geostationary Earth Orbit.

2. **Literature Review**. The author look upon from reference journals and reference books in regards to relevant materials to aid the research. This includes research on the ways the transfer is done, latest development in trajectory optimization, and also research regarding definitions and space program in general.

3. **Building Mathematical Tools**. Based upon literature review and research, the Author builds mathematical tools and programs to do the calculations. These tools are built using scientific computing, and built in a way that enable various input so the research data can be varied.

4. **Creating A Preliminary Space Mission**. With mathematical tools ready, the Author then can begin creating a preliminary design for a space mission with the sequence of first launching to Low Earth Orbit and then to Geostationary Earth Orbit.

5. **Comparison of Various Trajectory Methods**. The scenarios to achieve this orbit is then compared between each other to see which one results in the most optimal fuel cost. The fuel post in this regard is limited to the fuel needed from the initial launch Low Earth Orbit to Geostationary Earth Orbit. The initial fuel from the launch pad to the parking Orbit is not considered.

6. **Calculation of Possible Payload Mass**. Based upon the most optimal launch profile, the possible payload mass with the selected launcher can be determined using the Rocket Equation.

3.2 Scientific Computing

This research utilize computer to calculate the required parameters, hence the name of scientific computing. This approach was chosen by the author as manual calculation will be tedious and ineffective. To accomplish this, the author uses several program/software as shown below.

3.2.1 Python Programming Language

Python is selected as the primary language for this research due to several reasons. The first one is its long-standing usage within the scientific community, making plug-ins and library for scientific usage widely available. The second reason is due to Python's user-friendliness, as the Author was not trained in using Python before and have to learn from the beginning for the sake of this program.

3.2.2 Anaconda Platform

Anaconda is a software distribution package made by Anaconda Inc. It is an open source package designed to make Python accessible to everyone, from individual users to enterprises. (*About Anaconda Inc*) Anaconda accomplishes this by providing various required scientific package all in one simple installation. The built in manager is called the Anaconda Navigator. Within the program, users can manage over 7500 scientific plug-in and packages to choose which one suit their needs, with the base Anaconda package containing 250 packages. (Anaconda Inc, 2020) Using Anaconda Navigator, the author manages all the required scientific packages required for this project.

3.2.3 NumPy Package

NumPy is an add-on package that enables Python to compute numerically. It added a lot of mathematical functions to Python with its library. It is open source and one of the most important scientific library for Python. (*About Numpy*)

Features of NumPy are the backbone of this project, which requires a lot of NumPy's built in calculation regime.As NumPy is included inside the Anaconda distribution, the Author can use it straight away.

3.2.4 SciPy Ecosystem

SciPy is a collection of additional scientific libraries for Python. It is a complete tool box for computational science, which when coupled with NumPy, provides a high performance and varied computing capability. (*About SciPy*)

The Author uses SciPy to aid in calculations so the Author doesn't need to write each individual calculation from scratch.

3.2.5 MatPlotLib Plotter

Matplotlib is a library to enable visualization of computation through Python. Its result are clear and concise, yet also customizable to fit the user's specific requirement. Matplotlib is also expandable to include new features.(*Matplotlib*)

The Author uses Matplotlib to visualize the calculations which has been done. With Matplotlib, results can be displayed in a clear and concise manner.

3.3 Building Math Toolbox - Converting from TLE into Orbital Elements

3.3.1 Introduction to LAPAN A2

LAPAN A2 is a micro-satellite developed by Lembaga Penerbangan dan Antariksa Nasional (LAPAN) of Indonesia. It is classified as a microsatellite and is the second one from LAPAN. It is built upon LAPAN's first microsatellite, LAPAN-TUBSAT, which was made with cooperation with Technical University Berlin, hence the name. Based upon experience building and operating LAPAN-TUBSAT, LAPAN-A2 was developed indigenously, all made in house in LAPAN facilities, which were upgraded to accommodate the LAPAN-A2 program. (*LAPAN-A2*)

The selection of LAPAN-TUBSAT as its basis and its size as a microsatellite is due to cost constrains. LAPAN hoped that the development of these microsatellite, even though only utilize small amount of resources, can be a stepping stone for a space program in the future that can help the advancement of Indonesia as a whole. (Hardhienata, Triharjanto, and Mukhayadi, 2011)

The main goal of the program is prove LAPAN's capability in designing, building and operating a microsatellite all within Indonesia. As its main mission, LAPAN-A2 is to observe the Indonesian vast archipelago using its built in camera, with a resolution of 6 meters. In addition, LAPAN-A2 is also equipped with a Automatic Identification System (AIS) to identify maritime traffic around Indonesia, providing much needed patrol capability along Indonesian waters. Another payload inside LAPAN-A2 is an Automatic Packet Reporting System (APRS), used for communication in case of natural disaster in the archipelago and also a amateur radio voice repeater. These two devices are operated in conjunction with Indonesian Amateur Radio Organization or ORARI.(Hardhienata, Triharjanto, and Mukhayadi, 2011)

For its orbit, as it's primary mission is to monitor Indonesia, it is placed at Low Earth Orbit along the equator, with a small 8 degree inclination. It was launched as a secondary payload to Indian observatory satellite called ASTROSAT. (Hardhienata, Triharjanto, and Mukhayadi, 2011)

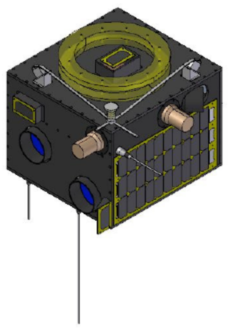


FIGURE 3.1: 3D Model of LAPAN A2. Image credit: Hardhienata, Triharjanto, and Mukhayadi, 2011

3.3.2 Obtaining LAPAN A2 data from Celestrak

First, data regarding LAPAN-A2 must be obtained. This data is obtained via public service at https://celestrak.com/. The data regarding LAPAN-A2 from Celestrak in Two-Line Element format is as follows:

```
LAPAN-A2
1 40931U 15052B   20168.74122108  .00000632  00000-0 -12038-5 0  9993
2 40931   5.9950 340.4753 0013975 268.9107  90.9697 14.76636552255154
```

This data was taken on 16th June 2020. The Celestrak website only takes into account the latest TLE data with no historical records, however this is enough for the starting point of our calculation as in the later steps this data will be propagated.

3.3.3 Conversion Regime

As the contents of TLE data cannot be used directly, it is must be converted into Classical Orbital Elements. The data is first interpreted as follows:

TABLE 3.1: Two Line Element of LAPAN A2.

Line 0		
Column	Data	Description
1-24	LAPAN-A2	Name of the tracked object
Line 1		
Column	Data	Description
1	1	Line Number
3-7	40931	Satellite Catalog Number
8	U	Satellite Classification, U for Unclassifed
10-17	15052B	International Designator
19-32	20168.74122108	Element Set Epoch (UTC)
34-43	.00000632	1st Derivative of Mean Motion with Respect to Time
45-52	00000-0	2nd Derivative of Mean Motion with Respect to Time, Decimal Point Assumed
54-61	-12038-5	B* Drag Term
63	0	Element Set Type
65-68	999	Element Number
69	3	Checksum
Line 2		
Column	Data	Description
1	2	Line Number
3-7	40931	Satellite Catalog Number
9-16	5.9950	Orbit Inclination (degrees)
18-25	340.4753	Right Ascension of Ascending Node (degress)
27-33	0013975	Eccentricity (decimal point assumed)
35-42	268.9107	Argument of Perigee (degrees)
44-51	90.9697	Mean Anomaly (degrees)
53-63	14.76636552	Mean motion (revolutions/day)
64-68	25515	Revolution Number at Epoch
69	4	Checksum

Based on the data above, the particular elements of interest is from the second line of the set, which contains data that is readily converted into COEs. However, the true anomaly and the semimajor axis are not expressed explicitly, thus the data must be converted to obtain the value of true anomaly and semimajor axis.

To obtain semi major axis, the value can be obtained from mean motion via Kepler's Third Law (Vallado and McClain, 2013):

$$n = \sqrt[2]{\frac{\mu}{a^3}} \tag{3.1}$$

As mean motion in TLE is expressed in revolution per day, it is first must be converted to mean revolution per second.

$$n = \frac{14.76636552}{24 hours * 3600 seconds} = 1.70907 * 10^{-4} rev/second \tag{3.2}$$

Rearranging the equation, we can obtain for a (semi major axis) in km:

$$a = \sqrt[3]{\frac{\mu}{(2 * \pi * 1.70907 * 10^{-4})^2}} \tag{3.3}$$

For $\mu = 396000 \frac{km^3}{s^2}$, the resulting value is $7002.803 km$ from Earth's center.

For conversion from mean anomaly to true anomaly, an internet online service is used. This is due to the amount of algorithm required which can complicate the program. Obtained from (Juergen Giesen, 2016), the resulting true anomaly is $91.1298 deg$.

The complete COE value for LAPAN-A2 based on the TLE data is as follows:

- Semi major axis = 7002.803 km
- Inclination = 5.9950 degrees
- Right Ascension of Ascending Node = 340.4753 degrees
- Argument of Perigee = 269.9108 degrees
- Eccentricity = 0.0013975
- True Anomaly = 91.1298 degrees

All of this calculation were done using Python programming language, with the entire program included in the Appendix.

3.4 Building Math Toolbox - Conversion Between State Vectors and Orbital Elements

After obtaining COEs, the next step is the ability to convert between state vectors with orbital elements and back and forth. This capability is required because different parts of the research requires different orbit definition method, so the ability to change them is essential.

The basis of this conversion regime is based on Vallado's Fundamental of Astrodynamics and Application (Vallado and McClain, 2013). The steps are shown below, with the program code shown in the Appendix.

3.4.1 From Orbital Elements into State Vectors

For calculation of state vectors, it is more useful to describe an orbit size using semi-parameter instead of semi-major axis. A semi-parameter is the distance from the primary focus perpendicular to the orbit. For a circular orbit, the semi-parameter is simply the radius because the two of the foci is located at the exact some point, but for elliptical orbit this is not the case. Thus, it is better to use semi-parameter to account for this shape. (Vallado and McClain, 2013)

The equation for semi-parameter of an elliptical orbit is given by:

$$p = a * (1 - e^2) \tag{3.4}$$

After obtaining the semi-parameter, the conversion regime can begin.

1. Initially, all of the orbital elements with angle as measurement have to be converted from degrees to radians. This is due to NumPy processing which computes in radians.

2. The values are then put into 2 matrices. The first matrix is the position matrix:

$$\hat{r}_{PQW} = \begin{bmatrix} \frac{p\cos(v)}{1+e\cos(v)} \\ \frac{p\sin(v)}{1+e\cos(v)} \\ 0 \end{bmatrix} \tag{3.5}$$

   and the second matrix is the velocity matrix:

$$\hat{v}_{PQW} = \begin{bmatrix} -\sqrt{\frac{\mu}{p}}\sin(v) \\ \sqrt{\frac{\mu}{p}}(e+\cos(v) \\ 0 \end{bmatrix} \tag{3.6}$$

3. The matrices are then rotated using a transformation matrix to obtain matrices that correspond to geocentric equatorial coordinate system.

4. Lastly, the matrices are rearranged to fit a typical R and V matrix.

3.4.2 From State Vectors into Orbital Elements

Likewise, a conversion regime from state vectors into orbital elements, also adapted from Vallado's Fundamental of Astrodynamics and Application (Vallado and McClain, 2013) is as follows:

1. All of the state vector are put into two NumPy array, the position vector into an array, and likewise for the velocity vector. The magnitude of these two vector are then calculated.

2. Calculate the angular momentum using the cross product of the vector. Then, find the magnitude of the resulting vector.

3. Set another vector 'k' which purpose is to aid in calculation, which contain only 'one' in k direction and zero in other direction (i and j).

4. The node vector is calculated. The node vector is the cross product of unit vector 'k' with the angular momentum vector. The magnitude of the node vector is also calculated.

5. Eccentricity is then calculated using the vector form of eccentricity, as the initial position and velocity is known. This vector is given by:

$$\hat{e} = \frac{(v^2 - \frac{\mu}{r})\hat{r} - (\hat{r} \cdot \hat{v})\hat{v}}{\mu} \tag{3.7}$$

   The magnitude of this vector serves as the value of eccentricity for this orbit.

6. The next step is computing semi major axis. To compute semi major axis, specific mechanical energy must first be found:

$$\xi = \frac{v^2}{2} - \frac{\mu}{r} \tag{3.8}$$

   After obtaining specific mechanical energy, the size of semi-major axis can computed by:

$$a = -\frac{\mu}{2\xi} \tag{3.9}$$

7. The last step is the calculation of all of the angles: the Right Ascension of Ascending Node (Ω), Argument of Periapsis (ω), Inclination (i), and True Anomaly (v), which are given the equation below respectively:

$$\cos(\Omega) = \frac{n_I}{|\hat{n}|} \qquad IF(n_J < 0)\ THEN \Omega = 360deg - \Omega \tag{3.10}$$

$$\cos(\omega) = \frac{\hat{n} \cdot \hat{e}}{|\hat{n}| \cdot |\hat{e}|} \qquad IF(e_K) < 0\ THEN \omega = 360deg - \omega \tag{3.11}$$

$$\cos(i) = \frac{h_K}{|\hat{h}|} \tag{3.12}$$

$$\cos(v) = \frac{\hat{r} \cdot \hat{r}}{|\hat{e}||\hat{r}|} \qquad IF(\hat{r} \cdot \hat{v} < 0)\ THEN v = 360deg - v \tag{3.13}$$

3.5 Building Math Toolbox - Orbit Propagation

The state vectors only describe the position and velocity of a satellite at that instant. Likewise, the classical orbital elements of an orbit while is helpful in visualization the shape of an orbit, doesn't really account for the actual trajectory of the satellite. To know exactly how the satellite moves through its orbit, the state vectors must be propagated. This propagation is done by solving for the Two-body equation given by:

$$\ddot{\hat{r}} = -\frac{G(m_{\oplus} + m_{sat})}{r^2}\frac{\hat{r}}{r} \tag{3.14}$$

3.5.1 Orbital Propagation using ODE Solver

The Author solved the two-body equation using SciPy's built in ODE-Solver, called "solve_ ivp". The solve _ ivp function can use various algorithm to solve an ODE, such as Runge-Kutta Fourth Order Method, Runge Kutta of Eighth Order, and LSODA, which is an algorithm derived from an earlier programming language (FORTRAN). Each of this method has their own strength and weakness depending on the problem. (SciPy Developers, 2020). The Author chose to use LSODA and ODE45 (Runge-Kutta Fourth Order) as they are the universal choice.

To use the solver, a function that describes the problem must be made. This essentially means breaking down the two-body problem into its elements. The steps are as follows:

1. Find the magnitude of the position vector. The magnitude is given by:

$$r = \sqrt[2]{x^2 + y^2 + z^20} \tag{3.15}$$

2. Find the derivative of each of the velocity vector element.

$$\ddot{x} = (-\frac{\mu}{r^3})x \tag{3.16}$$

$$\ddot{y} = (-\frac{\mu}{r^3})y \tag{3.17}$$

$$\ddot{z} = (-\frac{\mu}{r^3})z \tag{3.18}$$

3. Arrange the result in a NumPy array, with the position element first (x, y, z), followed by the acceleration $(\ddot{x}, \ddot{y}, \ddot{z})$. Return the completed array.

4. Include the array in the solver using SciPy's solve _ ivp function, selecting 'LSODA' or ODE45' ODE solving algorithm.

3.5.2 Orbital Propagation in Consideration for J2 Perturbation

Recall that when any perturbation is considered, the two-body equation becomes:

$$\ddot{\hat{r}} = -\frac{G(m_{\oplus} + m_{sat})}{r^2}\frac{\hat{r}}{r} + p \tag{3.19}$$

With p as acceleration due to perturbation. In case of J2 perturbation, the acceleration according to Vallado (Vallado and McClain, 2013) are as follows:

$$a_I = -\frac{3J_2\mu {R_{\oplus}}^2 r_I}{2r^5}(1 - \frac{5{r_K}^2}{r^2}) \tag{3.20}$$

$$a_J = -\frac{3J_2\mu {R_{\oplus}}^2 r_J}{2r^5}(1 - \frac{5{r_K}^2}{r^2}) \tag{3.21}$$

$$a_K = -\frac{3J_2\mu {R_{\oplus}}^2 r_K}{2r^5}(3 - \frac{5{r_K}^2}{r^2}) \tag{3.22}$$

with the value of J2 corresponds to the J2 constant:

$$J_2 = 0.00108263$$

Therefore, the final acceleration value, after accounting for J2 perturbation, becomes:

$$\ddot{x} = (-\frac{\mu}{r^3})x - \frac{3J_2\mu {R_{\oplus}}^2 r_I}{2r^5}(1 - \frac{5{r_K}^2}{r^2}) \tag{3.23}$$

$$\ddot{y} = (-\frac{\mu}{r^3})y - \frac{3J_2\mu {R_{\oplus}}^2 r_J}{2r^5}(1 - \frac{5{r_K}^2}{r^2}) \tag{3.24}$$

$$\ddot{z} = (-\frac{\mu}{r^3})z - \frac{3J_2\mu {R_{\oplus}}^2 r_K}{2r^5}(3 - \frac{5{r_K}^2}{r^2}) \tag{3.25}$$

The new acceleration value are then placed into an array with the position value first and the acceleration value later, in x, y, and z order. The array is later returned to the solve _ ivp function as per previous calculation regime.

3.6 Building Math Toolbox - Orbit Maneuvering

As this research focuses on maneuver from Low Earth Orbit to Geostationary Earth Orbit, various orbit maneuver methods will be used. Thus, the algorithms on how to execute them are outline below. These maneuvers are based upon algorithms found in Vallado's book. (Vallado and McClain, 2013).

For these research, all orbit are considered as circular cases.

3.6.1 Calculation of Orbit Change Using Hohmann Transfer

The step undertaken to calculate the trajectory from one orbit to another orbit, specifically from LEO, is follows:

1. Declare all the Classical Orbital Elements of the initial Low Earth Orbit.
2. As the orbit is considered circular, the orbit size is measured as radius. Convert the initial radius of the orbit to canonical (in reference to the Earth) unit. The conversion is as follows:
$$r_{init} = \frac{r_{satellite} + r_{Earth}}{r_{Earth}} \tag{3.26}$$
Likewise, the intended final orbit is also converted:
$$r_{final} = \frac{r_{final} + r_{Earth}}{r_{Earth}} \tag{3.27}$$
3. Calculate the time unit of Earth for use in canonical unit calculation.
$$TU = \sqrt[2]{\frac{{r_{Earth}}^3}{G * M}} \tag{3.28}$$
4. Then, find the velocities of the satellite at both the initial point and final point. These velocities are calculated upon Kepler's law.
$$v_{init} = \sqrt[2]{\frac{\mu}{r_{init}}} \tag{3.29}$$
$$v_{final} = \sqrt[2]{\frac{\mu}{r_{final}}} \tag{3.30}$$
5. As the transfer orbit between the initial and final orbit is elliptical, find its semi-major axis, simply by finding the average between the two:
$$a_{trans} = \frac{r_{init} + r_{final}}{2} \tag{3.31}$$
6. Calculate the transfer velocities. There are two burns in a Hohmann transfer, one burn in the initial orbit, and a second burn which took place exactly half the period of the transfer orbit, circularizing the orbit. The first orbit is called v_{transA} and v_{transB}, respectively. The formula is as follows:
$$v_{transA} = \sqrt[2]{\frac{2\mu}{r_{init}} - \frac{\mu}{a_{trans}}} \tag{3.32}$$

$$v_{transA} = \sqrt[2]{\frac{2\mu}{r_{final}} - \frac{\mu}{a_{trans}}} \tag{3.33}$$

7. Find the Δv of these two burns.

$$\Delta_a = |v_{transA} - v_{init}| * \frac{r_{Earth}}{TU} \tag{3.34}$$

$$\Delta_b = |v_{final} - v_{transB}| * \frac{r_{Earth}}{TU} \tag{3.35}$$

8. Calculate the total time of transfer using the time it took to transverse the transit orbit:

$$t = \pi * a_{trans}{}^{\frac{3}{2}} \tag{3.36}$$

9. Return the resulting Δ_a ,Δ_b, and t value back to the main program.
10. Convert the resulting Classical Orbital Elements into state vectors using previous algorithm.
11. Propagate the resulting orbit and plot the result using Matplotlib.

3.6.2 Calculation of Orbit Change Using Bi-elliptic Transfer

As a Bi-elliptic transfer is essentially two Hohmann transfer in a row, in total there will be three burns. The steps are as follows:

1. Convert all of the radius of the orbits into canonical units.

$$r_{init} = \frac{r_{satellite} + r_{Earth}}{r_{Earth}} \tag{3.37}$$

$$r_{transfer} = \frac{r_{transfer} + r_{Earth}}{r_{Earth}} \tag{3.38}$$

$$r_{final} = \frac{r_{final} + r_{Earth}}{r_{Earth}} \tag{3.39}$$

2. Find the Time Unit for use in canonical equations. For Earth, 1 Time Unit equals to 806.80415 seconds.
3. Calculate the velocity at both the initial orbit and final intended orbit.

$$v_{init} = \sqrt[2]{\frac{\mu}{r_{init}}} \tag{3.40}$$

$$v_{final} = \sqrt[2]{\frac{\mu}{r_{final}}} \tag{3.41}$$

4. Calculate the semi-major axis of two transfer orbits, as there are two transfer orbit in a bi-elliptic transfer:

$$a_{trans1} = \frac{r_{init} + r_{transfer}}{2} \tag{3.42}$$

$$a_{trans2} = \frac{r_{transfer} + r_{final}}{2} \tag{3.43}$$

5. Transfer velocities are calculated through:

$$v_{trans1a} = \sqrt[2]{\frac{2\mu}{r_{init}} - \frac{\mu}{a_{trans1}}} \tag{3.44}$$

$$v_{trans2c} = \sqrt[2]{\frac{2\mu}{r_{final}} - \frac{\mu}{a_{trans2}}} \tag{3.45}$$

$$v_{trans1b} = \sqrt[2]{\frac{2\mu}{r_{transfer}} - \frac{\mu}{a_{trans1}}} \tag{3.46}$$

$$v_{trans2b} = \sqrt[2]{\frac{2\mu}{r_{transfer}} - \frac{\mu}{a_{trans2}}} \tag{3.47}$$

6. Total ΔV is calculated, first using canonical units:

$$\Delta V_{canonical} = |v_{trans1a} - v_{init}| + |v_{trans2b} - v_{trans1b}| + |v_{final} - v_{trans2c}| \tag{3.48}$$

7. Convert the canonical ΔV into m/s

$$\Delta V = \Delta V_{canonical} * \frac{r_{Earth}}{TU} \tag{3.49}$$

8. Calculate the time of flight by calculating the time required to transverse the two transfer orbits:

$$t = \pi * a_{trans1}{}^{\frac{3}{2}} + \pi * a_{trans2}{}^{\frac{3}{2}} \tag{3.50}$$

9. Return the value of the required $DeltaV$ to the main program.

10. Convert back the resulting COEs into state vectors, and propagate it using the ODE Solver.

11. Plot the resulting orbit using Matplotlib.

3.6.3 Non-co-planar Maneuver - Inclination Change Only

A change of plane in inclination only will require the burn to be conducted at exactly the crossing between the orbital plane and the equator plane. There will be two points where this happens: these are referred as nodes.

To calculate the burn, the calculation relies on a simple matter - at the nodes, the true anomaly of both orbits will be equal, in fact all of the COE are equal, except the inclination of course. This means the magnitude of velocity before and after burn is also the same. Because the velocity does not change, the velocity vector create an isosceles triangle with the following change of velocity:

$$\sin(\frac{\Delta i}{2}) = \frac{\Delta_i}{2v_{initial}\cos(\phi_{fpa}})) \tag{3.51}$$

Rearranging the equation, we can obtain the Δv required

$$\Delta v = 2v_{initial}\cos(\phi_{fpa})\sin(\frac{\Delta i}{2}) \tag{3.52}$$

3.6.4 Non-co-planar Maneuver - Right Ascension of Ascending Node Change Only

Another possible change is the change in right ascension of the ascending node. This burn are done at the intersection between the initial orbit and the final orbit. First, we define a new angle ϑ which is the angle of the rotation of the velocity. Then, we can find this angle with this equation:

$$\cos(\vartheta) = \cos^2(i_{initial}) + \sin^2(i_{initial})\cos(\Delta\Omega) \tag{3.53}$$

Upon obtaining this angle, the Δv can be calculated through:

$$\Delta v_{\Omega} = 2v_{initial}\sin(\frac{\vartheta}{2}) \tag{3.54}$$

3.6.5 Non-co-planar Maneuver - Both Changes

When considering for maneuver that changes both inclination and right ascension of ascending node, the concept is similar to the change of RAAN. However, the difference is in the calculation of the angle ϑ. The equation is as follows:

$$\cos(\vartheta) = \cos(i_{initial})\cos(i_{final}) + \sin(i_{initial})\sin(i_{final})\cos(\Delta\Omega) \tag{3.55}$$

And the Δv is obtained by:

$$\Delta v_{\Omega} = 2 v_{initial} \sin(\frac{\vartheta}{2}) \tag{3.56}$$

3.7 Building Math Toolbox - Lambert's Problem Solver

For this research, the Lambert Solver Algorithm that is going to be used is based upon Izzo's paper. (Izzo, 2015). This algorithm is used due to its simplicity and its fitness for this particular problem, which is a single revolution transfer orbit. An implementation of Izzo's algorithm is used by the European Space Agency (Izzo, 2019) and also Poliastro, an open-source astrodynamics library for Python. (Juan Luis Cano Rodríguez, 2019).

The Lambert Solver requires two algorithm to work. The first algorithm is the main Lambert solver code (with $P1$, $P2$, μ and t as input, while the second algorithm is the iterator that is used in the first algorithm.

The steps to implement the first algorithm is as follows:

1. Obtain the 2 position vector($\vec{r_1}, \vec{r_2}$), the time of flight between them, and the gravitational parameter. The time of flight and the gravitational parameter must be larger than zero. These 4 variables serve as input to the solver.

2. Find the vector $\vec{c}$ which is the chord between the two points, and find the magnitude of the all the vectors.

$$\vec{c} = \vec{r_2} - \vec{r_1} \tag{3.57}$$

$$c = |\vec{c}|, r_1 = |\vec{r_1}|, r_2 = |\vec{r_2}| \tag{3.58}$$

3. Calculate the semiparameter

$$s = \frac{1}{2}(r_1 + r_2 + c) \tag{3.59}$$

4.

$$\hat{i}_{r,1} = \vec{r_1}/r_1, \hat{i}_{r,2} = \vec{r_2}/r_2 \tag{3.60}$$

$$\hat{i}_h = \hat{i}_{r,1} \times \hat{i}_{r,2} \tag{3.61}$$

5. Find λ parameter

$$\lambda = \sqrt{1 - c/2} \tag{3.62}$$

6. Begin switch case

   if $(r_{11} r_{22} - r_{12} r_{21}) < 0$ **then**

$$\lambda = -\lambda \tag{3.63}$$

$$\hat{i}_{t,1} = \hat{i}_{r,1} \times \hat{i}_h, \hat{i}_{t,2} = \hat{i}_{r,2} \times \hat{i}_{r,2} \tag{3.64}$$

else

$$\hat{i}_{t,1} = \hat{i}_h \times \hat{i}_{r,1}, \hat{i}_{t,2} = \hat{i}_h \times \hat{i}_h \tag{3.65}$$

end if

$$T = \sqrt{\frac{2\mu}{s^3}t} \tag{3.66}$$

7. Enters the iteration regime. The ”findxy” function below is name of the second algorithm’s function. The x_{list} and y_{list} are arrays.

$$x_{list}, y_{list} = findxy(\lambda, T) \tag{3.67}$$

$$\gamma = \frac{\mu s}{2}, \rho = \frac{r_1 - r_2}{c}, \sigma = \sqrt{(1-\rho^2)} \tag{3.68}$$

for each x, y in x_{list}, y_{list} **do**

$$V_{r,1} = \gamma[(\lambda y - x) - \rho(\lambda y + x)]/r_1 \tag{3.69}$$

$$V_{r,2} = -\gamma[(\lambda y - x) - \rho(\lambda y + x)]/r_2 \tag{3.70}$$

$$V_{t,1} = \gamma\sigma(y + \lambda x)/r_1 \tag{3.71}$$

$$V_{t,2} = \gamma\sigma(y + \lambda x)/r_2 \tag{3.72}$$

$$\vec{v}_1 = V_{r,1}\hat{i}_{r,1} + V_{t,1}\hat{i}_{t,1} \tag{3.73}$$

$$\vec{v}_2 = V_{r,2}\hat{i}_{r,2} + V_{t,2}\hat{i}_{t,2} \tag{3.74}$$

end for, end of the iteration regime

The following is the $findxy$ algorithm, which through iteration shall calculate all the x and y. The x and y in this definition are angles, not coordinate, defined more clearly in the paper (The Lambert problem is still a geometrical problem). The process according to Izzo (Izzo, 2015) is as follows:

1. Ensure that the parameter $|\lambda|$ is smaller than zero, and T is smaller than zero.

2. Define M, the number of revolutions of orbit in which the transfer happen.

$$M_{max} = floor(T/\pi) \tag{3.75}$$

3. Find T_{00}, where T_0 is the value of T when $x = 0$ and T_{00} meaning T_0 for single revolution

$$T_{00} = \arccos \lambda + \lambda \sqrt{1 - \lambda^2} \tag{3.76}$$

4. Begin switch case, accounting for different orbit sizes and also the limit of number of revolutions for the trajectory

```
if $T < T_{00} + M_{max}\pi$ and $M_max > 0$ then
begin the Halley iteration, from $x = 0, T = 0$ and find $T_{min}(M_{max})$
if $T_{min} > T$ then
    $M_{max} = M_{max} - 1$    (3.77)
end if
end if
```

5. Obtain the value of T when $T(x = 1)$

$$T(x = 1) = T_1 = \frac{2}{3}(1 - \lambda^3) \tag{3.78}$$

6. Compute value of x_0, which equation depends of various case of T

7. After obtaining the value of x_0, begin the Householder iteration from there and find values of x and y.

8. Begin iteration

```
while$M_{max} > 0$ do
calculate $x_{0l}$ and $x_{0r}$ with $M = M_{max}$
begin Householder iteration from $x_{0l}$ and find $x_r, y_r$
begin Householder iteration from $x_{0r}$ and find $x_l, y_l$
$M = M_{max}$
end while
```

The program to implement these algorithm in a scientific computing environment is based upon Poliastro Python Library.

3.8 Mission Design - Selection of Suitable Launch Vehicle and Launch Site

The author needs to select a suitable launch vehicle for the mission, and as a consequence, the launch site available for the corresponding launch vehicle. For this research, the individual launch performance or capability is not the primary selection criteria, but rather the availability of launch vehicle's specification is the ultimate factor, as the performance data is immensely needed.

Based upon the specification included in the Launch Vehicle's User Manual, the initial parking orbit that is located at LEO can be determined. This is the starting point for design of the orbit transfer to Geostationary Earth Orbit. The final orbit is a circular Geostationary Orbit, in which its exact latitude and longitude is not in the scope of this research.

Once the satellite it is its initial Low Earth Orbit, there are various scenarios to achieve Geostationary Earth Orbit. These scenarios are then compared to see which one has the lowest fuel cost.

3.9 Mission Design - Scenario One

The outline of the first scenario is as follows:

1. From Low Earth Orbit, the orbit is enlarged to Geosynchronous Earth orbit via 2 possible methods: Hohmann and Bi-elliptic.
2. Then, using non-co-planar maneuver, the GSO orbit is shifted to the equator to form a Geostationary Earth Orbit

3.9.1 Calculation of Maneuver from Initial Orbit (LEO) to GSO via Hohmann and Bi-Elliptic

Based upon the Launch Vehicle documentation, the final orbit will be exactly a circular orbit at an altitude of 35 786 km at the same plane of the the initial Low Earth (Parking) Orbit.

Using algorithm based upon Vallado (Vallado and McClain, 2013), this circular to circular, coplanar orbit transfer is calculated, from the parking orbit altitude to geosynchronous orbit altitude.

The first method is using Hohmann transfer. This simply follows the original formulation, and the final Δv cost can be immediately known.

The second method is using Bi-elliptic transfer, with a transfer orbit higher than the desired final orbit (35 768 km). This method will also utilize the algorithm in Vallado's book. As the altitude of the transfer orbit can be varied, the Author has chosen the following transfer orbit:

- **35 800 km**. This altitude represent the starting point, with just a little bit higher than the target orbit of 35 768 km, accounting for launch vehicle insertion accuracy.
- **40 768 km**. 5000 km above Geosynchronous Earth Orbit.
- **45 768 km**. 10000 km above Geosynchronous Earth Orbit.
- **55 768 km**. 20000 km above Geosynchronous Earth Orbit.

3.9.2 Calculation of Maneuver from GSO to GEO via Non-Co-Planar Maneuver

Upon arrival to a Geosynchronous Earth orbit, the amount of Δv required to change the orbital plane to equator is calculated using the earlier algorithm.

3.10 Mission Design - Scenario Two

The outline of the first scenario is as follows:

1. From Low Earth Orbit, the non-co-planar maneuver is immediately done to shift the orbit to equatorial plane.
2. Then, using either Hohmaan Transfer or Bi-elliptical transfer, the orbit is enlarged to a circular orbit at an altitude of 35 768 km - the Geostationary Earth Orbit.

3.10.1 Calculation from Inclined Plane to Equatorial Plane

Regardless of the inclination of the initial parking plane, the plane is shifted using non-co-planar maneuver burn to shift it to the equatorial plane (inclination angle of zero).

3.10.2 Calculation of Orbit Enlargement using Hohmann and Bi-elliptic Transfer

Once the satellite is in the orbital plane, the orbit is enlarged using both the Hohmann and Bi-elliptic transfer. The transfer orbit for the Bi-elliptic method is the same altitude with the previous scenario.

- **35 800 km**. The starting point for bi-elliptic calculation.
- **40 768 km**. 5000 km above Geostationary Earth Orbit.
- **45 768 km**. 10000 km above Geostationary Earth Orbit.
- **55 768 km**. 20000 km above Geostationary Earth Orbit.

3.11 Mission Design - Scenario Three (via Lambert Solver)

The third method in this research is using the Lambert Solver based upon Izzo's (Izzo, 2015) paper, as shown earlier in the chapter. As the Lambert Solver can yield various results depending on the constraints, the constrains for this research's Lambert solver is as follows:

1. The number of revolutions for the Lambert Solver is strictly within one revolution of the orbit.
2. The starting point $P1$ is exactly the same as the starting point for other methods, and $P2$ is exactly differentiated by the difference of altitude of $P1$ and $P2$.
3. The time of flight between the two point is differentiated as follows: **0.1x, 0.5x, 1x, 1.5x, and 2x** the time of flight using Hohmann transfer method.

3.12 Mission Analysis - Total Delta V Comparison

Based upon all the scenarios and their sub-cases, the total Δv for each case are calculated. They are first compared to each other for each scenario. Then, the best case from each scenario is compared again, to compare which scenario works best.

3.12.1 Propagation of Orbit With and Without J2 Perturbation

The maneuver are then all propagated using the earlier orbit propagator. For each case, the calculation is done twice, one accounting for J2 Perturbation, and one without. The propagation is then visualized using Matplotlib plotter

3.13 Mission Analysis - Possible Payload Calculation

Finally, using the transfer orbit case with the lowest propellant cost, the payload can that be brought to GEO with that particular launch vehicle can be calculated using the rocket equation:

$$\Delta V = I_{sp} g_0 ln \frac{m_0}{m_f} \tag{3.79}$$

where the value of I_{sp} can be obtained from the launch vehicle's specification.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Mathematical Tools Confirmation

To ensure consistency of the mathematical tools (programs) of this research, the functions used to solve the problem must first be checked. The result of function checking is shown below.

4.1.1 Checking of the Conversion Regime

As the ability to convert from state vectors to classical orbital elements and vice-versa is crucial, the Author needs to verify whether the algorithm is satisfactory. To do so, the author obtained LAPAN-A2's TLE, convert it into COE, convert it into state vectors, and then convert it back into COE. The result of the initial COE and the final COE is then compared to see whether it differs or not.

From LAPAN A2 TLE, the classical orbital elements is as follows:

- **Semi-major Axis**: 7018.095459732759 km
- **Inclination**: 5.9950 deg
- **Eccentricity**: 0.0013975
- **Right Ascension of the Ascending Node**: 340.4753 deg
- **Argument of Perigee**: 268.9107 deg
- **True Anomaly**: 91.1298 deg
- **Semi-parameter**: 7018.081753348462 km

With the code, the resulting state vectors is as follows:

```
rx =  [6655.98110129] ry =  [-2225.7413041] rz =  [13.31194546]
 vx =  [2.38547486] vy =  [7.10516067] vz =  [0.78697455]
```

The state vectors are then reverted back to classical orbital elements and the end result is:

```
Semimajor =  7018.095459732762  km
 Inclination =  5.994999999999943  deg
 Eccentricity =  0.0013974999999999939
 RAAN =  340.47529999999995  deg
 AOG =  269.91080000001494  deg
 True Anomaly =  91.12979999998507  deg
```

As evident, the conversion algorithm has successfully convert an orbit description, from TLE, to COE, to RV, and back to COE again, all within accuracy.

4.1.2 Propagation of LAPAN A2 Orbit With and Without J2

To check the code of orbit propagator, the Author once again uses the LAPAN A2 satellite as an example orbit. Initial orbit data were determined based upon Celestrak's database for LAPAN A2 as of June 16th, 2020. The result of the propagator code and visualized using matplotlib is as follows:

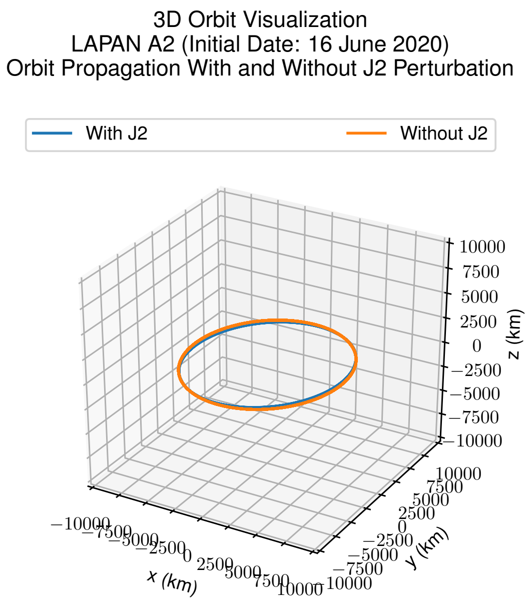


FIGURE 4.1: 3D Orbit Visualization of LAPAN A2 (16th June 2020), With and Without J2 Perturbation - On A Single Graph

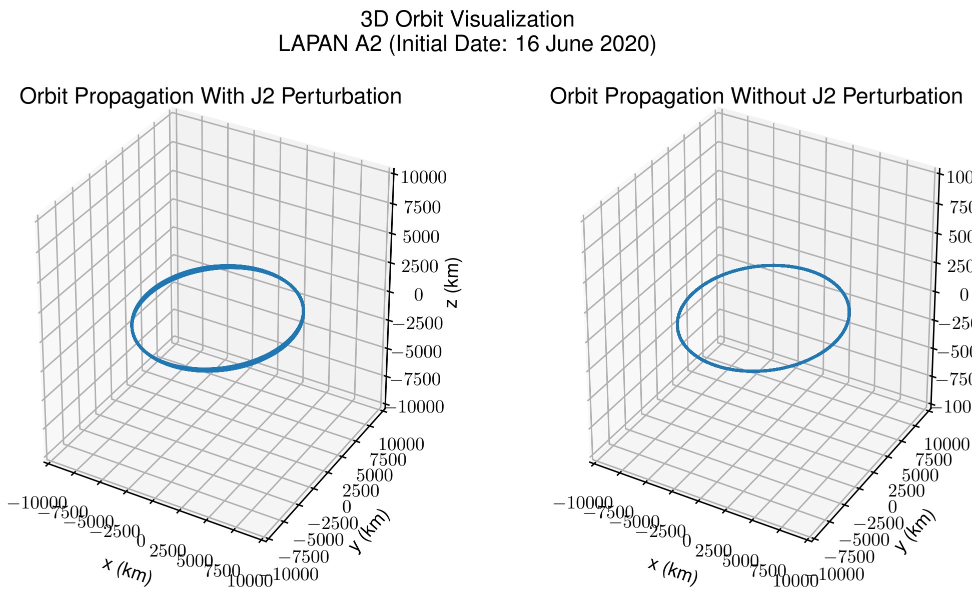


FIGURE 4.2: 3D Orbit Visualization of LAPAN A2 (16th June 2020), With and Without J2 Perturbation - Side-by-Side Comparison

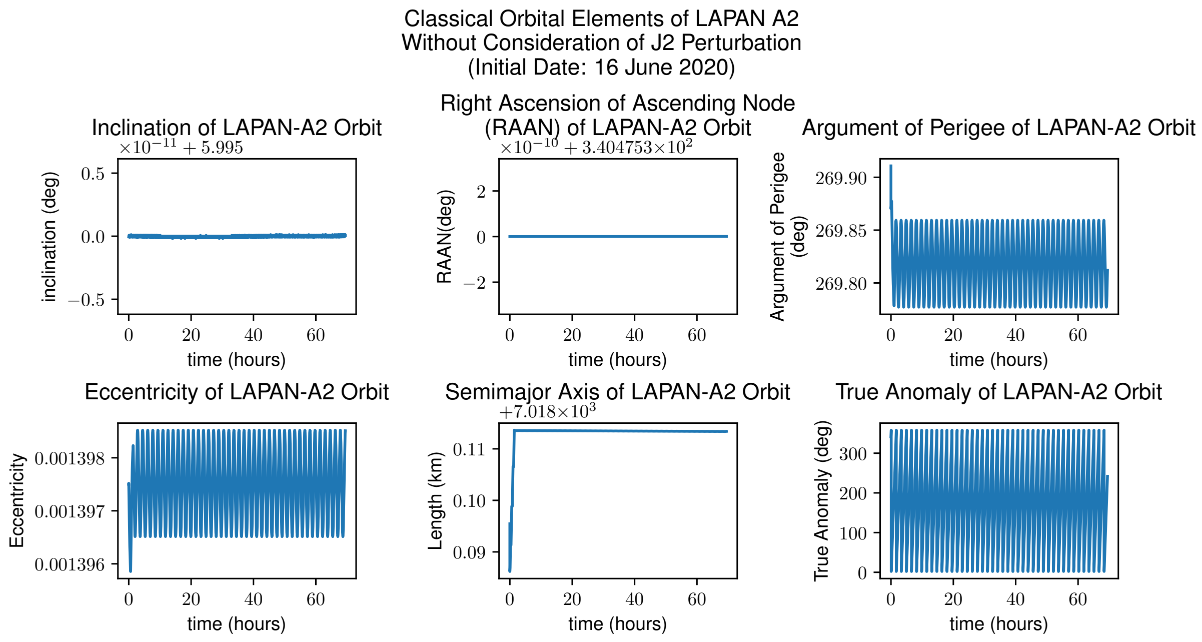


FIGURE 4.3: Classical Orbital Elements of LAPAN A2 (16th June 2020) - Without J2 Perturbation

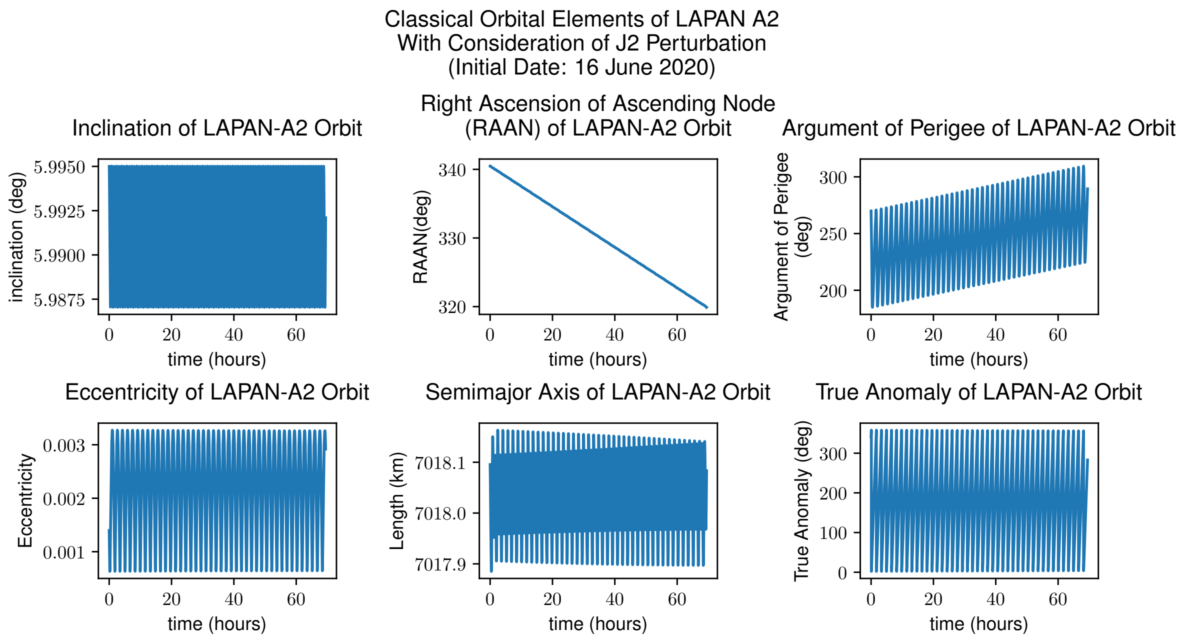


FIGURE 4.4: Classical Orbital Elements of LAPAN A2 (16th June 2020) - With J2 Perturbation

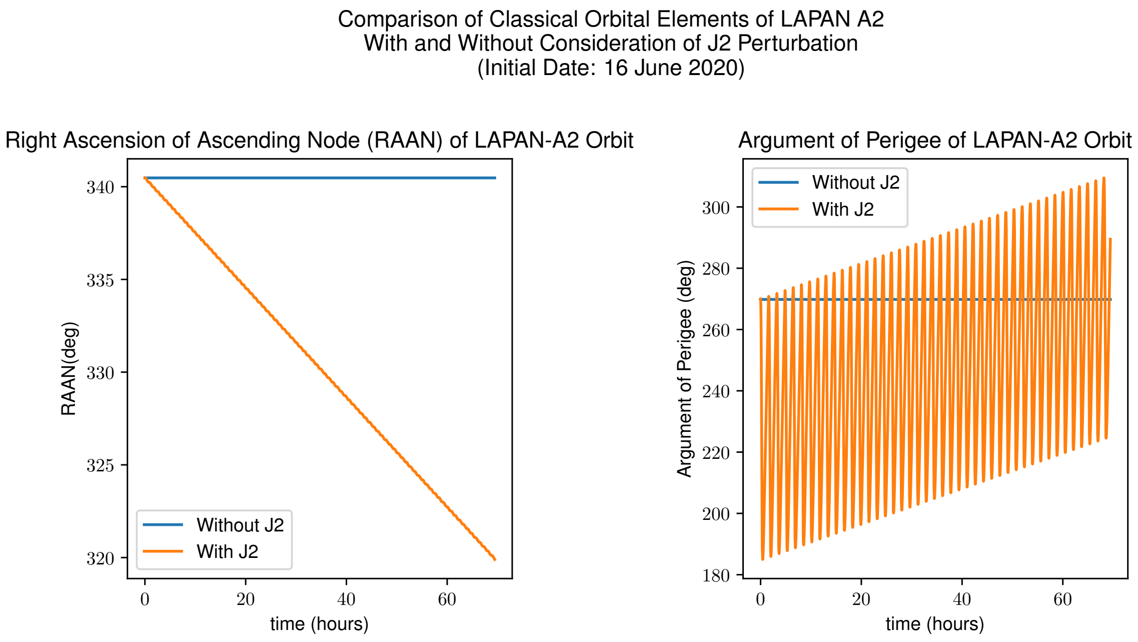


FIGURE 4.5: Comparison of Right Ascension of Ascending Node and Argument of Perigee of LAPAN A2 (16th June 2020), With and Without J2 Perturbation

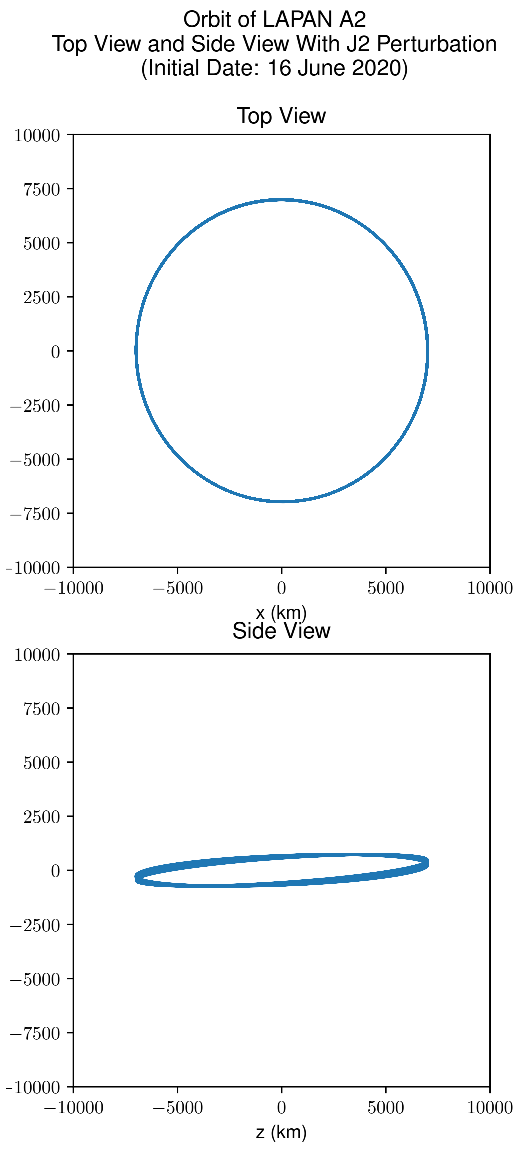


FIGURE 4.6: Top and Side View of LAPAN A2 Orbit (16th June 2020), With J2 Perturbation

The result of the orbit propagator is consistent with the literature (Vallado and McClain, 2013) (Curtis, 2014) denoting that J2 perturbation affects an orbits Right Ascension of Ascending Node and Argument of Perigee. Therefore, the propagator successfully accounts for J2 perturbation.

4.2 Mission Design - Launch Vehicle

4.2.1 Introduction to Epsilon Rocket

In order to conduct this research, the Author needs to select a suitable launch vehicle so its initial Low Earth Orbit can be determined. The following launch vehicles are ones which are considered as small launch vehicle - a type of vehicle that fits within limited budget. The main factor in selecting a launch vehicle is the amount of data regarding its specification that can be publicly accessed. For this research, the Author has chosen the Epsilon Launch Vehicle from Japan Aerospace Exploration Agency. The data regarding this rocket is publicly available and detailed.

The Epsilon Launch Vehicle (Epsilon LV) is one of Japan Aerospace Exploration Agency (JAXA)'s flagship launch vehicle. Its primary focus is to launch small satellites, with the initial goal of providing Japan with small scientific mission capability without relying on outside sources. The Epsilon LV is also being pushed as a commercial alternative. (Agency, 2018)

Epsilon LV is a solid-propellant rocket which is built upon JAXA's proven technologies, such as a former M-V rocket (another solid-fuel rocket) and JAXA's primary launch vehicle, the H-II. In fact, Epsilon is a combination of proven M-V and H-II LV parts, with the first stage coming from a H-IIA booster, and the second and the third stage coming from the M-V LV. So far, this has been proven - the Epsilon LV is a very reliable rocket with 100 % success rate as of 2020. (Agency, 2018)

Another goal of the Epsilon program is to reduce launch cost and reduce deployment time. The Epsilon LV costs only a third of its M-V predecessor, and boast an incredible 7 day time from the installation of the first stage to post-launch processing, compared to 42 days of M-V LV and 40 days of Pegasus LV (USA). The Epsilon LV is also available for both single ride or multi-ride (ride-sharing) configuration, which enable customers wider usage capability. (*Epsilon Launch Vehicle*)

Operation of the Epsilon LV is conducted on Uchinoura Space Center, located in Kagoshima Prefecture, Japan. The Uchinoura Space Center is one of Japan's primary spaceport, focusing on the Epsilon rocket and sounding rocket launches. Another primary space port within Japan

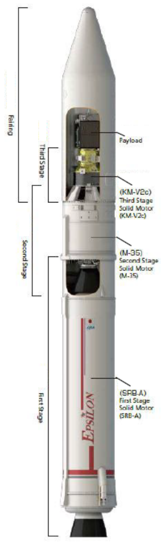

Fairing
Third Stage
Second Stage
First Stage
Payload
(KM-V2c) Third Stage Solid Motor (KM-V2c)
(M-35) Second Stage Solid Motor (M-35)
(SRB-A) First Stage Solid Motor (SRB-A)
EPSILON

FIGURE 4.7: Diagram of Epsilon Launch Vehicle.) Image credit: Agency, 2018

being Tanegashima Space Center, which is currently reserved for heavy launch vehicle. Uchinoura Space Center has a historic and proven lineage of launching rockets, providing a complete launch service of its customers. The coordinate for Epsilon Launch Pad is **31°15'03.5"N 131°04'56.7"E.**

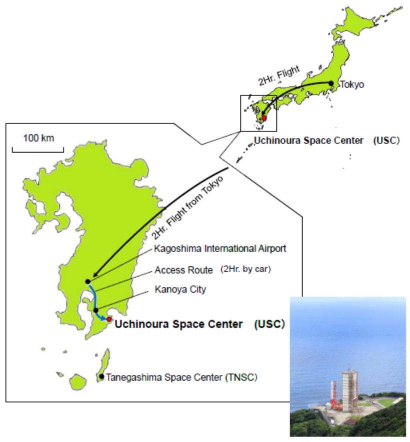

2Hr. Flight
Tokyo
Uchinoura Space Center (USC)
100 km
2Hr. Flight from Tokyo
Kagoshima International Airport
Access Route (2Hr. by car)
Kanoya City
Uchinoura Space Center (USC)
Tanegashima Space Center (TNSC)

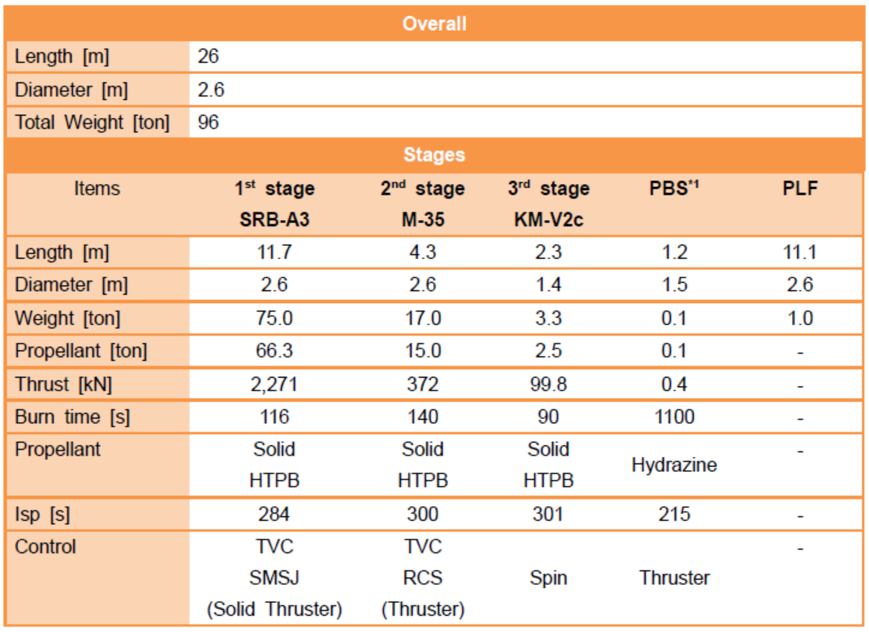
FIGURE 4.8: Uchinoura Space Center, the Launch Site for Epsilon Launch Vehicle. Image credit: Agency, 2018

FIGURE 4.9: Launch of Epsilon Launch Vehicle from Uchinoura Space Center.
Image credit: *Uchinoura Space Center*

For this research, the Epsilon launch vehicle was selected to be the baseline for the possible mission due to information availability is much superior compared to other rocket on its class. This is useful in regard to mission planning, which requires specific and accurate payload capability according to various mission profiles.

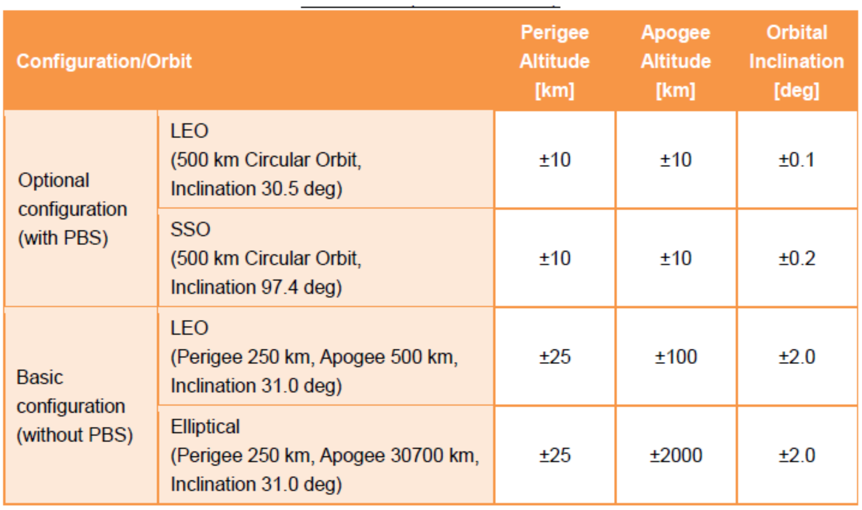
4.2.2 Epsilon Rocket Specification

For this research, the mission is assumed to be a single-ride (the new satellite as the primary and only payload). The specification of the LV is as follows:



Overall					
Length [m]	26				
Diameter [m]	2.6				
Total Weight [ton]	96				
Stages					
Items	1st stage SRB-A3	2nd stage M-35	3rd stage KM-V2c	PBS[*1]	PLF
Length [m]	11.7	4.3	2.3	1.2	11.1
Diameter [m]	2.6	2.6	1.4	1.5	2.6
Weight [ton]	75.0	17.0	3.3	0.1	1.0
Propellant [ton]	66.3	15.0	2.5	0.1	-
Thrust [kN]	2,271	372	99.8	0.4	-
Burn time [s]	116	140	90	1100	-
Propellant	Solid HTPB	Solid HTPB	Solid HTPB	Hydrazine	-
Isp [s]	284	300	301	215	-
Control	TVC SMSJ (Solid Thruster)	TVC RCS (Thruster)	Spin	Thruster	-

FIGURE 4.10: Specification of Epsilon Launch Vehicle. Image credit: Agency, 2018



Configuration/Orbit		Perigee Altitude [km]	Apogee Altitude [km]	Orbital Inclination [deg]
Optional configuration (with PBS)	LEO (500 km Circular Orbit, Inclination 30.5 deg)	±10	±10	±0.1
	SSO (500 km Circular Orbit, Inclination 97.4 deg)	±10	±10	±0.2
Basic configuration (without PBS)	LEO (Perigee 250 km, Apogee 500 km, Inclination 31.0 deg)	±25	±100	±2.0
	Elliptical (Perigee 250 km, Apogee 30700 km, Inclination 31.0 deg)	±25	±2000	±2.0

FIGURE 4.11: Accuracy of the Epsilon Launch Vehicle. Image credit: Agency, 2018

4.3 Initial and Final Orbit Definition

4.3.1 Initial LEO (Parking Orbit) Definition

Based upon the specification of the Epsilon Launch Vehicle above, the configuration with PBS (Post-Boost Stage) for LEO is located at 500 km circular orbit at inclination of 30.5 deg. The launch vehicle have an accuracy of ± 10 km in both apogee and perigee and $\pm 0.1 deg$ for the orbital inclination. Therefore the initial Low Earth Orbit is defined as follows:

- **Semi-major Axis**: 6 878.1 km
- **Inclination**: 30.5 deg
- **Eccentricity**: 0
- **Right Ascension of the Ascending Node**: 0 deg
- **Argument of Perigee**: 0 deg
- **True Anomaly**: 0 deg

4.3.2 Final Geostationary Orbit Definition

The final target orbit in Geosynchronous orbit is in the same plane with the original LEO orbit. Therefore the difference is in it's semi major axis (or radius, as this is a circular orbit).

- **Semi-major Axis**: 42 164.1 km
- **Inclination**: 0 deg
- **Eccentricity**: 0
- **Right Ascension of the Ascending Node**: 0 deg
- **Argument of Perigee**: 0 deg
- **True Anomaly**: 0 deg

4.4 Scenario One Result

4.5 Scenario Two Result

4.6 Scenario Three (Lambert Solver) Result

4.7 Comparison of Each Scenario

4.8 Payload Calculation

4.9 Final Discussion

CHAPTER 5

CONCLUSIONS AND FUTURE WORKS

5.1 Conclusion

- cool
- cooler
- coolest

5.2 Future Work

BIBLIOGRAPHY

Chris55 (Sept. 26, 2015). *Perihelion-Aphelion*. URL: https://commons.wikimedia.org/wiki/File:Perihelion-Aphelion.svg.

Vallado, David A. and Wayne D. McClain (2013). *Fundamentals of astrodynamics and applications*. 4. ed. Space technology library 21. OCLC: 869869951. Hawthorne, Calif: Microcosm Press. 1106 pp. ISBN: 978-1-881883-18-0.

Curtis, Howard D. (2014). *Orbital mechanics for engineering students*. Third edition. Elsevier aerospace engineering series. OCLC: ocn852806044. Amsterdam ; Boston: Elsevier, BH, Butterworth-Heinemann is an imprint of Elsevier. 751 pp. ISBN: 978-0-08-097747-8.

0.39 (Sept. 12, 2004). *Orbital state vectors*. URL: https://commons.wikimedia.org/wiki/File:Orbital_state_vectors.png.

Lasunncty (Oct. 10, 2007). *Orbit1*. URL: https://commons.wikimedia.org/wiki/File:Orbit1.svg.

Federal Aviation Administration (2018). “Describing Orbits”. In: *Advanced Aerospace Medicine On-line*. Federal Aviation Administration.

National Aeronautics and Space Administration. *Catalog of Earth Satellite Orbits*. URL: https://earthobservatory.nasa.gov/features/OrbitsCatalog.

European Space Agency. *Low Earth Orbit*. URL: https://www.esa.int/ESA_Multimedia/Images/2020/03/Low_Earth_orbit.

Caleb Henry (2020). *SpaceX seeks FCC permission for operating all first-gen Starlink in lower orbit*. URL: https://spacenews.com/spacex-seeks-fcc-permission-for-operating-all-first-gen-starlink-in-lower-orbit/.

European Space Agency (2020b). *Types of Orbits*. URL: https://www.esa.int/Enabling_Support/Space_Transportation/Types_of_orbits.

— (2020a). *Geostationary Orbit*. URL: https://www.esa.int/ESA_Multimedia/Images/2020/03/Geostationary_orbit.

Pettit, Don (2012). *The Tyranny of Rocket Equation*. URL: https://www.nasa.gov/mission_pages/station/expeditions/expedition30/tryanny.html.

National Aeronautics and Space Administration. *Ideal Rocket Equation*. URL: https://www.grc.nasa.gov/WWW/K-12/rocket/rktpow.html.

Vallado, David A and Paul J Cefola (2012). "IAC-12-A6.6.11 TWO-LINE ELEMENT SETS – PRACTICE AND USE". In: p. 15.

Transilvania University of Braşov, Braşov, Romania et al. (June 24, 2016). "SATELLITE TRACKING USING NORAD TWO-LINE ELEMENT SET FORMAT". In: *SCIENTIFIC RESEARCH AND EDUCATION IN THE AIR FORCE* 18.1, pp. 423–432. DOI: 10.19062/2247-3173.2016.18.1.58. URL: http://www.afahc.ro/ro/afases/2016/MATH&IT/CROITORU_OANCEA.pdf (visited on 06/30/2020).

Izzo, Dario (Jan. 2015). "Revisiting Lambert's problem". In: *Celestial Mechanics and Dynamical Astronomy* 121.1, pp. 1–15. ISSN: 0923-2958, 1572-9478. DOI: 10.1007/s10569-014-9587-y. URL: http://link.springer.com/10.1007/s10569-014-9587-y (visited on 04/01/2020).

Conway, Bruce A (2010). *Spacecraft trajectory optimization*. OCLC: 664572578. Cambridge; New York: Cambridge University Press. ISBN: 978-0-511-77802-5 978-0-511-90945-0 978-0-511-90666-4 978-0-511-90391-5. URL: http://www.books24x7.com/marc.asp?bookid=36775 (visited on 04/01/2020).

Kechichian, Jean Albert (July 1997). "Optimal Low-Earth-Orbit-Geostationary-Earth-Orbit Intermediate Acceleration Orbit Transfer". In: *Journal of Guidance, Control, and Dynamics* 20.4, pp. 803–811. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/2.4116. URL: https://arc.aiaa.org/doi/10.2514/2.4116 (visited on 06/28/2020).

Darby, Christopher L. and Anil V. Rao (July 2011). "Minimum-Fuel Low-Earth Orbit Aeroassisted Orbital Transfer of Small Spacecraft". In: *Journal of Spacecraft and Rockets* 48.4, pp. 618–628. ISSN: 0022-4650, 1533-6794. DOI: 10.2514/1.A32011. URL: https://arc.aiaa.org/doi/10.2514/1.A32011 (visited on 06/28/2020).

Tsuda, Yuichi et al. (Oct. 2013). "System design of the Hayabusa 2—Asteroid sample return mission to 1999 JU3". In: *Acta Astronautica* 91, pp. 356–362. ISSN: 00945765. DOI: 10.1016/j.actaastro.2013.06.028. URL: https://linkinghub.elsevier.com/retrieve/pii/S009457651300218X.

Matousek, Steve (Nov. 2007). "The Juno New Frontiers mission". In: *Acta Astronautica* 61.10, pp. 932–939. ISSN: 00945765. DOI: 10.1016/j.actaastro.2006.12.013. URL: https://linkinghub.elsevier.com/retrieve/pii/S0094576507000124.

Fitrianingsih, E and R Armellin (Apr. 2018). "Analytical Approach to the Fuel Optimal Impulsive Transfer Problem Using Primer Vector Method". In: *Journal of Physics: Conference Series* 1005, p. 012043. ISSN: 1742-6588, 1742-6596. DOI: 10.1088/1742-6596/1005/1/012043. URL: https://iopscience.iop.org/article/10.1088/1742-6596/1005/1/012043 (visited on 05/04/2020).

Anaconda Inc. *About Anaconda Inc*. URL: https://www.anaconda.com/about-us.

— (2020). *Anaconda Documentation*. URL: https://docs.anaconda.com/anaconda/.

Numpy Inc. *About Numpy*. URL: https://numpy.org/about/.

SciPy Developers. *About SciPy*. URL: https://www.scipy.org/about.html.

Matplotlib Development Team. *Matplotlib*. URL: https://matplotlib.org/.

European Space Agency. *LAPAN-A2*. eoPortal Directory. URL: https://earth.esa.int/web/eoportal/satellite-missions/l/lapan-a2.

Hardhienata, Soewarto, Robertus Heru Triharjanto, and Mohamad Mukhayadi (2011). "LAPAN-A2 : Indonesian Near-Equatorial Surveilance Satellite". In: p. 10.

Juergen Giesen (2016). *Solving Kepler's Equation of Elliptical Motion*. URL: http://www.jgiesen.de/kepler/kepler.html.

SciPy Developers (2020). *Scipy Integrate Solve_ivp*. URL: https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.solve_ivp.html.

Izzo, Dario (Feb. 22, 2019). *esa/pykep: Bug fixes and more support on Equinoctial Elements*. Version v2.3. DOI: 10.5281/ZENODO.2575462. URL: https://zenodo.org/record/2575462 (visited on 07/18/2020).

Juan Luis Cano Rodríguez (2019). *Izzo*. URL: https://docs.poliastro.space/en/stable/api/safe/iod/izzo.html?highlight=lambert#poliastro.iod.izzo.lambert.

Agency, Japan Aerospace Exploration (July 2018). *Epsilon Launch Vehicle User's Manual*.

Japan Aerospace Exploration Agency. *Epsilon Launch Vehicle*. URL: https://global.jaxa.jp/activity/pr/brochure/files/rocket07.pdf.

— *Uchinoura Space Center*. URL: https://www.jaxa.jp/projects/pr/brochure/files/centers04.pdf.

Appendix A: Python Code

```python
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.pyplot as plt

def rv2coe(rx,ry,rz,vx,vy,vz):
    """this function converts from RV to orbital elements"""
    #this is the messy part where the computer actually computes
    #turning the inputed data into NumPy array
    RIJK = np.array([rx, ry, rz])
    VIJK = np.array([vx, vy, vz])

    Rmag = np.linalg.norm(RIJK)
    Vmag = np.linalg.norm(VIJK)

    # calculting angular momentum
    h= np.cross(RIJK,VIJK)
    hmag = np.linalg.norm (h)

    #set unit vector K for usage in calculation
    kunit = np.array ([0,0,1])
    # calculating node vector and its magnitude
    n = np.cross(kunit,h)
    nmag = np.linalg.norm(n)

    #Calculation of eccentricity
    #start by defining mu, and separating the equation into brackets to
    reduce complexity
    mu = 398600
    bracket1 = (Vmag**2 - (mu/Rmag))
    bracket2 = np.dot(RIJK,VIJK)
    bracket3 =(1/mu)
    #calculation of vector quantity of eccentricity and its magnitude
    evector = bracket3*((bracket1*RIJK)-(bracket2*VIJK))
    emag = np.linalg.norm(evector)


    #determination of orbit shape
    if emag < 1:
        shape = "not parabolic"
    else:
        shape = "parabolic"

    #Calculation of specific mechanical energy
    sme = ((Vmag**2/2) - (mu/Rmag))

    #Calculation of  semi major axis
    semimajor = -mu/(2*sme)

    #Calculation of semiparameter
    p = (hmag**2)/mu

    #Calculation of angles
    #inclination angle
    cosi = (h[2])/hmag
    i = np.arccos(cosi)*(180/np.pi)

    #ascending node
    cosomega = n[0]/nmag
    omega = np.arccos(cosomega)*(180/np.pi)
    if n[2]<1:
        checkedomega = 360 - omega
    else:
        checkedomega = omega

    #argument of perigee
    cosArPeri = (np.dot(n,evector)/(nmag*emag))
    ArPeri = np.arccos (cosArPeri)*(180/np.pi)
    if evector[2] < 0:
        checkedArPeri = 360 -ArPeri
    else:
        checkedArPeri = ArPeri

    #true anomaly
    cosv = (np.dot(evector,RIJK))/(emag*Rmag)
    v = np.arccos(cosv)*(180/np.pi)
    if np.dot(RIJK,VIJK) < 0:
        checkedv = 360 - v
    else:
        checkedv = v

    #special cases angle
    cosArPeriTrue = evector[0]/emag
    ArPeriTrue = np.arccos(cosArPeriTrue)*(180/np.pi)
    if evector[1]<0:
        checkedArPeriTrue = 360 - ArPeriTrue
    else:
        checkedArPeriTrue = ArPeriTrue
    tilde = checkedomega+checkedArPeri

    #argument of latitude
    cosu = (np.dot(n,RIJK))/(nmag*Rmag)
    u = np.arccos(cosu)*(180/np.pi)
    if RIJK[2]<0:
        checkedU = 360-u
    else:
        checkedU = u

    #true longitude
    coslambda = RIJK[0]/Rmag
    lambdacoef = np.arccos(coslambda)*(180/np.pi)
    if RIJK[1]<0:
        checkedLambda = 360-lambdacoef
    else:
        checkedLambda = lambdacoef

    #return the result of the function
    return semimajor, i, emag, checkedomega, checkedArPeri, checkedv



def COE2RV(p,e,i,longitudeomega,argumentomega,v):
    """this function converts from orbital elements to RV"""
    #convert from degrees to radians
    longitudeomegarad = longitudeomega/(180/np.pi)
    argumentomegarad =  argumentomega/(180/np.pi)
    vrad = v/(180/np.pi)
    irad = i/(180/np.pi)

    #calculate the elements for matrices of R and V

    mu = 398600

    r1upper = p*np.cos(vrad)
    r1lower = 1+(e*np.cos(vrad))
    r1 = (r1upper/r1lower)
    r2 =((p*np.sin(vrad))/(1+(e*np.cos(vrad))))
    r3 = 0

    v1 = -(np.sqrt(mu/p))*np.sin(vrad)
    v2 = (np.sqrt(mu/p))*(e+np.cos(vrad))
    v3 = 0

    #creating the transformation matrix
    #elements are numbered from top left, go right, until bottom right
    t11 =(np.cos(longitudeomegarad)*np.cos(argumentomegarad))-(np.sin(
   longitudeomegarad)*np.sin(argumentomegarad)*np.cos(irad))
    t12 =(-np.cos(longitudeomegarad)*np.sin(argumentomegarad))-((np.sin(
   longitudeomegarad))*np.cos(argumentomegarad)*np.cos(irad))
    t13 =(np.sin(longitudeomegarad)*np.sin(irad))
    t21 =(np.sin(longitudeomegarad)*np.cos(argumentomegarad))+(np.cos(
   longitudeomegarad)*np.sin(argumentomegarad)*np.cos(irad))
    t22 =(-np.sin(longitudeomegarad)*np.sin(argumentomegarad))+(np.cos(
   longitudeomegarad)*np.cos(argumentomegarad)*np.cos(irad))
    t23 =(-np.cos(longitudeomegarad)*np.sin(irad))
    t31 =np.sin(argumentomegarad)*np.sin(irad)
    t32 =np.cos(argumentomegarad)*np.sin(irad)
    t33 =np.cos(irad)

    transform = np.array([[t11,t12,t13],[t21,t22,t23],[t31,t32,t33]])

    #creating the R and V matrix
    matrixR = np.array([[r1],[r2],[r3]])
    matrixV = np.array([[v1],[v2],[v3]])

    #transforming the R and V matrix
    finalr = np.dot(transform, matrixR)
    finalv = np.dot(transform, matrixV)

    #return the results
    return finalr, finalv

def gradienttwobody(t, ybar):
    """
    IVP solver for orbit propagator
    Keyword Arguments:
    t -- time
    ybar -- array of initial values
    """
    mu = 398600.4
    x = (ybar[0])
    y = (ybar[1])
    z = (ybar[2])
    sqrtr = ((x**2)+(y**2)+(z**2))
    r = np.sqrt(sqrtr)

    x1 = ybar[3]
    x2 = ybar[4]
    x3 = ybar[5]
    x1dot = (-(mu)/(r**3))*x
    x2dot = (-(mu)/(r**3))*y
    x3dot = (-(mu)/(r**3))*z

    ybar_dot = np.array([x1,x2,x3,x1dot,x2dot,x3dot])

    return ybar_dot

def hohmannCircular(initialKm, finalKm):
    """
    Hohmann transfer calculator for circular orbits
    Keyword arguments:
    initialKm -- initial orbit altitude in Km
    finalKm -- final desired after burn altitude in km
    """
    # Variable declaration
    EarthRadius = 6378.137  # in km
    mu = 1
    TU = 806.80415 # in seconds

    # Convert from km into canonical units
    rInitial = ((initialKm + EarthRadius)/EarthRadius)
    rFinal = ((finalKm + EarthRadius)/EarthRadius)

    # Finding initial velocity at points in the transfer
    vInitial = np.sqrt((mu/rInitial))
    vFinal = np.sqrt((mu/rFinal))

    # Transfer Orbit semi-major axis
    aTrans = ((rInitial+rFinal)/2)

    # Finding transfer velocities
    vTransA = np.sqrt( ((2*mu)/rInitial) -(mu/aTrans) )
    vTransB = np.sqrt( ((2*mu)/rFinal) -(mu/aTrans) )

    # Final deltaV
    deltaVCanonical = np.abs(vTransA - vInitial) + np.abs(vFinal - vTransB)
    deltaVTotal = deltaVCanonical*(EarthRadius/TU)

    # Finding Time of Flight
    timeOfFlight = np.pi* (aTrans**(3/2))

    return deltaVTotal, timeOfFlight

def BiEllipticCircular (initialKm, transferKm, finalKm):
    """
    BiElliptical tranfer calculator for circular orbits
    Keyword arguments:
    initialKm -- initial orbit altitude in km
    transferKm -- altitude of transfer point, must be bigger than final
    altitude, in Km
    finalKm -- final desired after burn altitude in km
    """
    # Variable declaration
    EarthRadius = 6378.137  # in km
    mu = 1
    TU = 806.80415 # in seconds

    # Convert from km into canonical units
    rInitial = ((initialKm + EarthRadius)/EarthRadius)
    rTransfer = ((transferKm + EarthRadius)/EarthRadius)
    rFinal = ((finalKm + EarthRadius)/EarthRadius)

    # Finding initial velocity at points in the transfer
    vInitial = np.sqrt((mu/rInitial))
    vFinal = np.sqrt((mu/rFinal))

    # Transfer Orbit semi-major axis
    aTrans1 = (rInitial+rTransfer)/2
    aTrans2 = (rTransfer+rFinal)/2

    # Finding transfer velocities
    vTrans1a = np.sqrt( ((2*mu)/rInitial) - (mu/aTrans1)  )
    vTrans2c = np.sqrt( ((2*mu)/rFinal) - (mu/aTrans2) )
    vTrans1b = np.sqrt( ((2*mu)/rTransfer) - (mu/aTrans1) )
    vTrans2b = np.sqrt(  ((2*mu)/rTransfer) - (mu/aTrans2) )

    # Final deltaV
    deltaVCanonical = np.abs(vTrans1a - vInitial) + np.abs(vTrans2b -
    vTrans1b ) + np.abs(vFinal-vTrans2c)
    deltaVTotal = deltaVCanonical*(EarthRadius/TU)

    # Finding Time of Flight
    timeOfFlight = np.pi* (aTrans1**(3/2)) + np.pi * (aTrans2**(3/2))
    return deltaVTotal, timeOfFlight
```

```

def rv2coeCircular (rx,ry,rz,vx,vy,vz):
    """
    RV2COE function for circular orbits
    """
    #this is the messy part where the computer actually computes
    #turning the inputed data into NumPy array
    RIJK = np.array([rx, ry, rz])
    VIJK = np.array([vx, vy, vz])

    Rmag = np.linalg.norm(RIJK)
    Vmag = np.linalg.norm(VIJK)

    # calculting angular momentum
    h= np.cross(RIJK,VIJK)
    hmag = np.linalg.norm (h)

    #set unit vector K for usage in calculation
    kunit = np.array ([0,0,1])
    # calculating node vector and its magnitude
    n = np.cross(kunit,h)
    nmag = np.linalg.norm(n)

    #Calculation of eccentricity
    #start by defining mu, and separating the equation into brackets to
    reduce complexity
    mu = 398600
    bracket1 = (Vmag**2 - (mu/Rmag))
    bracket2 = np.dot(RIJK,VIJK)
    bracket3 =(1/mu)
    #calculation of vector quantity of eccentricity and its magnitude
    evector = bracket3*((bracket1*RIJK)-(bracket2*VIJK))
    emag = np.linalg.norm(evector)


    #determination of orbit shape
    if emag < 1:
        shape = "not parabolic"
    else:
        shape = "parabolic"

    #Calculation of specific mechanical energy
    sme = ((Vmag**2/2) - (mu/Rmag))

    #Calculation of  semi major axis
    semimajor = -mu/(2*sme)

    #Calculation of semiparameter
    p = (hmag**2)/mu

    #Calculation of angles
    #inclination angle
    cosi = (h[2])/hmag
    i = np.arccos(cosi)*(180/np.pi)

    #ascending node
    cosomega = n[0]/nmag
    omega = np.arccos(cosomega)*(180/np.pi)
    if n[2]<1:
        checkedomega = 360 - omega
    else:
        checkedomega = omega

    #argument of perigee
    cosArPeri = (np.dot(n,evector)/(nmag*emag))
    ArPeri = np.arccos (cosArPeri)*(180/np.pi)
    if evector[2] < 0:
        checkedArPeri = 360 -ArPeri
    else:
        checkedArPeri = ArPeri

    #true anomaly
    cosv = (np.dot(evector,RIJK))/(emag*Rmag)
    v = np.arccos(cosv)*(180/np.pi)
    if np.dot(RIJK,VIJK) < 0:
        checkedv = 360 - v
    else:
        checkedv = v

    #special cases angle
    cosArPeriTrue = evector[0]/emag
    ArPeriTrue = np.arccos(cosArPeriTrue)*(180/np.pi)
    if evector[1]<0:
        checkedArPeriTrue = 360 - ArPeriTrue
    else:
        checkedArPeriTrue = ArPeriTrue
    tilde = checkedomega+checkedArPeri

    #argument of latitude
    cosu = (np.dot(n,RIJK))/(nmag*Rmag)
    u = np.arccos(cosu)*(180/np.pi)
    if RIJK[2]<0:
        checkedU = 360-u
    else:
        checkedU = u

    #true longitude
    coslambda = RIJK[0]/Rmag
    lambdacoef = np.arccos(coslambda)*(180/np.pi)
    if RIJK[1]<0:
        checkedLambda = 360-lambdacoef
    else:
        checkedLambda = lambdacoef

    #return the result of the function
    return semimajor, i, emag, checkedomega, checkedArPeri, checkedLambda

def hohmannCircularDetailed(initialKm, finalKm):
    """
    Hohmann transfer calculator for circular orbits
    Shows results in deltaV a, deltaV b, time of flight
    Keyword arguments:
    initialKm -- initial orbit altitude in Km
    finalKm -- final desired after burn altitude in km
    """
    # Variable declaration
    EarthRadius = 6378.137  # in km
    mu = 1
    TU = 806.80415 # in seconds

    # Convert from km into canonical units
    rInitial = ((initialKm + EarthRadius)/EarthRadius)
    rFinal = ((finalKm + EarthRadius)/EarthRadius)

    # Finding initial velocity at points in the transfer
    vInitial = np.sqrt((mu/rInitial))
    vFinal = np.sqrt((mu/rFinal))

    # Transfer Orbit semi-major axis
    aTrans = ((rInitial+rFinal)/2)

    # Finding transfer velocities
    vTransA = np.sqrt( ((2*mu)/rInitial) -(mu/aTrans) )
    vTransB = np.sqrt( ((2*mu)/rFinal) -(mu/aTrans) )

    # Final deltaV
    deltaVA = (np.abs(vTransA - vInitial))*(EarthRadius/TU)
    deltaVB = (np.abs(vFinal - vTransB))*(EarthRadius/TU)

    # Finding Time of Flight
    timeOfFlight = np.pi* (aTrans**(3/2))

    return deltaVA, deltaVB, timeOfFlight

def gradienttwobody_J2(t, ybar):
    """
    IVP solver for orbit propagator, taking into account J2
    Keyword Arguments:
    t -- time
    ybar -- array of initial values
    """

    mu = 398600.4             # for Earth
    j2_const = 0.00108262668
    er  = 6371                # Earth radius in km

    x = (ybar[0])
    y = (ybar[1])
    z = (ybar[2])
    sqrtr = ((x**2)+(y**2)+(z**2))
    r = np.sqrt(sqrtr)

    x1 = ybar[3]
    x2 = ybar[4]
    x3 = ybar[5]

    const = ((3*j2_const*mu*(er**2))/(2*(r**5)))
    ai = -const*x*(1 - (5*(z**2))/(r**2))
    aj = -const*y*(1 - (5*(z**2))/(r**2))
    ak = -const*z*(3 - (5*(z**2))/(r**2))
    x1dot = (-(mu)/(r**3))*x + ai
    x2dot = (-(mu)/(r**3))*y + aj
    x3dot = (-(mu)/(r**3))*z + ak

    ybar_dot = np.array([x1,x2,x3,x1dot,x2dot,x3dot])

    return ybar_dot
```