ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

DEVELOPMENT OF A COMPREHENSIVE SIMULATION SOFTWARE FOR SPACECRAFT MISSIONS

M.Sc. THESIS

Emirhan Eser Gül

Institue of Science and Technology

Defense Technologies

JANUARY 2023

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

DEVELOPMENT OF A COMPREHENSIVE SIMULATION SOFTWARE FOR SPACECRAFT MISSIONS

M.Sc. THESIS

Emirhan Eser Gül
(514191007)

Institue of Science and Technology

Defense Technologies

Thesis Advisor: Prof. Dr. Alim Rüstem ASLAN

JANUARY 2023

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

UZAY ARACI GÖREVLERİ İÇİN KAPSAMLI BİR SİMÜLASYON YAZILIMI GELİŞTİRİLMESİ

YÜKSEK LİSANS TEZİ

Emirhan Eser Gül
(514191007)

Fen Bilimleri Enstitüsü

Savunma Teknolojileri Programı

Tez Danışmanı: Prof. Dr. Alim Rüstem ASLAN

OCAK 2023

Emirhan Eser Gül, a M.Sc. student of ITU Graduate School student ID 514191007 successfully defended the thesis entitled "DEVELOPMENT OF A COMPREHENSIVE SIMULATION SOFTWARE FOR SPACECRAFT MISSIONS", which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Prof. Dr. Alim Rüstem ASLAN** Istanbul Technical University	.....
Jury Members :	**Prof. Dr. Name SURNAME** Istanbul Technical University	.....
	Prof. Dr. Name SURNAME Sabanci University	.....
		.....

Date of Submission : **30 December 2022**
Date of Defense : **21 January 2023**

"Life, forever dying to be born afresh, forever young and eager, will presently stand upon this Earth as upon a footstool, and stretch out its realm amidst the stars."
-H.G. Wells

FOREWORD

I am honored to present this thesis, which I have prepared under the guidance of my esteemed professor, Prof Dr. Alim Rüstem Aslan. I am deeply grateful to him for his invaluable guidance, support, mentorship, and encouragement throughout my studies. It has been an absolute pleasure working with him over the past 5 years, and I am forever grateful for the opportunities, knowledge and expertise he has shared with me.

I would like to express my sincere gratitude to all professors of Faculty of Aeronautics and Astronautics, especially Asst. Prof. Cuma Yarım whose support I could always rely on. Without the invaluable knowledge about orbital mechanics he has shared with me, this work would not have been possible.

I would also like to thank my colleagues, members of the Space Systems Design and Testing Laboratory, especially Boğaç Karabulut, for their support and assistance during the course of my research. The sleepless nights I spent in the laboratory with Onur Öztekin and Aybüke Ağırbaş, who have been my friends since undergraduate years, were some of the highlights of my student years.

I am also deeply grateful to my father Cevdet Gül, my mother İnci Gül, and my brother Emrehan Gül for their unwavering support and love throughout my life. Their moral and material support has been invaluable, and I could not have achieved this without their constant encouragement and belief in me.

I am fortunate to have my girlfriend, Ecem, who has always been there for me through all of my decisions and ordeals. Her love and understanding have persistently kept me motivated and inspired, and without doubt has played an integral role in my success.

Finally, I would like to give my sincerest appreciation to my best friends, Burak and Öykü for being a constant source of support and heartening throughout my journey, even in the darkest of times. Their friendship means the world to me, and I am so thankful to have them by my side.

I hope that this thesis will contribute to the future of space missions and make a meaningful impact on the wider community. To quote my favorite writer, "Laudato Si Ad Astra", as all the inspiration and motivation for the work I put in this field comes from the glimmering stars that make the face of heaven so incredible to behold.

January 2023 Emirhan Eser Gül

TABLE OF CONTENTS

ABBREVIATIONS

AU : Astronomical Unit
BCRF : Barycenter of the Solar System
BER : Bit Error Rate
CEP : Celestial Ephemeris Pole
CIO : Celestial Intermediate Origin
CIP : Celestial Intermediate Pole
CNN : Convolutional Neural Networks
CRTBP : Circular Restricted Three Body Problem
CSSI : Center for Space Standards and Innovation
EOP : Earth Orientation Parameters
GCRF : Geocentric Celestial Reference Frame
GEO : Geosynchronous Orbit
GSD : Ground Sampling Distance
GUI : Graphical User Interface
HPOP : High-Precision Orbit Propagator
IAU : International Astronomical Union
ICRS : International Celestial Reference System
IERS : International Earth Rotation and Reference Systems Service
ISWA : Integrated Space Weather Analysis System
ITRS : International Terrestrial Reference System
IXRD : Improved X-Ray Detector
JD : Julian Date
LEO : Low-Earth Orbit
LNA : Low Noise Amplifier
LOS : Line of Sight
LSTM : Long-Short Term Memory
LTAN : Local Time of the Ascending Node
MEME : Mean Equator and Mean Equinox
MSE : Mean Square Error
RAAN : Right Ascension of the Ascending Node
RMS : Root-Mean Square
RNN : Recurrent Neural Networks
SRP : Solar Radiation Pressure
SSO : Sun-Synchronous Orbit
TEME : True Equator and Mean Equinox
TIO : Terrestrial Intermediate Origin
TLE : Two-Line Element
TOF : Time of Flight
UHF : Ultra High Frequency
UTC : Coordinated Universal Time
VHF : Very High Frequency
VLBI : Very-long Baseline Interferometry

SYMBOLS

a	: Semi-major axis
e	: Eccentricity
i	: Inclination
Ω	: Right ascension of the ascending node
ω	: Argument of perigee
θ	: True anomaly
h	: Altitude
E	: Eccentric Anomaly
M	: Mean Anomaly
λ	: Longitude
ϕ	: Latitude
f	: Flattening of Earth
β	: Flight path angle
R_E	: Radius of Earth
r_p	: Radius of periapsis
r_a	: Radius of apoapsis
G	: Universal Gravitational Constant
μ	: Gravitational parameter
J_2, J_3	: Second and third gravitational zonal harmonics of the Earth
ρ	: Density
C_d	: Drag coefficient
C_r	: Radiation pressure coefficient
$\mathbf{c}$	: Speed of light
w_E	: Earth rotation rate
t	: Time
ΔV	: Velocity change

LIST OF TABLES

LIST OF FIGURES

DEVELOPMENT OF A COMPREHENSIVE SIMULATION SOFTWARE FOR SPACECRAFT MISSIONS

SUMMARY

The growth of satellite mission, especially CubeSats, in terms of complexity and capabilities has required the development of dedicated orbit simulation software for mission planning and analysis. This thesis presents the development and uses of a simulation software that will be used to aid in the design of spacecraft missions. The process of developing the software architecture is described in stages from software requirement analysis to test and verification of the final implementation.

During a space mission, a spacecraft may be placed in a variety of orbits for different purposes. Preliminary mission design needs to consider all mission phases to meet the needs of more complex missions. To effectively design an orbit, it is important to clearly define the purpose of the orbit and regularly review and reassess this purpose as mission requirements evolve or become more defined. It is also important to consider alternative orbit designs, as there may be multiple options that are viable. For example, a single large satellite in a geosynchronous orbit or a group of smaller satellites in low-Earth orbit may both be effective for communication purposes. Multiple different designs are often compared to find the orbit that best accommodate the mission requirements.

There are various criteria that have to be considered according to the mission, such as determining the communication links between satellites and ground stations, and finding the time intervals when there is a pass or eclipse, which allow determining the requirements of communication and power systems. For an Earth-observing satellite, the orbit that has the most revisit time for desired locations and properties of the optical system such as the field of view should be determined.

The aim of this work is to make use of the software tools to create a simulation software that provides a framework for efficient analysis and planning of satellite missions that include earth observation, communication, and scientific objectives in order to helps the mission design process by giving the ability to make fast and reliable decisions regarding the satellite system requirements.

The developed software implements multiple orbit propagators, with the most prominent being the High-precision Orbit Propagator (HPOP) which takes into account all of the forces that can be modelled so far. However, physics-based models alone are insufficient for accurately predicting orbits and avoiding collisions, as demonstrated by previous collisions caused by such predictions. Our knowledge of the physical world is not sufficient enough to create perfect models as it is near impossible to predict some perturbations precisely, such as solar activity which is only an approximation based on statistical data, as well as the atmosphere models and the area of the satellite that drag

force affects, which also change depending on the attitude model. In order to improve the accuracy of these models, a machine-learning approach that utilizes the past flight data is proposed. Models of orbit prediction errors can be learned directly from a large amount of historical data, allowing for predictions without explicitly modeling forces or perturbations. Hence, a neural-networks model was trained and its impact was demonstrated.

The software is developed using various programming languages. The user interface is programmed in JavaScript, using HTML and CSS. Orbital analyses and other computation heavy tasks were performed in C++ as it has the benefits of modular design, less resource use and fast execution speed, as well as good portability. Python was used for model training and artificial intelligence methods due to the enormous number of scientific libraries it includes. Electron framework was used to provide cross-platform compatibility. For real-time data visualization in both 2D and 3D, the Cesium framework was implemented using Bing as data provider for satellite imagery and terrain modelling. The simulation results show that the software succeeds in attaining high execution speed and precision.

The results indicate that the proposed solution can be useful in reducing time and effort put into the mission design process as well as increase the rate of success for both Earth and interplanetary missions. The developed software can be used for real-world mission design and operations, as a tool for education and engineering studies, and public engagement.

The structure of the thesis is as follows. First the mathematical background and celestial relationships that are widely used in mission planning are explained along with the algorithms used to implement them into the software. These include coordinate system transformations, various orbital elements, and time systems. Then orbital propagation is explained starting from two-body motion, which is the basis for all equations of motion, followed by adding perturbations and other forces acting on the satellite to facilitate the high-precision numerical propagator. Analytical propagators that are widely used are explained as well. Then, the design and implementation of a neural-networks model that is trained to improve the accuracy of the numerical propagators is described. The next chapter focuses on the simulation environment and the capabilities of the software. It is possible to easily create mission-specific orbits such as SSO, GEO, and Molniya, predict the visibility of satellites from different locations on the ground, determine the eclipse intervals, compute communication link budget, analyze on-orbit power generation, and perform basic maneuvers within the simulation environment. The software principles, architecture, development process, and user interface design is thoroughly explained as well. Finally, verifications using real data and satellite observations are performed and results are presented, which show that the developed software is ready for real mission use, and possible future developments are discussed.

UZAY ARACI GÖREVLERİ İÇİN KAPSAMLI BİR SİMÜLASYON YAZILIMI GELİŞTİRİLMESİ

ÖZET

Uydu görevlerinin, özellikle Küp Uyduların, karmaşıklık ve kapsam açısından yüksek oranda artması, görev planlama ve analiz için özelleştirilmiş bir yörünge simülasyon yazılımınının geliştirilmesi ihtiyacını ortaya çıkarmıştır. Bu tez, uzay aracı görevlerinin tasarımına yardımcı olacak bir simülasyon yazılımın geliştirilmesi ve kullanımını sunmaktadır. Yazılım mimarisinin geliştirilme süreci, yazılım gereksinimleri analizinden, son uygulamanın test ve doğrulamasına kadar aşamalar şeklinde açıklanmıştır.

Uzay görevleri sırasında, bir uzay aracı farklı amaçlar için çeşitli yörüngelere yerleştirilebilir. Görev ön tasarımı, karmaşık görevlerin ihtiyaçlarını karşılamak için tüm görev aşamalarını dikkate almayı gerektirir. Bir yörüngeyi etkili bir şekilde tasarlamak için, yörünge amacını net bir şekilde tanımlamak ve görev gereksinimlerinin evrilmesi ya da daha net bir hale gelmesiyle bu amacı sık sık gözden geçirip yeniden değerlendirmek önemlidir. Ayrıca, bir görev için birden çok geçerli seçenek olabilir, bu nedenle alternatif yörünge tasarımlarını da dikkate almak gerekmektedir. Örneğin, iletişim amaçları için, yer eşzamanlı yörüngeye yerleştirilen tek bir büyük uydu ile alçak Dünya yörüngesine yerleştirilen bir grup daha küçük uydunun her ikisi de etkili olabilir. Çoğu zaman, farklı tasarımları birbirleriyle karşılaştırmak ve görev gereksinimlerine en uygun yörüngeyi tespit etmek gerekmektedir.

Bunların yanı sıra, göreve bağlı olarak, tasarım esnasında göz önünde bulundurulması gereken türlü kriter bulunmaktadır. Görev için en uygun yörüngenin belirlenmesi, uydunun yörünge üzerinde ne kadar güç üretimi gerçekleştirebileceğinin analizi, kullanılacak iletişim sistemlerinin güç ve veri aktarma hızı gereksinimlerinin belirlenmesi, atmosfer sürtünmesi ve güneş ışınım basıncı gibi olguların yörünge ömrüne olan etkilerinin hesaplanması gibi faktörler tasarımı önemli ölçüde etkilemektedir. İnsanların bütün bu hesaplamaları yapması mümkün olsa da göz önünde bulundurulması gereken çok fazla değişken olması ve tasarımdaki bir değişikliğin diğer kararları etkilemesi sonucu birçok hesabın tekrar yapılması gerekmesi hem hata yapmayı çok olası kılmakta hem de gereken iş gücünü çok arttırmaktadır. Genellikle uydu tasarımı üzerine çalışılırken bu tarz analizlere yeterince zaman ayrılamamaktadır.

Bu çalışmanın amacı, yazılım araçlarını kullanarak Dünya gözlemi, iletişim ve bilimsel hedefleri içeren uydu görevlerinin etkin bir şekilde analizi ve planlaması için altyapı sağlayacak bir simülasyon yazılımı geliştirmektir. Bu sayede, uydu sistemleri gereksinimleriyle ilgili hızlı ve güvenilir kararlar verilerek görev tasarım sürecinin kolaylaştırılabilmesi hedeflenmektedir.

Geliştirilen yazılım, şu ana kadar geliştirilmiş en hassas model olan Yüksek Doğruluklu Yörünge İlerleticisi (HPOP) dahil, birden çok yörünge ilerletici içermektedir. Ancak, son yıllarda meydana gelen çarpışmaları göz önünde bulundurunca, fizik temelli modellerin yörüngeleri tek başlarına yeterince doğru bir şekilde tahmin edemedikleri ve çarpışmaların önlenmesi için yeterli olmadıkları görülmektedir. Fiziksel dünya hakkındaki bilgimiz mükemmel modeller oluşturmak için yetersizdir, örneğin güneş aktivitesi gibi yalnızca istatistiksel verilere dayalı bazı bozulmaların doğru bir şekilde tahmin edilmesi neredeyse imkansızdır, benzer şekilde atmosfer modelleri ve sürükleme kuvvetinin etkilediği uydu alanı da yönelim modellerine göre değişmektedir. Bu çalışma kapsamında, bu modellerin doğruluğunu iyileştirmek için geçmiş uçuş verilerini kullanan bir makine öğrenimi yaklaşımı önerilmektedir. Elimizde bulunan yüksek miktardaki tarihi uydu verisi kullanılarak, tahmin edilen yörüngelerdeki hata modelleri öğrenilebilir ve böylece ön görülemeyen bozuntu ve kuvvetlerin açıkça modellenmelerine ihtiyaç duyulmaksızın yörünge tahminleri iyileştirilebilir. Bu doğrultuda bir yapay sinir ağı modeli eğitilmiş ve etkisi gösterilmiştir.

Yazılım, çeşitli programlama dilleri kullanılarak geliştirilmiştir. Kullanıcı arayüzü JavaScript, HTML ve CSS kullanılarak programlanmıştır. Yörünge analizleri ve diğer yüksek işlem gücü gerektiren hesaplamalarda, modüler tasarım, daha az kaynak kullanımı ve yüksek işlem hızı gibi avantajlardan dolayı C++ tercih edilmiştir. Yapay zeka yöntemleri ve model eğitimi için içerdiği çok sayıdaki bilimsel kütüphane ve destekten dolayı Python kullanılmıştır. Platformlar arası uyumluluk sağlamak için Electron motoru, gerçek zamanlı 2B ve 3B veri görselleştirmesi için Cesium motoru, uydu görüntüleri ve arazi modelleme işleri için veri sağlayıcı olarak Bing kullanıldı. Simülasyon sonuçları, yazılımın yüksek işlem hızı ve doğruluğa sahip olduğunu ortaya koymaktadır.

Sonuçların gösterdiğine göre, bu çalışmada ortaya çıkarılan çözüm görev tasarım sürecine konulan zaman ve uğraşı azaltıp, başarı oranlarını arttırma potansiyeline sahiptir. Geliştirilen yazılım gerçek görevlerin tasarımı ve yürütülmesi için kullanılmanın yanı sıra mühendislik çalışmaları ve eğitim için bir araç olarak da kullanılabilecektir.

Tezin yapısı şu şekildedir. Önce, görev planlamasında sıkça kullanılan matematiksel bağıntılar ve göksel ilişkiler açıklanmış, bu bağıntıları yazılıma eklemekte kullanılan algoritmalara yer verilmiştir. Bunlar koordinat sistemleri dönüşümleri, çeşitli yörünge elemanları ve uzay alanında kullanılan zaman sistemleri dönüşümlerini içermektedir. Daha sonra, hareket denklemlerinin temelini oluşturan iki-cisim denkleminden yola çıkarak yazılımda yer alan çeşitli yörünge ilerleticiler anlatılmıştır. Bu denklemin üzerine bozuntular ve diğer kuvvet modelleri eklenerek yüksek doğruluklu yörünge ilerletici formüle edilmiş, yaygın olarak kullanılan analitik ve sayısal ilerleticiler hakkında bilgiler verilmiştir. Sonrasında tez kapsamında geliştirilen ve yörünge ilerleticilerin doğruluğunu arttırmak için kullanılması hedeflenen yapay sinir ağı modelinin teorisi ve uygulanışı açıklanmıştır. Sonraki bölümde simülasyon ortamı ve yazılımın kabiliyetleri anlatılmaktadır. SSO, GEO, Molniya gibi göreve odaklı yörüngelerin hızlıca tanımlanması, uyduların Dünya üzerindeki çeşitli noktalardan görüldüğü zaman aralıklarının tespit edilmesi, uydunun güneş ışığı almadığı

zamanların belirlenmesi, iletişim link bütçesi hesaplanması, yörünge üzerindeki güç üretimi ve basit yörünge manevralarının gerçekleştirilmesi gibi simülasyon kapsamındaki özellikler hakkında bilgiler verilmiştir. Yazılım geliştirilirken takip edilen ilkeler, yazılım mimarisinin tanımlanması, yazılım geliştirme süreci ve kullanıcı arayüzü tasarımı detaylıca anlatılmıştır. Nihayetinde, gerçek veri ve uydu dinlemeleri kullanılarak doğrulamalar gerçekleştirilmiş ve elde edilen sonuçlar sunularak yazılımın gerçek görevler için kullanılmaya hazır olduğu gösterilmiş, gelecekte yapılabilecek geliştirmeler tartışılmıştır.

1. INTRODUCTION

This work is conducted in Istanbul Technical University – Space Systems Design and Testing Laboratory (SSDTL), where a total of 7 CubeSats were developed by graduate and undergraduate students and inserted into orbit.

Developing a satellite has a lot of cost associated with it, usually exceeding 100 million dollars due to the various sophisticated and convoluted technologies they include. There are various constraints on space missions including power, communication, sunlight, weather, sensor resolution and payload constraints that has to be addressed during the mission design [1]. Using proper mathematical models and software technologies in order to manage and allocate the resources of the satellite and develop optimized mission plans, we can utilize the resources effectively and avoid further costs.

As satellite missions, particularly CubeSats, have become more sophisticated and capable, specialized orbit simulation programs have been developed to assist with mission planning and analysis. A simulation software that provides a framework for efficient analysis and planning of satellite missions that include earth observation, communication, and scientific objectives helps the mission design process by giving the ability to make fast and reliable decisions regarding the satellite system requirements.

1.1 Purpose of Thesis

In 2009, SSDTL was established in order to develop the first CubeSat of Turkey, ITUpSat-1. The decade following its success, 5 more CubeSats were developed and placed in orbit. The latest project, SharjahSat-1 3U CubeSat started development in 2019 and came to fruition as of 2022. During the development of these projects, the necessity of a software to aid in the mission planning process was realized.

The goal of this study is to research and design, given the objectives of the satellite project at hand, a reliable and straightforward method to decide critical system parameters of spacecraft based on various constraints while maintaining easy integrability to facilitate further development for future space missions that are not only limited to Earth-orbiting satellite systems.

The number of objects orbiting the Earth grow exponentially, and fast and precise predictions are needed for modern space surveillance. Traditional Special Perturbations orbit propagators provide accurate results, but are awfully slow as performing numerical integration of the osculating equations of motion is a costly process, usually in more than 100 steps per revolution of the spacecraft. On the other hand, fully-analytical General Perturbations models yield fast results but the equations they are based on use a good deal of approximations, hence they have a large margin of error. Hence, a non-analytical, AI-powered orbital propagator was developed using deep learning and reinforcement learning methods and its advantages against the available methods were examined and discussed.

The requirements for the software were derived from the objectives of the SharjahSat-1 mission, which can be inspected in two categories.

1. iXRD Payload (Primary Payload): An X-Ray detector to:
    - Extend knowledge on miniaturized X-Ray detectors and their capabilities to contribute to Space Weather research.
    - Detect hard X-rays from very bright galactic X-ray sources such as very bright black hole and neutron stars.
    - Observe and study the bright sun flares and development of solar coronal holes, responsible for driving the stellar wind at an early phase.
2. Optical Payload (Secondary Payload): A camera system with a GSD of at least 50 meter/pixel to:
    - Acquire the images of regions on Earth from low orbit.
    - Take photos of SAASST and its surroundings.

These objectives require the design of an orbit with high visit times of SAAST, determination of camera field of view, determination of communication link budgets, and propagation of orbit considering different attitude models such as point tracking.

Key tasks for the software are then:

- Accurately model the orbital propagation based on the physical parameters of the orbit for the desired time interval. Atmospheric models, radiation pattern of the sun, gravity models are all utilized in this process.
- Determine communication links between the satellite and a ground station and/or another satellite.
- Determine the power generation on-orbit based on hardware parameters and eclipse durations.
- Visualize the orbit in both 3D and 2D views.

1.2 Literature Review

Spacecraft simulation and mission analysis software are important tools for the design and operation of spacecraft and their missions. These tools allow engineers and mission planners to model and predict the behavior of spacecraft under a variety of conditions, and to optimize the performance and efficiency of the spacecraft and its mission.

There have been numerous developments in this field over the years. One key development has been the increased use of advanced algorithms and mathematical models to more accurately and efficiently simulate spacecraft behavior, as our understanding of force models and accuracy of observed quantities increased. This has been facilitated by the availability of powerful computing systems and the increasing use of data from real-world spacecraft missions.

Another important development has been the integration of various tools and software packages into comprehensive mission analysis platforms. These platforms allow users to analyze all aspects of a spacecraft mission, from orbit determination and trajectory

optimization to the effects of various force models and the performance of subsystems such as propulsion and power systems.

In recent years, there has also been a trend towards the development of user-friendly, intuitive software interfaces that make it easier for non-experts to use these tools. This has made it possible for a wider range of individuals, including students and researchers, to access and utilize these tools.

While almost all satellite operators develop their own software from scratch, there are many open-source and commercial products available as well. The most prominent open-source spacecraft simulation software at the moment are GMAT and Orekit.

GMAT (General Mission Analysis Tool) was developed at the Goddard Space Flight Center by a team of NASA with contributions from the public and private intuitions. It is widely used for preliminary mission design and real-world mission operations. Mission analysis and trajectory optimization of a spacecraft can be achieved using this tool. It can be used to design the path a spacecraft will follow, optimize the maneuvers it will make along the way, and predict its future trajectory. GMAT can also be used to determine the orbit of a spacecraft, visualize and communicate mission parameters, and explore the range of possible mission options (also known as the "trade space") [2].

Another established flight software is Orekit by CS Group. Orekit is a software library that offers a wide range of low-level tools and algorithms for use in space flight dynamics. It includes basic concepts such as time, frames, orbital parameters, orbit propagation, attitude, celestial bodies, force models, and JPL ephemerids, as well as comprehensive support for each of these concepts. Orekit also provides several high-level features, including attitude modes, frame handling, propagation and outfitter tools, time scale tools, and Earth orientation parameters. These tools and features can be used to analyze and simulate various aspects of spacecraft behavior and mission planning [3].

Many other tools exist on the market at the moment as well, like AGI STK, FreeFlyer, and Patrius. For the scope of this work, only open-source solutions were inspected.

1.3 Overview

The software is developed using various programming languages. JavaScript was used for user interface alongside HTML and CSS. Orbital analyses and other computation heavy tasks were performed in C++ as it has the benefits of modular design, less resource use and fast execution speed, as well as good portability. Python was used for model training and artificial intelligence methods due to the enormous number of scientific libraries it includes. Electron framework was used to provide cross-platform compatibility. For real-time data visualization in both 2D and 3D, the Cesium framework was implemented using Bing as data provider for satellite imagery and terrain modelling. The software demonstrates a high level of precision and performs quickly according to the results.

Chapter 2 describes the celestial relationships and the mathematical foundation as well as implementation of algorithms used in the software. The user has the choice to obtain orbital elements in different coordinate systems and orbital elements, whose relationships and conversion procedures were detailed. Different time systems that are widely used in space missions are also described.

Chapter 3 explains the orbital propagators, starting from the fundamental two-body motion equation and building upon it to finally achieve the High-Precision Orbit Propagator which considers all force models modelled so far. Both the analytical and numerical propagators are examined, and the integrators used behind the latter are briefly discussed. Then, in Chapter 4, the theory and development of the neural-networks based model which uses historical orbit data to improve the accuracy of numerical propagators is explained along with its results.

Chapter 5 gives detailed information about the capabilities of the constructed simulation environment. It involves creation of specific orbit types which are pre-defined in the software, such as sun-synchronous and geosynchronous orbits that are widely used in real missions. The process of determining ground station passes and eclipse events, computation of communication link budgets, power generation analysis, and calculation of orbital maneuvers are also explained in this chapter.

In Chapter 6, the software development process is outlined. The development principles and how they are followed, design of software architecture, user interface development, and general process are explained.

Finally, in Chapter 7, outputs are verified by using real data, by comparison with other software, and by performing satellite observations in ITU Ground Station. The flight-readiness of the software is demonstrated and the results as well as planned future improvements are discussed.

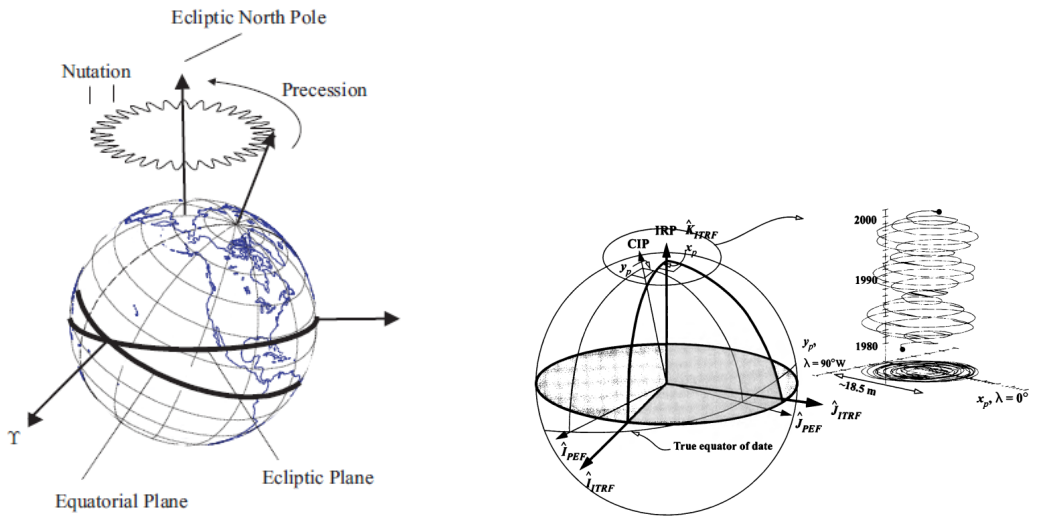
2. CELESTIAL RELATIONSHIPS

2.1 Coordinate Systems

The orientation of the orbital elements in relation to a central body is defined using a coordinate system. The software is able to convert a state vector from one coordinate system to another by applying the appropriate translation and rotation operations. In this section, the relations governing the implemented coordinate system transformations are defined. For simplicity, only equations for converting to ICRF or J2000 systems are given, from which all other systems can be obtained.

2.1.1 ICRF

The International Astronomical Union (IAU) adopted the International Celestial Reference System (ICRS) as the standard reference system starting 1998. The associated reference frame, called the International Celestial Reference Frame (ICRF) is realized using an algorithm consisting of multiple models as shown in Figure 2.1.



(a) Precession & Nutation Effects [4] (b) Polar motion effect [5]

Figure 2.1 : Effects of perturbating forces on Earth's motion

The most recent algorithm uses the the IAU2006 nutation and P03 precession models, and the Earth rotation angle and became operational in the beginning of 2009. The origin of this frame is located at the barycenter of the solar system (BCRF) within a relativistic framework, and its axes are realized by observations of extragalactic radio objects from VLBI measurements, in order to ensure the frame has no net rotation.

ICRF is the most accurate representation of an inertial reference frame that has been created so far. It is an improvement on the J2000 frame, and most recent star catalogs and ephemerides for celestial bodies are typically expressed using the ICRF frame as a reference.

The geocentric counterpart of this system is the Geocentric Celestial Reference Frame (GCRF). Within the software, selecting "ICRF" automatically moves the origin to the working central body, so it actually represents GCRF when selecting Earth.

2.1.2 ITRF

The International Terrestrial Reference Frame (ITRF) is an Earth-centered system fixed to the rotating earth. Its origin is at the center of mass of the Earth, and its axes are attained by coordinates of defining observing stations on the surface of the Earth, and since tectonic motion of plates affect these stations, this system is periodically re-adjusted. Conversion between ITRF and ICRF is performed using several models that incorporate IERS Earth Observation Parameters using IAU-2000A theory, with 2006 update for precession.

- **Precession (*P*)**

  The gradual change in the orientation of Earth's axis of rotation and the location of the equinox [6]. It is caused by a combination of factors, including the gravitational forces of the planets (planetary precession) which cause the orientation of the ecliptic to change steadily, and the torque produced on Earth's irregular mass distribution by the Sun and Moon (luni-solar precession), resulting in a wobbling motion of the Celestial Ephemeris Pole (CEP) around the ecliptic north pole. Together, these two forms of precession are known as general precession.

- **Nutation (N)**

  The short-term and periodic variation of Earth's equator and vernal equinox, primarily caused by the torque exerted on Earth's equatorial bulge by the Moon [7]. It is made up of a combination of movements with various periods, the longest of which has a period of 18.6 years, and is related to the regression of the Moon's orbit's node. Nutation effects also include the torque exerted by the gravitational pull of Solar System bodies on the oblate Earth.

- **Polar Motion (Π)**

  Movement of the rotation axis with respect to the Earth's crust [7]. The Celestial Intermediate Pole (CIP) is the axis of Earth rotation, and it is normal to the equator.

- **Earth Rotation (Θ)**

  Describes the rotation of Earth about its own axis [8].

  The Earth Rotation Angle is the angle between the Celestial Intermediate Origin (CIO) and the Terrestrial Intermediate Origin (TIO), and is related to UT1. It was previously known as the stellar angle [9].

The transformation can be defined

$$r_{ICRF} = P(t)N(t)\Theta(t)\Pi(t)r_{ITRF} \tag{2.1}$$

And for velocity, we have to include the rotation rate of Earth, which is related to the Earth rotation angle and is given by

$$w_E = 7.292155146706979 \times 10^{-5} \left\{ 1 - \frac{L_D}{86400} \right\} \tag{2.2}$$

Where L_D, length of day, represents the rate at which UT1 is changing at a particular moment in time. It is maintained by the IERS. The velocity conversion is then

$$v_{ICRF} = P(t)N(t)\Theta(t)w_E^x\Pi(t)r_{ITRF} + P(t)N(t)\Theta(t)\Pi(t)v_{ITRF} \tag{2.3}$$

Where

$$w_E^x = \begin{pmatrix} 0 & -w_E & 0 \\ w_E & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \tag{2.4}$$

The rotation of the Earth is a 3-rotation through the Earth rotation angle, θ_{ERA}.

$$\theta_{ERA} = 2\pi(0.7790572732640 + 1.00273781191135448(JD_{UT1} - 2451545.0)) \tag{2.5}$$

$$\Theta(t) = R_z(\theta_{ERA}) \tag{2.6}$$

Here, JD_{UT1} is the Julian date expressed in UT1.

Polar motion matrix PI is calculated by [10]

$$R_z(-s')R_y(x_p)R_x(y_p) \tag{2.7}$$

Where x_p and y_p are the time-dependent coordinates of the Celestial Ephemeris Pole interpolated from Earth Orientation Parameters (EOP) files provided by the IERS. The variable s' is the TIO locator given by

$$s' = -0.0015" \left(\frac{a_c^2}{1.2} + a_a^2 \right) T_{TT} \approx -0.000047" T_{TT} \tag{2.8}$$

Here, a_c is the **Chandler wobble**, and a_a is the **annular wobble** of the pole. T_{TT} is the terrestrial time.

The combined Precession-Nutation matrix can be computed as follows.

$$PN = \begin{bmatrix} 1 - aX^2 & -aXY & X \\ -aXY & 1 - aY^2 & Y \\ -X & -Y & 1 - a(X^2 + Y^2) \end{bmatrix} R_z(s) \tag{2.9}$$

$$a = \frac{1}{2} + \frac{1}{8}(X^2 + Y^2) \tag{2.10}$$

X and Y are the coordinates of the CIP unit vector in the ICRS, and s is the CIO locator. Computation of these values are done by evaluating series expansions with thousands of terms, so they are pre-computed at 1 day intervals and interpolated to reduce computational complexity.

2.1.3 J2000

J2000, J2K, or EME2000 is the inertial frame defined by the Mean Equator and Mean Equinox (MEME) of the J2000 epoch (1 Jan 2000 12:00:00.000 TDB). The z-axis is aligned with the celestial North Pole, and the x-axis is aligned with the mean equinox.

It is realized by the FK5 IAU76 theory, which uses the precession (1976 IAU Theory), nutation (1980 IAU Theory), sidereal time, and an adjustment to the equation of the equinoxes. Until the emergence of the ICRF, the J2000 axes were considered the best.

The Frame Bias matrix, B, is used to rotate from ICRF to J2000 [10]. It is obtained by three sets of rotations, $R_x(-\eta_0)R_y(\xi_0)R_z(\delta\alpha_0)$, where η_0 and ξ_0 are the offsets from the ICRS pole, and $\delta\alpha_0$ is the shift in the origin. $\eta_0 = -6.8192$ mas, $\xi_0 = -16.6170$ mas, $\delta\alpha_0 = -14.6$ mas, all converted to radians.

$$[\mathbf{B}] = \begin{bmatrix} 1-\frac{1}{2}(\delta\alpha_0^2+\xi_0^2) & \delta\alpha_0 & -\xi_0 \\ -\delta\alpha_0-\eta_0\xi_0 & 1-\frac{1}{2}(\delta\alpha_0^2+\xi_0^2) & -\eta_0 \\ \xi_0-\eta_0\delta\alpha_0 & \eta_0+\xi_0\delta\alpha_0 & 1-\frac{1}{2}(\eta_0^2+\xi_0^2) \end{bmatrix} \tag{2.11}$$

Note that the frame bias is not always a precise transformation, because if it were, there would not be a benefit of using a non-rotating origin.

2.1.4 True/Mean of Date/Epoch

The True of Date (TOD) and Mean of Date (MOD) systems are intermediate coordinate systems appearing in FK5 reduction.

A sequence of Euler rotations are used for transformation between J2000 and Mean Of Date. The rotation angles are calculated by applying cubic polynomials of time since the J2000 epoch (as measured in JED) according to the angles and rates of 1976 IAU Theory of Precession, taken from the US Naval Observatory circular No. 163.

In case of transformation between Mean of Date and True of Date axes, the nutations in longitude and obliquity (according to 1980 Nutation model), and the mean obliquity are used, followed by the update to the equation of the equinoxes.

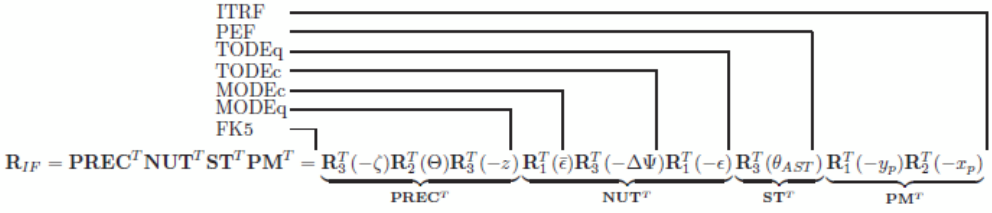


Figure 2.2 : FK5 Reduction Components [4].

There is also an additional property associated with these frames as follows.

- **of Date.** The epoch of the coordinate system is always the same as the epoch of the associated time of ephemeris generation. Hence, the nutation parameters are determined at each step.
- **of Epoch.** The epoch of the coordinate system is constant, so the nutation matrix does not change at steps. This approach is not usually used due to the error it introduces.

2.1.5 B1950

The B1950 (Besselian year 1950) axes were considered the most accurate representation of inertial axes until the J2000 frame. These axes are derived from the FK4 star catalog and its method for defining the MEME. The epoch is the start of the Besselian year 1950, which is 31 Dec 1949 22:09:46.866 (A Besselian year is the time it takes for the mean solar right ascension to increase by 24 hours [11]). The B1950 axes can be obtained by applying a constant rotation to the J2000 axes using the following equation by Seidelmann [7].

$$r_{J2000} \approx \begin{bmatrix} 0.9999256794956877 & -0.0111814832204662 & -0.0048590038153592 \\ 0.0111814832391717 & 0.9999374848933031 & -0.0000271625947142 \\ 0.0048590037723143 & -0.0000271702937440 & 0.9999881946023742 \end{bmatrix} r_{B1950} \tag{2.12}$$

2.1.6 TEME

The true equator, mean equinox (TEME) system is used in SGP4 model, and is related to the uniform equinox [7].

Some of the definitions of this system is ambiguous and is not recommended to use for modern applications. The implemented frame within the software follows the conventions and relations to other frames that are set out in [5].

$$r_{J2000} = P(t)N(t)R_z(-Eq_{Equinox1982})r_{TEME} \tag{2.13}$$

Where $Eq_{Equinox1982}$ signifies the equation of equinox according to the IAU 1982 model.

$$Eq_{Equinox1982} \approx \Delta\psi_{1980}\cos(\varepsilon_{1980}) \tag{2.14}$$

2.2 Orbital Element Types

In this chapter, definitions of Orbital Element Sets associated with Coordinate Types used in the software are given along with procedures to convert each of them to a simple state vector. The state of a spacecraft, meaning its orbit's shape and orientation, can be defined with six independent quantities. The collection of six different parameters are called an element set, or a state vector if they are the position and velocity vectors, r_0 and v_0 There are various orbital element sets which can be used for different applications, but they all represent initial conditions of a two-body orbit. Some of these sets also include time as the seventh element. The variety in these sets originate from complications arising from some peculiar geometries and usually tend to remove singularities. While most of the element sets can be used for any arbitrary celestial body, the Fixed Spherical, SGP4, and sets that use the Mean Element Theory can only be used when the central body is Earth.

2.2.1 Cartesian

The position and velocity vectors are most commonly expressed in rectangular Cartesian coordinate frame, which is referred to as the Earth-Centered Inertial (ECI) frame. Both vectors have three components along the principal axes of the inertial reference frame that originates at the center of the Earth.

Table 2.1 : Cartesian Element Set.

Element	Description
x	Position along the x-axis
y	Position along the y-axis
z	Position along the z-axis
$\dot{x}$	Velocity along the x-axis
$\dot{y}$	Velocity along the y-axis
$\dot{z}$	Velocity along the z-axis

2.2.2 Keplerian

Keplerian element set, also known as Classical Orbital Elements, is the most widely used collection of parameters when defining an orbit, as it is possible to intuitively discern the general shape of the orbit by looking at the parameters.

Table 2.2 : Keplerian Element Set Description.

Element	Description
a	**Semi-major axis.** Half the length of the longer axis of the orbital ellipse. Defines the size of the orbit
e	**Eccentricity.** Describes the shape of the ellipse ($e \in \mathbb{R}, 0 \leq e < 1$, where 0 = circular orbit).
i	**Inclination.** Angle between the orbital and equatorial planes. Inclinations greater than 90° indicate retrograde motion; its revolution around the Earth is in the opposite direction of Earth's rotation.
Ω	**Right ascension of the ascending node.** The angle between the positive X axis (vernal equinox) and the node line (point on the orbit at which the satellite crosses the equator from south to north).
ω	**Argument of perigee.** Angle from the ascending node to the direction of perigee (eccentricity vector), measured in the orbit plane and in satellite's motion's direction. ($w \in \mathbb{R}, 0 \leq e < 360°$)
θ	**True anomaly.** Angle between the eccentricity and the position vectors, measured in the direction of satellite motion and in the orbit plane.

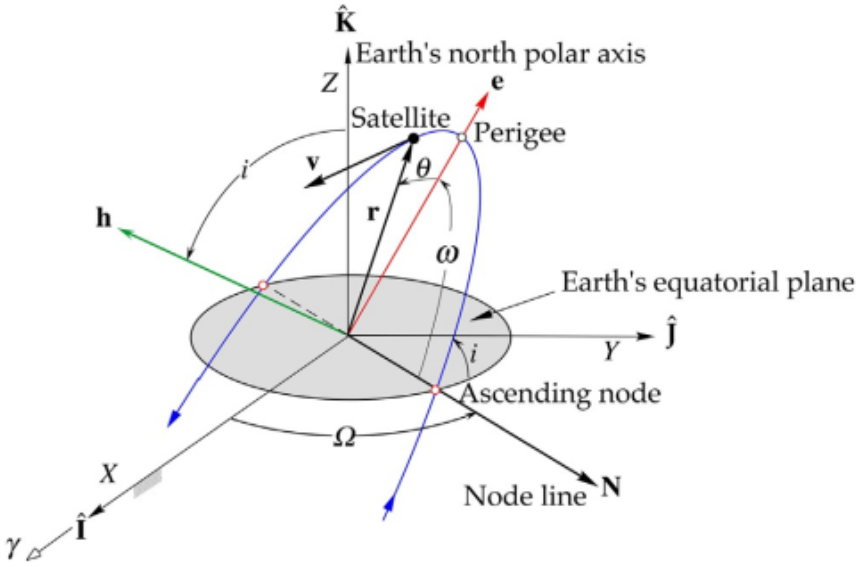


Figure 2.3 : Keplerian Element Set [12].

The software allows entering alternative parameters that can be converted from one into another as the user might choose to use them instead. These parameters are given in Table 2.3. The equations involved are given below.

$$r_a = a(1+e) \tag{2.15}$$

$$h_a = a*(1+e) - R_E \tag{2.16}$$

$$T = 2\pi\sqrt{\frac{a^3}{\mu}} \tag{2.17}$$

$$N = \sqrt{\frac{\mu}{a^3}}86400\,/\,(2\pi) \tag{2.18}$$

$$r_p = \frac{a*(1-e^2)\,/\,(1-e))}{2\,/\,(1-e)-1} \tag{2.19}$$

$$h_p = r_p - R_E \tag{2.20}$$

$$\lambda_{asc} = \tan^{-1}(C_{21}\,/\,C_{11}) + \Omega \tag{2.21}$$

$$M(\theta) = \begin{cases} \theta & (e=0) \\ 2\tan^{-1}\left(\sqrt{\frac{1-e}{1+e}}\tan\frac{\theta}{2}\right) - e\frac{\sqrt{1-e^2}\sin(\theta)}{1+e\cos(\theta)} & (e\neq 0) \end{cases} \tag{2.22}$$

$$E = 2\tan^{-1}\left(\sqrt{\frac{(1-e)}{(1+e)}}\tan(\theta\,/\,2)\right) \tag{2.23}$$

$$u = \omega + \theta \tag{2.24}$$

$$t_p = M\cdot T\,/\,2\pi \tag{2.25}$$

$$t_{asc} = \frac{T}{2\pi}[M(\theta) + M(\omega) - u] \tag{2.26}$$

Where C_{21} and C_{11} are the elements of ICRF to ITRF matrix. Details are given in Chapter 2.1.1.

Table 2.3 : Keplerian Elements Alternatives.

Parameter to Replace	Element	Description	Equation
a	Apogee Radius (r_a)	Distance from the center of the central body to the farthest point in orbit.	2.15
	Apogee Altitude (h_a)	Distance from the surface of the central body to the maximum orbital radius .	2.16
	Period (T)	Amount of time it takes for one complete revolution around the central body.	2.17
	Mean Motion (n)	Number of orbits per time unit.	2.18
e	Perigee Radius (r_p)	Distance from the center of the central body to the closest point in orbit.	2.19
	Perigee Altitude (h_p)	Distance from the surface of the central body to the minimum orbital radius.	2.20
Ω	Longitude of the Ascending Node (λ_{asc})	Earth-fixed longitude in ITRS frame where the satellite has crossed the inertial equator from south to north at, or prior to the initial condition of the orbit.	2.21
θ	Mean Anomaly (M)	The angle from the eccentricity vector to the hypothetical position vector obtained by assuming the satellite is constantly moving in its average angular rate.	2.22
	Eccentric Anomaly (E)	The angle between the central body and the auxiliary circle of the orbit, where a line perpendicular to the semi-major axis intersects the satellite's position on the ellipse.	2.23
	Argument of Latitude (u)	Sum of the Argument of Perigee and the True Anomaly.	2.24
	Time Past Perigee (t_p)	The elapsed time since the last perigee pass assuming two-body motion.	2.25
	Time Past Ascending Node (t_{asc})	The elapsed time since the last ascending node crossing assuming two-body motion.	2.26

The procedure to obtain state vector from Keplerian elements follows the classical Euler angle sequence $[R_z(\gamma)][R_x(\beta)][R_z(\alpha)]$ and is given in Algorithm 1.

Algorithm 1: Keplerian Elements to State Vector [12]

1 $r_{\bar{x}} = \frac{h^2}{\mu}\frac{1}{1+e\cos\theta}\begin{Bmatrix}\cos\theta\\ \sin\theta\\ 0\end{Bmatrix} \quad v_{\bar{x}} = \frac{\mu}{h}\begin{Bmatrix}-\sin\theta\\ e+\cos\theta\\ 0\end{Bmatrix}$

2 $[Q] =$
$$\begin{bmatrix} -\sin\Omega\cos i\sin w+\cos\Omega\cos w & -\sin\Omega\cos i\cos w-\cos\Omega\sin w & sin\Omega\sin i\\ \cos\Omega\cos i\sin w+\sin\Omega\cos w & \cos\Omega\cos i\cos w-\sin\Omega\sin w & -\cos\Omega\sin i\\ sini\sin w & sini\cos w & \cos i\end{bmatrix}$$

3 $r_x = [Q]r_{\bar{x}} \quad v_x = [Q]v_{\bar{x}}$

The reverse transformation is also given in Algorithm 2 as it is used a lot.

Algorithm 2: State Vector to Keplerian Elements [12]

1 Calculate the magnitudes of vectors $\vec{r} = x\hat{i}+y\hat{j}+z\hat{k}$ and $\vec{v} = v_x\hat{i}+v_y\hat{j}+v_z\hat{k}$.
$r = \sqrt{\vec{r}\cdot\vec{r}} \qquad v = \sqrt{\vec{v}\cdot\vec{v}}$

2 Compute the radial velocity
$v_r = \vec{r}\cdot\vec{v}/r$

3 Compute the specific angular momentum and its magnitude.
$$\vec{h} = \vec{r}\times\vec{v} = \begin{vmatrix}\hat{i} & \hat{j} & \hat{k}\\ x & y & z\\ v_x & v_y & v_z\end{vmatrix} \qquad h = \sqrt{\vec{h}\cdot\vec{h}}$$

4 Compute the node line vector and its magnitude.
$$\vec{N} = \vec{k}\times\vec{h} = \begin{vmatrix}\hat{i} & \hat{j} & \hat{k}\\ 0 & 0 & 1\\ h_x & h_y & h_z\end{vmatrix} \qquad N = \sqrt{\vec{N}\cdot\vec{N}}$$

5 Using these quantities, we can obtain each Keplerian element.
$$a = \frac{h^2}{\mu}\frac{1}{(1-e^2)}$$
$$e = \sqrt{1+\frac{h^2}{\mu^2}\left(v^2-\frac{2\mu}{r}\right)}$$
$$i = \cos^{-1}(h_z/h)$$
$$\Omega = \begin{cases}\cos^{-1}(N_x/N) & (N_Y \geq 0)\\ 2\pi-\cos^{-1}(N_x/N) & (N_Y < 0)\end{cases}$$
$$\omega = \begin{cases}\cos^{-1}(\frac{\vec{N}\cdot\vec{e}}{Ne}) & (e_z \geq 0)\\ 2\pi-\cos^{-1}(\frac{\vec{N}\cdot\vec{e}}{Ne}) & (e_z < 0)\end{cases}$$
$$\theta = \begin{cases}\cos^{-1}\left[\frac{1}{e}\left(\frac{h^2}{\mu r}-1\right)\right] & (v_r \geq 0)\\ 2\pi-\cos^{-1}\left[\frac{1}{e}\left(\frac{h^2}{\mu r}-1\right)\right] & (v_r < 0)\end{cases}$$

2.2.3 Equinoctial

Mathematical singularities arise during state vector to orbital elements transformation, when eccentricity and inclination are approaching zero. This can be observed for near-circular orbit applications which low-altitude remote sensing and geostationary satellites use in order to provide constant relative velocity and constant distance, with the latter further needing to be placed in a near-equatorial plane (i.e. $i \approx 0$) [13]. The argument of perigee is a problematic element for circular orbits as there is no defined perigee (i.e., the line of apsides is not well-defined when eccentricity approaches zero). Consequently, small orbital changes that change the location of perigee can lead to great errors when calculating ω (argument of perigee). Similarly, the line of nodes become indeterminate as the inclination approaches zero, hence the Ω (RAAN) equations become singular.

The Equinoctial set, which is also known as the Non-singular elements, removes the singularities at $e = 0$, and $i = 0$ and $i = 90^\circ$ using the longitude of periapsis, which is measured in two different planes when the orbit of the spacecraft is non-equatorial, but introduces a singularity for true retrograde orbits ($i = 180^\circ$). In order to circumvent this issue, a seventh parameter called the retrograde factor f is used which is -1 for retrograde and +1 for posigrade orbits.

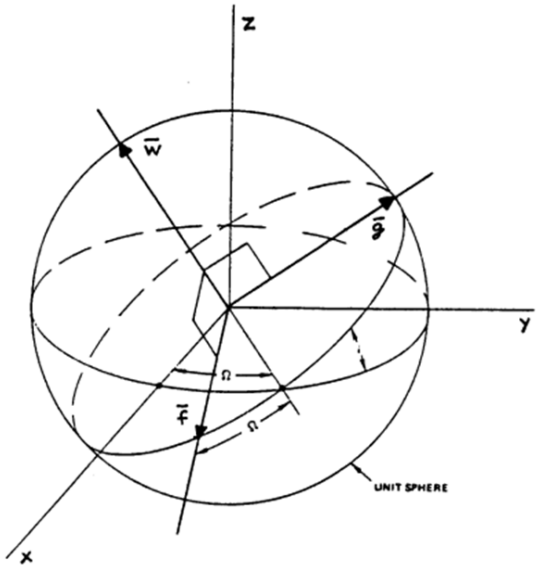


Figure 2.4 : Equinoctial Coordinate Frame

Table 2.4 : Equinoctial Element Set.

Element	Description
a	**Semi-major axis.** Half the length of the longer axis of the orbital ellipse.
h	**Eccentricity vector component.** The eccentricity vector is in the equinoctial reference frame, and points from the central body to perigee and has a magnitude of e. $h = e\sin(f\Omega + \omega)$
k	**Eccentricity vector component.** $k = e\cos(f\Omega + \omega)$
p	**Ascending node vector component.** The ascending node vector is defined in the equinoctial reference frame. It points from the central body to ascending node and has a magnitude of i. $p = \left[\tan\frac{i}{2}\right]^f \sin\Omega$
q	**Ascending node vector component.** $q = \left[\tan\frac{i}{2}\right]^f \cos\Omega$
λ_m	**Mean Longitude.** Specifies the position of the spacecraft within its orbit at epoch. $\lambda_m = \Omega + \omega + M(\theta)$

Algorithm 3 describes the necessary steps to transform Equinoctial Set into Cartesian elements.

Algorithm 3: Equinoctial to State Vector Algorithm [14]

1 Find the equinoctial reference frame basis vectors (f,g,w) as shown in Figure 2.4.

$$f = \frac{1}{1+p^2+q^2}\begin{bmatrix} 1-p^2+q^2 \\ 2pq \\ -2fp \end{bmatrix}$$

$$g = \frac{1}{1+p^2+q^2}\begin{bmatrix} 2fpq \\ (1+p^2-q^2)f \\ 2q \end{bmatrix}$$

$$w = \frac{1}{1+p^2+q^2}\begin{bmatrix} 2p \\ -2q \\ (1-p^2-q^2)f \end{bmatrix}$$

2 Find the eccentric longitude F and true longitude L, the auxiliary longitudes associated with this set. Kepler's Equation must be solved to obtain F. Calculating L requires the auxiliary quality b

$$\lambda = F + h\cos F - k\sin F$$

$$b = \frac{1}{1+\sqrt{1-h^2-k^2}}$$

$$\sin L = \frac{(1-k^2b)\sin F + hkb\cos F - h}{1-h\sin F - k\cos F}$$

$$\cos L = \frac{(1-h^2b)\cos F + hkb\sin F - k}{1-h\sin F - k\cos F}$$

3 Compute the position and velocity in the equinoctial frame.

$$r = a(1-h\sin F - k\cos F) = \frac{a(1-h^2-k^2)}{1+h\sin L + k\cos L}$$

$$X = r\cos L \quad Y = r\sin L$$

$$\dot{X} = -\frac{\sqrt{\mu\,/\,a}(h+\sin L)}{\sqrt{1-h^2-k^2}}$$

$$\dot{Y} = -\frac{\sqrt{\mu\,/\,a}(k+\cos L)}{\sqrt{1-h^2-k^2}}$$

4 Compute the state vector.

$$r = Xf + Yg \qquad v = \dot{X}f + \dot{Y}g$$

2.2.4 Spherical

The Spherical coordinate type uses polar coordinates instead of rectangular. Spherical system is also called as the ADBARV system due to the symbols of its elements. If

a fixed coordinate system is used, longitude and latitude replace right ascension and declination, respectively.

Table 2.5 : Spherical Element Set.

Element	Description
α	**Right Ascension.** The angle from the X axis (vernal equinox) to the projection of the position vector of the satellite onto the equatorial plane, measured towards the Y axis. $\alpha = \tan^{-1}\left(\frac{y}{x}\right)$
δ	**Declination.** The angle between the inertial equatorial plane and the spacecraft position vector, measured towards the Z axis. $\delta = \sin^{-1}\left(\frac{z}{r}\right)$
β	**Flight Path Angle.** The angle between the radius and velocity vectors is the vertical flight path angle, whereas its complement is the horizontal one. $\beta = \cos^{-1}\left(\frac{\vec{r}\cdot\vec{V}}{\lvert\vec{r}\cdot\vec{V}\rvert}\right)$
A	**Azimuth.** The angle, measured in the spacecraft instantaneous geocentric horizontal plane, that is between the projection of the velocity vector onto the said plane and the local northerly direction. Contrary to most disciplines, it is positive when measured clockwise from due north. $A = \tan^{-1}\left(\frac{\hat{A}\times\hat{P}\cdot\vec{r}/r}{\hat{A}\cdot\hat{P}}\right)$ where $\hat{W} = \frac{\vec{r}\times\vec{V}}{\lvert\vec{r}\times\vec{V}\rvert}$ $\hat{A} = \frac{\hat{W}\times\bar{r}}{\lvert\hat{W}\times\bar{r}\rvert}$ $\hat{P} = \frac{(\vec{r}\times\hat{k})\times\vec{r}}{\lvert(\vec{r}\times\hat{k})\times\vec{r}\rvert}$
r	**Radius.** Magnitude of the position vector. $r = \sqrt{x^2+y^2+z^2}$
V	**Velocity.** Magnitude of the velocity vector. $V = \sqrt{\dot{x}^2+\dot{y}^2+\dot{z}^2}$

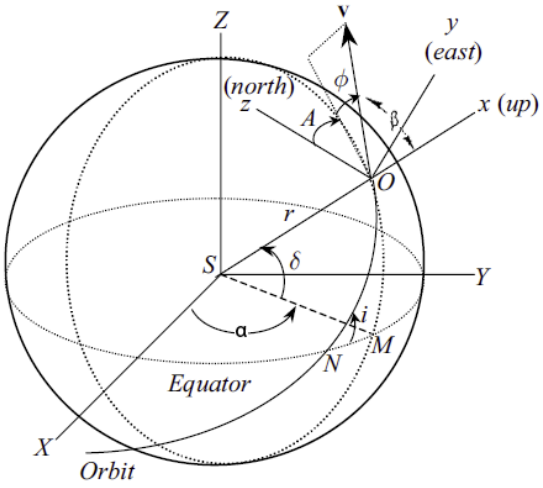


Figure 2.5 : Spherical Coordinate System.

It can be seen in Figure 2.5 that a right spherical triangle can be formed by the arcs of MO, ON, and NM, which yields the following equation:

$$\cos i = \cos\delta \sin A \tag{2.27}$$

When $A = 90°$, the minimum inclination is achieved which is exactly δ. Therefore, the smallest inclination the orbit of a launched spacecraft can possibly have is equal to the latitude of the launch site [15].

Algorithm 4 describes the steps to obtain Cartesian elements from Spherical elements.

Algorithm 4: Spherical Elements to State Vector Algorithm [16].

1 The $\hat{x}$ axis becomes colinear with the radius vector using the rotation sequence $[R_z(\alpha)][R_y(-\delta)]$ .Multiplying with the appropriate rotation matrices yield the following.
$x = r\cos\delta\cos\alpha$
$y = r\cos\delta\sin\alpha$
$z = r\sin\delta$

2 In order to align the $\hat{x}$ axis along the velocity vector direction, the sequence of rotation is given by $[R_z(\alpha)][R_y(-\delta)][R_x(-A)][R_y(-\beta)]$.
$\dot{x} = V[\cos\alpha(-\cos A\sin\beta\sin\delta + \cos\beta\cos\delta) - \sin A\sin\beta\sin\alpha]$
$\dot{y} = V[\sin\alpha(-\cos A\sin\beta\sin\delta + \cos\beta\cos\delta) + \sin A\sin\beta\cos\alpha]$
$\dot{z} = V[\cos A\cos\delta\sin\beta + \cos\beta\sin\delta]$

2.2.5 Fixed Spherical

When the problem requires the orbit to be referenced relative to the Earth, a modified version of the spherical coordinate system that is fixed to and rotates with the body can

be used. This system uses the Earth-fixed position parameters with inertial velocity parameters. It is similar to the LDBARV system in literature where geographic longitude replaces right ascension, but the declination is replaced with latitude as well in this system.

In order to calculate the geodetic parameters, we must first perform ICRF to ITRS transformation on the position vector. The transformation matrix is given in Chapter 2.1.1. The fixed position vector will be referred as $\vec{r'} = x'\hat{i} + y'\hat{j} + z'\hat{k}$ below, and R_E is the Equatorial radius of the Earth whereas f is the flattening.

Table 2.6 : Fixed Spherical Element Set.

Element	Description
h	**Geodetic Altitude.** Position of the spacecraft relative to the reference ellipsoid considering Earth is an oblate spheroid as shown in Figure 2.7. $h = r' - R_E \left[1 - f \sin^2 \delta - \frac{f^2}{2} \sin^2 2\delta \left(\frac{R_E}{2} - \frac{1}{4} \right) \right]$
λ	**Celestial Longitude.** Longitude of the spacecraft's subsatellite point, measured from the vernal equinox. Also known as right ascension. The reason it is used instead of geodetic longitude is to remove the dependency to the orbit epoch, so that the right ascension at Greenwich meridian does not have to be calculated. $\lambda = \tan^{-1}\left(\frac{y'}{x'}\right), (-180^\circ \leq \lambda \leq 180^\circ)$
ϕ	**Geodetic Latitude.** The angle between the normal to the reference ellipsoid that passes through the equatorial plane and the position of the satellite as shown in Figure 2.7. $\phi = \delta + \sin^{-1}\left[\frac{R_E}{r'}\left(f \sin 2\delta + f^2 \sin 4\delta \left(\frac{R_E}{r'} - \frac{1}{4}\right)\right)\right]$
FPA (β)	**Flight Path Angle.** The angle between the velocity and radius vectors as described in Table 2.5.
A	**Azimuth.** Velocity azimuth angle as described in Table 2.5.
V	**Velocity.** Magnitude of the velocity vector.

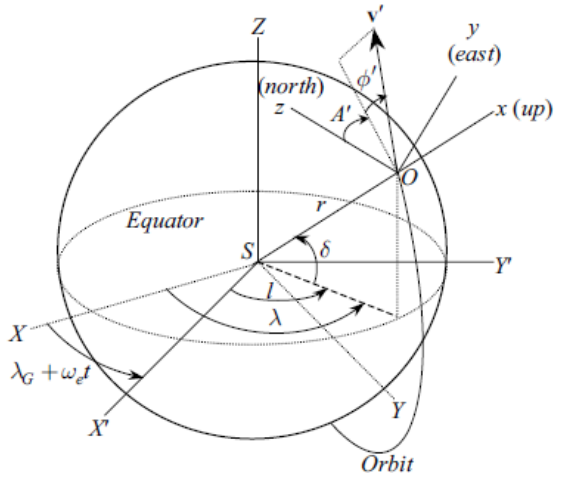


Figure 2.6 : Fixed Spherical Coordinate Frame.

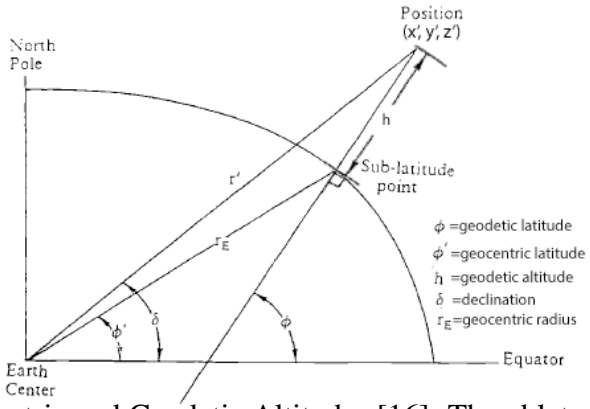


Figure 2.7 : Geocentric and Geodetic Altitudes [16]. The oblateness of Earth causes the difference.

Algorithm 5 describes the steps to obtain Cartesian elements from Fixed Spherical elements.

Algorithm 5: Fixed Spherical Elements to State Vector Algorithm

1 Find the geocentric latitude and radius of the sub-latitude point.
$\phi' = \tan^{-1}[(1-f)^2 \tan\phi]$, $-90^\circ \leq \phi \leq 90^\circ$
$r_E = \dfrac{R_E(1-f)}{\sqrt{1-f(2-f)\cos^2\phi'}}$

2 Position vector in the fixed reference frame is obtained.
$x^* = r_E \cos\phi' + h\cos\phi$
$y^* = 0$
$z^* = r_E \sin\phi' + h\sin\phi$

3 Rotate in z-axis.
$x' = x^* \cos\lambda$
$y' = x^* \sin\lambda$
$z' = z^*$

4 Using the inverse of the ICRF-ITRS transformation matrix, revert back to the inertial frame.
$r = [U_{ITRS}^{ICRF}]^{-1} r'$

5 Calculate velocity in the local horizontal plane.
$\dot{x}' = V\sin\beta$
$\dot{y}' = V\sin A\cos\beta$
$\dot{z}' = V\cos A\cos\beta$

6 The Euler Angle representation $[R_z(\alpha)][R_y(\pi/2-\delta)][R_y(-\pi/2)]$ can be used to find the orientation of the local horizon frame in relation to the Cartesian frame . This yields the rotation matrix:
$C_{LH} = \begin{pmatrix} \cos\delta\cos\alpha & \cos\delta\sin\alpha & \sin\delta \\ -\sin\alpha & \cos\alpha & 0 \\ -\sin\delta\cos\alpha & -\sin\delta\sin\alpha & \cos\delta \end{pmatrix}$ where $\delta = \sin^{-1}\frac{z}{r}$,
$\alpha = \tan^{-1}\frac{y}{x}$

7 Obtain the velocity vector in the Cartesian frame.
$v = [C_{LH}]^{-1} v'$

2.2.6 Delanuay

These variables, introduced by Delaunay who used them to elaborate his Lunar Theory [17], consist of canonical action-angle variables and are widely used in general perturbation theories. The first three variables represent the angles and the last three represent the conjugate actions, relatively.

Table 2.7 : Delaunay Variables.

Element	Description
M	**Mean Anomaly.** Same as Keplerian set.
ω	**Argument of Perigee.** Same as Keplerian set.
Ω	**Right Ascension of the Ascending Node.** Same as Keplerian set.
L	**Orbital Energy Term.** $L = \sqrt{\mu a}$
G	**Magnitude of Angular Momentum.** Angular momentum vector is perpendicular to the orbital plane. $G = L\sqrt{1-e^2}$
H	**Z-component of Angular Momentum.** $H = G\cos i$

Reverse transformation from the Delaunay variables is a linear process. We can obtain the Kepler elements and use them to calculate the state vector.

$$a = L^2/\mu \qquad e = \sqrt{1-(G/L)^2} \qquad i = \cos^{-1}(H/G)$$

2.2.7 Others

Poincaré Variables

This is the canonical equivalent of the Equinoctial set, ready to be used in problems requiring Hamiltonian dynamics [5]. They have the advantage of being singular even for circular and equatorial orbits, and can easily be organized into symplectic pairs of coordinates and momenta [18].

$$\lambda_M = \Omega + \omega + M \qquad \mathscr{L} = \sqrt{\mu a}$$
$$g = \sqrt{2\mathscr{L}(1-\sqrt{1-e^2})}\cos(\omega+\Omega) \qquad \mathscr{G} = -g\tan(\omega+\Omega)$$
$$h = \sqrt{2\mathscr{L}\sqrt{1-e^2}(1-\cos i)}\cos\Omega \qquad \mathscr{H} = -h\tan\Omega$$

Variations of Equinoctial Set

There are two modifications to the equinoctial set. The first one defines the ascending node vector components as

$$p = \left[\sin\frac{i}{2}\right]^f \sin\Omega \qquad q = \left[\sin\frac{i}{2}\right]^f \cos\Omega$$

This variant is simpler, but less beneficial when working with perturbational equations [19]. The other variant uses the auxiliary parameter g instead of a, and uses the true longitude λ_t, instead of mean longitude λ_m.

$$g = a(1-e) \qquad \lambda_t = \Omega + \omega + \theta$$

Mean Element Sets

The Mean element theory is used to approximate motion by isolating the slowly varying effects, such as the motion of Ω due to J2 gravity term, from the effects of fast motions such as the change of true anomaly. The "mean" in this context is not a numerical average of sampling, but instead refers to averaging the differential equations of motion. The fast terms are averaged and used to estimate the motion of slowly varying ones. These sets can only be used for Earth-centered orbits and are useful for applications sensitive to gravitational perturbations.

The Brouwer-Lyddane Mean element set [20,21] has two versions called "Short" and "Long". The short one account for only the short period terms and the J2 oblateness term as the sole perturbation force. The long one also considers the long period effects and other oblateness terms up to J5. The most visible difference is usually seen at the mean eccentricity due to oscillation.

Kozai-Izsak Mean Elements [22] are extremely similar to the Short version of the Brouwer-Lyddane set, with the only difference coming from formulation. They can be used in place of the classical Keplerian elements when averaging over short periods. The same cannot be said for the long one as other perturbing forces (e.g. SRP, drag, third-bodies) can have a dominating influence.

2.3 Time Systems

The equations of motion in astrodynamics require a unique time system, and all satellite operations depend on precise timekeeping. There are four fundamental time systems; universal, sidereal, atomic, and dynamical. The first two are mathematically related and based on the Earth's rotation, while the last two are much more precise and independent. Different applications require different time systems, hence, the ability to convert between different systems is of utmost importance. The software is equipped

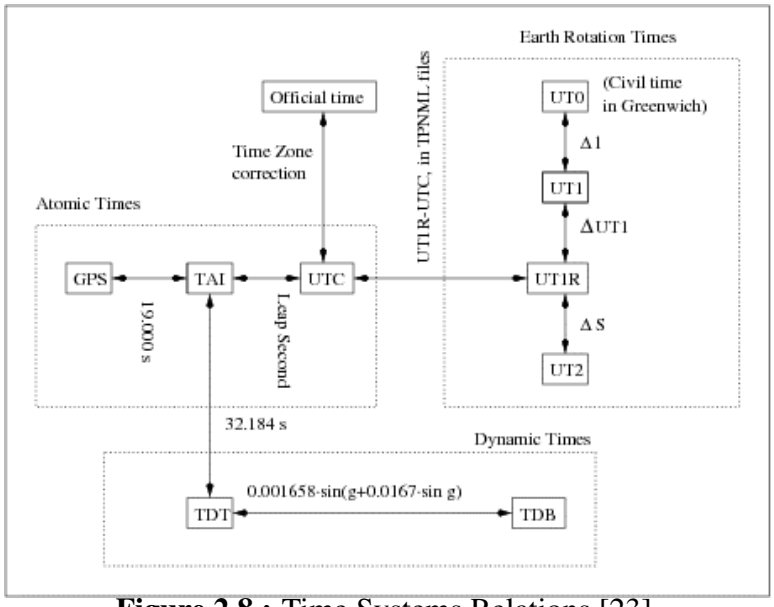


Figure 2.8 : Time Systems Relations [23].

with the necessary algorithms to convert from one time system to another, as well as the ability to represent time in different formats.

2.3.1 Universal Time

The Solar time is measured by the successive passes of the Sun over a longitude, and for this the Greenwich meridian was chosen as the 0 degree point for a fixed reference to define the start of a day. The irregularities in the solar motion due to factors like the eccentricity of the orbit of the Earth and accompanied seasonal variations, a convention called Universal Time that used a fictitious mean Sun (assuming uniform motion as a function of the sidereal time) was adopted [5]. The three variations of the UT along with UTC are described below.

1. **UT0**

   Observation of UT at a specific ground station. Defined as 12h plus the Greenwich Hour Angle GHA, it is related to UT1 as follows

$$UT1 = UT0 + \Delta l$$

   where Δl is the pole movement in longitude and is given as

$$\Delta l = \frac{1^s}{15}(x_p \sin\lambda - y_p \cos\lambda)\tan\phi)$$

where x_p and y_p are instantaneous pole coordinates and λ and ϕ are latitude and longitude of the observation site.

2. **UT1**

   UT1 is obtained by adding $\Delta UT1$ to the UTC, which is the correction factor for the periodic variations caused by the zonal tidal deformations of the polar moment of inertia, as described by the following equation [24].

$$\Delta UT1 = \sum_{i=1}^{N} \left[A_i \sin \left(\sum_{j=1}^{5} k_{ij} \alpha_j \right) \right]$$

   The most accurate values, which use 62 periodic components during calculation, can be obtained from the Earth Orientation Bulletin Series D by IERS [25]. The angular rate of UT1 in the present day has been defined to closely follow Newcomb's convention for mean solar time, based on the mean motion of the Sun reduced from 19th-century observations.

3. **UT2**

   Correcting UT1 or seasonal variations yield UT2; $UT1 + \Delta S$, where ΔS is the periodic seasonal variation on the earth rotation in seconds and is given as

$$\Delta S = 0^s022 \sin 2\pi t - 0^s120 \cos 2\pi t - 0^s0060 \sin 4\pi t + 0^s0070 \cos 4\pi t$$

   Where t is the date in Besselian years.

   $t = 2000.00 + MJD - 51544.02 \,/\, 365.2422$

4. **UTC**

   Coordinated Universal Time. It is linked to TAI by an integer second offset that is regularly updated to keep in close agreement with UT1, up to 0.9s difference. The list of leap seconds can be obtained from the IERS Bulletin C series.

$$UTC = TAI - LeapSeconds$$

2.3.2 Dynamical Time Systems

Dynamical time is derived from the dynamical motion of celestial bodies using relativistic corrections. They are used for astronomical calculations.

(a) **Terrestrial Dynamical Time (TT)**

Used to be known as TDT and is the successor of Ephemeris Time, is independent of geocentric ephemerides or equations of motion, and represents the time that would be measured by a perfect clock on the surface of the Earth geoid.

$$TT = TAI + 32.184s$$

(b) **Dynamical Barycentric Time (TDB)**

The independent variable of barycentric solar system ephemerides, and differs from TT only by periodic terms which causes a difference of 2ms.

$$TDB = TDT + \frac{2r_s}{AU\dot{n}_e} e_e \sin E + \text{other}$$

Where r_s is the Schwarzschild radius (1.478 km), n_e is the Earth's mean motion around the Sun ($1.991e^{-7} rad/s$) and e_e is the Earth's orbital eccentricity (0.016708617). "Other" includes small effects contributed by the third bodies, especially the Moon, and the diurnal motion of the Earth. E is the eccentricity anomaly of the Earth, which can be approximated to change the equation to include the mean anomaly of Earth, M_e, instead. The final approximated equation is then [26]

$$TDB = TDT + 0.001658 \sin M_e + 0.00001385 \sin 2M_e$$

Where $M = 6.240035939 + 628.3019560 T_{TDT}$ in radians and T_{TDT} is in Julian Centuries.

(c) **Barycentric Coordinate Time (TCB)**

The relativistic time coordinate of the Barycentric Celestial Reference System which describes the motion of solar-system objects in a non-rotating relativistic frame centered at the solar-system barycenter. TCB is linearly related to TDB with a relation that is chosen to match the rate of TDB to TT

for the time span covered by the JPL Development Ephemerides.

$$TCB = TDB + L_B(JD - 2443144.5) * 86400s + P$$

Where $L_B = 1.55051976772e^{-8}$ and $P \approx 6.55e^{-5}s$ [27]

(d) **Geocentric Coordinate Time (TCG)**

The relativistic time coordinate of the Geocentric Celestial Reference System.

$$TCG = TT + L_G(JD - 2443144.5) * 86400s$$

Where $L_G = 6.969290134e^{-10}$

2.3.3 Julian Date

The amount of time passed from the epoch January 1, 4713 B.C., 12:00. A Julian Day, *JD*, starts at noon every day so that the astronomers can make their observations in a single day. Given the date, Julian Date between March 1, 1900 and February 28, 2100 can be obtained from equation 2.28.

$$JD = 367 \cdot Y - \text{INT}\left[\frac{7(Y + \text{INT}\left(\frac{M+9}{12}\right))}{4}\right] + \text{INT}\left(\frac{275 \cdot M}{9}\right) + d + 1721013.5 + \frac{\left(\frac{s}{60} + min\right)}{60} + h}{24} \quad (2.28)$$

If the required date is not in this interval, algorithm 6 by Meeus [28] can be used instead. Algorithm 7 is also used in the software to get UTC date from a given Julian Date.

Algorithm 6: Calendar Date to Julian Date Conversion

```
1 Given Y = Year, M = Month from 1 to 12, D = Day with decimals
    if M == 1 or M == 2 then
      Y = Y - 1 and M = M + 12
    end if
2 A = INT(Y/100)          B = 2 - A + INT(A/4)
3 JD = INT(365.25(Y + 4716)) + INT(30.6001(M+1)) + D + B – 1524.5
```

Algorithm 7: Julian Date to Calendar Date Conversion [28]

1 JD = JD0 + 0.5 Z = INT(JD) F = FRAC(JD)

2

if Z <2299161 **then**

A = Z

else

$$\alpha = \text{INT}\left(\frac{Z - 1867216.25}{36524.25}\right)$$

$$A = Z + 1 + \alpha - \text{INT}(\alpha/4)$$

end if

3 $B = A + 1534$ $C = \text{INT}\left(\frac{B - 122.1}{365.25}\right)$

$D = \text{INT}(365.25C)$ $E = \text{INT}\left(\frac{B - D}{30.6001}\right)$

4 Day = B - D - INT(30.6001E) + F

Month = (E - 1) % 13

Year = C - 4716 if Month >2

= C - 4715 otherwise

For high precision, the software does not store the whole JD in a single variable but separates the integer and decimal parts in two different 64-byte registers. Other commonly used Julian Date variations are given below.

- **Modified Julian Date.** Same as Julian date, with a fixed offset. It starts each day at 00:00 hours rather than 12:00 hours, and uses a simplified numbering system for days.

$$MJD = JD - 2,400,000.5$$

- **Julian Ephemeris Date.** A measure of elapsed TDT days from a reference point of noon on November 24, 4713 BC in the Gregorian calendar. It is based on the TT time scale.

- **Julian Centuries.** Most astronomical relations use the Julian centuries from a specific epoch.

$$JC = (JD - 2451545)/36525$$

2.3.4 GPS Time

GPS time operates on an atomic scale like TAI with an offset value. It does not consider the rotation of Earth, so it does not include leap seconds or other adjustments that are periodically made to UTC. When GPS time was first established in 1980, it was synchronized with UTC, but it has since deviated from UTC and now maintains a constant offset from TAI [29].

$$GPS = TAI - 19s$$

GPS time is measured as the amount of time that has passed since 6 January 1980 00:00:00:00 UTC, which is known as the GPS epoch. It is reported in terms of the number of weeks that have passed since the epoch and the number of seconds into the current week. Leap seconds should be accounted for when converting between UTC and GPS time.

2.3.5 Others

In addition to the fundamental time systems, various other time display formats can be used in the software.

- **Local Time.** Using a specified time zone with an offset from the UTC for civilian timekeeping.
- **Canonical Times.** Earth Canonical Unit is the time it would take a fictional satellite to travel one radian in a circular orbit of Earth's equatorial radius ($\sqrt{R_E^3/\mu_E}$).
  Sun canonical unit equals one sidereal year ($\sqrt{(AU)^3/\mu_S}$).
- **GPS Z-count.** Elapsed time is measured in units called Z counts, which are equal to 1.5 seconds. The Z count is the number of complete weeks that have passed since the GPS epoch, plus the number of 1.5 second increments into the current week.
- **Epoch.** Time elapsed relative to the given scenario epoch. Can be in seconds, days, hours, or years. Alternatively, "Mission Elapsed Time" can be used to see seconds passed since a user-defined epoch.

3. ORBITAL PROPAGATORS

In orbital dynamics, the most fundamental equation is the law of universal gravitation, which describes how two bodies are attracted to each other.

$$F_g = MmG/r^2 \tag{3.1}$$

F_g = gravitational force between bodies
M, m = masses of two bodies
G = universal gravitational constant
r = distance between the center of mass of two bodies

While there is a myriad amount of celestial bodies in the universe, the motion of the spacecraft is usually governed by a single body. Starting from the simple two-body model, we will go through different levels of orbital propagators.

3.1 Two-Body

In two-body propagator, we make the following assumptions:

1. Attraction to a single central body governs the motion, hence the chosen coordinate system is inertial.
2. The mass of the satellite is negligible compared to that of the central body.
3. The bodies are only acted by gravitational and centrifugal forces.
4. The bodies are perfectly spherical and their masses are concentrated at the centers (uniform density).

For most satellite orbits, this assumption produces highly accurate results in the short-term, as the influence of the Earth on the vehicle is about 1000 factors larger than any other influence. However, there are several perturbations such as the oblateness of

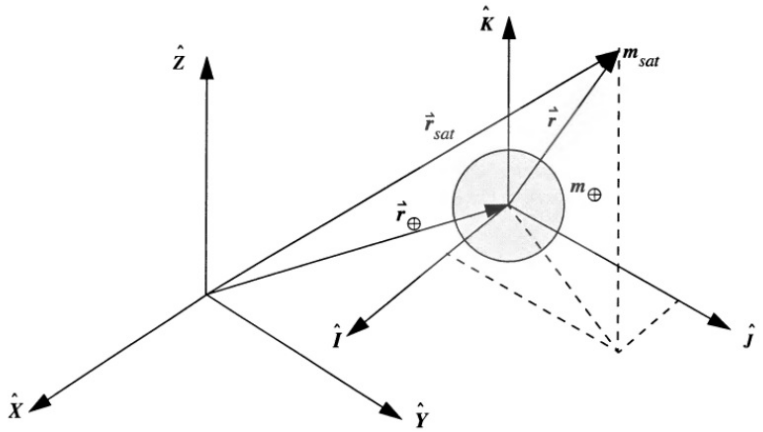


Figure 3.1 : 2-Bodies in Inertial Reference Frame

the Earth, atmospheric drag, and solar radiation that notably affect the orbital motion in long-term. Hence, this model is only good for a first approximation.

Two-body propagator, like other propagators, works by representing the state vector of the satellite as a function of time. That is, given the initial position and velocity of an orbiting body, we can determine the position and velocity in a later time. The most straightforward approach for this is to apply the equations of motion based on the universal law of gravitation, that is

$$F_g = m_1 a_1 = m_1 m_2 G/r^2 \tag{3.2}$$

We can express acceleration as the second derivative of position and cancel out the mass.

$$\ddot{R}_1 = Gm_2 \frac{\mathbf{r}}{r^3} \tag{3.3}$$

Which leads us to

$$\ddot{X}_1 = Gm_2 \frac{X_2 - X_1}{r^3} \qquad \ddot{Y}_1 = Gm_2 \frac{Y_2 - Y_1}{r^3} \qquad \ddot{Z}_1 = Gm_2 \frac{Z_2 - Z_1}{r^3} \tag{3.4}$$

$$\ddot{X}_2 = Gm_1 \frac{X_1 - X_2}{r^3} \qquad \ddot{Y}_2 = Gm_1 \frac{Y_1 - Y_2}{r^3} \qquad \ddot{Z}_2 = Gm_1 \frac{Z_1 - Z_2}{r^3} \tag{3.5}$$

Now if we have the initial position and velocity vectors, we can use a numerical integration method such as Runge-Kutta to obtain the state vectors in desired time steps. These equations give us the motion of both bodies in an inertial frame of reference. We can modify them as follows to obtain the relative motion of the

secondary mass with respect to the first one.

$$\ddot{x} = -\frac{\mu}{r^3}x \qquad \ddot{y} = -\frac{\mu}{r^3}y \qquad \ddot{z} = -\frac{\mu}{r^3}z \tag{3.6}$$

While this method is sufficient for two-body motion, we should be aware that in numerical integration, any error from calculating the last state is carried over into the next calculation. Over time, this error will build up and get worse. Hence, we will use the exact solution to this problem using Lagrange coefficients, where the error comes from only the initial calculations and remains constant [30].

Using some algebra and the conservation of momentum, Lagrange came up with the following equations that express the future state vector using initial position and velocity vectors.

$$\boldsymbol{r} = f\boldsymbol{r}_0 + g\boldsymbol{v}_0 \tag{3.7}$$

$$\boldsymbol{v} = \dot{f}\boldsymbol{r}_0 + \dot{g}\boldsymbol{v}_0 \tag{3.8}$$

Where the Lagrange coefficients f and g and their derivatives, $\dot{f}$ and $\dot{g}$, are given by

$$f = \frac{x\dot{y}_0 - y\dot{x}_0}{h} \qquad g = \frac{-xy_0 + yx_0}{h} \tag{3.9}$$

$$\dot{f} = \frac{\dot{x}\dot{y}_0 - \dot{y}\dot{x}_0}{h} \qquad \dot{g} = \frac{-\dot{x}y_0 + \dot{y}x_0}{h} \tag{3.10}$$

These equations can be expressed in terms of the change in true anomaly, using the identities:

$$\dot{x}_0 = -\frac{\mu}{h}\sin\theta_0 \qquad \dot{y}_0 = \frac{\mu}{h}(e + \cos\theta_0) \tag{3.11}$$

We obtain

$$f = 1 - \frac{\mu r}{h^2}(1 - \cos\Delta\theta) \qquad g = \frac{rr_0}{h}\sin\Delta\theta \tag{3.12}$$

$$\dot{f} = \frac{\mu}{h}\frac{1-\cos\Delta\theta}{\sin\Delta\theta}\left[\frac{\mu}{h^2}(1-\cos\Delta\theta) - \frac{1}{r_0} - \frac{1}{r}\right] \qquad \dot{g} = 1 - \frac{\mu r_0}{h^2}(1-\cos\Delta\theta) \tag{3.13}$$

Now we have to find the relation between the true anomaly and time. However, the equations differ for different type of orbits (elliptic and hyperbolic). Thus, we have to rewrite Kepler's equation in terms of a universal variable that is valid for all kinds

of orbits, namely universal anomaly χ [12]. Universal anomaly can be calculated iteratively using Newton's method such as:

$$\chi_{i+1} = \chi - \frac{\frac{r_0 v_{r0}}{\sqrt{\mu}}\chi_i^2 C(z_i) + (1 - ar_0)\chi_i^3 S(z_i) + r_0\chi - \sqrt{\mu}\Delta t}{\frac{r_0 v_{r0}}{\sqrt{\mu}}[1 - \alpha\chi_i^2 S(z_i)] + (1 - ar_0)\chi_i^2 C(z_i) + r_0} \tag{3.14}$$

Where

$$\alpha = \frac{1}{a} \quad z_i = \alpha\chi_i^2$$

and C(z) and S(z) are Stumpff functions that can be expanded as

$$S(z) = \begin{cases} \frac{\sqrt{z} - \sin\sqrt{z}}{(\sqrt{z})^3} & (z > 0) \\ \frac{\sin\sqrt{-z} - \sqrt{z}}{(\sqrt{z})^3} & (z < 0)\,(z = \alpha\chi^2) \\ \frac{1}{6} & (z = 0) \end{cases} \tag{3.15}$$

$$C(z) = \begin{cases} \frac{1 - \cos\sqrt{z}}{z} & (z > 0) \\ \frac{\cosh\sqrt{-z} - 1}{-z} & (z < 0)\,(z = \alpha\chi^2) \\ \frac{1}{2} & (z = 0) \end{cases} \tag{3.16}$$

Here, $z < 0$ is for hyperbolas, $z = 0$ is for parabolas, and $z > 0$ is for ellipses. A good first estimation is given by Chobotov [16] as

$$\chi_0 = \sqrt{\mu}|\alpha|\Delta t \tag{3.17}$$

Integrating the universal anomaly to Lagrange functions, they become

$$f = 1 - \frac{\chi^2}{r_0} C(\alpha\chi^2) \qquad g = \Delta t - \frac{1}{\sqrt{\mu}}\chi^3 S(\alpha\chi^2) \tag{3.18}$$

$$\dot{f} = \frac{\mu}{r r_0}[\alpha\chi^3 S(\alpha\chi^2) - \chi] \qquad \dot{g} = 1 - \frac{\chi^2}{r} C(\alpha\chi^2) \tag{3.19}$$

Now we can obtain the universal anomaly from the time interval, plug it into the equation, and propagate our orbit analytically. The algorithm implementation is as follows

Algorithm 8: Two-Body Algorithm.

$r_0 = norm(R0)$
$v_0 = norm(V0)$
$v_{r0} = \frac{dot(R0,V0)}{r_0}$
$\alpha = \frac{2}{r_0} - \frac{v_0^2}{\mu}$
The sign of α determines the type of orbit
$\chi_0 = \sqrt{\mu}|\alpha|\Delta t$
while ratio $>$ tolerance **do**
  Calculate the ratio in equation 3.14
  $\chi_{i+1} = \chi_i - ratio_i$
end while
Obtain f and g from equations 3.18
Obtain $\dot{f}$ and $\dot{f}$ from equations 3.19
Obtain final state from equations 3.7 and 3.8

Since there are no perturbations in two-body model, the orbit will never decay and keep its orientation forever. An example orbit created by the two-body model is shown in Figure 3.2.

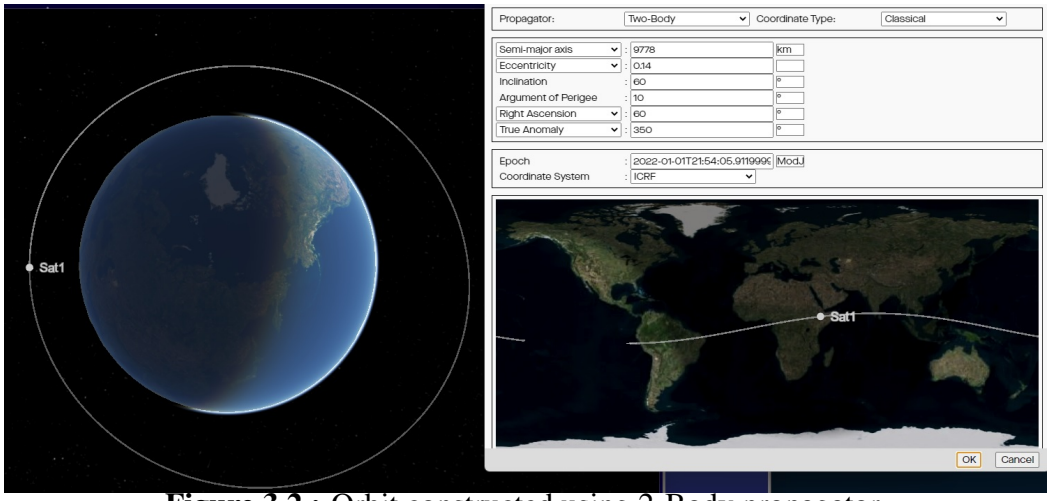


Figure 3.2 : Orbit constructed using 2-Body propagator.

3.2 N-Body

In 2-Body model, we assumed the only influence on the motion of the spacecraft was gravitational force of Earth. While this assumption is mostly sufficient for most Earth-orbiting satellite applications, gravitational pull from other Celestial bodies can cause enough interference to slightly alter the orbital path of any satellite orbiting Earth. Furthermore, for the analysis of interplanetary trajectories and Halo-orbits that are focused around Lagrange points, we must be able to solve the Restricted Three

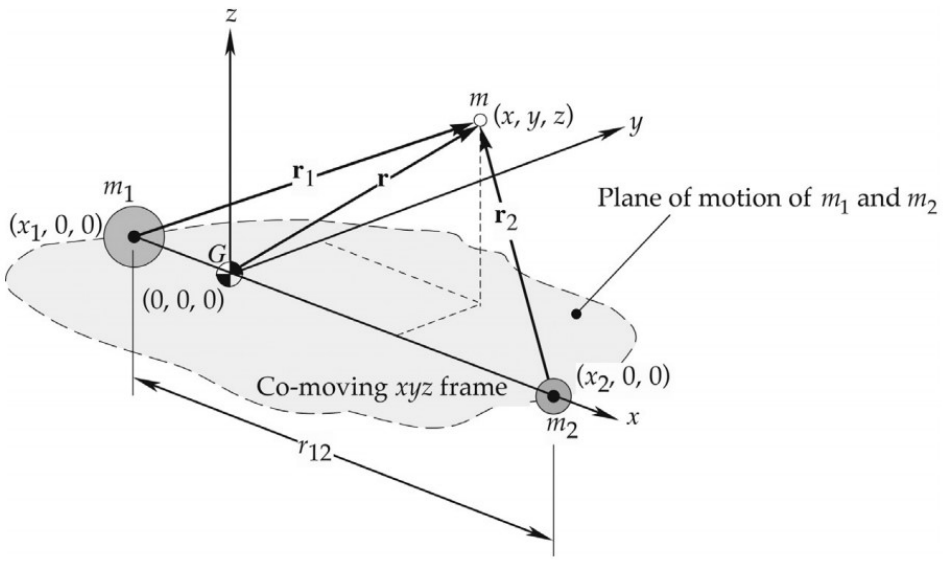


Figure 3.3 : 3-Body Problem

Body Problem (RTBP). The problem herein is to determine the motion of an object in a system of two or more large masses such that the mass of the object is negligible compared to other bodies. A special case of this problem is called Circular Restricted Three Body Problem (CRTBP) where the two masses revolve in circular orbits around their mutual center of gravity. Contrary to the Two-Body problem, this motion does not have a closed-form solution, but we can formulate equations of motion using a rotating frame of reference [31].

First, let's find a generalized solution for n-body problem. From Equation 3.2, we can express the gravitational force exerted by every object to each other as

$$F_{ij} = \frac{Gm_i m_j}{||r_{ji}^3||}\hat{r}_{ji} \tag{3.20}$$

The accelerations can be expressed

$$a_{ij} = \sum_{j=1\ (j\neq i)}^{n} \frac{Gm_j r_{ji}}{||r_{ji}^3||} \tag{3.21}$$

Then we can use an ordinary differential equation solver to numerically integrate the equations of motion given that we have the initial state of the masses. Now, let's consider the CRBP. Again using 3.2, we can write the forces exerted on the third body

m as

$$F_1 = -\frac{Gm_1 m}{r_1^2}\hat{u}_{r1} \tag{3.22}$$

$$F_2 = -\frac{Gm_2 m}{r_2^2}\hat{u}_{r2} \tag{3.23}$$

From Newton's second law, we can obtain the acceleration of the body from

$$m\ddot{r} = F_1 + F_2 \tag{3.24}$$

Plugging in

$$\ddot{r} = -\frac{\mu_1}{r_1^3}r_1 - \frac{\mu_2}{r_2^3}r_2 \tag{3.25}$$

Where

$$\mu_1 = Gm_1 \ \mu_2 = Gm_2$$

We can express the relative positions r_1 and r_2 as

$$\begin{aligned} r_1 = (x - x_1)\hat{i} + y\hat{j} + z\hat{k} = (x + \pi_2 r_{12})\hat{i} + y\hat{j} + z\hat{k} \\ r_2 = (x - \pi_1 r_{12})\hat{i} + y\hat{j} + z\hat{k} \end{aligned} \tag{3.26}$$

Where

$$\pi_1 = \frac{m_1}{m_1 + m_2} \ \pi_2 = \frac{m_2}{m_1 + m_2}$$

Considering that the velocity of the center of mass and the angular velocity of the circular orbit are constant, the 5-term relative acceleration formula can be combined into

$$\ddot{r} = (\ddot{x} - 2\Omega\dot{y} - \Omega^2 x)\hat{i} + (\ddot{y} + 2\Omega\dot{x} - \Omega^2 y)\hat{j} + \ddot{z}\hat{k} \tag{3.27}$$

Plugging 3.27 and 3.26 into 3.25, we obtain the equations of motion for CRTBP.

$$\ddot{x} - 2\Omega\dot{y} - \Omega^2 x = -\frac{\mu_1}{r_1^3}(x + \pi_2 r_{12}) - \frac{\mu_2}{r_2^3}(x - \pi_1 r_{12}) \tag{3.28}$$

$$\ddot{y} + 2\Omega\dot{x} - \Omega^2 y = -\frac{\mu_1}{r_1^3}y - \frac{\mu_2}{r_2^3}y \tag{3.29}$$

$$\ddot{z} = -\frac{\mu_1}{r_1^3}z - \frac{\mu_2}{r_2^3}z \tag{3.30}$$

In the implementation of the N-Body propagator, the user is allowed to select the Celestial objects to include its gravitational pull effect during the calculation of the

motion of spacecraft. Due to integration error and other perturbations, this propagator is not recommended for Earth-orbiting satellites, but the interplanetary trajectory calculations are mostly performed using these equations.

3.3 Standard General Perturbations (SGP4)

The SGP4 model is used to predict the movements of objects in near earth orbits that have a period of less than 225 minutes. This model takes into account various factors that can affect the orbit of an object, including the shape of the Earth, atmospheric drag, radiation, and the gravitational influence of other celestial bodies such as the sun and moon. The SDP model, on the other hand, is used for objects in orbit with a period of more than 225 minutes, or an altitude of 5,877.5 km in a circular orbit. Both models are used to accurately predict the movements of objects in space and are important tools in the fields of space exploration and satellite tracking. [32].

The model uses Two-Line Element (TLE) sets that are maintained and updated in the space catalog.

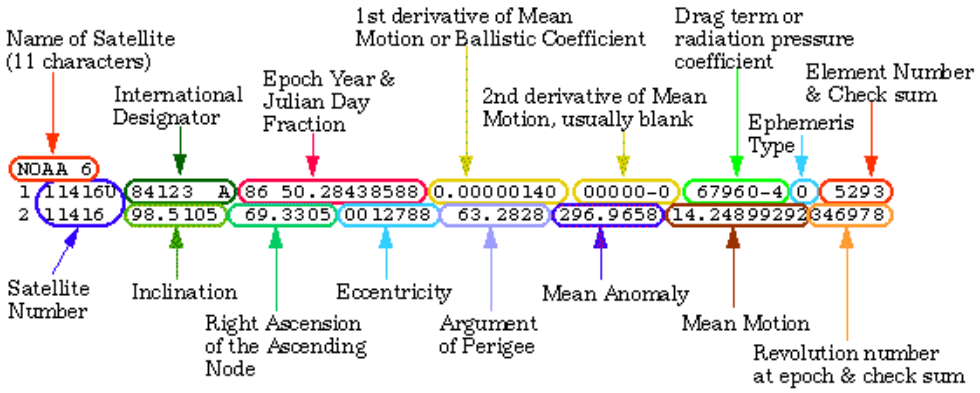


Figure 3.4 : Structure of TLE Data

The SGP4 model is effective at predicting the movements of objects in near earth orbit for a limited period of time, but its accuracy decreases over longer periods due to an error of about 1 km at the epoch, which increases by about 1-3 km per day. [32]. This means that it is important to regularly update the TLE data, which provides information on the orbit of an object, in order to make more accurate predictions. It's worth noting that while TLE data includes information on the classical orbital

elements, it is not suitable for use with other orbit propagators besides SGP4 and SDP4. This is because the TLE data represents mean orbital elements that are calculated to fit a set of observations, similar to how arithmetic and geometric means differ [33].

The working principle of the SGP4 model is given in Figure 3.5. The algorithm is complex and can be found in [34]. The overview of steps is given below.

- Calculate constants from the input elements and determine the six classical orbit elements.
- Add the secular effects of gravity on the mean motion, longitude of ascending node, and argument of perigee.
- Add the secular effect of drag on longitude of ascending node.
- Add the deep-space secular effects and long-period resonance effects to all orbital elements.
- Add the luni-solar periodic terms to the orbital elements and then long-term periodics

$$a_{xN} = e\cos w$$
$$a_{yN} = e\sin w + \frac{A_{3,0}\sin i_0}{8k_2\alpha\beta^2}(e\cos w)\left(\frac{3+5\theta}{1+\theta}\right)$$

  Where

$$A_{3,0} = -J_3R_E^3 \qquad k_2 = \frac{1}{2}J_2R_E^2$$

- Obtain eccentric anomaly E by solving Kepler's equation.
- Compute the quantities needed for short-term periodics.

$$e_L = (a_{xN}^2 + a_{yN}^2)^{1/2}$$
$$u = \arctan\frac{\sin\frac{a}{r}\left[\sin(E+w) - a_YN + \frac{a_xN(e\sin E)}{1+\sqrt{1-e_L^2}}\right]}{\cos\frac{a}{r}\left[cos(E+w) - a_xN + \frac{a_yN(e\sin E)}{1+\sqrt{1-e_L^2}}\right]}$$
$$r = a(1-\cos E)$$
$$\dot{r} = \mu_E\sqrt{a}e\sin E/r$$

$\Delta\Omega$, Δr, Δi and other quantities are found from equations that include the gravitational zonal harmonic of the Earth.

- Add the short-term periodics to obtain the osculating quantities

$$u_k = u + \Delta u$$
$$i_k = i_0 + \Delta i$$
$$\dot{r_k} = \dot{r} + \Delta \dot{r}$$
$$r\dot{f}_k = r\dot{f} + \Delta r\dot{f}$$

  and similarly for other Kepler elements.

- Calculate unit orientation vectors by performing coordination transforms using new inclination and longitude of ascending node.

$$U = M \sin u_k + N \cos u_k$$
$$V = M \cos u_k - N \sin u_k$$

  where M and N are transformation matrices.

- Find new position and velocity from

$$r = r_k U$$
$$\dot{r} = \dot{r_k} U + (r\dot{f})_k V \tag{3.31}$$

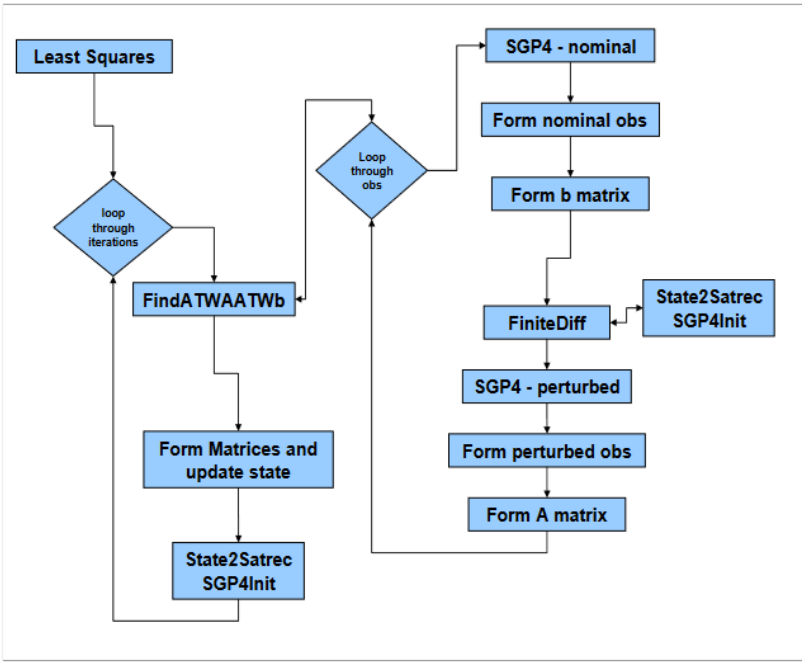


Figure 3.5 : Structural Organization of SGP4 [35].

In order to include the SGP4 propagator, open-source software was implemented from Center for Space website [35]. The two-line element data is parsed, given to the function as an input and the resulting position vectors are used to propagate orbit.

3.4 J2 and J4 Propagators

J2 and J4 propagation models are variations of the two-body propagator to include some amount of perturbation effects. In general, when the duration of analysis is short (e.g., couple of weeks), J2 Perturbation is used. On the other hand, if a long term analysis is required then J4 Perturbation is preferred. These models produce approximate solutions based upon mean Keplerian elements (see Mean Element Theory 2.2.7) considering the effects of secular gravitational and drag perturbations. The algorithm 9 shows the implementation of J2 propagator. The first time derivative of the mean motion divided by two ($\dot{n}_o$) and the second time derivative divided by six ($\ddot{n}_o$) are used to include drag effects, but they can be set to 0 if there is no estimate for these values available.

Algorithm 9: J2 Propagation [5]

1 Use Algorithm 2 to obtain Classical Orbit Elements $(a_o, e_o, i_o, \Omega_o, \omega_o, \theta_o)$ from the given state vector.

2 Calculate mean anomaly (M_o using true anomaly (θ_o) and eccentricity (e_o) as given in Table 2.3.

3 Compute the auxiliary variables.

$$p_o = a_o(1-e_o^2) \qquad n_o = \sqrt{\mu/a_o^3} \qquad \eta = n_o R_E^2 J_2 / p_o^2$$

4 Calculate the time derivatives of orbital elements.

$$\Delta a = -\frac{2a_o}{3n_o}\dot{n_o}$$

$$\Delta e = -\frac{2(1-e_o)}{3n_o}\dot{n_o}$$

$$\Delta\Omega = -\frac{3\eta}{2}\cos i_o$$

$$\Delta\omega = \frac{3\eta}{4}(4-5\sin^2 i_o)$$

$$\Delta M = \frac{3\eta}{4}\sqrt{1-e_o^2}(2-3\sin^2 i_o)$$

5 Update for perturbations.

$$a = a_o + \Delta a \Delta t$$

$$e = e_o + \Delta e \Delta t$$

$$\Omega = \Omega_o + \Delta\Omega\Delta t \quad [0, 2\pi]$$

$$\omega = \omega_o + \Delta\omega\Delta t \quad [0, 2\pi]$$

$$M = M_o + \Delta M_o \Delta t + n_o \Delta t + \frac{\dot{n_o}}{2}\Delta t^2 + \frac{\ddot{n_o}}{6}\Delta t^3 \quad [0, 2\pi]$$

6 Obtain true anomaly back from mean anomaly by solving Kepler's equation to get Eccentric Anomaly.

$$\theta = 2\tan^{-1}\left(\sqrt{\tfrac{1+e}{1-e}}\tan(E/2)\right)$$

7 Obtain the state vector from the updated orbit elements using Algorithm 1

The J4 propagator uses Algorithm 9 with different equations to include up to fourth zonal harmonic as given below.

$$\Delta\Omega = -\frac{3\eta}{2}\cos i_o + \frac{3\eta J_2 R_E^2}{32p_o^2}\cos i_o(12 - 4e_o^2 - (80 + 5e_o^2)\sin^2 i_o + \frac{15J_4 R_E^4 n\cos i}{32p^4}(8 + 12e^2 - (14 + 21e^2)\sin^2 i)$$

$$\Delta\omega = \frac{3\eta}{4}(4 - 5\sin^2 i_o) + \frac{9\eta J_2 R_E^2}{384p_o^2}(56e_o^2 + (760 - 36e_o^2)\sin^2 i_O - (890 + 45e_o^2)\sin^4 i_O - \frac{15J_4 R_E^4 n}{128p^4}(64 + 72e_o^2 - (248 + 252e_o^2)\sin^2 i_o + (196 + 189e_o^2)\sin^4 i_o)$$

$$\Delta M = \frac{3\eta}{4}\sqrt{1 - e_o^2}(2 - 3\sin^2 i_o)$$

3.5 High-Precision Numerical Orbital Propagator (HPOP)

The HPOP model takes into account a considerable number of force models and perturbations that affect the motion of the satellite. The differential equations of motion are integrated numerically to propagate the orbit. Numerical integration can be performed with various techniques which will be discussed later in this chapter. Due to the amount of force models used, highly accurate ephemerides can be generated with this method, for any type of orbit.

3.5.1 Force Models

- **Oblateness of the Earth**

  The Earth is shaped like an oblate spheroid, which means it is slightly flattened at the poles and bulges at the equator. This bulge exerts a force on satellites that pulls them back towards the equatorial plane if their orbit is not already aligned with the equator. While this force is relatively small compared to the force of the Earth's gravity, it can still be easily detected and can affect the satellite's orbit. As a result, the line of nodes shifts a few degrees each day. In addition to this long-term perturbation, other perturbations caused by the Earth's non-spherical shape can also affect the orbit of a satellite, particularly those in LEO. In 2-Body model, we assumed the Earth was a perfect sphere with homogeneous mass distribution, and used 3.3 to represent the acceleration on the satellite. Now, we will use the gradient

of the corresponding gravity potential U as

$$\ddot{R} = \nabla U \qquad\qquad U = G\int \frac{\rho(s)d^3s}{|r-s|} \tag{3.32}$$

Expanding the denominator in a series of Legendre polynomials as

$$\frac{1}{|r-s|} = \frac{1}{r}\sum_{n=0}^{\infty}\left(\frac{s}{r}\right)^n P_{nm}(cos\gamma) \qquad\qquad cos\gamma = \frac{r.s}{rs} \tag{3.33}$$

Where $P_n(u)$ is the Legendre polynomial of degree n and order m. Then representing the coordinates in longitude and geocentric latitude we have

$$x = r\cos\phi\cos\lambda$$

$$y = r\cos\phi\sin\lambda$$

$$z = r\sin\phi$$

We can express the gravity potential of the Earth as

$$U = \frac{GM}{r}\sum_{n=0}^{\infty}\sum_{m=0}^{n}\frac{R^n}{r^n}P_{nm}(\sin\phi)(C_{nm}\cos(m\lambda) + S_{nm}\sin(m\lambda)) \tag{3.34}$$

Where C_nm and S_nm are coefficients obtained after using the addition theorem of Legendre polynomials. The series of degree, order, and normalized numerical coefficients (C and S) to the spherical harmonics define a gravity model. A file consisting of these coefficients is used within the software to compute the oblateness effects. Some of the available gravity models for Earth are listed in Table 3.1. Note that some of the gravity models include tidal effects as well, which can be either tide-free or zero-tide.

Table 3.1 : Gravity Models.

Model	Degree & Order	Description	Tide Model
EGM96	360-360	**Earth Gravity Model 96.** Produced by a collaboration of NASA GSFC, National Imagery and Mapping Agency (NIMA), and Ohio State University.	Tide-Free
EGM2008	2159-2159	Updated version of EGM96 with slightly different GM and radius properties.	Tide-Free
JGM-2	70-70	**Joint Gravity Model v2.** A Geopotential model developed by GSFC and CNES.	Zero-Tide
JGM-3	70-70	**Joint Gravity Model v3.** A Geopotential model developed by GSFC.	Zero-Tide
GGM03	360-360	**Grace Gravity Model 3.** Based on flight data of GRACE by University of Texas.	Zero-Tide

- **Third body attractions**

  Both the Sun and the Moon exert a gravitational force on the satellite which are most prominent in geostationary orbits where the effects of the oblateness of Earth have a similar order of magnitude. The direction of the resulting acceleration changes depending on the alignment of the Earth, the Sun, and Moon. Other planets also cause gravitational attraction, with Jupiter and Venus taking the lead, but their effect is extremely small compared to luni-solar forces [13].

  The lunar and solar forces can be modeled using the same way in n-body propagator.

$$\ddot{R} = \mu \left(\frac{R-r}{|R-r|^3} - \frac{R}{|R|^3} \right) \tag{3.35}$$

  where R is the position of the perturbing body and r is the position f the satellite. Since both the Sun and the Moon are much farther than most satellites, we can use the following approximation when the satellite is pulled away from Earth

$$\ddot{R} \approx \frac{2\mu}{R^3} r$$

If it is perpendicular to the co-linear line connecting the Earth and the perturbing mass, then instead

$$\ddot{R} \approx -\frac{\mu}{R^3} r$$

Now all we have to do is find the coordinates of Sun and the Moon for the given time. While there are ways to calculate them using old measurements and assumptions, we can use the solar system ephemerides provided by the Jet Propulsion Laboratory (JPL). The lunar coordinates r_M are directly given, but we have to use the following equation to calculate the geocentric position of the Sun r_S

$$r_S = \hat{r}_S - r_{EMB} + \frac{1}{1+\mu*} r_M \tag{3.36}$$

Where $\hat{r}_S$ is the position of the Sun relative to the solar barycenter, r_{EMB} is its position relative to the Earth-Moon barycenter, and $\mu* \approx 81.3$ is the ratio of the masses of the Earth and the Moon. We can also expand upon this equation to include effects of other celestial bodies. The general collective equation for acceleration on the spacecraft caused by point mass k is given by

$$\ddot{R} = -\frac{\mu}{r^3}\mathbf{r} + G\sum_{\substack{k=1 \\ k\neq j}}^{n} m_k \left(\frac{\mathbf{r}_k - \mathbf{r}}{|\mathbf{r}_k - \mathbf{r}|^3} - \frac{\mathbf{r}_k}{|\mathbf{r}_k|^3} \right) \tag{3.37}$$

Where the term $-\frac{\mu}{r^3}\mathbf{r}$ is the acceleration due to central body, $\frac{\mathbf{r}_k - \mathbf{r}}{|\mathbf{r}_k - \mathbf{r}|^3}$ is the force of the k^{th} body on the spacecraft, and $\frac{\mathbf{r}_k}{|\mathbf{r}_k|^3}$ is the force of the k^{th} body on the central body [4].

- **Atmospheric Drag**

  Atmospheric particles colliding with the satellite result in a decrease in the kinetic energy and angular momentum of the orbit, causing the semi-major axis and the eccentricity to decrease. This has a major impact on the lifespan of a satellite, but does not significantly alter its orbital plane because the main component of drag is always directed opposite to the satellite's velocity vector. Drag force decreases linearly with atmospheric density, which shows an exponential decay as the altitude increases. Thus, drag is mostly a concern for low-altitude satellites.

$$\ddot{R} = -\frac{1}{2} C_d \frac{A}{m_s} \rho v_{rel}^2 \mathbf{e}_v \tag{3.38}$$

where

$$\mathbf{v}_{rel} = \mathbf{v} - \boldsymbol{\omega} \times \mathbf{r} + \mathbf{v}_w \tag{3.39}$$

- $\boldsymbol{\omega}$ = Angular velocity vector of the central body
- C_d = Drag coefficient
- A = Cross sectional area normal to v_{rel}
- ρ = Atmospheric density
- m_s = Mass of the spacecraft
- v_w = Velocity of the local wind
- e_v = Unit vector of the spacecraft relative velocity

The drag coefficient is related to various parameters such as the composition of the atmosphere, the weight and temperature of particles, surface material of the impinged area, etc. Hence, C_d has to be estimated as it is extremely difficult to have a priori knowledge about it. For spherical bodies, a rough approximation of $C_d = 2$ can be used, and a value between 2.0 and 2.3 is assumed for convex shapes.

In terms of area-to-mass ratio, the attitude of the spacecraft has to be known to determine this value at each step. If the satellite is always keeping its attitude, for example if it's always nadir-pointing, then this can be a constant value.

Another problematic parameter is the atmospheric density at the given altitude. There are a number of different atmospheric models available with differences around 20%. A brief comparison of the available models are given in Table 3.2. According to some studies [36,37], the statistical accuracies of these models are around 15% with no apparent improvement over the past decades.

Table 3.2 : Atmosphere Models.

Model	Description	Range
1976 Standard	A simple look-up table using 1976 standard atmosphere model.	86-1000 km
Harris-Priester	Considers solar flux and diurnal bulge.	0-1000 km
Jacchia 1971	Considers divisional and seasonal variations of the atmospheric composition.	100-2500 km
Jacchia 1960-1970	Predecessors to the 1971 model.	90-2500 km
Jacchia 1977	An updated version of 1971, revised in 1981 [38]. Requires a lot more computational power but does not significantly improve accuracy	100-2500 km
Jacchia-Roberts	Uses analytical methods for increasing the performance of 1971 model	>100 km
CIRA 1972	Uses numerical integration instead of interpolating polynomials. Otherwise similar to Jacchia 1971	>90 km
MSIS 1986	Empirical density model based on satellite data by Hedin, replacing CIRA [39]	90-1000 km
MSISE 1990	Updated version of MSIS 1986	0-1000 km
NRLMSISE 00	Developed by US Naval Research Laboratory using data from satellites	0-1000 km

Another critical point in drag calculation is the prediction of Solar and Geomagnetic Indices, as solar and geomagnetic activities significantly affect the atmosphere. Orbit determination can be done using measurements, but for propagation the predictions of these indices are required. The Center for Space Standards and Innovation (CSSI) provides observations and predictions (from NOAA) for F10.7 flux and geomagnetic indices, which can be used with Jachhia Roberts and MSIS models [40]. The short-term predictions of the solar flux is done considering the solar activity caused by synodic solar rotation with a period of 27-days. However, for mission planning, long-term effects caused by the 11-year solar cycle have to be considered as well. The solar dynamo model by Schatten [41] is used to predict average values in this case, and provided as a file by ISWA/GSFC [42].

- **Solar Radiation Pressure**

  Energetic particles from the sun impinge on the satellite, transferring their impulse. The intensity of this radiation changes depending on solar activity, which periodically peaks every 11 years. Eccentricity and longitude of perigee are mainly affected from SRP. This effect is most prominent for satellites with large solar panels.

  The equation of acceleration due to SRP, assuming that the surface normal points in the Sun's direction, is simply given by [43]

$$\ddot{R} = \nu P_s C_R \frac{A}{m_s} \frac{\mathbf{r}_{vs}}{r_{vs}^3} r_{AU}^2 \tag{3.40}$$

  - ν = Eclipse factor. $\nu = \left(1 - \frac{p}{100}\right)$ where p is the percent shadow.
  - P_s = Solar radiation pressure at 1 AU ($P_s \approx 4.56 \cdot 10^{-6} Nm^{-2}$)
  - C_R = Radiation pressure coefficient and is related to reflectivity ε as $C_R = 1 + \varepsilon$
  - $\mathbf{r}_{vs}$ = Sun vector given as $\mathbf{r} - \mathbf{r}_s$
  - r_{AU} = One astronomical unit.
  - A = SRP area of spacecraft
  - m_s = Mass of the spacecraft

  Note that this model is not sufficient for applications that require high-precision. The complex structure of the satellite and the surface properties have to be incorporated in this case, often using a finite element method.

- **Relativistic Effects**

  The size of the effects of general relativity is given by the Schwarzschild radius of Earth $(2\mu)/c^2 \approx 1$ cm. If this level of accuracy is desired, then relativistic effects should be considered [13]. The Newtonian equation of motion can be further corrected by applying post-Newtonian terms which account for three contributions, the Schwarzschild solution, geodesic precession, and Lense-Thirring precession.

$$\ddot{R} = \frac{\mu}{c^2 r^3}\left(\left(4\frac{\mu}{r} - v^2\right)\mathbf{r} + 4(\mathbf{r}\cdot\mathbf{v})\mathbf{v}\right) + 2(\Omega \times \mathbf{v}) + 2\frac{\mu}{c^2 r^3}\left(\frac{3}{r^2}(\mathbf{r}\times\mathbf{v})(\mathbf{r}\cdot\mathbf{J}) + (\mathbf{v}\times\mathbf{J})\right) \tag{3.41}$$

Where

$$\Omega = \frac{3}{2}\mathbf{v}_{B/S} \times \left(\frac{-\mu \mathbf{r}_{B/S}}{c^2 r_{B/S}^3} \right)$$
$$\mathbf{J} = \mathbf{R}_B^{I/F} \begin{bmatrix} 0 & 0 & \frac{2}{5} R_B^2 \omega_B \end{bmatrix}^T$$

- c = Speed of light
- J = Angular momentum per unit mass of the central body
- $r_{B/S}$ = Position of the central body with respect to the Sun.
- $v_{B/S}$ = Velocity of the central body with respect to the Sun.
- $\mathbf{R}_B$ = Mean equatorial radius of the central body.
- ω_B = Spin rate
- $\mathbf{R}_B^{I/F}$ = Fixed to inertial rotation matrix of the body.

- **Others**

  There are several smaller perturbations in addition to those discussed so far, typically causing accelerations in the order of $10^{-15} - 10^{-12} km/s^2$. These are mostly solid Earth tides, and the radiation pressure caused by Earth's albedo.

Earth Radiation Pressure

The solar radiation that hits the surface of the Earth is reflected and scattered, causing an optical albedo that exerts a slight pressure on the spacecraft. Similar to the SRP, it only occurs during daytime, and varies according to weather conditions as well as surface characteristics. The acceleration due to this phenomenon is given by the following equation, where the albedo and emissivity can be expressed considering J_2 terms [44].

$$\ddot{R} = \sum_{j=1}^{N} C_R \left(\nu_j a_j \cos\theta_j^E + \frac{1}{4}\varepsilon_j \right) P_s \frac{A}{m} \cos\theta_j^S \frac{dA_j}{\pi r_j^2} e_j \tag{3.42}$$

- dA_j = Earth area elements
- ν_j = Earth element shadow functions
- θ_j^E = Angle of the Earth surface normal to the incident radiation

- θ_j^S = Angle of the satellite surface normal to the incident radiation
- ε = Emissivity
- a = Albedo
- e_j = Unit vector pointing from the surface element to the spacecraft

Earth Tides

It is a well known fact that the gravitational attraction of third bodies, most notably the Sun and the Moon, cause oceans to rise and fall, a phenomenon called ocean tides. These luni-solar effects are not limited to the water which is quite malleable, but they also cause elastic deformation of the Earth as it is a rigid body and not a point mass. These are called solid tides, and cause the gravity field of the Earth to vary with time.

An important concept that has to be known when talking about tides is the Love number, which can be defined as the ratio of the tidal potential and the resulting perturbed gravity potential. Tidal models include Love numbers of various degrees to be used in the following equation that results in the time-dependent corrections to the geopotential coefficients C and S.

$$\begin{Bmatrix} \Delta C_{nm} \\ \Delta S_{nm} \end{Bmatrix} = 4k_n \left(\frac{GM}{GM_E} \right) \left(\frac{R_E}{s} \right)^{n+1} \sqrt{\frac{(n+2)(n-m)!^3}{(n+m)!^3}} P_{nm}(\sin\phi) \begin{Bmatrix} \cos(m\lambda) \\ \sin(m\lambda) \end{Bmatrix} \tag{3.43}$$

- k_n = Love numbers of degree n
- λ = Earth-fixed longitude of the tide generating body
- ϕ = Earth-fixed latitude of the tide generating body
- s = Geocentric distance to the disturbing body

One interesting observation is that since tidal potential is disproportional to the cube of the distance, the lunar tides are more prominent than the solar tides, about twice as much. Another type of tides are pole tides, which are caused by the movement of Earth's poles (known as the Chandler wobble) contributing to the centrifugal potential. This results in a small displacement of the Earth's crust [45]. Finally, the ocean tides are also one of the factors affecting satellite geodesy, albeit much less

than solid tides. An ocean tide potential that can be expended in terms of spherical harmonics can be used to express their effect by mapping them to the geopotential coefficients as well.

Spacecraft Thrust

The propulsion systems used on the spacecraft, whether for attitude control or maneuvering, directly affect the orbital motion. The thrust scale factor and duty cycle are parameters related to the computation of acceleration. One thing to note is that accelerations due to thrust result in discontinuities in the equations of motion, so the numerical integration algorithm that's used to calculate the motion how to be restarted before and after the thrusters are used.

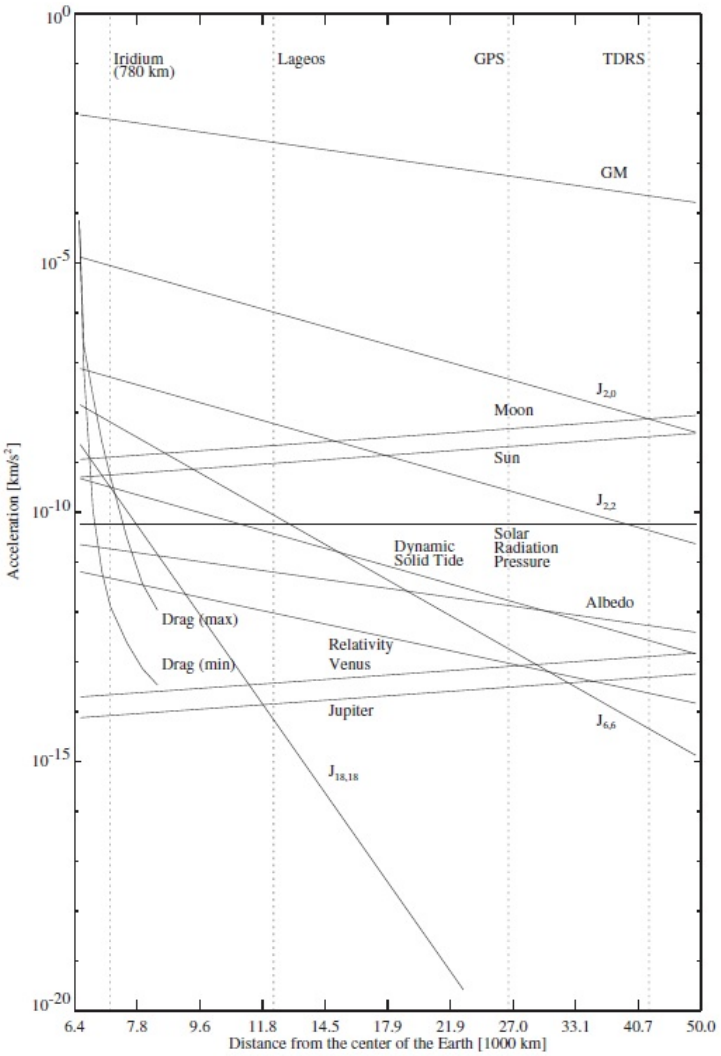


Figure 3.6 : Perturbations Order of Magnitudes [13].

3.5.2 Integrators

To obtain the highly accurate satellite orbit calculations that are currently needed, it is necessary to use numerical methods to solve the equation of motion [46]. There are multiple different choices when it comes to numerical integration with each of them having their strengths and weaknesses, which makes it impossible to have one method that performs the best under every condition. Hence, multiple solvers were included in the software that the user can choose from. In this chapter the fundamental principles and properties of the implemented integrators are described and evaluated according to the task at hand.

- **Runge-Kutta**

  Runge-Kutta is a single step method that breaks the interval into several smaller steps and calculates the estimates of the integration result at each step, h. In the classical 4^{th} order method, the increment function is given by the weighted mean of four slopes, which are evaluations of the function f at different steps. There is no need to compute the derivatives, hence the Runge-Kutta methods are fairly simple to implement and can be applied to a large set of use-cases.

  The generalized s-stage RK formulation is as follows. The estimation of the integration result at state i is given by

  $$k_i = f(t + c_i h, y(t) + \sum_{j=1}^{i-1} a_{ij} k_j) \tag{3.44}$$

  Where a_{ij} contains a set of coefficients unique to the instance of RK at hand and c_i is the sum of the coefficients at row i. Using the stage calculation result and a different set of coefficients, b_j, the total integration step can be calculated by

  $$y(t+h) = y(t) + \sum_{j=1}^{s} b_j k_j \tag{3.45}$$

  This equation can be used twice, once for integration order of p and once for $p+1$, to obtain estimates for different orders. The difference between these two solutions can be used to estimate the error of truncation of the pth-order formula.

  After estimating the error, the integration step can be adjusted to a size that is more suitable for the desired level of accuracy. If the solution obtained in the step has

lower accuracy than required, a smaller step-size has to be used to repeat this step, which can be calculated by

$$h^* = \sigma h \frac{\alpha}{\varepsilon}^{1/(m-1)} \tag{3.46}$$

Where

- σ = Safety factor to avoid unnecessary iteration; defaults to 0.9.
- α = Desired accuracy.
- ε = Obtained accuracy.
- m = Truncation order of the series expansion.

The step size parameter may be increased as well, in this case the term $m-1$ becomes m.

In some cases, it may be required to keep the maximum step size at a constant value. Hence, the software uses a "maximum allowed step" parameter as input from user.

Although this type of step size control is able to adapt the current step size to the characteristics of the differential equation, it doesn't eliminate the need for the user to provide an initial estimate for the starting step size.

- **Adams-Bashford-Moulton**

  This is a Predictor-Corrector, which is a multi-step method that attains high accuracy but requires memory as it uses past data points. The software employs a 4^{th} order Adams-Bashford predictor and Adams-Moulton corrector pair based on Bate, Mueller and White [47].

  The predictor uses the derivative at the current state as well as three previous states to extrapolate the new state as follows

$$y_{i+1}^{*j} = y_i^j + \frac{h}{24}\left[55f_n^j - 59f_{n-1}^j + 37f_{n-2}^j - 9f_{n-3}^j\right] \tag{3.47}$$

  Then the corrector is applied on this derivative information, using the information at the original and two past states to yield a final state.

$$y_{i+1}^j = y_i^j + \frac{h}{24}\left[9f_{n+1}^{*j} + 19f_n^j - 5f_{n-1}^j + 1f_{n-2}^j\right] \tag{3.48}$$

The estimated accuracy of the solution is

$$ee = \frac{19}{270}|y_{i+1}^{*j} - y_{i+1}^{j}| \tag{3.49}$$

- **Gauss-Jackson**

  This method is optimized for integrating systems of second order equations. Its predictor component alone is typically more accurate than the predictors in other methods, and it also includes a corrector. It has a strong ability to minimize the impact of accumulated round-off error [47].

$$y_i = h^2\left((\sum \ddot{y}_i)^2 + \frac{1}{12}\ddot{y}_i - \frac{1}{240}\partial^2\ddot{y}_i + \frac{31}{60480}\partial^4\ddot{y}_i - \frac{289}{36288}\partial^6\ddot{y}_i\right) \tag{3.50}$$

  Where

$$\partial^i y(t) = \partial^{i-1} y(t + \frac{h}{2}) - \partial^{i-1} y(t - \frac{h}{2})$$

4. DEEP LEARNING BASED ORBIT PROPAGATOR

Classical trajectory prediction algorithms produce estimations by assuming that the dynamic models of the system are known and, to make predictions simulations of these dynamic models are used. In real world application, there are a number of factors limiting the accuracy of the assumed dynamics models, such as orbit determination errors caused by the constraints of the measuring instruments. Our knowledge of the physical world is not sufficient enough to create perfect models as it is near impossible to predict some perturbations precisely, such as solar activity which is only an approximation based on statistical data, as well as the atmosphere models and the area of the satellite that drag force affects, which also change depending on the attitude model. All of these uncertainties cause errors during orbital propagation, which add up at each step along with integration errors. To solve this problem, machine learning methods can be used.

Machine learning (ML) is a field of artificial intelligence that involves using mathematical models to enable computers to learn and improve automatically from experience, without being specifically programmed to do so. ML models can perform a specific task without direct instruction. Furthermore, models generalize the patterns of the data and the patterns can be used to produce predictions. Today, a wide variety of applications are available such as computer vision, self-driving cars, data mining, email, malware filtering, search-engines. The new, advanced methods combined with rapidly evolving computational capabilities, has created an environment where aerospace problems can be solved.

The idea in this study is to improve the orbit prediction obtained by high-fidelity numerical models such as HPOP using the estimation of error, so that the orbit is not predicted from scratch but improved upon using past information of thousands of Earth-orbiting objects.

4.1 Background and Models

4.1.1 Long-Short Term Memory (LSTM) Structure

Differences of LSTM neural network structure from fully connected neural networks; it can work much better if the data depends on time or previous data [48]. For example, a network structure that classifies a sentence as a positive or negative comment, rather than having to look at the words one by one, causes them to make much more accurate predictions about the sentence. In maneuvering or vehicle type prediction, looking at historical radar data and predicting a network with information about their order will give much better results. [48]

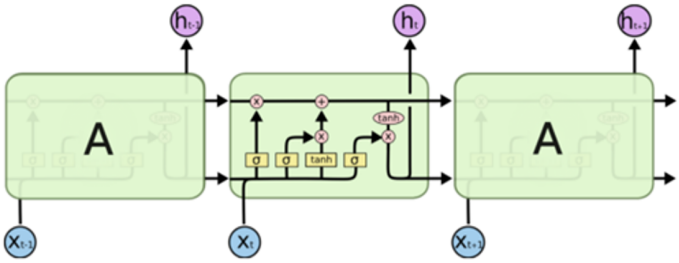


Figure 4.1 : LSTM Structure

In Figure [48], example of a LSTM cell can be seen. The X_t data, which comes as input from any time t, can produce 4 estimates in a cell.

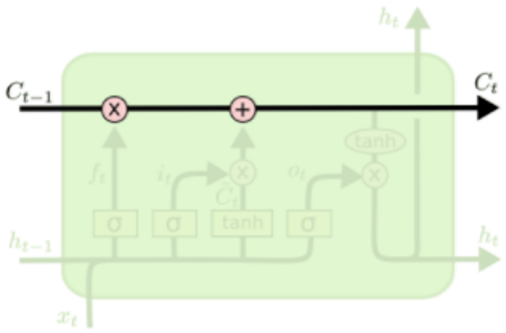


Figure 4.2 : LSTM Structure

The main logic behind the LSTM is the cell state, which is reminiscent of the conveyor belt and carries information between the states of the cell at different times. It is also

the structure that establishes the relationship between the satellite's historical data and its future [48]

4.1.2 Forget Gate

The forget gate is the gateway that decides which information to forget by processing the x_t data from a previous cell at the time of t (this zero can be selected for the first cell), with the h_{t-1} function. As a result, if the output from the sigmoid function is zero, forget this information and 1 is to keep this information. [48]

$$sigmoid(z) = \frac{1}{1+e^{-z}} \tag{4.1}$$

$$f_t = sigmoid(W_f[h_{t-1}, x_t] + b_f) \tag{4.2}$$

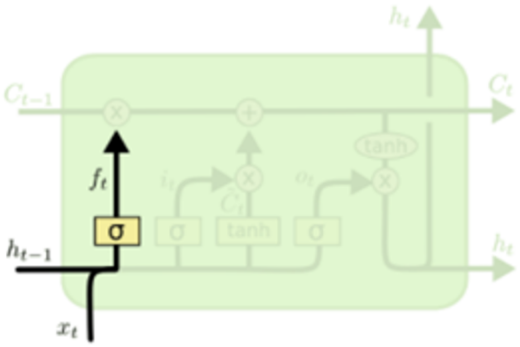


Figure 4.3 : Forget Gate of a LSTM Cell

4.1.3 Input Gate

It is the structure between the cell state and the output of the cell from the incoming data. Decides which part of the new incoming data to keep [48].

$$i_t = sigmoid(W_i[h_{t-1}, x_t] + b_i) \tag{4.3}$$

$$C_t = tanh(W_c[h_{t-1}, x_t]) + b_c) \tag{4.4}$$

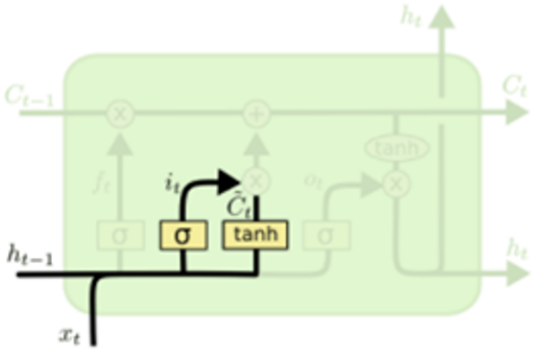


Figure 4.4 : Input Gate of a LSTM Cell

4.1.4 Output Gate

Finally, the cell state, the filtered input, and the output of the previous time step form the output of the cell for that time step. This output will also be the input of the next time step.

$$o_t = sigmoid(W_i[h_{t-1}, x_t] + b_o) \tag{4.5}$$

$$h_t = o_t * tanh(C_t) \tag{4.6}$$

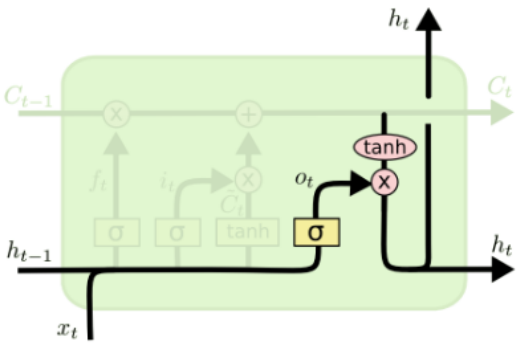


Figure 4.5 : Output Gate of a LSTM Cell

4.1.5 Dense Layer

The data from the LSTM is often not the desired probability distribution, and in addition to the LSTM structure, a layer of dense can be added to increase the capacity of the model.

$$relu(x) = \begin{cases} 0 & x = 0, \\ x & \text{x>0} \end{cases} \tag{4.7}$$

$$h = relu(Wx + b) \tag{4.8}$$

When the intermediate layers are activated with relu, by activating the last dense layer with the softmax function, the probability distribution is obtained [48].

4.1.6 Softmax Function

The softmax function is used to convert the outputs from the model as scores to meaningful probability estimates [48].

$$P(Y = k|X = x_i) = \frac{e^{sk}}{\sum_j e^{sj}} \tag{4.9}$$

$$s = f(x_i; W) \tag{4.10}$$

The matrix resulting from the Softmax function will give us a probability distribution with a sum of 1.

4.1.7 Loss Function

The loss function is used to measure the accuracy of the model. The loss function indicator of how far the estimation really is, in other words scores from the model. The loss score of the model close to zero means that model may be making good predictions.

4.1.8 Categorical Cross Entropy Function

Categorical cross entropy function is used when classifying multiple elements. [48]

$$Loss = -\sum_{i}^{C} t_i log(f(s)_i) \tag{4.11}$$

$f(s)_i$ is the output of the softmax function.

4.1.9 Mean Square Error

Mean square error (MSE) is used to measure the error of the regression problems. While making predictions on satellite state, error is calculated as follows. The more error there is, the more mean square error punishes as loss [48].

$$MSE = \frac{1}{n}\Sigma_{i=1}^{n}(y_{predict} - y_{real})^2 \tag{4.12}$$

4.1.10 Optimization

Initially, a randomly selected W matrix produces incorrect estimates and yields a high numerical result of the loss function. Finding a W matrix that will reduce the loss function (produce better estimates) is possible with optimization.

4.1.11 Stochastic Gradient Decent

Stochastic Gradient Decent S is an iterative process for optimizing an objective function with suitable smoothness properties.

Algorithm 10: Stochastic Gradient Decent Algorithm

Result: Local Minimum of The Loss Function

for *i = 1,2,3,...n* **do**

  $w = w - \nu \Delta Q_i(w)$

end

SGD and their variations are often used to train neural networks. In this project, SGD, ADAM and Adagrad optimizers are used.

4.2 Dataset

The main goal of the model is to reduce the discrepancy between the trajectory estimations and the grand truth. Therefore, we use real data from satellites to train the model. The huge database of historical TLE files (nearly 10GB) were used along with SGP4 propagator, and orbits were constructed by constantly updating the TLEs, on average once 12 hours, to achieve as much accuracy as possible. These data are then predicted with the HPOP for the exact time steps. The difference that starts to emerge between these two data is the target that the model tries to minimize.

$$\text{The Dataset 1: } HPOP(X,Y,Z,V_x,V_y,V_z) \tag{4.13}$$

$$\text{The Dataset 2: } Measurement(X,Y,Z,V_x,V_y,V_z) \tag{4.14}$$

$$\text{The Objective: min } MSE(HPOP - Measurement) \tag{4.15}$$

Satellite data orbiting in different low earth orbits were used to train the models. Starlink, ISS and many other freely available orbit data were used.

4.3 Model Architecture

There are many types of deep neural network architectures that can be used for time series analysis, including Recurrent Neural Networks (RNNs) and Long Short-Term Memory Networks (LSTMs). Convolutional Neural Networks (CNNs) are commonly used for spatial analysis of high-dimensional data such as images, but they can also be applied to one-dimensional data with long-term dependencies. Four different types of RNNs are implemented in this case, with the first two models being encoder-decoder architectures, which connect the encoder output directly to the decoder input at the latent space."

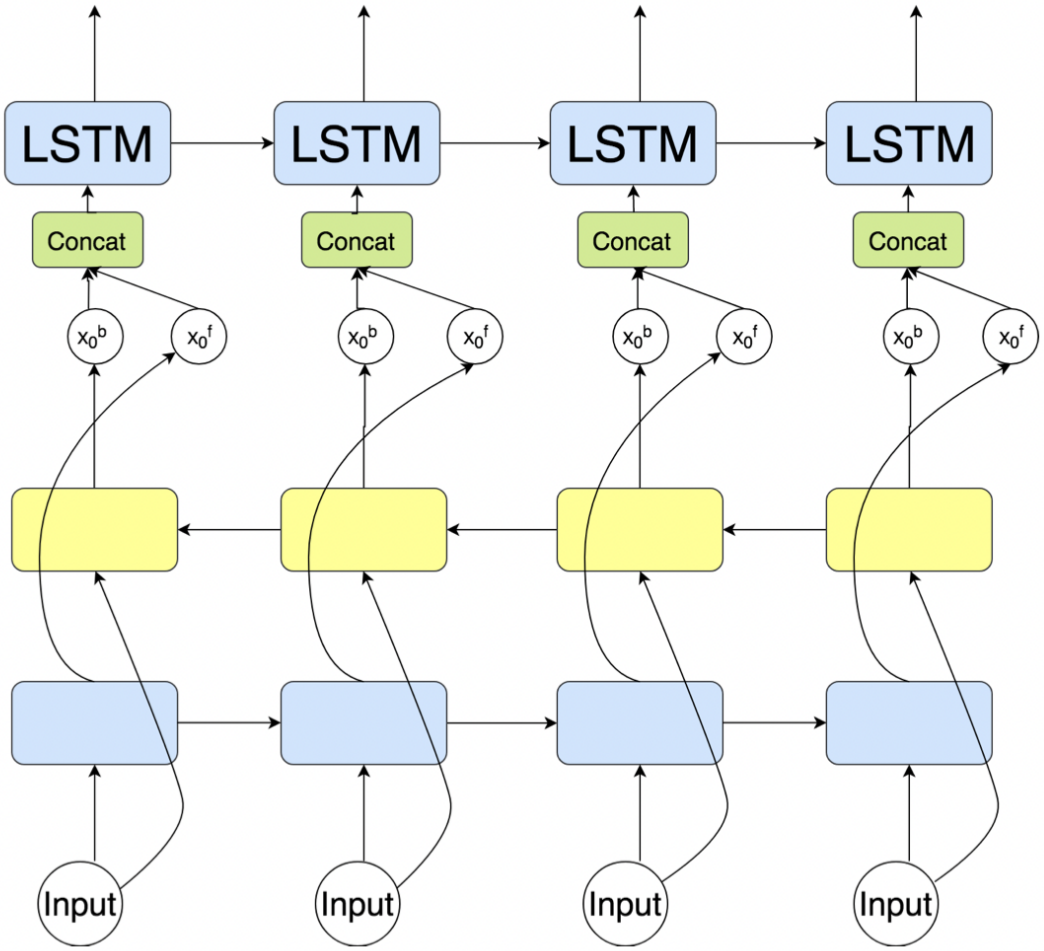


Figure 4.6 : LSTM Connection Architecture

The parameters of the prediction model are adjusted based on the results of simulations. It was found that adding more layers to the model does not necessarily

improve accuracy, but it does increase computation time. On another note, increasing the number of neurons in a single layer may improve the overall model accuracy.

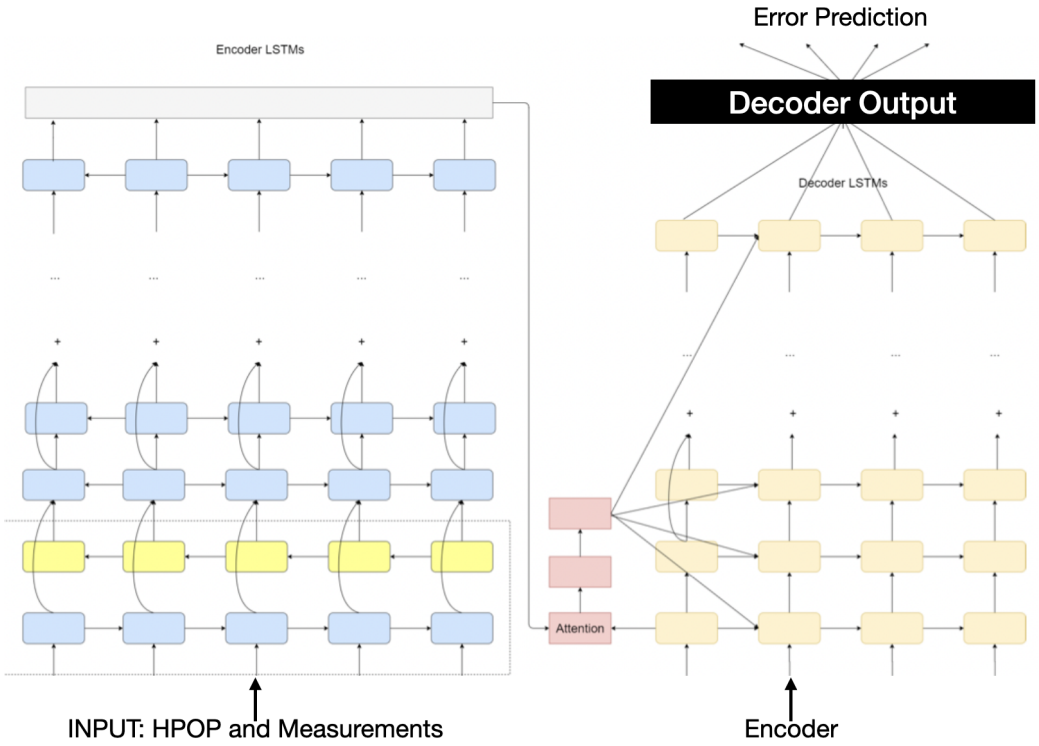


Figure 4.7 : Model Architecture

The activation function at the last layers of the prediction part is a linear function. A dropout method was used to prevent over-fitting.

Layer (type)	Output Shape	Param #	Connected to
encoder_input (InputLayer)	[(None, 10, 7)]	0	[]
LSTM1 (LSTM)	(None, 10, 256)	270336	['encoder_input[0][0]']
LSTM2 (LSTM)	(None, 32)	36992	['LSTM1[0][0]']
dense1 (Dense)	(None, 64)	2112	['LSTM2[0][0]']
dense2 (Dense)	(None, 64)	2112	['LSTM2[0][0]']
dense3 (Dense)	(None, 64)	2112	['LSTM2[0][0]']
dense4 (Dense)	(None, 64)	2112	['LSTM2[0][0]']
dense5 (Dense)	(None, 64)	2112	['LSTM2[0][0]']
dense6 (Dense)	(None, 64)	2112	['LSTM2[0][0]']
out1 (Dense)	(None, 1)	65	['dense1[0][0]']
out2 (Dense)	(None, 1)	65	['dense2[0][0]']
out3 (Dense)	(None, 1)	65	['dense3[0][0]']
out4 (Dense)	(None, 1)	65	['dense4[0][0]']
out5 (Dense)	(None, 1)	65	['dense5[0][0]']
out6 (Dense)	(None, 1)	65	['dense6[0][0]']

Total params: 320,390
Trainable params: 320,390

Figure 4.8 : Model Parameters

The model parameters can be seen at Figure 4.8

4.3.1 Implementation

Keras is a library written in python programming language that allows users to create and train deep learning models very quickly. All of the methods described above have been implemented in the Keras library. Although Keras is easy to use, it is not flexible enough to write complex layers. In order to develop better models, Tensorflow and Pytorch libraries are implemented.

4.4 Results

In this section 4.2, the results of trajectory estimation using the deep learning method are discussed. As mentioned in the Section 4.2, the main objective is to keep difference between model predictions and real data as low as possible.

The Figures 4.9, 4.10, 4.11 show the training process of various models.

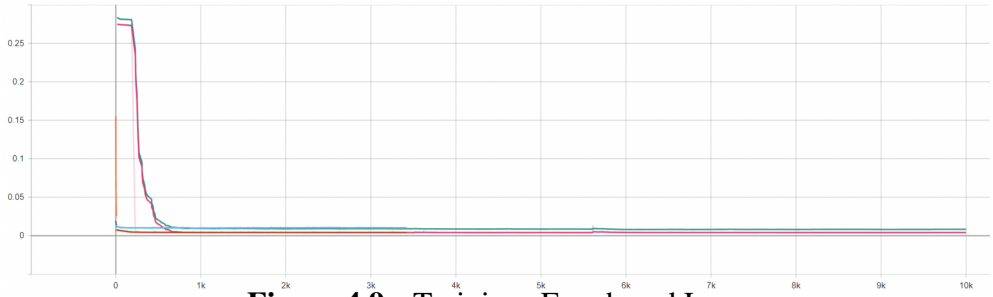


Figure 4.9 : Training: Epoch and Loss

The graphs show the normalized values, values are normalized 10^3 and 10^2 for position and velocity vectors, respectively. As can be seen in the graphs after about 1000 steps of training, the error is less than 10^{-3} which corresponds to several meters.

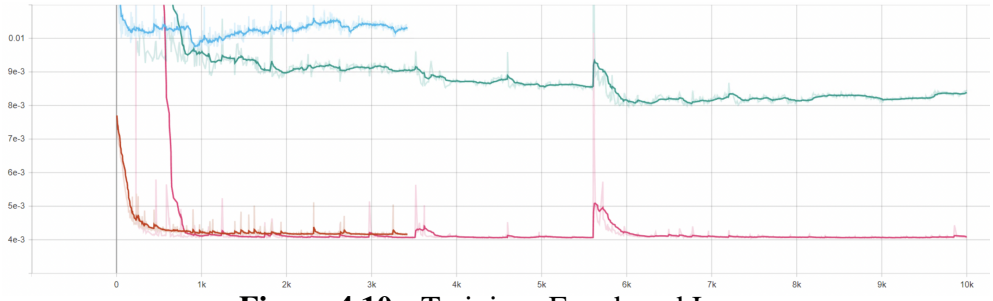


Figure 4.10 : Training: Epoch and Loss

Figure 4.10 is the zoomed version of Figure 4.9. As can be seen in the figure, all loss functions were converged to $5 \cdot 10^{-3}$ At this point, tests were run to find out which model worked better. In the Figure 4.11, it can be observed that the green model gives better results in the validation conditions.

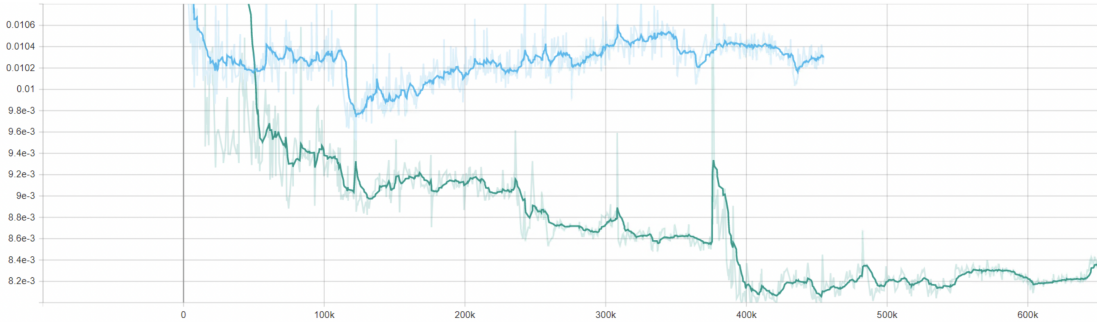


Figure 4.11 : Validation: Epoch and Loss

Also after training for a long time significantly decreases validation loss, on the other hand, training loss stays converged. The best performing model is deployed to the software framework to increase accuracy of the trajectory prediction.

5. SIMULATION ENVIRONMENT

In order to properly plan certain studies, it is important for engineers to be familiar with not just propagation, the most common type of orbit analysis, but also other analytical tools such as orbit types, maneuvers, communication and power analyses. This also includes understanding the concepts of reconnaissance and surveillance from space. Moreover, there are several specific types of orbits that can be useful for specialized purposes, such as sun-synchronous, which maintain a constant angle with the Sun, frozen orbits, which fix one or more orbital elements against perturbations, and repeat-groundtrack satellites, which provide periodic coverage over a particular ground location [49]. While these types of analyses provide a good starting point for mission planning, it is important to remember that more detailed studies will be necessary in order to fully understand the effects of perturbations on the orbit for any given mission [5]. The simulation environment is intended to aid during the mission planning phase, by providing the tools for engineers to determine critical parameters such as how much fuel to keep on-board, how much solar power can be generated, and whether the specified orbit is sufficient for providing a communication window with points of interest.

5.1 Orbit Tool

The orbit tool provides the framework to quickly create mission-specific orbits such as sun-synchronous, geosynchronous, repeat-ground track, and Molniya orbits.

5.1.1 Sun Synchronous Orbits

Sun synchronous orbits (SSO) allow a satellite to pass over a specific location on the surface of Earth's at the same local solar time during each revolution, by taking advantage of the oblateness of Earth. This can be achieved by tilting the orbit plane slightly relative to the equatorial plane of the Earth and choosing an appropriate orbital

period. SSOs are often used for Earth observation satellites because they allow for consistent lighting conditions on the ground, making it easier to observe and compare features over time. Additionally, SSOs can also be useful for communications satellites because they allow for predictable coverage patterns.

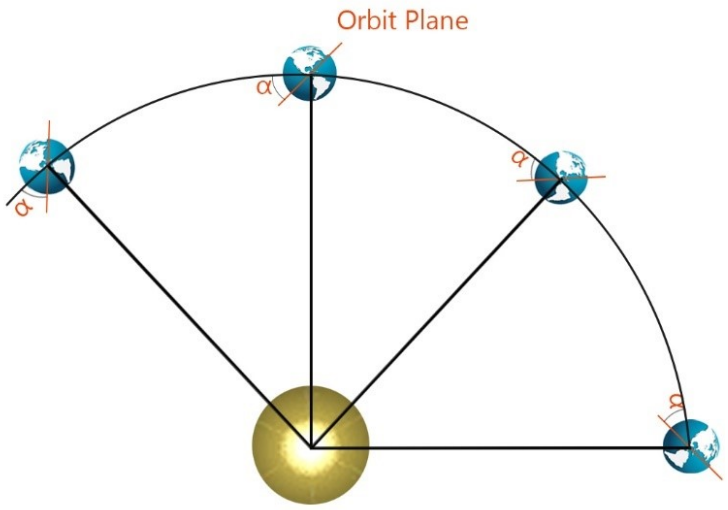


Figure 5.1 : Sun-Sycnhronous Orbit

In an SSO, the orbital plane rotates in alignment with the daily rotation of Earth around the Sun, which is equivalent to a rotation of 360° every 365.26 days. The advance rate of the ascending node is then

$$\dot{\Omega} = \frac{2\pi}{365.26 \times 24 \times 3600} = 1.991063853 \times 10^{-7} \text{rad/sec}$$

The change in right ascension due to oblateness is given by

$$\dot{\Omega} = -\left[\frac{3}{2}\frac{\sqrt{\mu_E} J_2 R_E^2}{(1-e^2)a^{7/2}}\right]\cos i \tag{5.1}$$

SSOs are usually defined by the local time of the ascending node (LTAN) (e.g., 10.30 a.m. orbit). An alternative is local time of descending node, which is equal to LTAN + 180°. The LTAN parameter can be converted into longitude of ascending node, λ_{asc}, which can be converted to RAAN as shown in Table 2.3.

$$\lambda_{asc} = LTAN - t_{epoch} \tag{5.2}$$

Hence, in orbit tool, either altitude or inclination can be specified, along with Node Definition (LTAN or LTDN) and the rest of the orbital parameters are computed according to Equations 5.1 and 5.2.

5.1.2 Critically-Inclined Orbits

In critically inclined orbits, perigee is fixed at a specific latitude and does not change over time. The rate of change of argument of perigee due to J_2 term is given by

$$\dot{\omega} = -\left[\frac{3}{2}\frac{\sqrt{\mu_E}J_2R_E^2}{(1-e^2)a^{7/2}}\right](\frac{5}{2}\sin^2 i - 2) \tag{5.3}$$

When $\dot{\omega} = 0$, the line of apsides remains stationary. The two inclination values that satisfy this condition are 63.4° and 116.6°, with the latter resulting in a retrograde orbit.

The software requires four parameters to create this kind of an orbit, the direction (posigrade or retrograde) to determine inclination, apogee and perigee altitudes, and longitude of the ascending node.

Molniya (lightning) orbits are type of critically inclined orbits, characterized by a high eccentricity. The apogee of these orbits are located in the Northern Hemisphere, so the satellite spends a significant amount of time at high latitudes. The Molniya telecommunications satellites are inserted into orbits with inclination of 63.4° and a period of 12 hours, which corresponds to a very large semi-major axis of 53,000 km. The half-day orbit creates repeating ground tracks, which are detailed below. The critical inclination of the orbit means that the argument of perigee remains constant at a preferred value (which is usually around 270°), maintaining the desired viewing geometry. However, the luni-solar effects and zonal harmonics of Earth gravity cause the argument of perigee to change ever-so-slightly, which is usually corrected through station-keeping maneuvers [16].

It is possible to create a Molniya orbit using the Orbit Tool with just three parameters, perigee altitude, argument of perigee, and longitude of apogee. The last parameter is useful because it tells over which longitude on Earth will the satellite spend most of its time. It can be found by setting Ω in such a way that after half a revolution, the sub-satellite point shall be at the desired longitude.

5.1.3 Geosynchronous Orbits

A satellite in a Geosynchronous orbit appears to remain stationary relative to a particular location on the surface of the planet. This is achieved by having an altitude of approximately 35,786 kilometers, which corresponds to an orbital period of one sidereal day. Geosynchronous orbits are often used for communications satellites because they allow for continuous coverage of a particular region. They are also useful for weather forecasting and military surveillance. However, geosynchronous orbits are affected by other perturbations as well, which can cause the satellite's position to drift over time.

Setting the eccentricity to 0, and altitude to 35,786 km, we only require two inputs from the user, inclination and the sub-satellite point (same as the longitude of the ascending node, in True of Date coordinate system), which is the key parameter in geosynchronous orbits.

5.1.4 Repeat Ground-Track Orbits

Orbits that periodically repeat their ground trace can be useful for applications that require identical viewing conditions at different times in order to monitor changes. The ground trace, or the path that the spacecraft follows over the surface of the planet, can be designed to repeat every day or to alternate between two different paths before repeating. This type of orbit can be useful for tasks such as monitoring crops or studying atmospheric conditions.

The key properties of these orbits are *revolutions to repeat*, k_{r2r}, and *days to repeat*, k_{d2r}. The user inputs these two parameters, inclination, eccentricity, and the longitude of the first ascending node, and Algorithm 11 is used to find the repeat ground-track orbit solution. Alternative to days to repeat, the average altitude can be given as well.

Algorithm 11: Repeat Ground Track Finding Algorithm [5]

1 Initialize variables

$k_{revpday} = \frac{k_{r2r}}{k_{d2r}}$

$n = k_{revpday}\omega_E$

$a = \left(\mu\left(\frac{1}{n}^2\right)\right)^{1/3}$

2 Loop until a is fixed at a value.

$p = a(1-e^2)$

$$\dot{\Omega} = -\left[\frac{3nJ_2}{2}\left(\frac{R_E}{p}\right)^2\right]\cos i$$

$$\dot{\omega} = \left[\frac{3nJ_2}{2}\left(\frac{R_E}{p}\right)^2\right](4-5\sin^2 i)$$

$$\dot{M}_0 = \left[\frac{3nJ_2}{2}\left(\frac{R_E}{p}\right)^2\right]\sqrt{1-e^2}(2-3\sin^2 i)$$

$n = k_{revpday}(\omega_E - \dot{\Omega}) - (\dot{M}_0 + \dot{\omega})$

$a = \left(\mu\left(\frac{1}{n}^2\right)\right)^{1/3}$

$$e = \frac{a - r_p}{a}$$

5.1.5 Orbital Maneuvers

Most of the satellites have to perform maneuvers in order to achieve their desired orbit after launch, to maintain their orbits, or to transfer to a completely different orbit. Even CubeSats can maneuver due to the developments in propulsion systems. The software at the moment only has the capability to perform impulse maneuvers, but the framework allows implementation of more advanced orbital maneuvers in the future, such as designing interplanetary trajectories, orbital rendezvous, and phasing.

An impulse maneuver can alter the shape of an orbit as well as the orbital plane. Co-planar maneuvers involve changes in both speed ($v_f - v_i$) and flight-path angle ($\Delta\beta = \beta_2 - \beta_1$) to create a new, co-planar trajectory. Plane change maneuvers, on the other hand, alter the orbital plane and do not involve changes in either speed or flight-path angle at the impulse point [15].

An impulsive co-planar maneuver is shown in Figure 5.2. It is apparent that the velocity change for such a maneuver can be found from

$$\Delta V = \sqrt{v_i^2 + v_f^2 - 2v_i v_f \cos\Delta\beta} \tag{5.4}$$

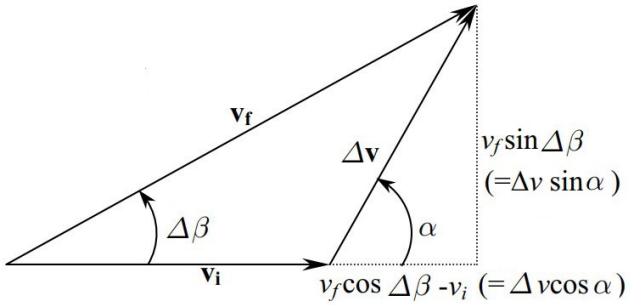


Figure 5.2 : Impulsive Coplanar Maneuver

For a plane change without a change in velocity, we can use

$$\Delta V = 2v_i \sin\frac{\delta}{2} \tag{5.5}$$

Where δ is the angle of plane change. This can be used for both inclination and RAAN changes, as shown in Figure 5.3.

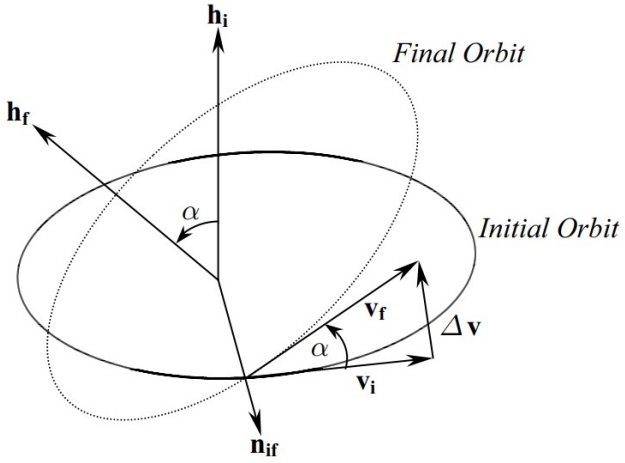


Figure 5.3 : Plane Change Maneuver [15].

If both inclination i and Ω are going to be changed simultaneously, then the resulting plane change angle can be found from

$$\cos\delta = \cos i_i \cos i_f + \sin i_i \sin i_f \cos(\Omega_f - \Omega_i) \tag{5.6}$$

If a combined maneuver (both velocity and plane change) is desired, then the general equation is given by

$$\Delta V = \sqrt{v_i^2 + v_f^2 - 2v_i v_f [\cos\Delta\beta - \cos\beta_f \cos\beta_i (1 - \cos\delta)]} \tag{5.7}$$

5.1.6 Lambert's Problem

Instead of determining an orbit by using a state vector at a specific time, we can find orbital elements by using position vectors at two different times. It is a boundary-value problem that requires a solution that meets conditions at two different points, and usually does not have a closed-form solution, requiring an iterative numerical solution process instead [15].

The known quantities are the initial and final position vectors, $r1$ and $r2$, and the time between them, $\Delta t = (t2 - t1)$. The quantities to be found are the orbital elements of the trajectory passing through points 1 and 2 with a given Δt. Note that the direction should be given as well.

Be aware that Lambert has a limitation. When the two position vectors are near-parallel, the transfer plane becomes not "well defined", and the resulting ΔV cost may be higher than normal.

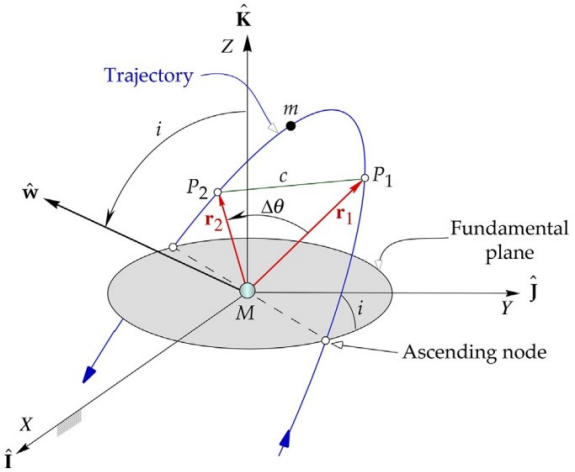


Figure 5.4 : Lambert's Problem [12].

An analytic approach can be used to solve the Lambert's problem. We will use the following function.

$$\sqrt{\mu}\Delta t = \left[\frac{y(z)}{C(z)}\right]^{3/2} S(z) + A\sqrt{y(z)} \tag{5.8}$$

Where $C(z)$ and $S(z)$ are Stumpff functions given in Equations 3.15 and 3.16, and

$$y(z) = |r_1| + |r_2| + A\frac{zS(z)-1}{\sqrt{C(z)}} \tag{5.9}$$

We will also make use of Lagrange coefficients given by

$$f = 1 - \frac{y(z)}{|r_1|} \tag{5.10}$$

$$g = A\sqrt{\frac{y(z)}{\mu}} \tag{5.11}$$

$$\dot{g} = 1 - \frac{y(z)}{|r_2|} \tag{5.12}$$

The complete algorithm to solve Lambert's problem is detailed in Algorithm 12.

Algorithm 12: Solution of Lambert's Problem

1 Find the change in true anomaly, $\Delta\theta$.

$$\Delta\theta = \begin{cases} \cos^{-1}\dfrac{r_1 \cdot r_2}{|r_1||r_2|} & \text{if prograde and } (r_1 \times r_2)_z \geq 0 \text{ OR retrograde and } (r_1 \times r_2)_z < 0 \\ 2\pi - \cos^{-1}\dfrac{r_1 \cdot r_2}{|r_1||r_2|} & \text{if prograde and } (r_1 \times r_2)_z < 0 \text{ OR retrograde and } (r_1 \times r_2)_z \geq 0 \end{cases}$$

2 Compute the auxiliary parameter A

$$A = \sin\Delta\theta \sqrt{\frac{|r_1||r_2|}{1-\cos\Delta\theta}}$$

3 Solve Equation 5.8 for z iteratively using Newton's method.

$$z_{i+1} = z_i - \frac{F(Z_i)}{F'(z_i)}$$

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

$$F'(z) = \begin{cases} \left[\dfrac{y(z)}{C(z)}\right]^{3/2} \left(\dfrac{1}{2z}\left[C(z) - \dfrac{3S(z)}{2C(z)}\right] + \dfrac{3S(z)^2}{4C(z)}\right) + \dfrac{A}{8}\left[3\dfrac{S(z)}{C(z)}\sqrt{y(z)} + A\sqrt{\dfrac{C(z)}{y(z)}}\right] & (z \neq 0) \\ \dfrac{\sqrt{2}}{40}y(0)^{3/2} + \dfrac{A}{8}\left[\sqrt{y(0)} + A\sqrt{\dfrac{1}{2y(0)}}\right] & (z = 0) \end{cases}$$

4 Find y after obtaining value of z using Equation 5.9.

5 Compute Lagrange functions using Equations 5.10 to 5.12

6 Velocities can be obtained by

$$v_1 = \frac{1}{g}(\mathbf{r_2} - f\mathbf{r_1})$$

$$v_2 = \frac{1}{g}(\dot{g}\mathbf{r_2} - \mathbf{r_1})$$

7 Obtain orbital elements using state vectors (r_1, v_1) and (r_2, v_2) by Algorithm 1.

5.2 Communication Analysis

The software has the capability to determine the time intervals where a satellite "sees" a ground station or another satellite. Access between objects can be restricted by the user in order to specify what is considered a valid access.

A ground station can be defined by supplying the following parameters;

- **Coordinates.** It can either be Cartesian (body-fixed, ITRF for Earth), Spherical, or Geodetic. The following transformations are performed according to the input type.

Geodetic:

$$C = \frac{R_E}{\sqrt{1 - e^2 \sin \psi^2}} \tag{5.13}$$

$$S = C(1 - e^2) \tag{5.14}$$

$$r_{ITRF} = \begin{pmatrix} (C+h)\cos\phi\cos\lambda \\ (C+h)\cos\phi\sin\lambda \\ (S+h)\sin\phi \end{pmatrix} \tag{5.15}$$

Spherical:

$$r_{ITRF} = \begin{pmatrix} R\cos\phi\cos\lambda \\ R\cos\phi\sin\lambda \\ R\sin\phi \end{pmatrix} \tag{5.16}$$

Where R is the radius, λ is longitude, and ϕ is latitude.

- **Altitude.** The height above sea level. This data is automatically derived from the Earth terrain data, but can be overridden.
- **Constraints.** Either one or a combination of the following; minimum/maximum elevation angle, minimum/maximum azimuth angle, minimum/maximum range.

5.2.1 Pass Predictions

In order to calculate the pass start and end times, the satellite is first propagated using the chosen orbital propagator. At each time step, the immediate ephemeris of the satellite is used to check whether there is a Line of Sight (LOS) with the chosen ground station, and if the constraints are satisfied. Algorithm 13 defines the steps to calculate the angles in topocentric horizon system.

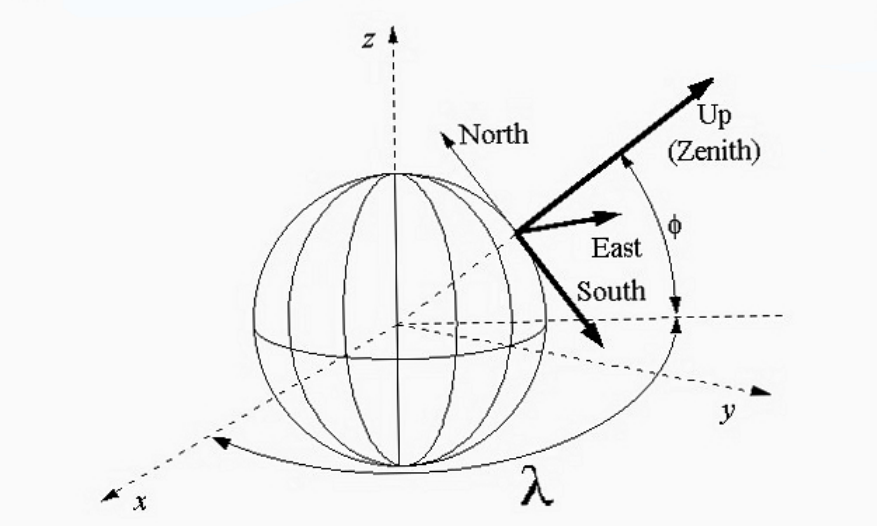


Figure 5.5 : Topocentric Horizon Coordinate System [33].

Algorithm 13: Pass Determination [5]

1 Compute geocentric equatorial position vector of the ground station, r_{GS} using equation 5.15

2 Calculate the relative position vector of the satellite
$\rho = r_S - r_{GS}$

3 Perform the transformation $R_y(90-\phi)R_z(\lambda)$ to topocentric horizon system.

4 The elevation angle, α can be found from
$$\alpha = \sin^{-1}\frac{\rho_z}{|\rho|}$$

5 The azimuth angle can be found using the following equations and adjusting the quadrant
$$\sin A = \frac{\rho_x}{\sqrt{\rho_x^2+\rho_y^2}}$$
$$\cos A = \frac{-\rho_y}{\sqrt{\rho_x^2+\rho_y^2}}$$

Alternatively, the atan2 function can simply be used to obtain azimuth angle.

$$A = atan2(-\rho_y, \rho_x) + \pi$$

After range, and elevation and azimuth angles are found for a step, the software simply compares if the elevation is positive, and that all the values are within limits.

In order to save computational effort and time, not every time step is checked. If both the elevation angle and its slope is negative, meaning that the satellite is getting away, time is forwarded 75% of orbital period. If the slope is positive, it is forwarded a couple of minutes or seconds depending on the current elevation angle. In terms of satellite to

satellite visibility, a different approach is used. Checking visibilities at every time step during numerical propagation can become a highly costly operation as the number of satellites increase, and the accuracy will be bound by the step size δt in this approach. Hence, an efficient and precise method that works in all kinds of orbits were adapted from Alfano, Negron, and Moore [50].

Consider the geometry of the visibility function in Figure 5.6. Satellites are either rising or setting in relation to each other, as long as the position vectors reside in the Earth-tangent plane. If a vector R, which originates from the center of the Earth and is perpendicular to the line connecting the two satellites, was shorter than or equal to the Earth's radius, it would be clear that the satellites would not be able to communicate directly with each other. However, due to atmospheric interference, it would be more realistic to assume that the magnitude of S is slightly larger than the Earth's radius [51]. Let us introduce a "bias factor", b, that counts for the thickness of the atmosphere. By default, 100m was used.

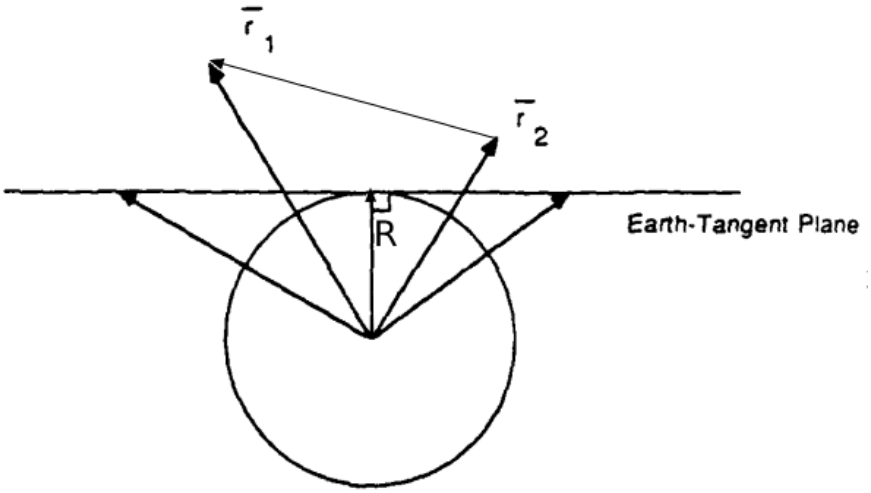


Figure 5.6 : Satellite to Satellite Visibility Geometry

The angle between satellite positions can be compared with the angles formed from the LOS tangent to the spherical surface ($R_E + b$) to evaluate visibility. Hence, the visibility function between two satellites whose positions are given as r_1 and r_2 in the Earth-centered inertial system can then be defined as;

$$\phi(t) = \cos^{-1}\left(\frac{R_E + b}{|r_1|}\right) + \cos^{-1}\left(\frac{R_E + b}{|r_2|}\right) - \cos^{-1}\frac{r_1 \cdot r_2}{|r_1||r_2|} \tag{5.17}$$

The results can be interpreted as:

$$\phi(t) \begin{cases} > 0 & \text{visible} \\ 0 & \text{rise or set} \\ < 0 & \text{no visibility} \end{cases}$$

In order to correct for the oblateness of the Earth, the $\hat{k}$ components of the position vectors r_1 and r_2 are rescaled by $1/\sqrt{1-e_E^2}$.

The parabolic blending method is then used to find the rise and set times as described in the work of Alfano, Negron, and Moore [50].

5.2.2 Field of View

Satellites used for communications, weather, navigation, or surveillance require Earth surface coverage. Hence, the software has the capability to calculate the coverage, or field of view, of satellites.

A satellite's coverage of a particular area at a given moment is usually determined by a cone-shaped field of view that intersects the Earth's surface and creates a circular area of coverage centered on the point directly below the satellite. The satellite's field of view is limited by the horizon, with higher minimum ground elevation angles being used to account for atmospheric conditions or obstructions on the ground. In some cases, the satellite's sensor may also have its own limitations in terms of its angular field of view or the distance from the satellite to the area being observed, which can further restrict the satellite's visibility. These factors can be described using equations that relate the satellite's coverage parameters [16].

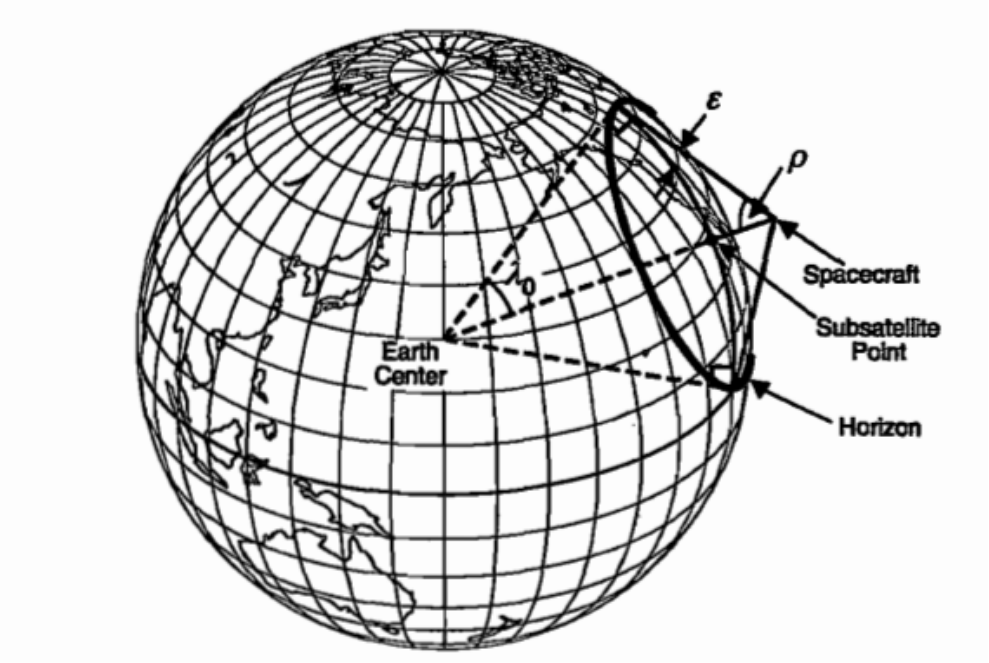


Figure 5.7 : Satellite Coverage Geometry [49].

The parameters indicated in Figure 5.7 are described below.

- ρ = Satellite field-of-view angle
- λ_0 = Earth central angle of coverage.
- ε = Ground elevation angle

$$\sin\rho = \cos\lambda_0 = \frac{R_E}{R_E + H} \tag{5.18}$$

$$\tan\varepsilon = \frac{\cos\lambda_0 - [R_E/(R_E + H)]}{\sin\lambda_0} \tag{5.19}$$

In units of distance, assuming r is range in kilometers and λ_0 in radians, we can use the approximation

$$r = \lambda_0 R_E \tag{5.20}$$

For increased precision, R_E at the precise latitude can be used.

The distance to the horizon, or maximum slant range, is given by

$$D_{max}^2 = R_E^2 + (R_E + H)^2 - 2R_E(R_E + H)\cos\lambda_0 \tag{5.21}$$

A correction factor should be used to take the oblateness of Earth into account to increase precision, which has been explained in works by Liu [52] and Collins [53], [49].

5.2.3 Link Analysis

One of the key points of spacecraft mission design is the calculation of the link budget for communication analysis. The software provides the tools to perform preliminary downlink/uplink link budget calculations. There are several inputs to be considered.

- **Orbit Parameters**

  The slant range is calculated as shown in Figure 5.8, which is used for path and atmospheric loss calculations. The mean altitude is calculated from orbital elements. Frequency is given as an input by the operator.

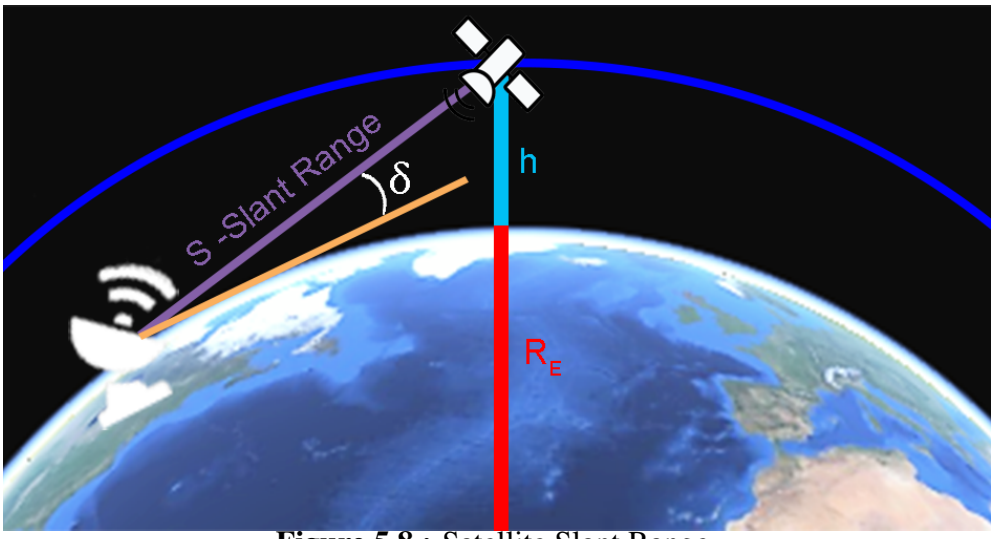


Figure 5.8 : Satellite Slant Range.

The slant range is given by

$$S = R_E \left[\sqrt{r^2/Re^2 - \cos^2 \delta} - \sin \delta \right] \tag{5.22}$$

Where r is the mean orbit radius, and δ is minimum elevation angle.

Path loss is given by:

$$L_s = 22.0 + 20 \log (S/\lambda) \tag{5.23}$$

Where λ is the wavelength found by c/f (c is the speed of light, f is frequency).

- **Transmitters**

  Input values are the transmitter power, P_t, and estimated transmitter line loss, L_l. These losses are usually very small, and include the cable loss based on cable length, connector losses, antenna mismatch losses, and filter insertion losses.

  Total power delivered to transmitter is computed using

$$P = P_t - L_l \tag{5.24}$$

- **Receivers**

  User inputs the estimated receiver line losses, L_{lr}, which usually involves multiplying the cable lengths by the cable loss/meter value provided by the manufacturer.

  Transmission line coefficient is found using

$$\alpha = 10^{(L_{lr}/10)} \tag{5.25}$$

  The system noise temperature, T_s, is calculated as follows.

$$T_s = T_a\alpha + T_0(1-\alpha) + T_{LNA} + (T_{cr}/(G_{LNA}/(10^{(L_c/10)}))) \tag{5.26}$$

  Where

  - T_a = Sky temperature
  - T_0 = Ground station feedline temperature OR spacecraft temperature
  - T_{LNA} = LNA Temperature
  - T_{cr} = Communications Receiver Front End Temperature
  - G_{LNA} = LNA Gain
  - L_c = Cable loss from LNA to communications receiver

  In a ground station, the temperature of the sky as measured by the antenna includes both noise from the colder sky and noise that is generated on the ground near the station, which is typically caused by computers in the area that are within radio range of the ground station [54].

The sky temperature is calculated as follows

$$T_{skyMin} = T_{galacticMin} + T_{noiseSource} \tag{5.27}$$

$$T_{skyMax} = T_{galacticMax} + T_{noiseSource} \tag{5.28}$$

The coldest and warmest galactic noise temperatures

$$T_{galacticMin} = 80((f/1000)/0.25)^{-2.75} + 2.7 \tag{5.29}$$

$$T_{galacticMax} = 380((f/1000)/0.25)^{-2.75} + 2.7 \tag{5.30}$$

Where f is the receiver frequency in MHz.

Noise source effective temperature is found from

$$T_{noiseSource} = 10^{((No+198.6-10\log(B*1000))/10)} \tag{5.31}$$

Where No is the estimated or measured noise level, and B is receiver bandwidth in KHz, which is given by

$$B = \frac{R_b}{SE * FEC}(1+r) \tag{5.32}$$

Where

- R_b = Bit rate
- SE = Spectral efficiency
- FEC = Forward Error Correction factor
- r = Filter roll-off factor

The values from ITU recommendations can be used to obtain noise levels [55].

- **Antenna Gain**

  Gain of the antenna and beamwidth are the main parameters used here. The user has to input the antenna type to use as the equations change accordingly. The available antenna types and their equations are given in Table 5.1. Empty values mean information from the manufacturer should be used.

Table 5.1 : Antenna Options.

Type	Inputs	Gain	Beamwidth
Yagi	Boom length (λ_b)	Table at [56], depending on λ_b	$\sqrt{\frac{40000}{10^{G/10}}}$
Helix	Circumference (c) Turn spacing (s) Turns (n)	$10\log(15c^2sn)$	$\frac{1}{c\sqrt{sn}}$
Parabolic Reflector	Diameter (d) Aperture Efficiency (AE)	$20.4+$ $20\log d+$ $20\log(AE/1000)+$ $10\log f$	$\frac{21}{(d/1000)f}$
Monopole	Gain, Beamwidth	-	-
Dipole	Gain, Beamwidth	-	-
Canned turnstile	Gain, Beamwidth	-	-
Quadrifilar Helix	Gain, Beamwidth	-	-
Patch	Gain, Beamwidth	Manufacturer	-

- **Antenna Pointing and Polarization Losses**

  This section addresses the loss values caused by pointing of antenna and spacecraft errors. The values we are considering may not be precise because they can be influenced by the performance of the spacecraft's attitude determination and control systems and the ground station's ability to accurately track the satellite. We are examining how the antenna's directivity or gain changes as we move away from the direction in which it has the highest gain. It is common to refer to the decrease in gain as the antenna is rotated and viewed from a distance as the antenna's "gain roll-off."

  Equation for pointing loss is

$$\xi = 2\theta_e \frac{79.76}{\delta} \tag{5.33}$$

$$L_p = -10\log\left(3282.81\frac{\sin\xi^2}{\xi^2}\right) \tag{5.34}$$

  Where θ_e is the estimated pointing error and δ is the beamwidth.

And for polarization loss

$$L_{pol} = 0.5\left(1 + \frac{(1-r_1^2)(1-r_2^2)\cos(2\theta_p) + 4r_1r_2}{(1+r_1^2)(1+r_2^2)}\right) \tag{5.35}$$

Where r_1 and r_2 are the axial ratios of TX and RX antennas, respectively, and θ_p is the polarization angle between antennas.

The axial ratio is a measure of the directional characteristics of an antenna, and is defined as the ratio of emitted (or received) power when the antenna is aligned with major axis to when it is aligned with the minor axis. It is a measure of the directional characteristics of an antenna.

- **Atmospheric and Ionospheric Losses**

  Loss due to atmosphere depends mostly on elevation angle. The look-up table from [57] is used to obtain losses due atmospheric absorption, which goes from 10.2 dB at 0 degrees to 0 dB at 90 degrees.

  In case of ionosphere, while the elevation angle of a satellite can have an impact on the signal absorption, this relationship is almost insignificant. This value can either be interpolated from the look-up table (usually lower than 1 dB), or manually given by the operator.

  The rain attenuation can be obtained from the tabulated values as shown in Figure 5.9.

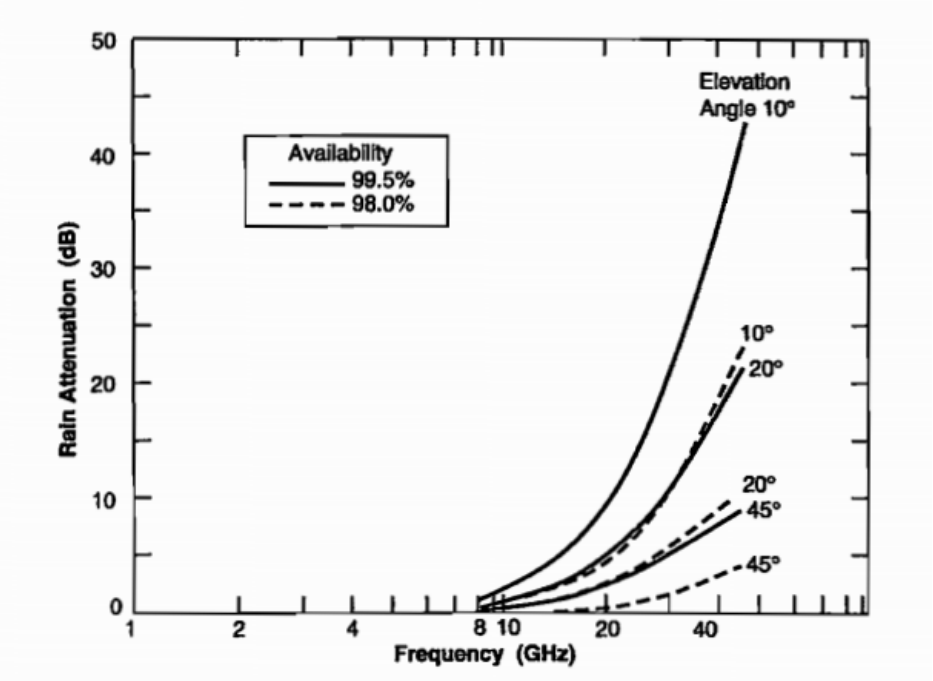


Figure 5.9 : Rain Attenuation Values [49].

- **Modulation and Encoding**

  Operator can select from the available options or manually enter the required E_b/N_o by looking at Figure 5.10 to attain desired BER for chosen modulation and coding scheme. Finally, the implementation losses are given (default 1 dB).

Table 5.2 : Modulation Options.

Modulation	Coding	BER	Required Eb/No (dB)
AFSK/FM	None	$1.0E-4$	21.0
G3RUH FSK	None	$1.0E-4$	16.7
Non-Coherent FSK	None	$1.0E-4$	13.4
Coherent FSK	None	$1.0E-4$	10.5
GMSK	None	$1.0E-5$	9.6
QPSK	None	$1.0E-6$	10.5
BPSK	None	$1.0E-6$	10.5
BPSK	Convolutional $R=1/2$, $K=7$	$1.0E-6$	4.8
BPSK	Conv. $R=1/2$, $K=7$ & R.S. (255,223)	$1.0E-6$	2.5
BPSK	Turbo Code (Parallel w. Interleaver)	$1.0E-6$	0.75

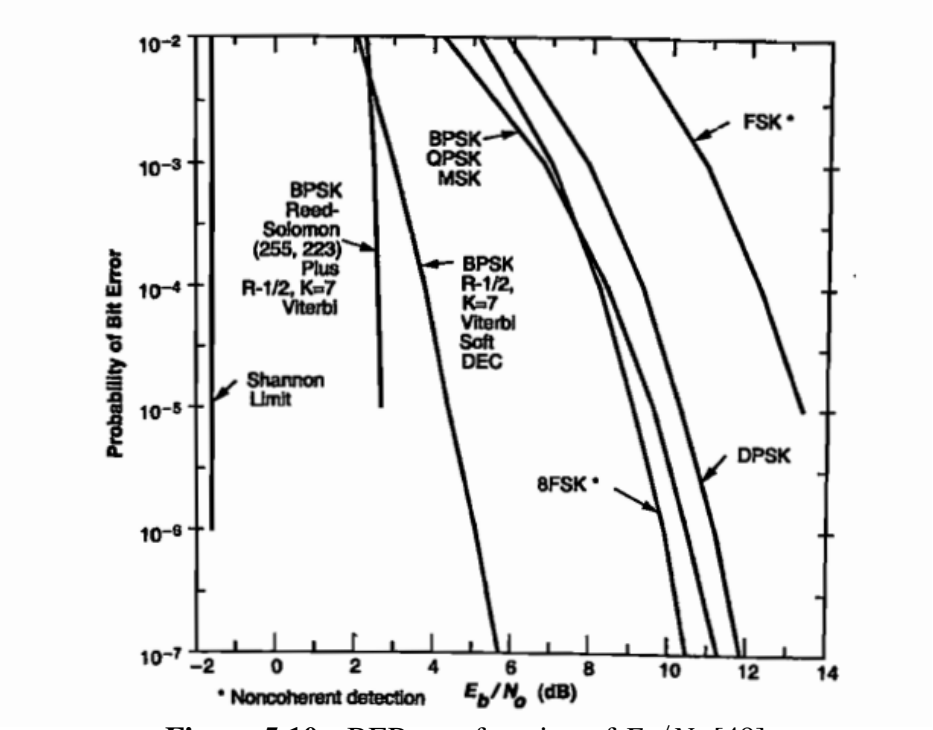


Figure 5.10 : BER as a function of E_b/N_o [49]

Finally, the link budget is calculated as follows;

$$RIP = P + G_t - L_{scp} - L_{pol} - L_s - L_{atm} - L_{ion} - L_{rain} \tag{5.36}$$

$$G/T = G_r - L_{lr} - 10\log T_s \tag{5.37}$$

$$S/N_o = RIP - L_p - 228.60 + G/T \tag{5.38}$$

$$E_b/N_o = S/N_o - 10\log R \tag{5.39}$$

Where

- RIP = Received Isotropic Power (dbW).
- P = Transmitter power delivered to antenna (Eq 5.24).
- G_t = Transmitter antenna gain (Table 5.1).
- L_{scp} = Spacecraft antenna pointing loss (Eq 5.34).
- L_{pol} = Polarization loss (Eq 5.35).
- L_s = Path (Space) loss (Eq 5.23).

- L_{atm} = Atmospheric loss.
- L_{ion} = Ionospheric loss.
- L_{rain} = Rain attenuation.
- G/T = Figure of merit (dB/K).
- G_r = Received antenna gain (Table 5.1).
- L_{lr} = Receiver line losses (input).
- T_s = System noise temperature (Eq 5.26).
- S/No = Signal-to-Noise Power Density (dBHz).
- L_p = Ground antenna pointing loss (Eq 5.34).
- R = Data rate in bps (input).

5.3 Power Analysis

Several factors affect the power generation capabilities of the solar array of spacecraft. These include the orbit and attitude, position of the sun, solar and lunar eclipses, and layout of the panels [58].

The following equation is used to calculate power generation of solar arrays

$$P = S \times \eta \times I_d \times L_d \times A_E \tag{5.40}$$

Where

- P = Power generated.
- S = Solar constant flux density ($1367 W/m^2$ [49]).
- η = Solar cell efficiency.
- I_d = Inherent degradation.
- L_d = Lifetime degradation.

- A_E = Effective panel area.

A rotation matrix that uses the attitude information is used to rotate the spacecraft model. The solar light that impinges on each surface element is then calculated based on the satellite's position and orientation. The effective area (i.e., area that sees the Sun) of each illuminated panel is then found, and equation 5.40 is used at every step of the propagation to determine the power produced by solar panels. Summing the individual panel powers yield the total generated power, which can be compared with the orbit-averaged total power to see how the total power generation capabilities change over long durations [59].

The software currently supports only one attitude model which allows only a fixed orientation. In the future, the software will support customizable attitude models that can be used to more accurately compute power generation.

5.3.1 Eclipse Periods

In mission planning, it is important to determine the intervals where the satellite does not receive any sunlight, for both force model computations regarding SRP and for determining the times when there won't be any power generation. Additionally, the attitude model of the spacecraft can be changed during eclipse times to save power. Hence, the capability to determine the times of eclipses is included in the software.

The vector R_S and r are the positions of the Sun and the satellite with respect to the center of the Earth, respectively. The satellite position vector can be decomposed in components parallel (d) and perpendicular (a) to the direction towards Sun. The angle ψ is the angle between the directions towards the Sun and the satellite.

$$\psi = \cos^{-1} \frac{R_S \cdot r}{|R||r|} \tag{5.41}$$

$$a = r \sin \psi \tag{5.42}$$

Then both conditions below have to be satisfied for an eclipse to occur

- Satellite is on the night-side of Earth ($\psi > 90$)
- Satellite hides behind Earth ($a < R_E$)

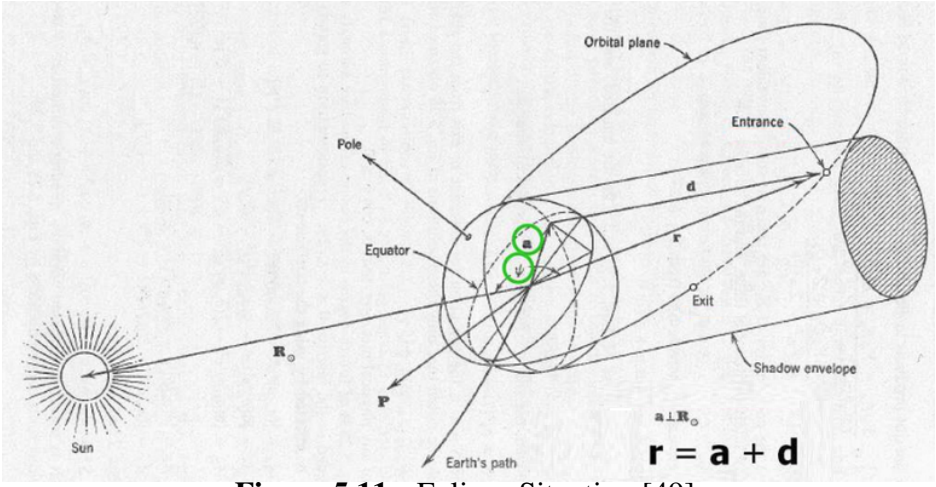


Figure 5.11 : Eclipse Situation [49].

The shadow function is defined as

$$S(\theta) = R_E^2(1 + e\cos\theta)^2 + p^2(\cos\psi)^2 - p^2 \tag{5.43}$$

Where θ is the true anomaly of orbit and p is the semi-latus rectum ($p = a(1-e^2)$). Table 5.3 describes how to evaluate the result of the shadow function.

Table 5.3 : Shadow Function Results.

	$S < 0$	$S = 0$	$S > 0$
$\psi < 90^\circ$	In sunlight	In sunlight	In sunlight
$\psi > 90^\circ$	In sunlight	Entering/Leaving shadow cone	In shadow cone

6. SOFTWARE ARCHITECTURE AND DEVELOPMENT

The mathematical specifications given so far were implemented in the Orbiter software [60]. The software is written in a combination of multiple programming languages and frameworks with an aim to achieve cross platform as well as web support. The software can be installed as a stand-alone program on any kind of operating system, or be used online through a website granted it is being served on a server. In order to achieve this, the Electron framework using JavaScript was used [61]. The graphical user interface (GUI) was created using web design tools such as HTML, CSS, SCSS, and JavaScript. The Cesium open-source module was used in the modelling of the Earth and visualization of the assets [62]. In the back-end, all calculations regarding orbital dynamics were coded in C++ due to several considerations.

- Speed: C++ is a compiled language, and a fast one at that, which means it can execute faster than interpreted languages like Python. This can be particularly important for computationally intensive tasks like.
- Control: C++ gives a lot of control over how the code runs, including how memory is allocated and used. This is helpful for optimizing the performance of the code.
- Libraries: There are many libraries available for C++ that can be helpful for scientific computing tasks. For example, the Eigen library includes a number of useful math functions and algorithms that comes in handy for orbital dynamics calculations.
- Community: There is a large community of C++ developers, so it is easy to find help and resources.

6.1 Principles

Several fundamental concepts were evaluated before starting the development, and most important themes to follow were decided as follows.

- Keep it simple, stupid (KISS): Try to keep the design as simple as possible, while still meeting the requirements of the software. McConnell states that managing complexity is the most critical technical matter in software development [63], as complex designs can be more difficult to understand and maintain.

- Modularity (Divide and Conquer): Divide the software into distinct components or modules, each with a specific responsibility. This can make it easier to understand and modify the software. Each type of resource has a specific set of interfaces that are implemented through base classes in C++. New components within each category are created by implementing classes that are derived from these base classes and defining the necessary methods to provide the desired functionality.

- Open/closed principle: The software should be designed to allow for extension without requiring modification. This means that new features can be added without altering existing code.

- Don't repeat yourself (DRY): Avoid duplicating code or logic in the software. Instead, try to find a way to reuse existing code or abstract common concepts into a shared component.

In the subject of writing code for astrodynamics, the following recommendations by Vallado were practiced as best as possible in order to avoid errors and achieve maintainability [5].

- Number, order, type: In a function, the number and order of the formal parameters (arguments passed between the function and its caller) must be consistent and must match the expected data types (e.g., Vector3, Vector3, Number) every time the function is called.

- Documentation: While documentation can take time and effort, it is often neglected when budgets or time are limited. However, failing to document the work can make it difficult for the programmer or others to reuse or modify the in the future.

- Variable names: Traditionally, variable names were kept to 6-8 characters in length, but it is usually counterproductive to use excessively long names either. Instead, use

common sense and aim for variable names that are similar to mathematical notation (e.g., CD for drag coefficient, BC for ballistic coefficient). Using descriptive and concise names can help make your code more readable and easier to understand. In short, the name of a variable, class, or function, should explain why it exists, what it does, and how it's used [64].

- Modularity: It is important to maintain the integrity of algorithms, even if doing so results in larger code size. When this is necessary, it is important to use documentation effectively to help make the code easier to understand and maintain.
- Testing: Code that has not been tested is inherently unreliable, so it is important to test and validate any code that you write. While there may be constraints on testing, it is important to at least try to do the following:
  - Test each occurrence of a dynamic event. For example, use different types of orbits (circular, elliptical, parabolic, and hyperbolic) to test a code that propagates orbits.
  - Test each computational method. Exercise each loop in the code, and fully execute the code itself.
  - Develop new test cases to cover emerging situations.

6.2 Architecture

Several different modules were created within both the front-end and back-end of the software, and while the modules are mostly available for independent use, some interactions between different modules were inevitable. For example, both the Coordinate System and Orbit Propagator modules use the same global values such as planetary information, so they require the Celestial Objects module, albeit at an abstract level.

The software architecture consists of several components that are organized into functional modules, or packages, and work together to simulate spacecraft missions. There are three main packages in the system: User Interface, the Orbit Engine, and Utilities. Each of these packages is further divided into smaller units, known as

subpackages, which provide more detailed views of the functions within each package. These different packages and subpackages interact with each other to model spacecraft in orbit and simulate the spacecraft's environment. Figure 6.1 provides an overview of this package grouping and subpackaging.

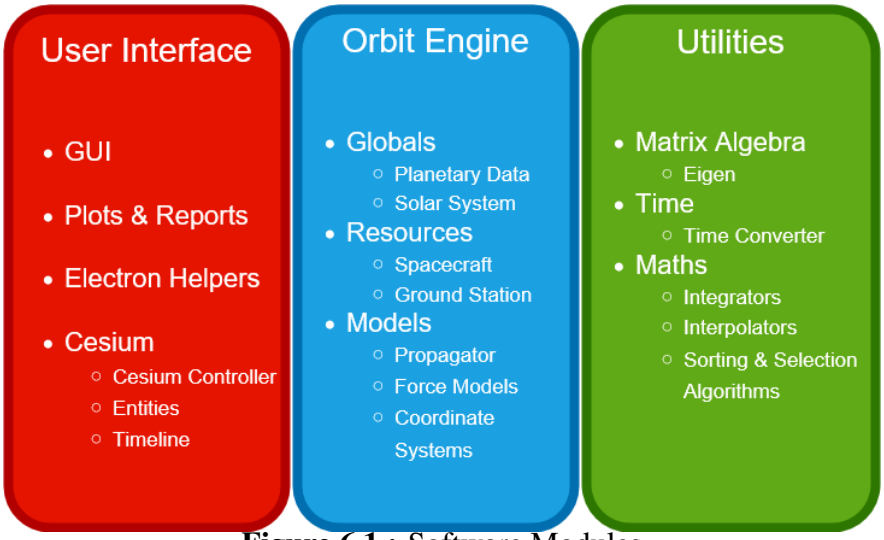


Figure 6.1 : Software Modules

6.2.1 User Interface

All 2-way communications between users and the software are contained in this package.

- **GUI**: Users interact with the software through a user interface. The GUI is coded like a website using HTML, CSS, and Javascript with the addition of Cesium.js module for the 3D visualization of space environment. The GUI provides an easy to use yet rich environment and is connected to the back-end in two ways. In the online version that is due to be hosted on a server, the back-end is accessed via REST API calls. In the installed version, the back-end part consisting of C++ modules are integrated into the software with a software library called "napi". The "node-gyp" package uses python to create a node modules from the native C++ code and the APIs can directly be called.

- **Plots & Reports**: This component handles preparing reports and graphics from the generated data, and formatting them according to the user's preferences. Different

units and timesteps can be chosen, and one report can include any number of outputs. The "plotly" Javascript library is used for plotting.

- **Electron Helpers**: This one exists for the installed version. Electron is a tool for creating desktop applications from web sites. This module handles Electron related properties such as window creations and attaching event handlers, such as context and application menus.

- **Cesium**: This package is for interfacing with the Cesium module. It is split into three parts, consisting of a controller, entities, and timeline. The Controller handles creating the Cesium viewer when a new 3D or 2D window is instantiated, and keeps track of the scene and entities.

  An Entity is a Cesium object that has a visual description in the scene. It can be a Satellite, a Ground Station, a Location, or a Sensor on a spacecraft. The Controller keeps track of these entities and updates their position and orientation at each timestep.

  Timeline tools is for managing the scenario time. User can adjust the UI timeline tool freely back and forth in the limits set during the initialization of the scenario, and this module keeps track of the current time, whether animation should happen, and in what speed.

6.2.2 Orbit Engine

This is the back-end part that handles all the astrodynamics computations. The components that are used to model the elements of the flight simulation are contained in this package. It can be broken up to 3 submodules.

- **Globals**: This subpackage includes all of the environmental data and tools needed to model the solar system. These include the Earth Orientation Parameters, space weather data, and planetary constants.

- **Resources**: These are the entities that also exist in the UI, such as spacecraft and ground station classes.

- **Models**: These are the physics computations that contain the force models, and orbit propagation.

6.2.3 Utilities

This package contains classes used to achieve functionality of other functions, and can be thought as the backbone of all computational modules.

- **Matrix Algebra**: Uses the Eigen C++ library to provide matrix calculus functionality.
- **Time**: This library performs conversions between various time systems, from UT1 to GPS Time as described in the Time Systems chapter. The IERS bulletin data is included here.
- **Maths**: All of the generic mathematical algorithms and solvers are programmed here. These include the numerical integration algorithms such as Runge-Kutta, interpolation algortihms (Lagranga etc), sorting and selection algorithms for big arrays, as well as other useful low level computations.

6.3 User Interface

The UI was designed in such a way that it is robust but easy to use for inexperienced users, has a clear and intuitive layout, with well-labeled buttons and easily discoverable features. The main window is organized in a way that makes it easy for users to find and access the various components, such as the asset library and object view.

There is a sidebar on the left for adding assets, two windows for the 3D and 2D views, and an object browser to navigate the available entities in the scenario, with options for selecting, renaming, and deleting them.. The asset library is organized into categories, such as satellite, ground station, and communication link.

Components can be selected by clicking or edited by double-clicking the tree view. A component can be zoomed in, in which case the camera automatically moves to an appropriate viewing distance and is centered on the satellite. This changes camera view from inertial to satellite-based, so the orbit itself appears to be moving.

The 3D and 2D windows are detachable, allowing users to move them around or even display them on separate monitors if desired.

Overall, the goal of the UI design was to provide users with all the tools and information they need to create their desired content, while also being intuitive and easy to navigate.

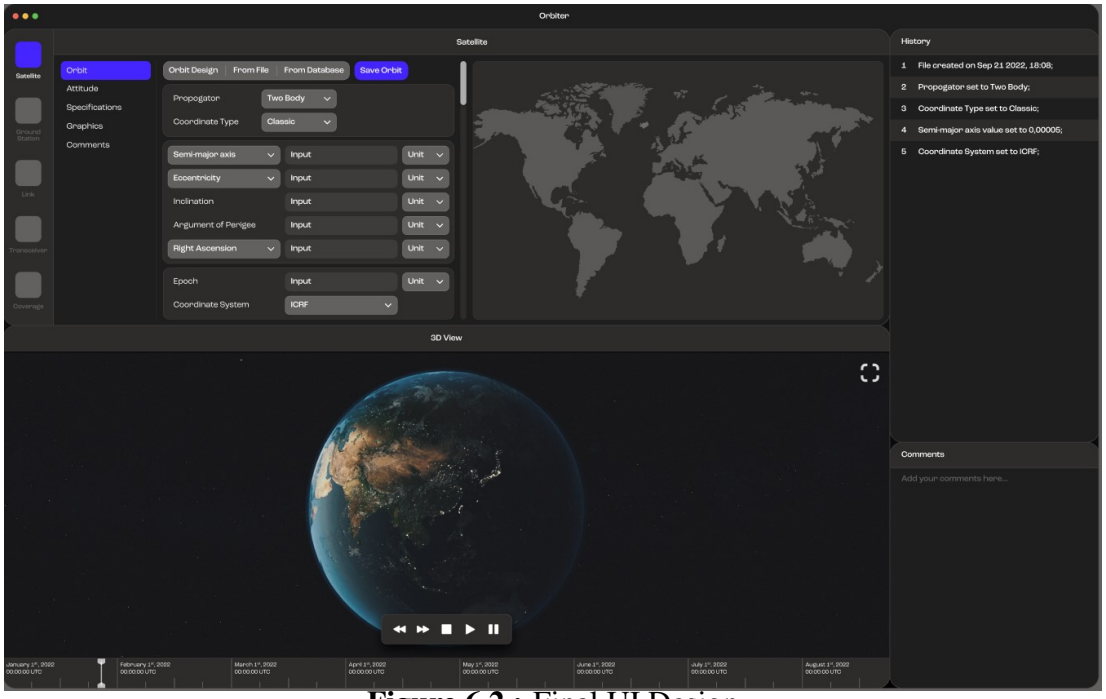


Figure 6.2 : Final UI Design

6.3.1 Ground Tracks

A ground track can be described as the projection of the orbit plane of a satellite onto the surface of the Earth. An important feature of satellite ground tracks is that the orbital inclination (i) can be deduced by one look, as the maximum and minimum geographical latitudes of the track are equal to i and $-i$, respectively. This osculation in latitudes cause the shape of the ground track in low earth orbits to resemble a sine curve on a Mercator projection.

The associated geographical latitude and longitude of the point where the line connecting the center of the planet to the spacecraft passes through the surface can be used to track the satellite. The latitude is equal to the declination, δ, and the longitude is defined as the angle between this point and the Greenwich meridian, counted positively from the east. The right ascension of the Greenwich meridian is known as the *Greenwich Hour Angle* and can be found from

$$\Theta(t) = 280.4606^\circ + 360.9856472^\circ \dot{d} \tag{6.1}$$

Where d is the time since the J2000 epoch, measured in days. Θ increases by 360° per revolution, which is equal to a sidereal day ($23^h56^m04^s$). Hence, *Greenwich Hour Angle* is equivalent to *sidereal time*.

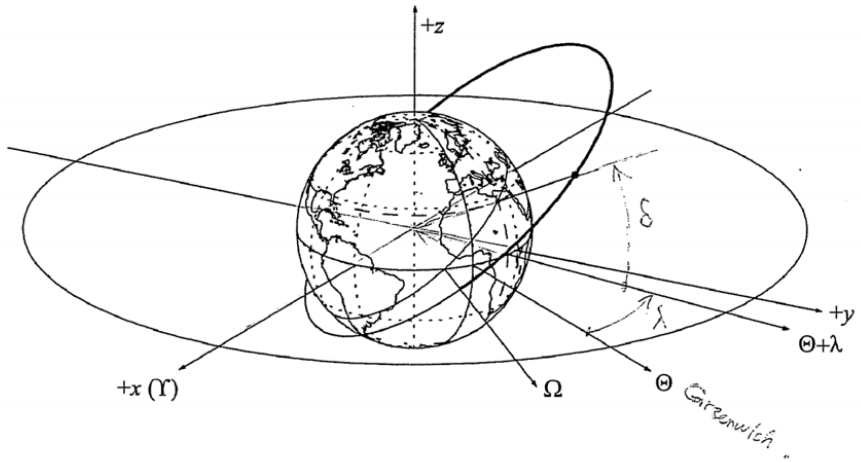


Figure 6.3 : Ground projection of an orbit [13].

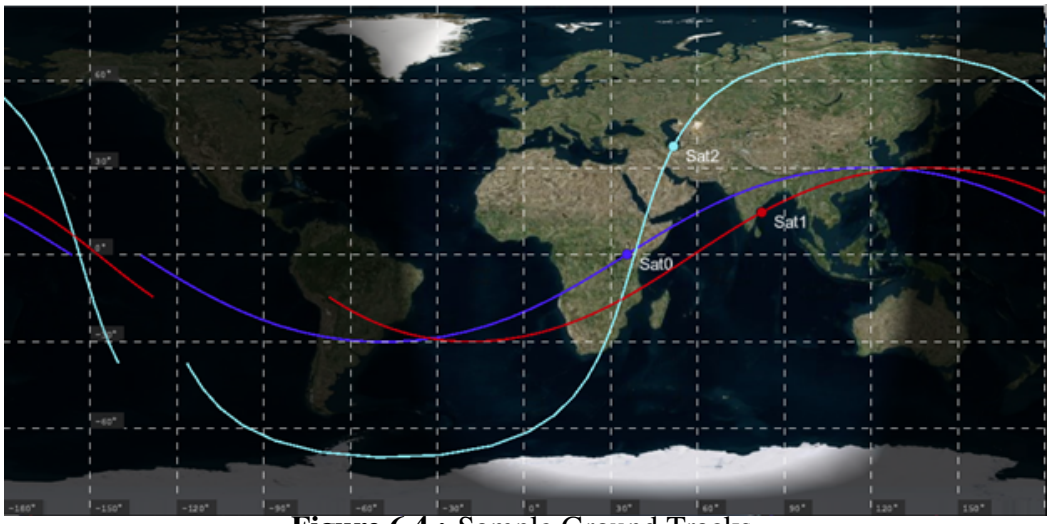


Figure 6.4 : Sample Ground Tracks.

Since the Earth rotates, the ground track shifts westwards in each revolution. Observing the ground tracks, one can also discern the orbital period in addition to the inclination. The distance between two successive equator crossings, in terms of latitude, gives the period when divided by the angular speed of the Earth. This can be observed in Figure 6.5, where the distance of 23.72° divided by $15.04°/h$ equals 1.577 hours.

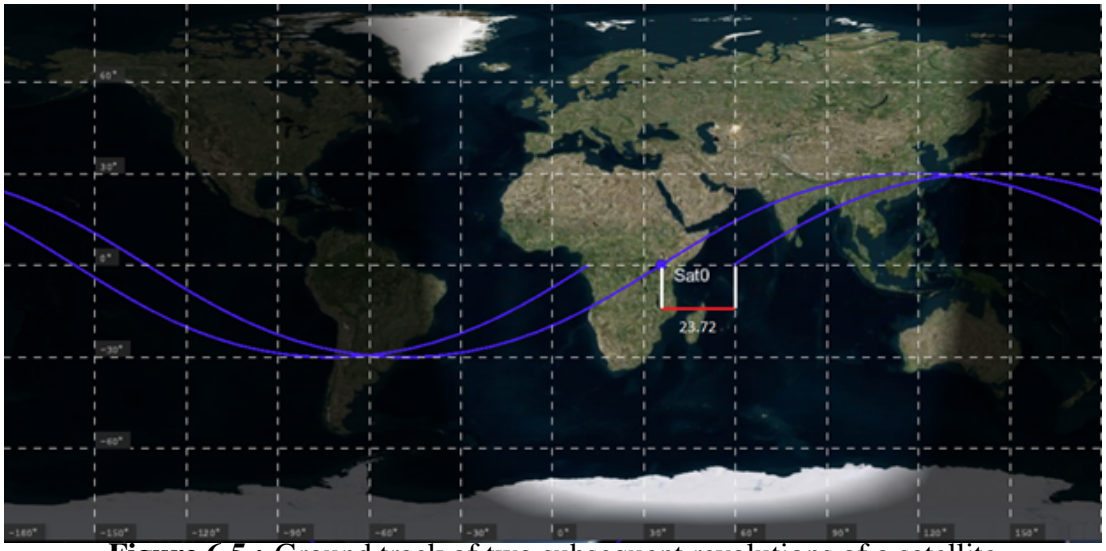


Figure 6.5 : Ground track of two subsequent revolutions of a satellite

Different kinds of orbits can result in various ground track shapes. Figure 6.6 show geostationary and Molniya orbits, respectively. Details about these kinds of orbits are given in Chapter 5.1.

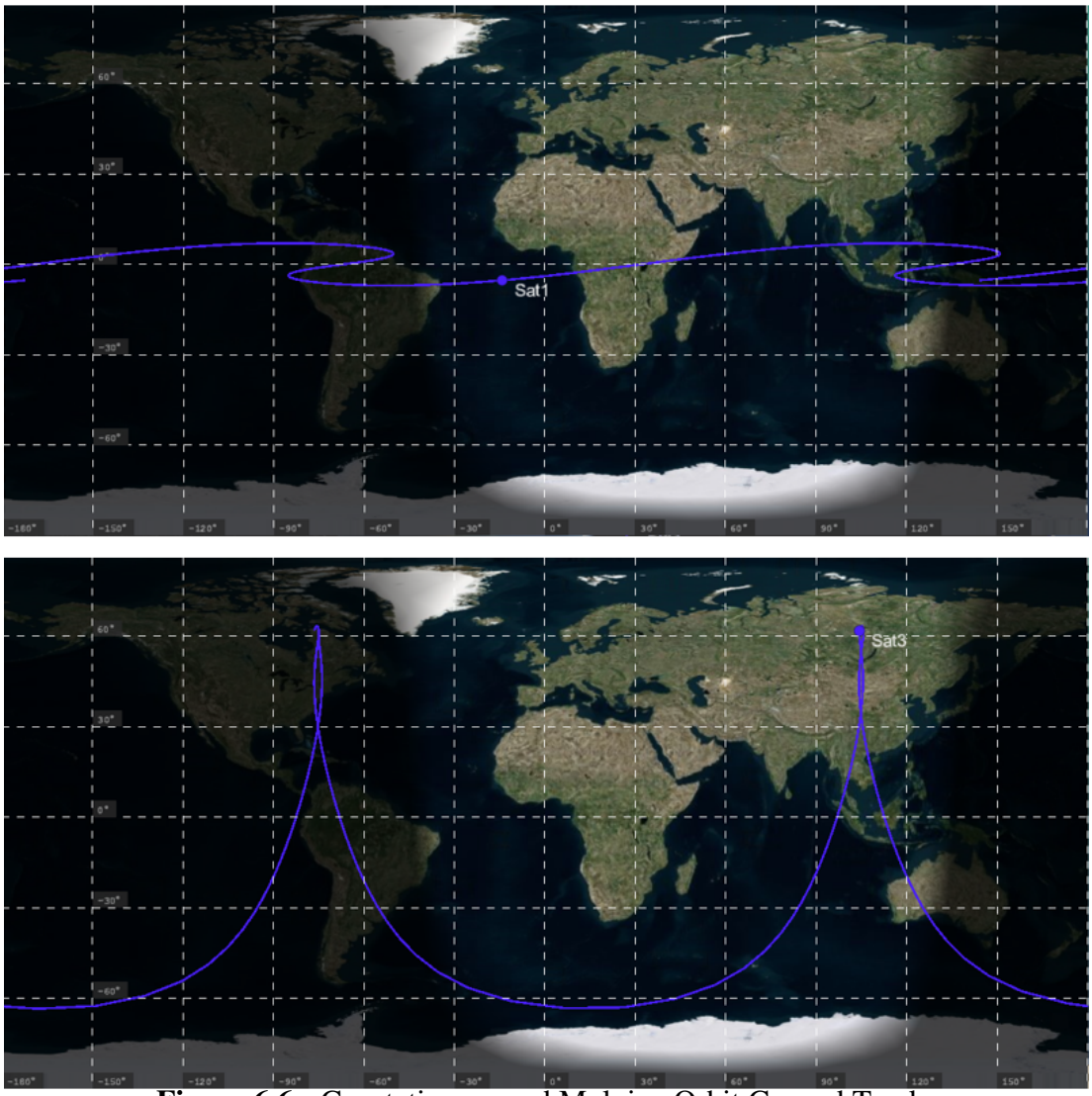


Figure 6.6 : Geostationary and Molniya Orbit Ground Tracks

Algorithm 14 describes the steps to plot the ground track of a satellite using its position vector $\vec{r} = x\hat{i} + y\hat{j} + z\hat{k}$.

Algorithm 14: Ground Track Algorithm [12]

1 Compute the rates of change of right ascension and argument of perigee due to oblateness. The second zonal harmonic (J_2) for Earth is $1.08263e^{-3}$.

$$\dot{\Omega} = \left[\frac{3}{2}\frac{\sqrt{\mu}J_2R_E^2}{(1-e^2)^2a^{7/2}}\right]\cos i$$

$$\dot{\omega} = \left[\frac{3}{2}\frac{\sqrt{\mu}J_2R_E^2}{(1-e^2)^2a^{7/2}}\right]\left(\frac{5}{2}\sin^2 i - 2\right)$$

2 Calculate the time since perigee passage from mean anomaly using the equation in 2.3.

$$t_0 = \frac{M}{2\pi}T$$

3 Calculate the right ascension (α) and declination (δ) at each step.

for $t = t_0; t < t_{end}; t \mathrel{+}= \Delta t$ **do**

$\mathrm{M} = \dfrac{2\pi}{T}t$

$E - e\sin E = M$ {Solve Kepler's equation to get E}

$\theta = 2\tan^{-1}\left(\sqrt{\dfrac{1+e}{1-e}}\tan\dfrac{E}{2}\right)$

$\Omega = \Omega_0 + \dot{\Omega}\Delta t$

$\omega = \omega_0 + \dot{\omega}\Delta t$

$\phi = \omega_E(t - t_0)$

$\{\mathbf{r'}\} = [R_z(\phi)]\,\{\mathbf{r}\}$

$l = x/|\vec{r}| \quad m = y/|\vec{r}| \quad n = z/|\vec{r}|$

$\delta = \sin^{-1} n$ {Declination is equal to the geographical latitude}

$$\alpha = \begin{cases} \cos^{-1}(l/\cos\delta) & (m > 0) \\ 2\pi - \cos^{-1}(l/\cos\delta) & (m \leq 0) \end{cases}$$

$\lambda = \alpha - \Theta(t)$ {Convert right ascension to geographical longitude}

end for

7. SOFTWARE VERIFICATION

The final software was subjected to a series of tests to ensure its reliability and accuracy. One of the key principles used in the software design was modularity, which involves breaking down complex systems into smaller, more manageable components. This made it easier to identify and fix bugs because each component could be tested independently, rather than having to test the entire system at once. Moreover, the software also used object-oriented programming principles, which allow for the reuse of code and the creation of more efficient and scalable system. The use of these software principles not only made it easier to identify and fix bugs, but it also made the testing process much faster. By decomposing the system into smaller elements and using reusable code, the testing process was streamlined, allowing for more efficient and effective debugging.

To validate the accuracy of the software, data from various satellites was obtained and compared to the results simulated by the software. The pass predictions of the software were then tested by listening to several satellites from the ITU Ground Station. The SharjahSat-1 CubeSat was used to validate the software's ability to analyze communication links. In addition to the software being tested, the same simulations were also run using the GMAT R2020 satellite simulator, which is the current industry standard and the latest version available. The comparisons were conducted using both LEO and GEO satellites.

7.1 Tests using Satellite data

The aim of this test was to compare the results obtained from the software simulation to real data, in order to asses the accuracy levels of the outputs. Two different cases were considered, one with numerical propagation that includes all force models except by thrust, so satellites without propulsion systems were chosen for analysis, and one with the improvement on the data with the developed deep-learning based model. Around

50 different satellites were considered for this test. Comparison data was created using TLEs and SGP4 model. The TLEs were updated approximately once every 6 hours to generate data, so the obtained state vectors reflect reality as much as possible. The approximate RSS for 50 test cases for different types of orbit are given in Table 7.1.

Table 7.1 : Test Results.

	Max Position Diff. RSS (m)	
Orbit Type	Numerical Propagation	DL Model
LEO (ISS)	2.112E-00	3.232E-01
LEO (ITUpSat-1)	6.853E+00	1.523E-03
LEO (SSO)	1.577E+00	9.454E-03
GEO	5.653E-02	2.896E-02

Figure 7.1 also show the results for one test case using ITUpSat-1 satellite.

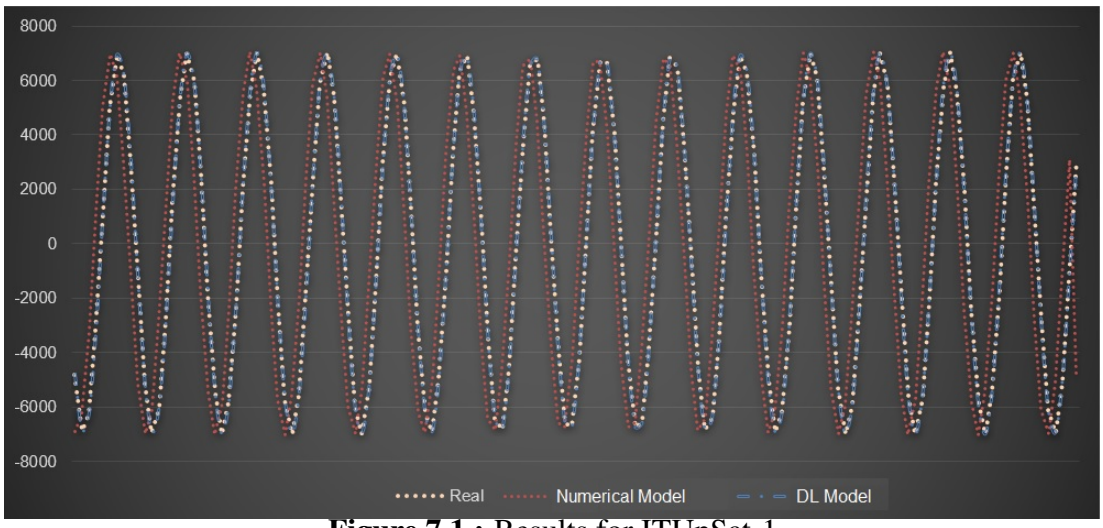


Figure 7.1 : Results for ITUpSat-1

7.1.1 Comparison with other software

The approach for testing the orbital dynamics model in the software involved comparing the results of numerical propagation to those obtained using GMAT and other available software. The tests covered both LEO and GEO regimes and included a variety of force models such as gravity potential, n-body attractions, drag, SRP, solid Earth tides, and relativity. The tests involved individually evaluating each dynamics model for selected orbit test cases, as well as testing the combined effects of all applicable models for other selected orbit test cases. Table 7.2 shows the initial state vectors in ICRF axis system for the chosen orbit types. The integrator used for these

tests was RKV89. Fixed values for SRP and drag were used for the tests. Table 7.3 provides the initial epochs, and duration and time step of simulations for the chosen test cases.

Table 7.2 : Comparison Test Initial State Vectors.

	Position (km)			Velocity (km/s)		
Orbit Type	X	Y	Z	V_x	V_y	V_z
LEO (ISS)	-82.6557	-5269.6561	4277.8336	6.0679	-3.01277	-3.588
LEO (ITUpSat-1)	3276.6883	1364.7791	-6133.4832	-4.873	-4.4179	-3.5915
LEO (SSO)	-2641.47	-6350.7	0	-0.906071	0.376865	7.5491
GEO	-24134.9	34576	0	-2.52114	-1.75981	0

Table 7.3 : Comparison Test Epochs and Durations.

Orbit Type	TOF (days)	Initial Epoch	Output Step (sec)
LEO (ISS)	1	21 Dec 2022 09:00:00.000	60
LEO (ITUpSat-1)	1	21 Dec 2022 09:00:00.000	60
LEO (SSO)	1	21 Dec 2022 09:00:00.000	60
GEO	7	21 Dec 2022 09:00:00.000	600

Table 7.4 shows data that compares the results of different orbit propagation tests. The data shows the maximum difference in position, measured in meters, between the test data and the actual results over the test duration. The velocity differences between the test results were also analyzed and were consistently much lower than the differences in position. The comparison of the most test results using the RSS method showed good agreement, with differences typically at the submeter level.

Table 7.4 : Comparison Results.

	Max Position Diff. RSS (m)		
Orbit Type	Orbiter	GMAT	Other
LEO (ISS)	7.666E-01	1.654E+00	8.844E-01
LEO (ITUpSat-1)	2.565E-03	1.167E+00	6.656E-01
LEO (SSO)	3.122E-02	9.875E-01	1.421E+00
GEO	3.221E-02	6.335E-04	8.177E-04

Since most of the data used to train the models come from LEO satellites, they have better results than GEO.

7.1.2 Experimental Results

In order to verify the software results with experimentation, two tests were performed. First, 10 consecutive passes of ITUpSat-1 were predicted using the software. Then, using the ground station in ITU Faculty of Aeronautics and Astronautics (Figure 7.3), the satellite ITUpSat-1 was tracked and listened during the predicted times.

It was shown that the pass predictions were correct as the satellite was transmitting beacon signals during the computed time intervals, as shown in Figure 7.2.

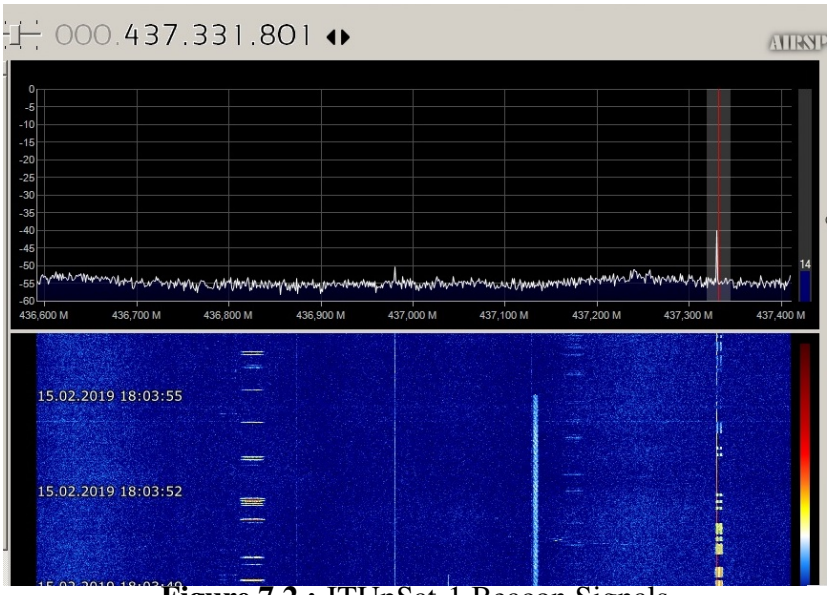


Figure 7.2 : ITUpSat-1 Beacon Signals

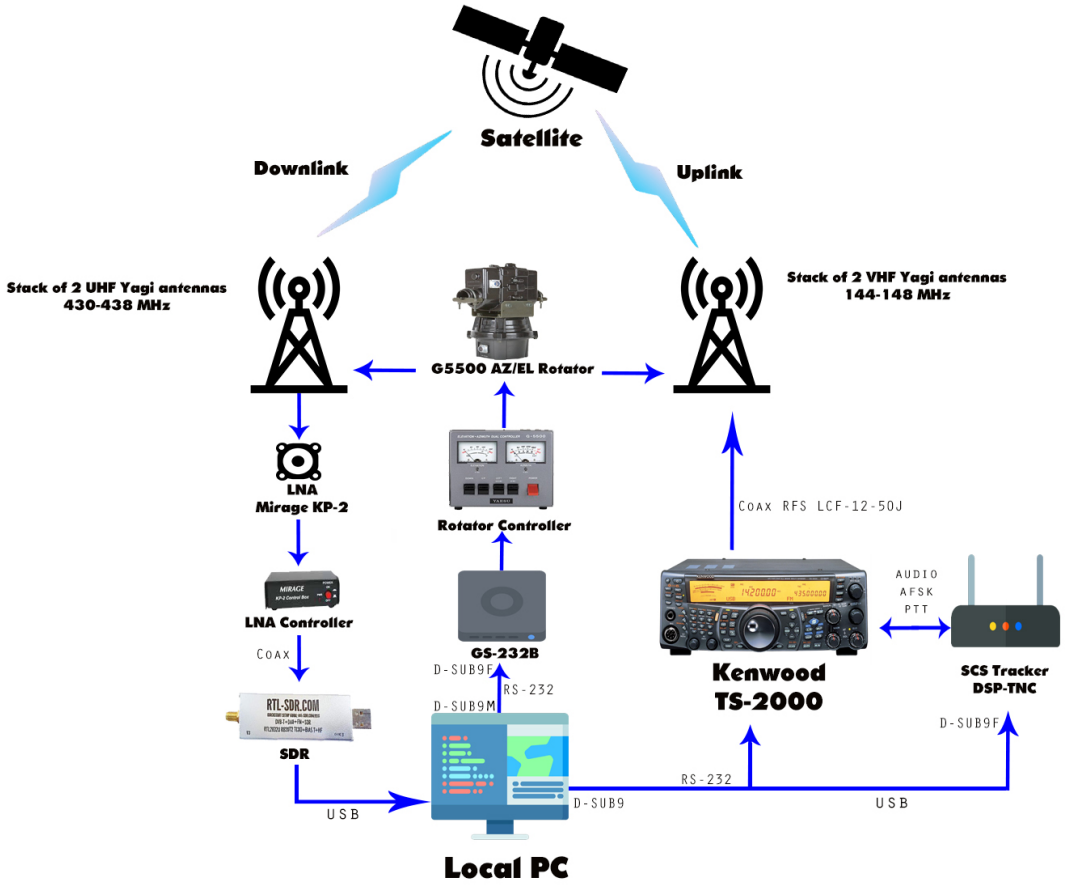


Figure 7.3 : ITU Ground Station Schematic

The actual pass (found from SGP4 using a very recent TLE) versus the pass predicted by the software is shown in Figure 7.4 as well.

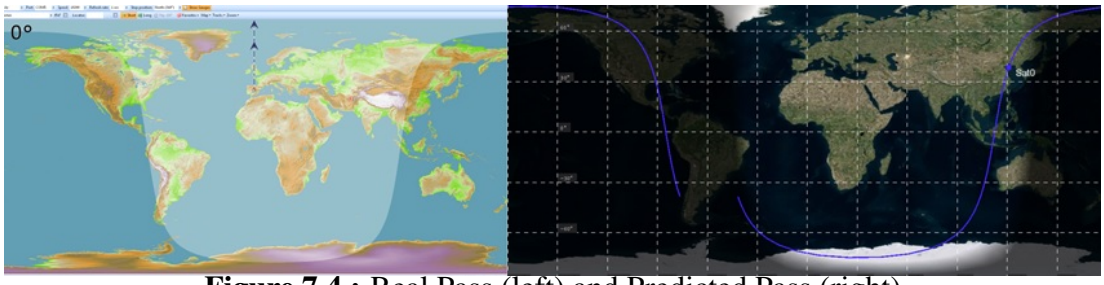


Figure 7.4 : Real Pass (left) and Predicted Pass (right)

The second test was verification of the link budget analysis. Sharjahsat-1 was used for this purpose. The communication system parameters were given as input to the software link budget calculator, as shown in Chapter 5.2.3, and the received signal power is found to be consistent with calculations, with an average accuracy of 98.65%. The signal received from the satellite is shown in figure 7.5.

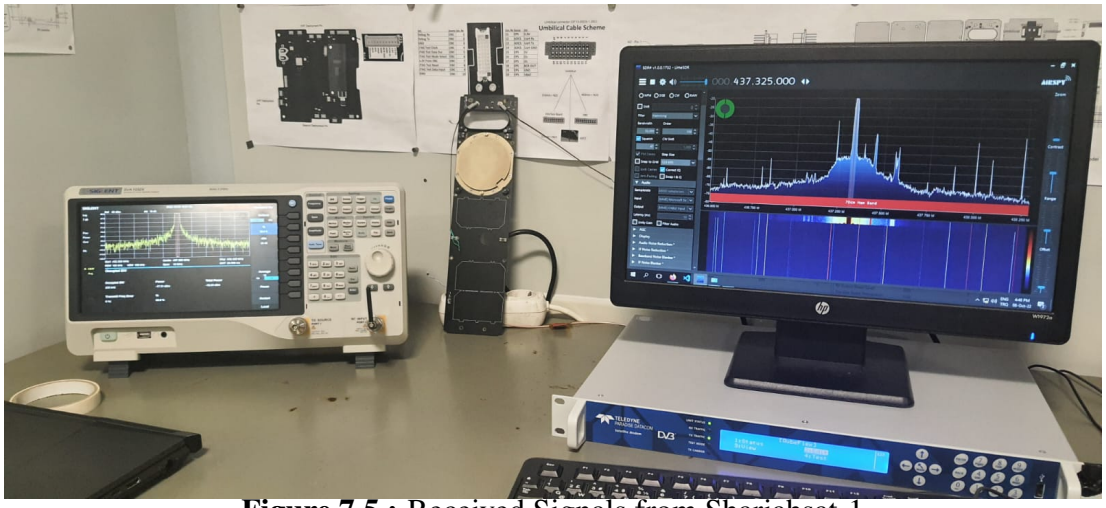


Figure 7.5 : Received Signals from Sharjahsat-1

7.2 UI Verification

This process involves testing the UI to ensure that it is intuitive and provides the necessary information and functionality to the user. Verification points included testing the layout and organization of the UI, checking for errors and inconsistencies, and ensuring that the UI is visually appealing and easy to navigate. Both developers and users were involved in the UI verification process to ensure that the software meets the needs of the intended audience. The user interface of the software was intuitive and easy for undergraduate students to navigate, as they were able to easily find what they were looking for. When the chapters of the thesis were shared with the users, the feedback indicated that it was effective in improving their understanding of space mission design and operations. Based on this feedback, the material will be refined and compiled into an educational glossary to be incorporated into the software. This glossary will serve as a reference for both software usage and for clarifying concepts related to mission planning for operators.

8. CONCLUSIONS

Spacecraft mission planning is an intricate process that depends on a variety of models which try to capture the complex governing forces in the harsh environment of space to solve the equation of motion. Astrodynamics calculations serve as only the starting point from which the critical parameters of the mission can be derived from, such as determination of ground station passes to find contact durations and eclipses to compute power system requirements.

The work performed in this thesis provides a computer platform for everybody interested in spacecraft mission operations. The software can be used at a basic level for educational purposes by simply visualizing orbits of different characteristics and showing how different assumptions during orbit propagation lead to different motions. Students can engage with the software to expand their understanding of astrodynamics, understanding which forces are in play and what factors are considered during mission planning phase. Organizations can use it on a deeper level and plan for real missions. It is intended that this work will aid in the future CubeSat missions developed in SSDTL as well as in other universities and space organizations.

The AI orbit propagator model developed within this thesis allows the myriad legacy spacecraft data to be used to increase the accuracy of the numerical models, a feature not available in any other kind of a software. As more and more missions are performed and more data is fed into the software, the accuracy is expected to increase even higher.

The software developed has an intuitive, easy to use interface that allows starters to begin experimenting without spending too much effort. The back-end is designed to be maintainable and easy to extend, as many more updates such as spacecraft sensor calculations are planned in the future. The thesis also explains the mathematical background and step-by-step process for performing the numerical integrations of the force models acting on spacecraft. These are the current default implementations used by the software.

Comparison with real data shows that the generated ephemerides align with real ephemerides with an accuracy of approximately 5% for LEO satellites. In this accuracy regime the results are on par with the commercial mission design software STK as well as the open-source GMAT software developed by NASA.

Finally, it must be emphasized that the release of the Orbiter software is not the end of this project, but a beginning. There are still numerous enhancements to be made to answer to the ever-growing requirements of the industry, such as an engine for orbital determination and for interplanetary missions.

8.1 Future Work

The software was designed to be able to be extended for various features related to space flight and satellite operations in the future. Some of these features include:

Other central bodies. Following the open/closed principle, all of the planetary values, such as gravitational parameter and radius, were not hard-coded to be of Earth's but taken from an object of the CelestialBody class. This will allow easily integrating other celestial bodies for faciliating and propagating orbits.

Hohmann and Bielliptic Transfers. The inclusion of common orbital transfer maneuvers, such as Hohmann transfers, to make it easier for users to execute these maneuvers.

Interplanetary trajectories. After the addition of different Celestial bodies such as the Moon, the next step for orbital maneuvers is to extend the capability to design interplanetary trajectories, using techniques such as patched conic approximation.

Rendezvous and Phasing. These will also be integrated into orbital maneuvers as most of the missions require these type of maneuvers that are critical for the mission. Since the software can propagate multiple orbits simultaneously at ease, it will not be difficult to perform the required calculations.

Attitude Models. There is currently no way to define or change the attitude behavior of the satellite in the software, which is required for complex orbital propagation. Different attitude laws will be added which the user can select from, such as Nadir-pointing, Point-tracking, fixed, etc.

Satellite sensors and instruments. There are various instruments used on satellites that can be analyzed such as cameras and radar systems. The software can be extended to perform complex calculations such as simulating the view of a camera system whose properties are defined.

Launch vehicles. Rocket dynamics can be incorporated into the simulator to perform a mission from start to finish. The user can find out if their rocket can carry the payload to the required orbit, and how much fuel is required on board etc. Additionally, the capability to analyze launch windows is a great necessity.

Aerial vehicles. Planes, UAVs, helicopters can all be included in the simulator by adding the appropriate flight dynamics algorithms. Communications between multiple vehicles etc. can be simulated.

Importing database files. There are different formats of satellite definitions and many databases online. It will be helpful to integrate different databases such as satellites, star catalogues, etc.

REFERENCES

[1] **Liang, X. and Liu, S.** (2020). *A C++ Based Modular Simulation Software of Satellites Earth Observation Mission Planning*, MATEC Web of Conferences 316.

[2] **The GMAT Development Team** (2020). *GMAT User Guide R2020A*, `https://gmat.atlassian.net/wiki/spaces/GW/overview`, date retrieved: 12.11.2021.

[3] **Maisonobe, L.**, **Pommier, V. and Parraud, P.** (2010). Orekit: An Open-source Library for Operational Flight Dynamics Applications.

[4] **GSFC** (2020). *General Mission Analysis Tool (GMAT) Mathematical Specifications*.

[5] **Vallado, D.** (2013). Fundamentals of Astrodynamics and Applications, Microcosm, Hawthorne, CA, 4. edition.

[6] **Lieske, J.H.**, **Lederle, T.**, **Fricke, W. and Morando, B.** (1977). Expressions for the Precession Quantities Based upon the IAU (1976) System of Astronomical Constants, *Astronomy and Astrophysics 58, 1*.

[7] **Seidelmann, P.K.** (1982). 1980 IAU Theory of Nutation: The Final Report of the IAU Working Group on Nutation, *Celestial Mechanics 27*, 79–106.

[8] **Aoki, S.**, **Guinot, B.**, **Kaplan, G.H.**, **Kinoshita, H.**, **McCarthy, D.D. and Seidelman, P.K.** (1982). The New Definition of Universal Time, *Astronomy and Astrophysics 105*, 359–361.

[9] **Union, I.A.** (2007). Nomenclature for Fundamental Astronomy (NFA).

[10] **Kaplan, G.H.** (2005). The IAU Resolutions on Astronomical Reference Systems, Time Scales, and Earth Rotation Models Explanation and Implementation, *United States Naval Observatory Circular No. 179*.

[11] **Green, R.M.** (1988). Spherical Astronomy, New York: Cambridge University Press. [3, 5].

[12] **Curtis, H.D.** (2014). Orbital Mechanics for Engineering Students 3rd Edition, Elseiver Aerospace Engineering Series, Butterworth-Heinemann, Oxford.

[13] **Montenbruck, O. and Gill, E.** (2005). *Satellite Orbits: Models, Methods and Applications*, Springer.

[14] **Danielson, D.**, **Sagovac, C. and Early, L.** *SEMIANALYTIC SATELLITE THEORY*, Naval Postgraduate School.

[15] **Tewari, A.** (2007). Atmospheric and Space Flight Dynamics, chapter 5, Modelling and Simulation in Science, Engineering, and Technology, Birkhauser, Boston, pp.123,129.

[16] **Chobotov, V.A.** (2002). *Orbital Mechanics*, AIAA Education Series, AIAA, 3. edition.

[17] **Delaunay, C.** (1860). Theorie du mouvement de la Lune, *Vol. I. Memoires de l'Academie des Sciences de l'Institut Imperial de France XXVIII:1*.

[18] **Fujimoto, K. and Scheeres, D.J.** (2012). *Correlation of Optical Observations of Earth-Orbiting Objects and Initial Orbit Determination*.

[19] **Battin, R.H.** (1987). *An Introduction to the Mathematics and Methods of Astrodynamics*, American Institute of Aeronautics and Astronautics Education Series.

[20] **Brouwer, D.** (1959). *Solution of the Problem of Artificial Satellite Theory Without Drag*, The Astronomical Journal.

[21] **Lyddane, R.H.** (1963). *Small Eccentricities or Inclinations in the Brouwer Theory of the Artificial Satellite*, The Astronomical Journal.

[22] **Kozai, Y.** (1959). *"The Motion of a Close earth satellite"*, The Astronomical Journal.

[23] **Subirana, J.**, **Juan Zornoza, J. and Hernández-Pajares, M.** (2011). *Transformations between Time Systems*, Technical University of Catalonia.

[24] **Yoder, C.**, **Williams, J. and Parke, M.** (1981). *Tidal Variations of Earth Rotation. Journal of Geophysical Research*, IERS Convention Center.

[25] **IERS**. *Earth Orientation Bulletin Series*, `https://hpiers.obspm.fr/iers/bul/buld/`.

[26] **Frank, M.** (1984). *The New FK5 Mean of J2000 Time and Reference Frame Transformations*, DVO Technical Report 84-2.

[27] **McCarthy, D. and Petit, G.** (2004). *IERS Conventions (2003). IERS Technical Note 32*, IERS Convention Center.

[28] **Meeus, J.** (1991). *Astronomical Algorithms*, Willmann Bell Inc, Richmond, VA.

[29] **Qinsy**. *UTC to GPS Time Correction*, `https://confluence.qps.nl/qinsy/latest/en/utc-to-gps-time-correction-32245263.html`, Retrieved September 23, 2022.

[30] **Rubinsztejn, A.** *Lagrange F & G solution*, `https://gereshes.com/2018/06/25/lagrange-fg-solution-the-2-body-problem/`, accessed: 2021-12-28.

[31] **Trim, N.**, (2009). Visualizing Solutions of the Circular Restricted Three-Body Problem, The State University of New Jersey, Camden, New Jersey.

[32] **Vallado, D.A.**, **Crawford, P.**, **Hujsak, R. and Kelso, T.** (2006). Revisiting Spacetrack Report #3, *Astrodynamics Specialist Conference*.

[33] **Kelso, T.** *Satellite Times Columns*, `https://celestrak.org/columns/`, accessed: 2022-11-20.

[34] **Hoots, F. and Roehrich, R.** (1988). *Models for Propagation of NORAD Element Sets*.

[35] **Vallado, D.A. and Crawford, P.** (2008). *SGP4 Orbit Determination*.

[36] **Gaposchkin, E.M. and Coster, A.J.** (1990). Evaluation of Thermospheric Models and the Precipitation Index for Satellite Drag, *Adv. Space Res., 10, 3303-3309*.

[37] **Marcos, F.A.**, **Killeen, T.L.**, **Burns, A.G. and Roble, R.G.** (1989). Evaluation of new atmospheric drag modeling techniques, *AAS 89-419*.

[38] **Jacchia, L.G. and Slowey, J.W.** (1981). *Analysis of Data for the Development of Density and Composition Models of the Upper Atmosphere*, Air Force Geophsyics Laboratory, AFGL-TR-81-0230.

[39] **Hedin, A.E.**, **Spencer, N.W. and Killeen, T.L.** (1988). *Empirical Global Model of Upper Thermosphere Winds Based on Atmosphere and Dynamics Explorer Satellite Data*.

[40] **Kelso, T.** *Space Weather Data Documentation*, `https://celestrak.org/SpaceData/SpaceWx-format.php`, accessed: 2022-11-29.

[41] **Schatten, K.H.**, **Scherrer, P.H.**, **Svalgaard, L. and Wilcox, J.M.** (1978). Using Dynamo Theory to predict the Sunspot Number during Solar Cycle 21, 411–414.

[42] **ISWA** (2022). `https://iswa.gsfc.nasa.gov/iswa_data_tree/model/solar/FDF/SchattenSolarFluxPrediction/`.

[43] **Long, A.e.a.** (1989). *Goddard Trajectory Determination System Mathematical Theory, Revision 1*, Lanham-Seabrook, MD.

[44] **Knocke, P.**, **Ries, J. and Tapley, B.** (1988). Earth Radiation Pressure Effects on Satellites, *Proceedings of the AIAA/AAS Astrodynamics Conference, 88-4292-CP*, pp.577–587.

[45] **McCarthy, D.D.** (1996). *IERS Conventions (1996)*, Central Bureau of IERS - Observatoire de Paris, Paris.

[46] **Gendt, G. and Sorokin, N.A.** (1978). *Probleme bei der numerischen Integration von Satellitenbahnen mit ho her Genauigkeit*, Vermessungstechnik, 26. Jg., Heft 9.

[47] **Bate, R.R.**, **Mueller, D.D. and White, J.** (1971). *Fundamentals of Astrodynamics*, Dover Publications.

[48] **Goodfellow, I.**, **Bengio, Y. and Courville, A.** (2016). *Deep Learning*, MIT Press.

[49] **Larson, W. and Wertz, J.** (2005). *Space Mission Analysis and Design*, Microcosm.

[50] **Alfano, S.**, **Negron, D. and Moore, J.** (1992). Rapid Determination of Satellite Visibility Periods, 281–296.

[51] **Sharaf, M.**, **Hayman, Z. and M.E., A.** (2012). Satellite to Satellite Visibility, 26–40.

[52] **Liu, K.** (1978). *Earth Oblateness Modeling*, D. Reidel Publishing Company.

[53] **Collins, S.** (1992). *Geocentric Nadir and Range from Horizon Sensor Observations of the Oblate Earth*, AIAA Paper No. 92-176 presented at the AAS/AISS Spaceflight Mechanics Meeting, Colorado Springs. CO.

[54] **AMSAT**. *Tools for Spacecraft and Communication Design*, `https://www.amsat.org/tools-for-calculating-spacecraft-communications-link-budgets-and-other-design-issues/`, date retrieved: 12.08.2022.

[55] **Recommendation, I.** (2022). Radio noise, 372–15.

[56] **A.R.R.L.** (1974). *The ARRL Antenna Handbook*, American Radio Relay League.

[57] **Ippolito, L.J.** (1986). *Radio Propagation in Satellite Communications*, Van Norstrand Reinhold.

[58] **Gorev, V.**, **Prokopiev, V.**, **Prokopiev, Y.**, **L.D., S. and Sidorchuk, A.** (2019). Calculating electric power generated by 3U CubeSat's photoconverters depending the orbit and orientation parameters.

[59] **Etchells, T. and Berthoud, L.** (2019). Developing a Power Modelling Tool for CubeSats.

[60] `www.orbytespace.com`.

[61] `www.electronjs.org`.

[62] `www.cesium.com/learn/cesiumjs-learn/`.

[63] **McConnell, S.** (2004). *Code Complete*, Microsoft Press 2nd Edition.

[64] **Martin, R.C. and Coplien, J.O.** (2009). *Clean Code: A Handbook of Agile Software Craftsmanship*, Prentice Hall.

APPENDICES

APPENDIX A : UI Screenshot
APPENDIX B : Part of Codebase

APPENDIX A : UI Screenshot

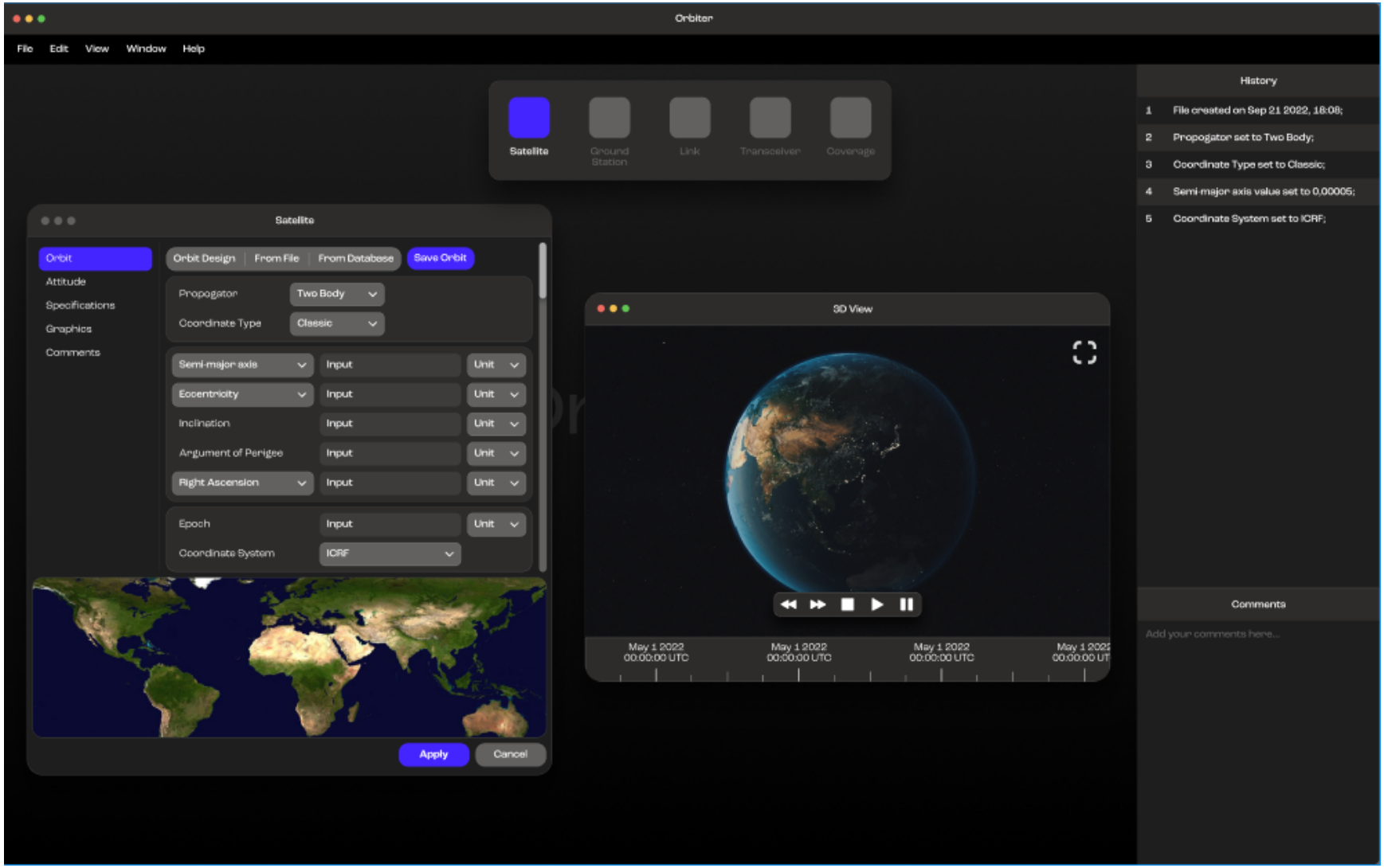


Figure A.1 : User Interface Design of Software.

APPENDIX B : Part of Codebase

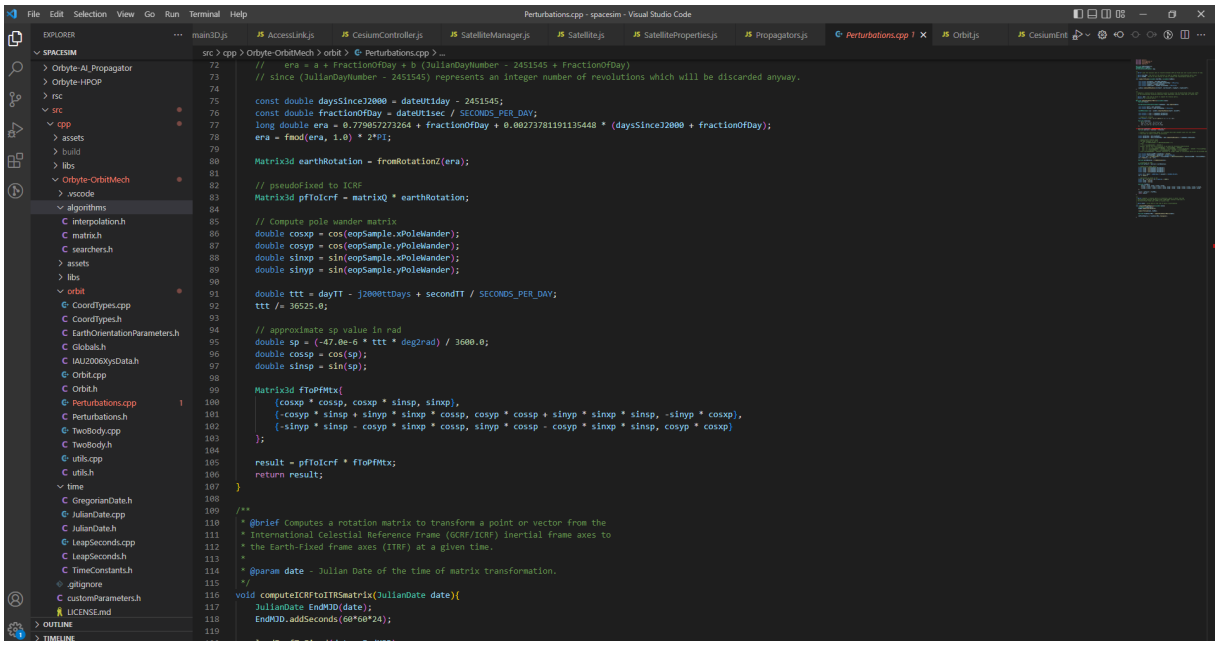


Figure B.1 : Screenshot from Codebase.

Figure B.2 : File Hierarchy Trees.

CURRICULUM VITAE

Emirhan Eser GÜL:

EDUCATION:

- **B.Sc.:** 2019, Istanbul Technical University, Faculty of Aeronautics and Astronautics, Department of Astronautical Engineering

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2022 Flight Dynamics Specialist and Software Developer, Leanspace, Strasbourg
- 2018-2022 Satellite Software & Communications Engineer, Space Systems Design and Testing Laboratory, Istanbul Technical University.
- 2020-2022 Founder and CEO, Orbyte Aerospace Technologies, Istanbul
- 2020-2021 Astronautical Engineer, Novart Defence and Space Technologies, Istanbul
- 2019-2020 Aircraft Systems Engineer, ATS Team, Istanbul
- 2019 AIAA Space Design Competition - 3rd Place Award

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- **Gül E.E.**, Karabulut B., Sarıca K., Aslan, A. R. (2022). On-Board Software Development for a 3U CubeSat. *11th Nano-Satellite Symposium*, October 17-19, 2022 Istanbul, Turkey. (Presentation Instance)

OTHER PUBLICATIONS, PRESENTATIONS AND PATENTS:

- Alsabt, I., Faroukh, Y., Alhammadi, A., AlKaabi, T., Alketbi, F., Alansaari, M., BinAshour, M., Fernini, I., **Gül E.E.**, Karabulut B., Aslan, A. R., Kalemci. E., Al Naimiy, H. (2022). Sharjah-Sat-1 Space-to-Ground Telecommunication Operations. *73rd International Astronautical Congress*, September 18, 2022 Paris, France.

- AlKaabi, T., Alsabt, I., Madara, S.R., BinAshour, M., Fernini, I., **Gül E.E.**, Karabulut B., Al Naimiy, H., Aslan, A. R. (2020). SAASST Ground Station: Satellite Tracking and Control for High Data Rates . *71th International Astronautical Congress*, October 12-14, 2022

- **Gül E.E.**, Demir U., Öztekin O., Karpat, M., Türkyılmaz, E. (2022). DERİN UZAY İSTASYONU - AY YÜZEYİ ARASI İNSAN VE KARGO TAŞIMA ARACI TASARIMI. *Ulusal Havacılık ve Uzay Konferansı*, September 9, 2022 Ankara, Turkey.