BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

SIMULATION ENVIRONMENT FOR OBJECTS IN LOW EARTH ORBIT

SIMULAČNÍ PROSTŘEDÍ OBJEKTŮ NA NÍZKÉ OBĚŽNÉ DRÁZE

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **ZBYNĚK POSPÍŠIL**
AUTOR PRÁCE

SUPERVISOR **Ing. JIŘÍ NOVÁK**
VEDOUCÍ PRÁCE

BRNO 2024

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Bachelor's Thesis Assignment

153125

Institut:	Department of Computer Graphics and Multimedia (DCGM)
Student:	**Pospíšil Zbyněk**
Programme:	Information Technology
Title:	**Simulation Environment for Objects in Low Earth Orbit**
Category:	Modelling and Simulation
Academic year:	2023/24

Assignment:

1. Study the basics of orbital mechanics and describe the influences causing changes of orbital elements.
2. Perform motion data acquisition of objects in low Earth orbit (satellites, space trash).
3. Develop a simulation environment with a Graphical User Interface (GUI) allowing prediction of position and velocity of objects in time.
4. Perform uncertainty analysis of the simulation model and compare the simulated trajectories with available data.
5. Evaluate the achieved results and discuss possible future advances of the project.

Literature:

- CURTIS, Howard D. *Orbital mechanics for engineering students*. Amsterdam; Boston: Elsevier Butterworth-Heinemann, 2005, ISBN 0-7506-6169-0.
- MAINI, Anil K a Varsha AGRAWAL. *Satellite Technology*. 3. New York: Wiley, 2014. ISBN 9781118636473.

Requirements for the semestral defence:
1. and 2. assignment points.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Novák Jiří, Ing.**
Head of Department:	Černocký Jan, prof. Dr. Ing.
Beginning of work:	1.11.2023
Submission deadline:	16.5.2024
Approval date:	9.11.2023

Abstract

This thesis is concerned with the problematic of predicting satellite's position in Low Earth orbit. It implements such behaviour with numerical integration methods and basic analytical methods, as well as with graphical user interface for plotting the results and enabling the user to manually adjust simulation's aspects. The input satellite data is taken in real time from public databases in TLE format. This work also provides an insight into various principles of orbital mechanics.

Abstrakt

Tato práce se zabývá problematikou předpovědi polohy satelitu na Nízkém orbitu Země a vizualizací těchto dat v samostatném programu. Takových cílů je dosaženo použitím jak analytických, tak i numerických integračních metod, analyzujících orbitální parametry získané z veřejných databází ve formátu TLE. Účelem práce je srozumitelně přiblížit tuto problematiku veřejnosti a nadšencům. Celý průběh simulace je zachycen v grafickém prostředí a umožňuje uživateli ladit podrobný běh simulace.

Keywords

Mathematical numerical simulation, Orbital mechanics, Determining position of satellite, Orbital perturbations, Analytical simulation, Coordinate conversion, Low Earth orbit, TLE, Ground tracks, Visualization

Klíčová slova

Matematická numerická simulace, Orbitální mechaniky, Určování pozice satelitu, Perturbace orbitu, Analytická simulace, Převody souřadnic, Nízká orbita Země, TLE, Poloha nad zemí, Vizualizace

Reference

POSPÍŠIL, Zbyněk. *Simulation Environment for Objects in Low Earth Orbit.* Brno, 2024. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Jiří Novák

Rozšířený abstrakt

Tato práce popisuje problematiku předpovědi dráhy satelitu na Nízké oběžné dráze Země. Jejím cílem je jednak vytvořit samostatný program, který dokáže získat informace o orbitálních parametrech satelitu ze dvou databází a druhak numerickými a analytickými metodami co nejpřesněji nasimulovat tato data v čase a následně je zobrazit v grafickém prostředí. Zároveň je záměrem této práce nastínit problematiku orbitálních mechanik nadšencům do kosmonautiky a srozumitelně ji vysvětlit.

Pozici satelitu na oběžné dráze lze popsat různými způsoby: vektorem ze středu Země, jeho pozicí pomocí zeměpisné šířky, zeměpisné délky a výšky nad povrchem, nebo tzv. orbitálními parametry. Každý způsob je vhodný k něčemu jinému. Tato práce nejdřív získá data ve formátu tzv. dvouřádkových elementů (TLE), které orbitální parametry obsahují. Jedná se o velmi univerzální způsob, jak polohu reprezentovat. Simulovat můžeme už je – analyticky – avšak mnohem přesnější je převést parametry do vektoru ze středu Země, tzv. inerciální soustavy. V ní je satelit reprezentován vektorem polohy a také vektorem jeho rychlosti. Tento způsob je vynikající k numerickému, přesnému určování budoucí trajektorie, protože narozdíl od orbitálních parametrů dokáže počítat s více vnějšími vlivy, které na satelit působí v dlouhodobém horizontu. Takovými vlivy jsou podle klesající důležitosti odpor atmosféry, gravitační vliv Slunce a Měsíce, vliv Slunečního záření, vliv šišatosti Země atd. Každý z těchto vlivů působí na satelit velmi malým zrychlením, které mění výsledný vektor rychlosti a tím i vektor polohy. Primárním vlivem je samozřejmě gravitace Země samotné.

Vektorem polohy ze středu Země ale nepopíšeme polohu běžnému člověku. Souřadnice se proto musí převést do známé zeměpisné šířky, zeměpisné délky a výšky nad povrchem. K tomu je ještě potřeba znát jak moc je Země otočená, což zjistíme tzv. epochou, tedy časovým údajem naměření TLE.

Druhá kapitola se zabývá teorii, vysvětlí funkci jednotlivých orbitálních parametrů, popisuje převody mezi výše zmíněnými souřadnými systémy a popisuje vlivy, které svým zrychlením působí na satelit. Tato práce počítá s vlivy atmosféry, šišatostí Země a gravitačním působením Slunce, Měsíce a všech planet Sluneční soustavy.

Třetí kapitola se zabývá implementací programu, tedy získáváním TLE dat ze dvou veřejných databází (Celestrak a Space-Track), popisu formátu TLE dat, architekturou simulátoru, numerickým metodám Euler a Runge-Kutt a nastíněním analytické simulace orbitálních elementů. Grafická část programu je implementována v jazyce Python v prostředí PyQt6, simulátor je pro větší rychlost a efektivitu napsán v jazyce C++.

Čtvrtá kapitola se zabývá validací modelů a porovnáváním výsledků simulátoru s pozdějšími referenčními měřeními satelitu a s referenčním modelem SGP4, kterému se práce snaží přiblížit.

Simulation Environment for Objects in Low Earth Orbit

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Mr. Ing. Jiří Novák. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Zbyněk Pospíšil
May 15, 2024

Acknowledgements

I would line to sincerely thank my supervisor Mr. Ing. Jan Novák for his support, all the advice and a lot of patience, especially during final months.

Special thanks belongs to Mr. Martin Douša, also for a lot of patience and willingness to teach others how to code.

Finally, I owe a debt of gratitude to my family and my closest friends for their continuous support, without which this work would not have been possible.

Contents

List of abbreviations

3LE Three-line element set. 29–32

BCRS Barycentric Celestial Reference System. 32

ECEF Earth-centred, Earth-fixed frame. 6, 13, 14, 16, 18, 33, 34, 37–39, 42, 44

ECI Earth-centred inertial frame. 5, 13, 14, 16–18, 21, 25, 30, 34, 37–39, 41, 42, 44

ERP Earth radiation pressure. 25, 44

GEO Geostationary Earth orbit. 21, 43

GMST82 Greenwich mean sidereal time 1982. 6, 14–16

GMT Greenwich Mean Time. 15

LEO Low Earth orbit. 5, 7, 19, 21, 26, 27, 30, 44

MEO Medium Earth orbit. 21, 43

NORAD North American Aerospace Defense Command. 32, 41, 51

ODE Ordinary differential equation. 34, 35

RAAN Right ascension of the ascending node. 10, 14, 52

RK4 Fourth Order Runge-Kutta. 34, 35

SGP Simplified General Perturbation model. 5, 6, 18, 20, 22, 24, 30, 39, 41–46

SRP Solar radiation pressure. 24, 25, 44

TEME True equator, mean equinox frame. 5, 13, 18, 30, 43, 44

TLE Two-line element set. 5, 6, 13, 18, 20, 24, 28–32, 34, 37, 39–44, 46, 51–53

USSA76 U.S. Standard Atmosphere 1976. 6, 20, 21

UT1 Universal Time. 15, 44

UTC Universal Coordinated Time. 15, 44

WGS84 World Geodetic System 1984. 6, 16–18

Nomenclature

Orbital Elements

Ω	Right ascension of the ascending node; RAAN	rad
ω	Argument of periapsis	rad
θ	True anomaly	rad
E	Eccentric anomaly	rad
e	Eccentricity	1
h	Specific angular momentum	$\mathrm{km}^2 \cdot \mathrm{s}^{-1}$
i	Inclination	rad
M_e	Mean anomaly of elliptical orbit	rad

Vectors

$\mathbf{r}$	Position vector in cartesian system
$\mathbf{v}$	Velocity vector in cartesian system
$\mathbf{a}$	Acceleration vector in cartesian system
$\mathbf{p}$	Perturbation vector in cartesian system

Orbital perturbations

μ_e	Earth's standard gravitational parameter	$\mathrm{km}^3 \cdot \mathrm{s}^{-2}$
μ_m	Moon's standard gravitational parameter	$\mathrm{km}^3 \cdot \mathrm{s}^{-2}$
$\mu_\odot$	Sun's standard gravitational parameter	$\mathrm{km}^3 \cdot \mathrm{s}^{-2}$
ω_e	Earth's angular rotation velocity	$\mathrm{rad} \cdot \mathrm{s}^{-1}$
ρ	Atmospheric density	$\mathrm{km} \cdot m^{-3}$
A	Cross-sectional area of satellite perpendicular to motion	m^2
B	Ballistic coefficient	1
B^*	B-Star	$\mathrm{kg} \cdot \mathrm{m}^2$

C_d	Drag coefficient	1
P_{sr}	Solar radiation pressure	$\mathrm{N} \cdot \mathrm{m}^{-2}$

Other Symbols

λ	Longitude; azimuthal angle of Earth	rad
ϕ	Azimuthal angle of spherical coordinate system	rad
ϕ	Geodetic latitude	rad
ϕ'	Geocentric latitude; polar angle of Earth	rad
θ	Polar angle of spherical coordinate system	rad
θ_{GMST82}	Greenwich mean sidereal time angle	rad
a	Semi-major axis	km
h	Height of the satellite	km
j	Julian day	
n	Mean motion	
P	Periapsis	km
R_e	Earth's equatorial radius	km
Υ	Vernal equinox	

Chapter 1

Introduction

Space is a fascinating place, that has attracted the attention of humans for centuries. All the way from basic astronomy in ancient Greece and China, through new insights made outside of Europe in the Middle Ages, and with revolution in knowledge starting with Nicolaus Copernicus's geocentric model, Johannes Kepler's theory of orbital mechanics and Galileo Galilei's improvements in observational astronomy, up to the knowledge boom in 20th century and first rockets. Advancements in this scientific field have become the matter of national pride for politics and a unique opportunity for scientists to develop new technologies, that improve our lives on daily basis.

This work attempts to summarize the technical and mathematical know-how of many documents, regarding the topic of basic satellite simulation. Such works are very scarce and often too complex.

The main focus of this thesis is to create an user-friendly program with graphical interface, which retrieves satellite's orbital parameters from public databases and uses author's model to propagate its movement in LEO. The intent is to be as precise as possible in coordinate conversions and subsequent simulations. The results will be compared with the reference SGP4 model, which is commonly used by state agencies and amateurs for quick but relatively imprecise orbital propagation. The author also aims to help to understand the topics of orbital mechanics and explain them to an average space fan.

Author of this document is a space enthusiast, who deeply admires recent advances in rocket technology. This work is dedicated to those, who dream of making a difference.

This thesis incorporates numerical orbit propagation in the form of vector-based simulation, as does the reference SGP4 model and simplified analytical propagation of orbital elements. Basics of kinematics alongside spherical coordinates can be found in section 2.1. Our model processes TLE data (explained in section 3.2), containing orbital elements in ECI TEME coordinate frame, specific to use with SGP models. Orbital or Keplerian elements, along the Kepler's laws of planetary motion are explained in section 2.2. This in-depth look also includes their variations, different angles (anomalies) or parameters. Section 2.3 begins with their transformation to perifocal coordinate frame in 2D space. This is then transformed into orbit around the Earth into ECI frame using Euler's zxz transformation matrix from eq. (2.14). As spacecraft's position is now determined with state vector – position vector and velocity vector – its propagation is done numerically with Euler or Runge-Kutta simulation methods (both in section 3.3). The foundation for correct orbital propagation

is formed by the two-body equation for relative motion (eq. (2.27)). This basic satellite-Earth interaction is expanded upon with perturbation effects, such as atmospheric drag (using USSA76 model), third body gravitation (Skyfield library) or Earth obliquity (all in section 2.4). Solar and Earth radiation pressures are described, but not implemented. Conversion of state vector into familiar ECEF WGS84 position of latitude longitude and height is achieved with TLE's measurement timestamp (epoch). This epoch is first converted into Julian date, using the J2000 constant and to GMST82 angle, describing the angle of Earth's rotation (see eq. (2.19)). The calculated latitude (see eq. (2.25)) is geodetic, respecting the bulged shape of Earth. Processes described in this paragraph are illustrated in fig. 3.3.

Regarding the design and implementation of the model, the UI is written in Python's library PyQt6 and the computation part in C++ for greater efficiency. The file structure is outlined in fig. 3.1 and in table 3.1. TLEs are obtained from Celestrak and Space-Track databases, that supplement each other. Position of the Sun, Moon and the planets of our Solar System is retrieved via Skyfield python library. Handling of the requests, as well as description of the TLE format is described in section 3.2. Simulator implementation is briefly summarized in section 3.3. Plotting of trajectories is done using PyQtGraph (described in section 3.4).

Finally, chapter 4 focuses on the validation of the model. Precision of coordinate conversions (in section 4.1) and comparison of propagated orbits with reference TLE measurements (in section 4.2) and with the reference SGP4 model (see section 4.3) are summarized at the end. Summarized are also SGP4's limitations.

Chapter 2

Theory of Orbital Mechanics

This chapter explains the theory behind kinematics of space travel, used reference frames and subsequent conversions between them and various orbital perturbations, that affect objects in Low Earth orbit (LEO). This chapter also tries to present the complex topic of orbital mechanics in an understandable way and consequently does not derive the principles nor the theory of provided equations. However their source is, conveniently, always listed. All equations are functionally equal to those used by the simulator.

2.1 Basics of kinematics

Kinematics is a subdomain of physics that studies and describes the geometry of motion. The causes for that motion are various, for example the gravitational pull or radiation pressure. However effects of such causes are the concern of dynamics. Kinematics explains the basics, mainly the positioning, velocity and acceleration of the studied object. The position of a point may be described in terms of cartesian, spherical, cylindrical, or curvilinear coordinates [3]. This work utilizes cartesian system with familiar vectors and spherical system for the "sphere" of the Earth with two angles – latitude, longitude and also a height.

The vectors in cartesian coordinates can be represented as lines, that start from origin, have direction and magnitude. Vectors are typically in 2D or 3D space with two or three coordinates respectively. All vectors in following chapters will be in bold.

The position in cartesian coordinates is given by position vector $\mathbf{r} \in \mathbb{R}^{\nVdash}$, with origin at the main reference point (e.g. a planet) and tip at the object's location. Following chapters will use metres m as primary units.

The velocity vector $\mathbf{v} \in \mathbb{R}^{\nVdash}$, representing object's velocity, originates from the tip of position vector and points to the new location of object in delta time. Commonly used SI units are naturally metres per second $\mathrm{m} \cdot \mathrm{s}^{-1}$. The vector can be mathematically expressed by eq. (2.1).

$$\mathbf{v} = \dot{\mathbf{r}} = \frac{\Delta \mathbf{r}}{\Delta t} \tag{2.1}$$

Finally, acceleration vector $\mathbf{a} \in \mathbb{R}^{\not\#}$ also originates from the tip of the position vector, however its direction and magnitude represent the change, over delta time, of the velocity vector. The SI units are metres per squared second $\mathrm{m} \cdot \mathrm{s}^{-2}$. Analogically, whilst utilizing velocity from eq. (2.1), acceleration can be mathematically expressed by eq. (2.2).

$$\mathbf{a} = \ddot{\mathbf{r}} = \dot{\mathbf{v}} = \frac{\Delta \mathbf{v}}{\Delta t} \tag{2.2}$$

The link between all three vectors can be furthermore easily explained in 2D space in fig. 2.1.

(a) Vectors in time t_0 (b) Vectors in time t_1

Figure 2.1: Relation between the different kinematics vectors in delta time.

Note that the tip of position vector $\mathbf{r}$ was moved by the velocity vector $\mathbf{v}$. Analogically, same applies for velocity and acceleration vectors. These principles, albeit in 2D space, are the very basics for simulating and predicting movement of objects in 3D, as the acceleration vector is known at any given moment in time. This is furthermore discussed in section 2.4.

The position in spherical coordinates is dictated by two angles and a distance from the origin – centre of the sphere. The distance r therefore defines the radius of the sphere. First angle, the polar angle θ lies between the z axis facing upwards (northward), the origin and an imaginary point. The second, azimuthal angle ϕ, lies between the x axis, the origin and the projection of our imaginary point onto xy plane, as illustrated in fig. 2.2.

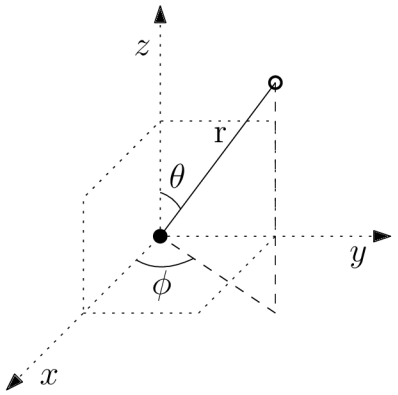

Figure 2.2: Showcase of spherical coordinates (θ, ϕ, r) defining a point.

When referencing our planet, polar angle θ is known as latitude ϕ and azimuthal angle ϕ is called longitude λ. The x axis passes through the Vernal equinox Υ. The distance r is measured from the Earth's centre.

2.2 Keplerian elements

Johannes Keppler, German astronomer, described the basics of planetary motion at the start of 17th century. His findings were based on the work of Nicolaus Copernicus and improved his three laws regarding circular orbit theory as such:

1. The orbit of a planet is an ellipse with the Sun at one of the two focal points.
2. A line between a planet and the Sun sweeps out equal areas during equal intervals of time.
3. The square of an object's orbital period is proportional to the cube of the length of the semi-major axis of its orbit.

Although these laws describe planetary motion, the planet and Sun can also be interchanged with the orbit of satellite and Earth. The laws state, that planet (or satellite) on elliptical path around the object it orbits, travels faster when it´s closer to the orbiting body, as it exchanges potential energy for kinetic and vice versa. In reality, Keppler´s findings don't account for the small acceleration exerted on the Sun from the gravitational pull of the orbiting planets in accordance with Newton´s second and third laws. This would be problematic with objects of comparable mass orbiting each other, but for most situations inside our Solar System, such effects are practically negligible [21].

In addition, the apsides are also commonly used to describe the nearest and farthest point of the orbiting object around its primary body. Any object that is orbiting planet Earth names them perigee and apogee, while for example orbits around the Sun have perihelion and aphelion respectively. When describing any arbitrary orbit, the terms periapsis and apoapsis are used.

Main orbital elements

Keplerian elements, also known as orbital elements, are formed by a set of six parameters, that describe the orbit of an object and its position on it. Although they are commonly used in orbital mechanics for forming satellite's or planet's orbit, they present an idealized mathematical view, that does not account for external disturbances – perturbations. Tracking programs therefore need to perform slight corrections.

Defining the size and shape of the orbit requires:

- eccentricity e – Unit of elongation from perfectly circular orbit. Circular orbits have $e = 0$, elliptical orbits $e \in (0, 1)$. Eccentricity equal or greater than 1 is used to describe parabolic or hyperbolic non-orbital trajectories of objects. $e \in R^+$, $e \in \langle 0, \infty)$; [1]
- specific angular momentum h — The cross product of relative position and relative velocity vectors the body divided by its mass, often replaced by semi-major axis a. $h \in \langle 0, \infty)$; $[\mathrm{km}^2 \cdot \mathrm{s}^{-1}]$

or alternatively

- semi-major axis a — The sum of apoapsis and periapsis distances divided by two. The centre of elliptical orbit. $a \in (0,\ \infty)$; [km]

With the orbit plane set, we can describe the orientation of the orbit with three Euler angles:

- right ascension of the ascending node Ω – Angle between the Vernal equinox's Υ direction and the ascending node of the orbit. Ascending node is the point, where orbiting satellite crosses the Earth's equator, moving from southern to northern hemisphere. Its counterpart is the Descending node. Also known as RAAN or RA of node. $\Omega \in \langle 0°,\ 360°)$
- inclination i — The angle between reference (for example equatorial) and orbital plane that it describes. Satellite orbits close to 0 degrees are referred to as equatorial, while orbits close to 90 degrees are regarded as polar. $i \in \langle 0°,\ 180°)$
- argument of periapsis ω — Orients the elliptic orbit in orbital frame, i.e. orients the orbit's periapsis, its lowest point. Also known as argument of perigee. $\omega \in \langle 0°,\ 360°)$

The last orbital element provides us with an exact position of the orbiting object:

- true anomaly θ — Angle on orbital plane between orbiting object and its periapsis with focus in the focal point (inside orbiting body). Increases non-linearly as objects move faster near periapsis and slower near apoapsis. Often replaced by mean anomaly M. $\omega \in \langle 0°,\ 360°)$

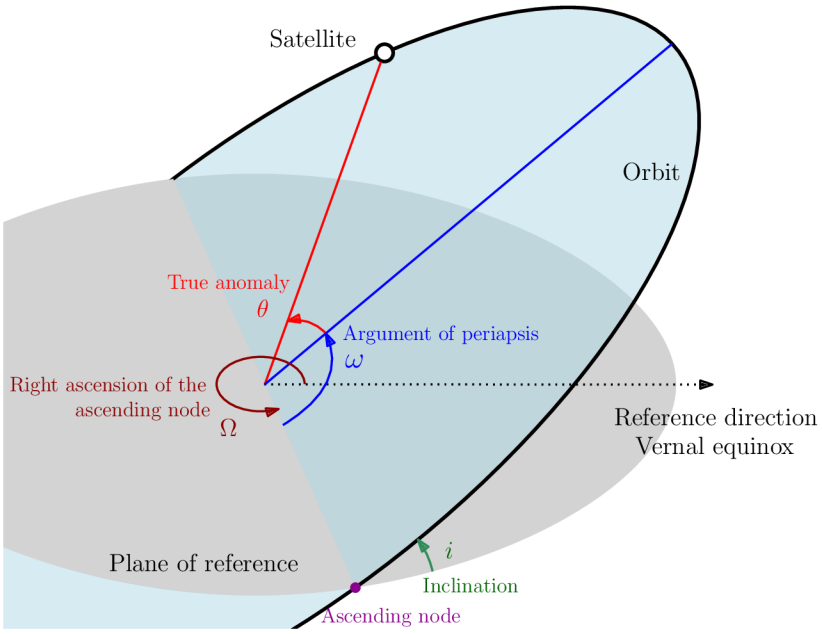

Figure 2.3: Keplerian elements that defy orbit's orientation in space.

As not every orbital parameter might be known, several equations will be presented, that explain various conversions. First, the true anomaly θ can be obtained via eq. (2.3),

$$\theta = 2 \cdot \arctan\left(\tan\left(\frac{E}{2}\right) \cdot \sqrt{\frac{1+e}{1-e}}\right) \tag{2.3}$$

where e is eccentricity and E is eccentric anomaly from eq. (2.7). Specific angular momentum h may also not be specified and needs to be obtained via eq. (2.4),

$$h = \sqrt{\mu_e} \cdot P \cdot (1+e) \tag{2.4}$$

where e stands for eccentricity, μ_e is the gravitational parameter of Earth and P is periapsis of the orbit, calculated from eq. (2.5),

$$P = a \cdot (1-e) \tag{2.5}$$

where a is the semi-major axis of the orbit. The format of data, that the simulator retrieves from public sources however omits even this. Instead, mean motion n is utilized, which describes how many orbits per day spacecraft undergoes.[1] Low values are consistent with satellite having both apsides in low altitudes, hence moving faster and vice versa. The semi-major axis can be obtained from eq. (2.6),

$$a = \sqrt[3]{\frac{\mu_e}{\left(n \cdot \frac{2\pi}{86{,}400}\right)^2}} \tag{2.6}$$

where μ_e stands for Earth's gravitational parameter and n for mean motion. Note the constant 86,400 representing number of seconds in 24 hours. All these equations will prove useful in conversions of orbital elements to other coordinate frames and in analytical propagation of the elements.

Anomalies

These anomalies (angles) can replace the last orbital element – true anomaly θ. Although not part of the basic six, they are commonly used in various equations and calculations.

- mean anomaly M_e – Fraction of the elapsed orbit since the start of periapsis. Sweeps the area of orbit in the same time as true anomaly θ, but its value increases linearly along an imaginary circle, as seen in fig. 2.4. It therefore does not always point to spacecraft, unlike true anomaly θ, except when it reaches periapsis or apoapsis. The subscript e means elliptic orbit. $M_e \in \langle 0,^\circ\ 360^\circ)$

[1] It can also be described by radians per second, needed for some equations. But Two-line elements (TLEs) describe it in orbits per day.

- eccentric anomaly E – Angle on orbital plane (focused in its centre) between orbited planet and a special point. This point lies on orbital plane on auxiliary circle with radius equal to semi-major axis (long half the ellipse) and on line that is both intersecting the satellite and is perpendicular to semi-minor axis (short half of the ellipse). $E \in \langle 0°,\ 360° \rangle$

Eccentric anomaly E can also be obtained using the Kepler's eq. (2.7),

$$E - e \cdot \sin E = M_e \tag{2.7}$$

where e is eccentricity and M_e mean anomaly. Finding the root of this equation is not trivial and requires the use of the Newton's approximation method.

Figure 2.4 illustrates the difference between mean anomaly M_e and true anomaly θ of a satellite on elliptic trajectory, indicated by arrows. Note the greater area swept by true anomaly near periapsis, as opposed to the uniformity of mean anomaly.

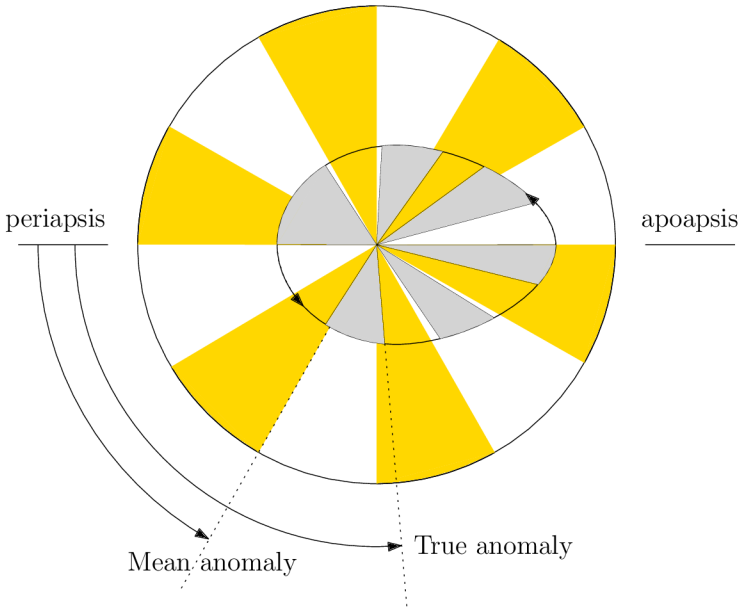

Figure 2.4: The difference between mean anomaly M_e and true anomaly θ.

Link to Epoch

Although we successfully defined the size and orientation of orbit and pinpointed the position of satellite, these parameters naturally change with time. The key missing link to reality is determining for what timestamp – epoch – are orbital elements valid. This topic will be explored in subsequent section 2.3.

2.3 Reference frames and Coordinate transformations

For projecting the orbit in three-dimensional space with Earth in the origin, inertial and fixed frames are used. Earth-centred inertial (ECI) frame is needed for describing and numerically predicting the movement of spacecraft using the state vector, whereas Earth-centred, Earth-fixed (ECEF) frame is useful for projecting satellites position onto rotating Earth's surface for a given Epoch. Note, that ECI and ECEF frames can be furthermore subdivided, as reference directions, original usage, or means of computation may slightly differ. This work will utilize True-Equator Mean-equinox ECI subframe (TEME), that is needed for working with TLE, as outlined in section 3.2.

Perifocal frame

This coordinate frame describes the orbit in two dimensions, in its "natural frame". This frame is centered at the focus point of the orbit, the planet. Its xy plane is the orbital plane and its x axis passes through satellite's periapsis. Perifocal frame is used as an inter-step in converting orbital elements into state vector.

Calculating the spacecraft's position and velocity vectors is performed as shown in matrix notation in eq. (2.8) and eq. (2.9),

$$\mathbf{r}_P = \frac{h^2}{\mu \cdot (1 + e \cdot \cos\theta)} \cdot \begin{Bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{Bmatrix} \tag{2.8}$$

$$\mathbf{v}_P = \frac{\mu}{h} \cdot \begin{Bmatrix} -\sin\theta \\ e + \cos\theta \\ 0 \end{Bmatrix} \tag{2.9}$$

where h represents specific angular momentum from eq. (2.4), e eccentricity and θ true anomaly from eq. (2.3). The Earth's gravitational parameter μ is also present. Note the extra arbitrary z coordinate, which is handy for subsequent operations and is equal to zero.

State vector

The state vector is composed of both position and velocity vector in 3-dimensional space, as outlined in section 2.1. For position, one of the ECI frames is generally used, positioning the spacecraft with xyz coordinates around a non-rotating origin in the centre of the Earth. Its x axis points in the direction of the Vernal equinox Υ, the xy plane is the Earth's equatorial plane and z plane its rotational axis. The positioning of Vernal equinox Υ is changed in 50-year intervals by about tenth of a degree.

Both components of a state vector can be described via eq. (2.10).

$$\mathbf{r}_S = \begin{Bmatrix} x \\ y \\ z \end{Bmatrix}, \mathbf{v}_S = \begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{Bmatrix} \tag{2.10}$$

Since predicting state vectors is the main concern of orbital mechanics, obtaining it from the six orbital elements first requires obtaining the position and velocity vectors in perifocal frame via eq. (2.8) and eq. (2.9). Note, that orbital elements that define the orbital's size, shape and spacecraft's position were needed.

As for the three Euler angles – elements describing the orientation of orbit – argument of periapsis ω, inclination i and RAAN Ω, we use them to rotate both perifocal vectors into ECI frame using zxz Euler transformation matrices in eq. (2.11), eq. (2.12) and eq. (2.13).

$$R_z(\omega) = \begin{bmatrix} \cos\omega & -\sin\omega & 0 \\ \sin\omega & \cos\omega & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

$$R_x(i) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos i & -\sin i \\ 0 & \sin i & \cos i \end{bmatrix} \tag{2.12}$$

$$R_z(\Omega) = \begin{bmatrix} \cos\Omega & -\sin\Omega & 0 \\ \sin\Omega & \cos\Omega & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.13}$$

Furthermore, all three separate coordinate transformations can be simplified into a single matrix T in eq. (2.14).

$$T_{P/S} = R_z(\omega) \cdot R_x(i) \cdot R_z(\Omega) = \\ \begin{bmatrix} \cos\Omega\cos\omega - \sin\Omega\cos i\sin\omega & -\sin\Omega\cos i\cos\omega - \cos\Omega\sin\omega & \sin\Omega\sin i \\ \cos\Omega\cos i\sin\omega + \sin\Omega\cos\omega & \cos\Omega\cos i\cos\omega - \sin\Omega\sin\omega & -\cos\Omega\sin i \\ \sin i\sin\omega & \sin i\cos\omega & \cos i \end{bmatrix} \tag{2.14}$$

Transformation of perifocal vectors r_P and v_P from eq. (2.8) and eq. (2.9) is achieved by multiplying both with transformation matrix $T_{P/S}$, resulting in eq. (2.15).

$$\begin{aligned} \mathbf{r}_S &= \mathbf{r}_P \cdot T_{P/S} \\ \mathbf{v}_S &= \mathbf{r}_P \cdot T_{P/S} \end{aligned} \tag{2.15}$$

Epoch and Earth's rotation

The main concern of conversion between ECI and Earth-centered, Earth-fixed system (ECEF) represented by latitude, longitude and height, is correction for rotation of the planet Earth. The reference moment in time is spacecraft's epoch, which is converted into Julian date, then into Greenwich mean sidereal time (GMST82) which itself is the desired angle of Earth's rotation.

- The spacecraft's epoch is composed of year, day and its fractional portion and is available alongside orbital elements in public databases. As mentioned in section 2.2, epoch determines for what timestamp are the elements valid.
- Julian date is the amount of days (with remainder fraction) that have passed since noon of Universal Time (UT1) on January 1st 4713 BC of the Julian Calendar. This point in time was chosen for mathematical convenience, as calculating with preceding years is simplified. Naturally, the transition of Julian to Gregorian calendar doesn't need to be accounted for. The Universal Time, historically known as Greenwich Mean Time (GMT) is similar to the familiar Universal Coordinated Time (UTC). In comparison, UTC differs negligibly from the UT1 by 'leap seconds', as the UT1 can vary due to Earth's rotation [18]. Note, that midnight in UT1 corresponds to a Julian date fraction of 0.5 [2]. For example, the widely-used reference epoch J2000 – the noon of January 1st 2000 – is Julian day 2,451,545.0.
- Greenwich mean sidereal time (GMST82) measures Earth's rotation with respect to distant stars and is defined as the hour angle between the Greenwich Meridian and the vernal equinox (the intersection of the planes of the earth's equator and the earth's orbit, the ecliptic) [31]. GMST82 is composed of the angle θ_{GMST82} in radians and its velocity $\dot{\theta}_{GMST82}$ in radians per day.

For simplicity, following equations will utilize certain time-related constants from the table 2.1.

constant	**value**	**unit**	**meaning**
J	2,451,545	days	J2000; days since 12:00 UTC, Jan 1st 4713 BC
D	36,525	days	approximated number of days in century
H	24	hours	hours in day
M	1440	minutes	minutes in day
S	86,400	seconds	seconds in day

Table 2.1: Useful time-related constants.

The general transition from epoch's year, day and time into Julian date j is performed as shown in eq. (2.16),

$$j = J + 0.5 + 365 \cdot y + \lfloor 0.25 \cdot y \rfloor - \lfloor 0.01 \cdot y \rfloor + \lfloor 0.0025 \cdot y \rfloor + l + d + \frac{h}{H} + \frac{m}{M} + \frac{s}{S} \quad (2.16)$$

where y is the measured year minus 2000. Also note the J2000 constant with half a day added, and constants H, M and S from table 2.1. Correction is necessary for leap years l. If y is a leap year, l will be -1 and 0 if otherwise. Variable d is the number of day in a year, regardless of month. Variables h, m and s mean hours, minutes and seconds respectively, in UTC time.

Since fraction of the day f is a known value included in epoch, eq. (2.16) can be furthermore simplified by eq. (2.17).

$$j = J + 0.5 + 365 \cdot y + \lfloor 0.25 \cdot y \rfloor - \lfloor 0.01 \cdot y \rfloor + \lfloor 0.0025 \cdot y \rfloor + l + d + f \tag{2.17}$$

Transformation from Julian date to GMST82 is done via converting Julian days j from eq. (2.17) into centuries c, that have passed since the J2000 timestamp, as shown in eq. (2.18),

$$c = \frac{(j - J)}{D} \tag{2.18}$$

where J and D are taken from table 2.1, and finally utilizing eq. (2.17) and eq. (2.18) in transformation to inter-step x and then to the rotation angle θ_{GMST82} (in eq. (2.19)) and to its velocity $\dot{\theta}_{GMST82}$ (in eq. (2.20)) [14],

$$\begin{aligned} x &= (((-6.2 \cdot 10^{-6} \cdot c + 0.093104) \cdot c + 8{,}640{,}184.812866) \cdot c + 67{,}310.54841) \cdot c \\ \theta_{GMST82} &= \left(\text{frac}(j) + \text{frac}\left(\frac{x}{S}\right) \right) \cdot 2\pi \end{aligned} \tag{2.19}$$

$$\dot{\theta}_{GMST82} = \left(1 + \frac{(-6.2 \cdot 10^{-6} \cdot 3 \cdot c + 0.093104 \cdot 2) \cdot c + 8{,}640{,}184.812866}{S \cdot D} \right) \cdot 2\pi \tag{2.20}$$

where function $\text{frac}(x)$ is defined as $\text{frac}(x) = x - \lfloor x \rfloor$. The other constants in the equation cannot be easily explained and are beyond the scope of this work. The definitions stem from IAU and are generally taken as convention [4][30][31].

Ground tracking

Transforming ECI state vector into ECEF spherical reference frame of latitude ϕ, longitude λ and height h (outlined in section 2.1) is necessity for ground-tracking a satellite – monitoring its position and path above Earth's surface. Although this task would be simple with a perfect sphere, Earth is an ellipsoid, thanks to its polar rotation and Moon's gravity, that is bulged on the equator by 21.38 km. This complicates calculations for precise measurement.

Throughout the time, universal standard used for on-Earth position measurements, the World Geodetic System (WGS84), was created. It belongs to the family of ECEF coordinate systems and is widely used in geography, cartography and most notably in navigation using GPS satellites. WGS84 also defines our planet as an ellipsoid – radii, distances and various other properties.

The point on Earth's surface above which the satellite is in zenith is referred to as a sub-point. On a perfect sphere, the Earth's centre, satellite's sub-point and satellite's position would be in a plane and alongside Earth's equatorial plane would form an angle called geocentric latitude ϕ'. As the Earth is bulged in reality, satellite's sub-point will be shifted slightly towards the Equator, forming a new angle, geodetic latitude ϕ, as illustrated in fig. 2.5 with satellite in local zenith.

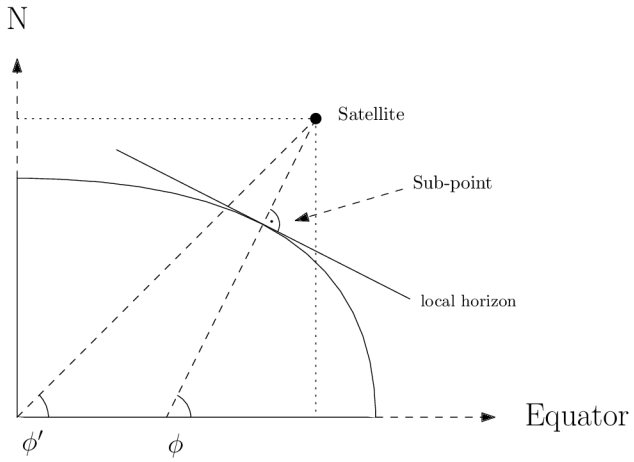

Figure 2.5: Difference between geocentric and geodetic latitudes.

Although the error is eliminated when measuring above either poles or the equator, the error can be as high as 3° at 45° geocentric latitude, which is for most cases impractical [14].

Calculating the longitude λ is not affected by oblateness of the Earth, hence simple eq. (2.21),

$$\lambda = \arctan\left(\frac{\mathbf{r}_y}{\mathbf{r}_x}\right) - \theta_{GMST82} \tag{2.21}$$

where $\mathbf{r}_x$ and $\mathbf{r}_y$ are x and y coordinates of ECI position state vector from eq. (2.10) and θ_{GMST82} is the angle by which the Earth has rotated, as calculated in eq. (2.19).

The geodetic latitude, however, cannot be calculated analytically and can only be approximated. First, the Earth's first eccentricity squared is calculated from the inverse flattening f of Earth's WGS84 ellipsoid [27], as in eq. (2.22).

$$e^2 = 2f - f^2 \tag{2.22}$$

Geodetic latitude ϕ is then obtained by iterating over the following eq. (2.23), eq. (2.24) and eq. (2.25),

$$\phi_n = \phi \tag{2.23}$$

$$C = \frac{1}{\sqrt{1 - e^2 \cdot \sin^2(\phi_n)}} \tag{2.24}$$

$$\phi = \arctan\left(\frac{\mathbf{r}_z + C \cdot e^2 \cdot R_e \cdot \sin(\phi_n)}{\sqrt{\mathbf{r}_x^2 + \mathbf{r}_y^2}}\right) \tag{2.25}$$

in which Earth's first eccentricity squared e^2 is obtained via eq. (2.22), then $\mathbf{r}_x$, $\mathbf{r}_y$ and $\mathbf{r}_z$ are x, y and z coordinates of ECI position state vector from eq. (2.10) and R_e is the

equatorial Earth's radius of 6378.137 km. The iteration is to be repeated until the difference $|\phi_n - \phi|$ is negligible [14].

Finally the height h of the satellite above surface is calculated as in eq. (2.26),

$$h = \frac{\sqrt{\mathbf{r}_x^2 + \mathbf{r}_y^2}}{\cos(\phi)} - R_e \cdot C \tag{2.26}$$

which uses variables as explained in the paragraph above. Note, that simple magnitude of ECI position state vector, that would normally measure object's distance from origin, won't suffice.

Conversion summary

Determining satellite's position on Earth from given orbital parameters is not trivial and was customized for data provided in Two Line Element (TLE) sets. Although results can generally be achieved in multiple ways, processing data from TLE has specific key principles, as they are accustomed for use in SGP family of perturbation models.

First, position and velocity vector in perifocal frame is formed using eq. (2.8) and eq. (2.9) respectively. Perifocal orbit has accurate dimensions, that need to be rotated and angled to ECI (TEME) frame of planet Earth via eq. (2.10), resulting in new position and velocity vector – collectively named state vector. State vectors can be predicted by simulation, or can further be transformed into ECEF (WGS84) frame of familiar latitude, longitude and height via eq. (2.25), eq. (2.21) and eq. (2.26) respectively.

2.4 Orbital perturbations

Predicting position of a satellite along with proper simulation of its trajectory is crucial in any space application. Although it is possible to determine and predict object's position with just orbital elements, this approach is not ideal for longer simulations, as not all effects can be expressed analytically. Real world physics have to account for gravitational forces of the Sun, Moon and the planets, obliquity of planet Earth, atmospheric drag, solar radiation etc. The small force of all these perturbations is combined into xyz acceleration vector, that affects the velocity and subsequently the position (eg. state vector) of orbiting object in delta time. These forces can sometimes be exploited to break free of spacecraft's Δv budget, as in the case of solar sailing [11].

The basics for these computations and predicting spacecraft's position vector $\mathbf{r}$ from eq. (2.10) involve the two-body equation for relative motion in eq. (2.27),

$$\ddot{\mathbf{r}} = -\mu_e \cdot \frac{\mathbf{r}}{r^3} \tag{2.27}$$

where μ_e is Earth's standard gravitational parameter, a constant that approximates to $398{,}600\,\mathrm{km}^3 \cdot \mathrm{s}^{-2}$ and $\mathbf{r}$ is spacecraft's distance from Earth's centre. Variable r is the magnitude of vector $\mathbf{r}$.

The practical application of eq. (2.27) in state vector prediction can be written as in eq. (2.28) and eq. (2.29),

$$\mathbf{a} = -\mu_e \cdot \frac{\mathbf{r}}{r^3} + \mathbf{p} \tag{2.28}$$

$$\begin{Bmatrix} \dot{\mathbf{r}} \\ \dot{\mathbf{v}} \end{Bmatrix} = \begin{Bmatrix} \mathbf{v} \\ \mathbf{a} \end{Bmatrix} = \begin{Bmatrix} \mathbf{v} \\ -\mu_e \cdot \dfrac{\mathbf{r}}{r^3} + \mathbf{p} \end{Bmatrix} \tag{2.29}$$

where changes to state vector in time are made by calculating the acceleration $\mathbf{a}$ for current position (in t_0) and applying this change to velocity (in t_0) and subsequently, the changed velocity to initial position, creating a new position in delta time (as explained in section 2.1). Note, that the additional vector $\mathbf{p}$ represents the sum of various perturbations described above and adds to the acceleration vector. This second order differential computation can be solved for example with (Leonhard) Euler's or commonly Runge-Kutta methods, which will be described in subsequent section 3.3.

Atmospheric drag

Although only a negligible fraction (0.0001%) of the Earth's atmosphere is above the 100 km line separating space from our planet, atmospheric drag models are critical for any satellite in LEO and in heights up to 1000 km [29]. As satellites move with high average orbital speeds of $29.78\,\mathrm{km}\cdot\mathrm{s}^{-1}$, exerted drag becomes significant and can deorbit satellite in 180 km orbit in matter of hours [17]. To put things into perspective, the reentry height for Apollo 13, as well as other missions including the Space Shuttle, was around 121 km. Accurate

modelling is more difficult than with other perturbation forces, due to different shapes and materials used by spacecrafts and due to atmospheric densities that depend on external factors.

The simulation will employ a model U.S. Standard Atmosphere 1976 (USSA76), which uses table values and exponential interpolation to find atmospheric density ρ up to 1000 km [19]. The model implemented in the code is taken from [7]. The following text will assume ρ as a function of altitude.

This chapter outlines two main methods for working with drag, either with the use of ballistic coefficient, or with so-called B-star. The ballistic coefficient is commonly used in aerodynamics and ballistics, whereas the B-star is more precise and customized for space applications. Its usage is however limited to SGP family of perturbation models or specialized programs. Both are provided in TLE sets alongside orbital elements.

The ballistic coefficient B measures how fast will object decelerate due to drag and can be distributed alongside orbital elements. High value implies low air resistance is to be expected and vice versa. It composes of core values, such as satellite's mass m, cross-sectional area A and drag coefficient C_d, as outlined in eq. (2.30).

$$B = \frac{C_d \cdot A}{m} \tag{2.30}$$

Drag is also measured against a rotating atmosphere, hence the cross product of Earth's angular velocity ω_e (in radians per second) and spacecraft's position vector $\mathbf{r}$ is subtracted from its velocity vector $\mathbf{v}$, forming a relative velocity vector $\mathbf{v}_{rel}$ in eq. (2.31).

$$\mathbf{v}_{rel} = \mathbf{v} - \omega_e \times \mathbf{r} \tag{2.31}$$

Note, that Earth's rotation speed is precisely 360.9856 degrees per 24 hours. Calculating the final drag acceleration vector $\mathbf{p}$ is done with the use of USSA76 function for atmospheric density ρ and with ballistic coefficient and relative velocity from eq. (2.30) and eq. (2.31) respectively, forming the modified drag eq. (2.32).

$$\mathbf{p}_{B-drag} = -\frac{1}{2} \cdot \mathbf{v}_{rel}^2 \cdot B \cdot \rho \cdot \hat{\mathbf{v}}_{rel} \tag{2.32}$$

The second approach is with the use of B-star (B*). It also estimates the effect of drag onto satellite, however it is provided (again, with orbital elements) as an average of multiple observations and incorporates additional corrections and adjustments, as for example the drag varies with orientation of the spacecraft [11]. Single observation is defined in eq. (2.33),

$$B^* = \frac{\rho_0}{2} \cdot \frac{C_d \cdot A}{m} = \frac{\rho_0}{2} \cdot B \tag{2.33}$$

where ballistic coefficient B, cross-sectional area A and drag coefficient C_d are known variables from eq. (2.30).
Note the use of reference air density ρ_0 of $2.461 \cdot 10^{-5}\,\mathrm{kg} \cdot \mathrm{m}^{-2} \cdot \text{Earth radii}^{-1}$. Calculating the final drag acceleration vector $\mathbf{p}$ is done via eq. (2.34),

$$\mathbf{p}_{B^*-drag} = \frac{\rho}{\rho_0} \cdot B^* \cdot \mathbf{v}_{rel}^2 \tag{2.34}$$

where $\mathbf{v}_{rel}$ is spacecraft's relative velocity from eq. (2.31), B^* is either taken from an average of multiple measurements or from eq. (2.33), ρ is the atmospheric density relative to altitude USSA76 and ρ_0 again the reference density of $2.461 \cdot 10^{-5}\,\mathrm{kg} \cdot \mathrm{m}^{-2} \cdot \text{Earth radii}^{-1}$ [12].

Gravity of other bodies

One of the other major perturbing forces is the gravitational effect of other bodies throughout the Solar System, with most notable examples being the gravity of our Moon and the Sun (and sometimes Jupiter). General rule implies, that the higher the orbit is, the greater effect will those other gravitational forces have, as Earth's gravity weakens. This is especially important for Medium (MEO) and Geostationary Earth orbits (GEO), where atmospheric drag is negligible to non-existent. Geostationary orbits are especially sensitive as even the smallest of deviations can render orbit no longer stationary. However for applications in LEO, the effect of other planets often is often omitted.

Furthermore, the gravity of the Moon also affects the Earth itself in the form of solid and ocean tides, forcing advanced perturbation models to deal with shifts in the Earth's gravity, as the centre of mass changes. Whereas ocean tides are well known, as the Moon's gravity causes sea levels to rise, solid tides are displacements of solid Earth's surface, which can be as high as 20 cm [32]. However the effects exerted on spacecraft are relatively small and naturally fade with growing distance from Earth.

The gravitational effect of single third-body object can be universally expressed by eq. (2.35),

$$\mathbf{p}_3 = \mu_o \cdot \frac{\mathbf{r}_o - \mathbf{r}_s}{|r_o - r_s|^3} - \mu_o \cdot \frac{\mathbf{r}_o - \mathbf{r}_e}{|r_o - r_e|^3} \tag{2.35}$$

where μ_o is the standard gravitational parameter of any celestial object o, then $\mathbf{r}_o$, $\mathbf{r}_s$ and $\mathbf{r}_e$ are the position vectors for third-body object, affected spacecraft and the Earth respectively, with r_o, r_s and r_e being the magnitudes of the vectors. The first, left part of eq. (2.35) represents satellite's gravitational attraction toward the object, whereas the right, second part subtracts object's gravitational pull exerted on the Earth, as to the primary body, the spacecraft orbits. Since in this case the Earth is also origin of our ECI coordinate frame, setting $\mathbf{r}_e = \{0, 0, 0\}$, eq. (2.35) can be simplified into eq. (2.36),

$$\mathbf{p}_3 = \mu_o \cdot \left(\frac{\mathbf{r}_o - \mathbf{r}_s}{|r_o - r_s|^3} - \frac{\mathbf{r}_o}{r_o^3} \right) \tag{2.36}$$

with known variables from eq. (2.35). Each such effect of any third-body object is then simply added to eq. (2.29).

The general μ_o from eq. (2.36) representing any object, is in this case subsidized by the gravitational parameters of various bodies, as listed in table 2.2.

Body	value $[\mathrm{km}^3 \cdot \mathrm{s}^{-2}]$	Body	value $[\mathrm{km}^3 \cdot \mathrm{s}^{-2}]$
Earth	$3.986004418 \cdot 10^5$	Mars	$4.282837 \cdot 10^4$
Sun	$1.32712440018 \cdot 10^{11}$	Jupiter	$1.26686534 \cdot 10^8$
Moon	$4.9028695 \cdot 10^3$	Saturn	$3.7931187 \cdot 10^7$
Mercury	$2.2032 \cdot 10^4$	Uranus	$5.793939 \cdot 10^6$
Venus	$3.24859 \cdot 10^5$	Neptune	$6.836529 \cdot 10^6$

Table 2.2: Standard gravitational parameters μ of various bodies.

Earth obliquity

As mentioned in section 2.3, Earth is not perfectly spherical, but rather oblate on the equator, as a consequence of centrifugal forces of planet, rotating around its polar axis. Moon's and Sun's gravity also played certain role, shaping the difference between polar and equatorial radii to 21.38 km. This slight deviation significantly affects satellite's orbits, as gravitational field varies: both poles have slightly stronger gravity, as they are closer to the center of mass, whereas higher mass near equators generally increases equatorial gravitation. Note, that such changes in gravity are symmetrical (zonal). Such effects are traditionally not expressed mathematically, but rather through observations of satellite's movements, forming a set of six dimensionless constants – zonal harmonics J (see table 2.3), that divide Earth into slices by latitude. The higher the subscript number, the smaller the slices are.

zonal harmonic	value	zonal harmonic	value
J_2	$1082.63 \cdot 10^{-6}$	J_5	$-0.15 \cdot 10^{-6}$
J_3	$-2.53266 \cdot 10^{-6}$	J_6	$0.54067 \cdot 10^{-6}$
J_4	$-1.61962 \cdot 10^{-6}$	J_7	$0.35236 \cdot 10^{-6}$

Table 2.3: List of zonal harmonics J.

Note, that J_2 has by far the greatest effect, as others are at least three orders of magnitude smaller. Zonal harmonics are also unique for every planet and affect gravitational perturbations in changes of orbit's latitude.

Although effects of Earth's oblateness are the most notable, gravity can also vary in specific regions. For such cases, sectorial and tesseral harmonics are introduced. Sectorial divide the planet by longitude and are caused by general asymmetric mass distribution. On the other hand, tesseral divide by both latitude and longitude into a grid-like pattern, that can furthermore represent mountain ranges, ocean trenches or thickness of crust. Note, that modelling these forces is the matter of specialized perturbation systems with great accurateness. For example, the reference SGP4 model only utilizes zonal harmonics J_2 to J_4 [25].

As zonal harmonic J_2 is by far the most important, its perturbing gravitational forces can be expressed by eq. (2.37), eq. (2.38) and eq. (2.39) derived from [13],

$$\frac{\partial \Phi}{\partial x} = -\frac{3}{2} \cdot \frac{J_2 \cdot \mu_e}{r^2} \cdot \left(\frac{R_e}{r}\right)^2 \cdot \frac{x}{r} \cdot \left[5 \cdot \left(\frac{z}{r}\right)^2 - 1\right] \tag{2.37}$$

$$\frac{\partial \Phi}{\partial y} = -\frac{3}{2} \cdot \frac{J_2 \cdot \mu_e}{r^2} \cdot \left(\frac{R_e}{r}\right)^2 \cdot \frac{y}{r} \cdot \left[5 \cdot \left(\frac{z}{r}\right)^2 - 1\right] \tag{2.38}$$

$$\frac{\partial \Phi}{\partial z} = -\frac{3}{2} \cdot \frac{J_2 \cdot \mu_e}{r^2} \cdot \left(\frac{R_e}{r}\right)^2 \cdot \frac{z}{r} \cdot \left[5 \cdot \left(\frac{z}{r}\right)^2 - 3\right] \tag{2.39}$$

where effects on x, y and z coordinates of familiar perturbation vector $\mathbf{p}$ are studied separately. Zonal harmonic J_2 from table 2.3 is used alongside Earth's standard gravitational parameter μ_e, magnitude r of spacecraft's position vector $\mathbf{r}$ and Earth's mean radius R_e. The Φ here represents general gravitational perturbations of non-spherical Earth. The eq. (2.37), eq. (2.38) and eq. (2.39) with their respective variables may be shortened into eq. (2.40) [7],

$$\mathbf{p}_{J2} = \frac{3}{2} \cdot \frac{J_2 \cdot \mu_e \cdot R_e^2}{r^4} \begin{Bmatrix} \left(5 \cdot \frac{z^2}{r^2} - 1\right) \frac{x}{r} \\ \left(5 \cdot \frac{z^2}{r^2} - 1\right) \frac{y}{r} \\ \left(5 \cdot \frac{z^2}{r^2} - 3\right) \frac{z}{r} \end{Bmatrix} \tag{2.40}$$

where lines of the matrix represent different solutions for x, y and z coordinates of spacecraft's position vector $\mathbf{r}$. The resulting perturbation vector $\mathbf{p}_{J2}$ summarizes effects of J_2. Higher zonal harmonics are, apart for changes in coefficients, calculated similarly and utilize the same variables as in eq. (2.40). This work also utilizes J_3 and J_4, as defined below in eq. (2.41) and eq. (2.42). For computing other zonal harmonics, see [26].

$$\mathbf{p}_{J3} = -\frac{1}{2} \cdot \frac{J_3 \cdot \mu_e \cdot R_e^3}{r^5} \begin{Bmatrix} 5\left(7\frac{z^3}{r^3} - 3\frac{z}{r}\right) \frac{x}{r} \\ 5\left(7\frac{z^3}{r^3} - 3\frac{z}{r}\right) \frac{y}{r} \\ 3\left(10\frac{z^2}{r^2} - \frac{35}{3}\frac{z^4}{r^4} - 1\right) \end{Bmatrix} \tag{2.41}$$

$$\mathbf{p}_{J4} = -\frac{5}{8} \cdot \frac{J_4 \cdot \mu_e \cdot R_e^4}{r^6} \left\{ \begin{array}{c} \left(3 - 42\frac{z^2}{r^2} + 63\frac{z^4}{r^4}\right)\frac{x}{r} \\ \left(3 - 42\frac{z^2}{r^2} + 63\frac{z^4}{r^4}\right)\frac{y}{r} \\ -\left(15 - 70\frac{z^2}{r^2} + 63\frac{z^4}{r^4}\right)\frac{z}{r} \end{array} \right\} \tag{2.42}$$

Solar radiation pressure

All orbiting objects in our solar system are subject to solar radiation pressure (SRP) – particles of photons hitting the surfaces of spacecrafts. Although photons are massless, their energy and momentum are not [7]. At the distance of Earth, the pressure P_{sr}, depicted in eq. (2.43) and calculated with Solar constant S and speed of light c amounts to

$$P_{sr} = \frac{S}{c} = 4.56 \cdot 10^{-6} \tag{2.43}$$

in $\mathrm{N} \cdot \mathrm{m}^{-2}$, or about $4.56\,\mu\mathrm{Pa}$. In contrast with atmospheric drag, which weakens with altitude, effect of both forces evens out at around 625 km [7]. Medium, geostationary and higher orbits around the Earth are generally impacted the most.

The direction of SRP is opposite to the vector spacecraft-to-Sun, in direction of travelling particles. In reality, Earth-to-Sun vector is also commonly used, as the Sun's parallax is negligible, even for spacecrafts in high orbits. The SRP naturally doesn't work, when satellite is behind Earth's shadow. Acceleration (perturbation) due to SRP can be calculated via eq. (2.44),

$$\mathbf{p}_{sr} = -v \cdot \frac{\mathbf{r}_\odot - \mathbf{r}}{|r_\odot - r|^3} \cdot P_{sr} \cdot \frac{C_r \cdot A}{m} \tag{2.44}$$

where v is so-called shadow function, being 0 when the spacecraft hides behind the Earth and 1 if otherwise. Then the first fraction represents vector Spacecraft-to-Sun (often conveniently replaced by Earth-to-Sun) and is supplemented with minus indicating opposite direction. P_{sr} from eq. (2.43) is solar pressure. The coefficient of reflectivity C_r lies between value 1 for black-coated satellites, absorbing all photons, and 2, being for white-coated satellites, mirroring photons back, doubling the force. A is satellite's absorbing (or cross-sectional) area and m its total mass.

Note, that the last fraction, containing details about the observed satellite, cannot be easily deduced. Absorbing area A is often simplified to cannonball model [9] of perfect sphere, however satellite's mass and colour are usually not publicly available. Simplified general perturbation models (such as SGP4) do not even model such effects, resulting in various workarounds with estimating SRP coefficients based on months of TLE data, by Chao, Campbell [5]. Some special satellites with big absorbing areas employ SRP as a form of controlled propulsion, called solar sailing [11].

Furthermore, as SRP can be reflected from the satellite, so it can from any planet, including the Earth. This effect is called Earth radiation pressure (ERP) or Earth's Albedo. Calculating this phenomena is beyond the scope of this work, as the amount of reflected particles depends on properties of surfaces, that particles are reflected by. Clouds have excellent reflecting properties alongside snow, whereas oceans or forests have not [23]. In reality, the ERP of a specific region is measured and modelled with specialized weather satellites.

Variation of orbital elements

So far, previous chapters covered the effects of various perturbation forces to acceleration of the spacecraft in ECI frame. As analytical orbit propagation calculates future position in orbital elements, their variation is to be expected. Calculation is more tricky, as instead of simple acceleration vector, the effects of drag, SRP or Earth's obliquity are expressed through set of equations, each unique to the affected orbital element. Although arithmetical simulation of this work includes only nodal and apsidal precessions (i.e. effects of Earth's obliquity), other perturbative effects are explained in [22].

In an ideal orbit without any disturbances, only the angle mean anomaly M_e would change, as the spacecraft changes its position on elliptic orbit. However due to Earth's obliquity, the satellite is shifting rapidly. Average differences (after one orbit) of semi-major axis a, eccentricity e and inclination i are equal to zero, meaning, that these parameters don't change at all. The only elements that change are mean anomaly M_e, right ascension of the ascending node (RAAN) Ω and argument of periapsis ω. This is usually referred to as nodal and apsidal precession respectively.

The changes of RAAN Ω, argument of periapsis ω and mean anomaly M_e in predetermined time step can be expressed via eq. (2.45), eq. (2.46) and eq. (2.47) respectively,

$$\dot{\Omega}_{J_2} = -\frac{3}{2} \cdot n \cdot J_2 \cdot \left(\frac{R_e}{P}\right)^2 \cdot \cos i \tag{2.45}$$

$$\dot{\omega}_{J_2} = -\frac{3}{2} \cdot n \cdot J_2 \cdot \left(\frac{R_e}{P}\right)^2 \cdot \left(2 - \frac{5}{2} \cdot \sin^2 i\right) \tag{2.46}$$

$$\dot{M}_{e\,J_2} = n + \frac{3}{2} \cdot n \cdot J_2 \cdot \left(\frac{R_e}{P}\right)^2 \cdot \left(1 - \frac{3}{2} \cdot \sin^2 i\right) \cdot \sqrt{1 - e^2} \tag{2.47}$$

where n is satellite's mean motion in $\mathrm{rad} \cdot \mathrm{s}^{-1}$, zonal harmonic J_2 is taken from table 2.3, R_e is equatorial radius of Earth, i is inclination and e is eccentricity of the trajectory. The denominator P stands for periapsis from eq. (2.5).

Perturbations summary

The determination of orbital perturbations is a complex topic, that involves much more factors, than those outlined in previous chapters. Alongside the mentioned atmospheric drag, obliquity of planet Earth, lunar gravity with solid and ocean tides or gravity of other bodies in general, solar radiation pressure and Earth radiation pressure, LEO's are also subject to the effects of general relativity or solar activity, alongside the unpredictable micrometeoroid impacts. Modelling such effects is however beyond the scope of this work.

Finally, the basic numerical approach from eq. (2.28) can be supplemented with

- eq. (2.34) for atmospheric drag ($\mathbf{p}_{B*-drag}$),
- eq. (2.36) for third-body gravitational perturbations ($\mathbf{p}_3$)
- eq. (2.40) for gravitational effects of oblate Earth ($\mathbf{p}_{J2}$)
- eq. (2.44) for solar radiation pressure, ($\mathbf{p}_{sr}$),

forming a more accurate solution of spacecraft's acceleration vector $\mathbf{a}$ for any given moment in eq. (2.48).

$$\mathbf{a} = -\mu_e \cdot \frac{\mathbf{r}}{r^3} + \mathbf{p}_{B*-drag} + \mathbf{p}_3 + \mathbf{p}_{J2} + \mathbf{p}_{sr} \tag{2.48}$$

Chapter 3

Design and Implementation

This chapter dives into the technical detail of simulator's implementation and its design features. The simulator (referred to as LEOSIM) is a standalone desktop application, which extracts data from various public databases and simulates movement of object in LEO with numerical methods. LEOSIM is subdivided into two parts – user interface, written in language Python, namely using graphical framework PyQt and into the computational part, written in language C++ for greater efficiency.

As this chapter references source files of LEOSIM, their basic structure of dependencies is shown in fig. 3.1, with description in table 3.1 for convenience. For greater detail about source files and various functions, see the attached Doxygen documentation.

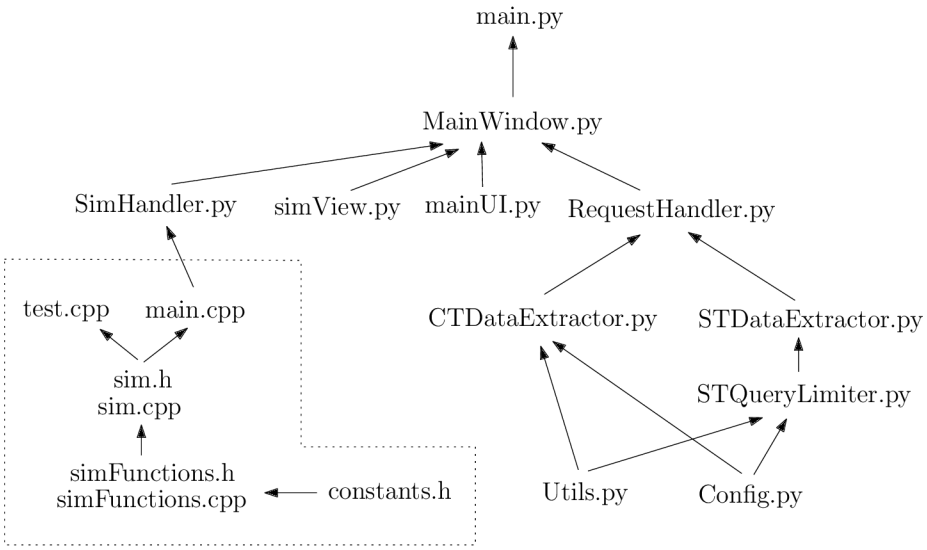

Figure 3.1: Structure of LEOSIM.

File	Purpose
`main.py`	Entry point of the application
`MainWindow.py`	Connects UI elements of the main window, its functionality and communication with other modules
`RequestHandler.py`	Module handles and provides common interface for all web database queries
`mainUI.py`	Module implements the UI, alongside its layout
`simView.py`	Module provides interactive graphical interface in PyQtGraph
`SimHandler.py`	Module communicates with the simulation model in C++
`CTDataExtractor.py`	Handles communication with the Celestrak database
`STDataExtractor.py`	Handles communication with the Space-Track database
`STQueryLimiter.py`	Implements query limiter for Space-Track API
`Utils.py`	Defines various utility classes, mainly Satellite and Epoch
`Config.py`	Contains base URLs and authentication credentials for Space-Track database
`main.cpp`	Main simulator file, that handles communication with Python interface
`sim.cpp`	Transforms specific simulation procedures to user-friendly form
`simFunctions.cpp`	Implements all conversion and propagation equations
`constants.h`	Provides physical constants for the equations
`test.cpp`	BOOST tests for comparing and validating the model

Table 3.1: Source files with their purpose.

3.1 User Interface Architecture

This chapter briefly summarizes the technical implementation of LEOSIM's user interface (UI) and its capabilities. The standalone graphical program is written using the cross-platform GUI toolkit Qt for Python, namely on version PyQt-6[1]. Layout was designed in PyQt-6 inbuilt designer app and transformed into Python code by PyQt-6 UI code generator. The UI is divided into two main parts: the interactive Earth with orbits of chosen satellites and the part for user inputs. The fig. 3.2, taken from LEOSIM, illustrates projecting the orbit of the ISS and AISSAT 1.

The table of satellites in the middle is filled with search results from the input field on the right side. Objects can be imported either via name, NORAD ID, date, group or filename. Different search methods import objects from different databases, as described in subsequent section 3.2. This also means, that objects imported in two line element (TLE) format will

[1] Avaliable from: https://wiki.python.org/moin/PyQt.

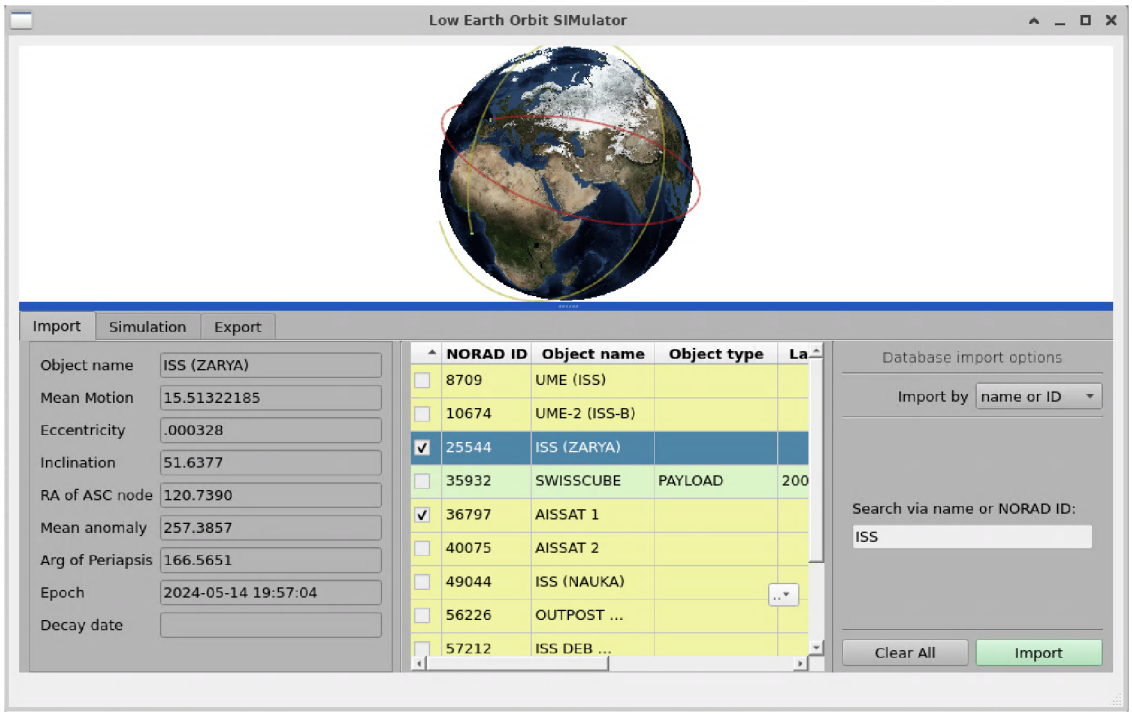

Figure 3.2: UI of LEOSIM.

miss statistical information and will be marked yellow, whereas complete objects will be green. Additionally, decayed objects, will be marked as red. Importing custom values from file is possible. In that case, both TLEs and 3LEs are accepted.

Upon selecting a satellite, its orbital parameters are shown on the left and its future positions are plotted, forming a single simulated orbit with current position marked as a small green point. Currently selected satellite has a red orbit, however others can be checked via the check box, enabling them to stay plotted, with yellow orbits. The table also provides a menu (two-dot button), which enables filtering based on object type or autocompleting satellite's info.

Selecting any satellite automatically fills its detailed info in the left part of the UI. Latitude and longitude can be found under the `simulation` tab. The background is white for illustrative purposes.

Whereas the `import` tab retrieves data from databases, the `simulation` tab enables predicting their movement and the `export` tab is used for exports. In the `simulation` mode, orbits of selected satellites can be propagated by fixed time step.

LEOSIM's graphical implementation is implemented in the `MainWindow.py` and `mainUi.py` source files.

3.2 Obtaining Satellite Data

This chapter describes the means of retrieving publicly available satellite data. Celestrak and Space-Track databases store TLE values, that represent basic information about object on LEO. Space-Track can further encode TLEs into XML format and provide additional details.

Two-line element set (TLE)

One of the ways to represent orbital data is with Two-line element sets (TLE). TLE is a standardized format from 1970's that encodes orbital parameters and a reference epoch (recall section 2.3) of their measurement into two short lines of 69 characters each. It also provides additional parameters for orbit propagation, such as B* (from section 2.4) and various statistical data. TLEs are created from measurements (and also using the SGP4/SDP4 model [12]) by Joint Space Operations Center, which is operated by US Air Force Space Command. Frequency of observations differ: active spacecrafts or objects affected by high drag can have their measurements updated several times a day, whereas space debris in higher orbits are usually updated only once or twice per week, unless there is a risk of collision with operational satellite [15]. Accuracy of measured TLE is dependant on object's altitude, its maneuvers, number of observations, that actually form the position and various other factors. The error therefore usually ranges from 1 to 2.5 km.

Although the format is largely standardized, some measured parameters, that happened to be equal to zero, may have been altogether omitted in some historic data. This may cause problems in the SGP4 models and in various fixed-parsing programs.

TLE's are designed to be used only with simplified perturbations models (such as SGP4), as the calculated orbital parameters are stored in a True-Equator Mean-Equinox coordinate frame (TEME), a subclass of ECI frames. This specific system varies slightly in how it handles the reference directions of Vernal equinox Υ (recall fig. 2.3) and equator plane, which is perpendicular to celestial ephemeris pole. "True" in this context implies, that all oscillations are accounted for, whereas "Mean" implies, that they are ignored. The original intent behind TEME was to provide an efficient, if approximate, coordinate system for various analytical theories [30], which however complicates its practical usage.

The second factor, which significantly limits TLE's use elsewhere, is that it contains mean orbital elements, created by removing periodic variations (such as gravity or drag effects) in a specific way [1]. In order to avoid large errors, these variations must be reconstructed (by the model) with exactly the same procedures, as when they were removed by the NORAD [12], when creating TLEs with SGP4 model.

In conjunction with SGP4, TLE's are intended and commonly used for quick analysis of potential collision risks, efficient monitoring of debris or for communication with satellites, using special tracking antennas. Note, that TLE's are issued only for Earth-orbiting objects (Moon excluded). Optionally, third line with object's name might be introduced, as creating Three-line element (3LE). In such cases, this line (shown below) is put before the other two:

```
ISS (ZARYA)
```

however TLE data is usually propagated as follows [6]:

```
1 25544U 98067A   24084.84536422  .00034327  00000-0  61923-3 0  9995
2 25544  51.6409  13.4998 0004450   2.7955  56.5150 15.49418300445462
```

The first line contains satellite's identification, international designation, timestamp (epoch), various statistical information and B* drag value. The second line consists mostly of various orbital parameters. For detailed overview of the values transmitted in TLE, see appendix A.

The format also features simple modulo check, that lowers the chance of distorted data being used. The last number in each of the two lines should correspond to modulo 10 of the sum of character's ASCII values in each line. This provides 90% detection rate for randomly distributed errors [30].

Celestrak

`celestrak.org` is a public sci-technical website with its roots in 1985, which provides popular science articles as well as professional literature on the topics of satellites in space. More importantly, it also contains a database of TLE, optionally 3LE data of all tracked objects. The newest TLE's of specified catalog number (CATNR) are retrieved directly via simple HTTP request [16]:

```
https://celestrak.org/NORAD/elements/gp.php?CATNR=25544&FORMAT=TLE
```

Requests that return multiple objects can be based on satellite's names or their affiliation to certain group (i.e. Starlink). Queries are not limited by rate. Getting historical data is only partially possible. Communication with the website in LEOSIM is implemented in the `CTDataExtractor.py` file.

Space-Track

`space-track.org` is a public website specialized for providing TLE's, as well as other Earth-orbit situational awareness data. The service is operated by U.S. Department of Defense and tracks more than 27 000 orbiting objects. Each object's data can be accessed in TLE, however responses in Extensible Markup Language (XML) format carry additional data, including type of object (i.e. payload, debris), code of launching country, launch date, distance of both apsides (recall section 2.2) and more, as shown in appendix B. The tradeoff is, that XML responses take incomparably more time to process than TLE and are not suitable for large chunks of objects. Getting archival measurements is however possible.

The queries are formed and accessed through application programming interface (API) with detailed parameters and filters. Their rate is however limited to 30 queries per minute and 300 requests per hour [28] and has to monitored by the `STQueryLimiter.py` file. Communication is implemented in the STDataExtractor.py file. Note, that user registration is compulsory. Accessing the website and authenticating is achieved with a testing account.

Skyfield

Skyfield is a library for Python, which computes positions for the stars, planets, and satellites in orbit around the Earth.[2] This work utilizes its API only for queries about the positions of all 8 planets of our solar system and also the Moon and the Sun. Positions are returned for given epoch as a xyz vector measured from the middle of Solar System, according to Barycentric Celestial Reference System (BCRS). Subtracting Earth's position from this vector then results in a position relative to Earth. Data from Skyfield is used in orbit propagation for calculating perturbation forces of third-body objects (recall section 2.4).

Handling requests

As mentioned in section 3.1, satellites in LEOSIM are obtained either by their name, NORAD ID, group, with general search by date interval, or with filename. Since both Celestrak and Space-Track websites have unique capabilities, both databases complete the table of satellites in different ways.

As name and group searches typically return most objects, queries are made in quick TLE (3LE) format in the Celestrak database. Searches by specific date require archive data and have to be taken from the Space-Track site. Note, that searches via TLE retrieve only basic information about imported satellites and are marked as yellow lines. This is however fully sufficient for orbit propagation. When the `Autocomplete satellites` option from the filter menu is checked and user selects any yellow row, request to Space-Track's API is sent with the relevant NORAD catalog number. Space-Track then replies in XML format, completing object's type, launch date and the country of origin. Line is then marked as green. Note, that searching by date may return historical objects, that have yet decayed. In that case, decay date is filled and the satellite is marked as red.

Due to rate limiting, any query sent to Space-Track is logged and assigned a timestamp. When the logging queue reaches its maximum limit of 30 requests per minute, timestamp of every new query is compared with the oldest, 30th entry. If both records differ by more than 60 seconds, the oldest query is removed and the new Space-Track request is handled. For simplification purposes, the limit of 300 requests per hour is not enforced. Rate limiter is implemented in the `STQueryLimiter.py` file, handling of all requests in general can be found in the `RequestHandler.py` file.

Additionally, imports of TLEs and 3LEs from text file is also supported. It is important to mention, that such data must be in specific format and that only either TLEs or 3LEs under each other will be recognized and properly imported. It is advised to obtain these lines through queries to one of the databases. File manipulation is implemented directly in the `MainWindow.py` file. For reference, the attached `3LE_input_data.txt` file contains few input samples.

[2] Available from: https://rhodesmill.org/skyfield/.

3.3 Simulator implementation

The computational part for all calculations regarding coordinate conversions, and orbital propagation is implemented in separate C++ files and connected to PyQt interface via `SimHandler.py` file and the subsequent `SimulatorHandler` class. This serves as an inter-step, which automatically compiles and runs the built C++ code as a subprocess. The compilation process happens only once, when orbit propagation is needed for the first time (typically when satellite is selected and its future orbit shown). User is notified of important events on the status bar at the bottom left corner.

Communication with the compiled program happens through command line arguments and the attached `stdout` and `stderr` output pipes. Launching the C++ program first requires argument with desired operation: either `--position`, which simply returns the ECEF cartesian position vector of spacecraft, or `--orbit`, which returns list of points, summarizing exactly one forward revolution around the Earth. Necessary arguments for orbit propagation then follow, including epoch data, orbital parameters, B* and position of the Sun and the Moon, as well as of every planet of the Solar System. Generally, ECEF vectors representing current and propagated positions are returned and passed directly into various PyQt plotting functions. Spacecraft's latitude and longitude is returned always.

Computational part of the simulator is implemented according to the theory, outlined in chapter 2. For convenience, the flow graph below highlights all basic principles.

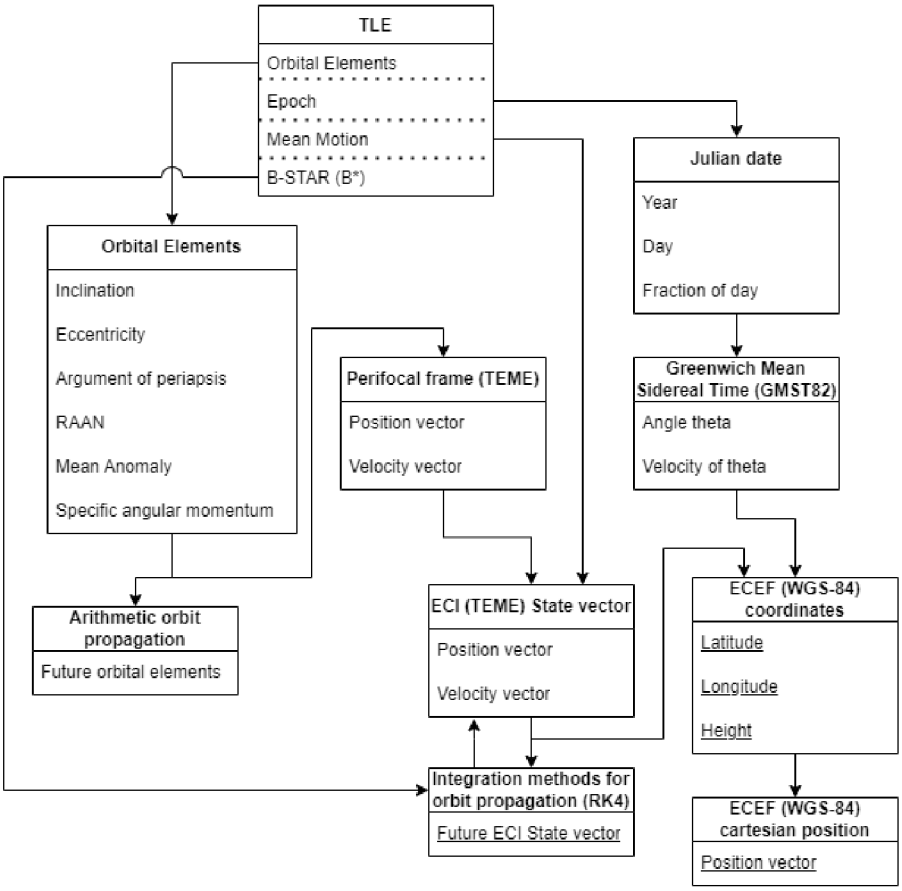

Figure 3.3: Simplified flow graph of the C++ computational part.

Figure 3.3 generally summarizes how various procedures interact with each other. The TLE box at the top symbolizes the entry point to the program. Provided TLE parameters are loaded into new object of `SatOrbit` class from `sim.h`. This object, representing satellite's orbit contains various functions for further manipulation – stepping the epoch time, getting spacecraft's ECI state vector, its ECEF coordinates, or ECEF position vector. Each of these procedures requires many intersteps, typical for various coordinate conversions. This class mainly serves as an interface for very specific functions implemented in `simFunctions.h` file.

Notable points of the program in fig. 3.3 are underlined. Propagating the orbit happens numerically with 4th order Runge-Kutta (RK4) method, producing a new state vector in delta time. After simulation time has been reached, the ECI vector is converted into familiar (spherical) coordinates of latitude, longitude and height (recall section 2.3) and occasionally into ECEF cartesian position vector for plotting the points of orbit.

The majority of functions solving single equations and the two main numerical methods, Euler and RK4, are implemented in the `simFunctions.h` file. Self-explanatory class definitions such as `PlanetPosition`, `GMST82` and `JulianDate` are also implemented here. For detailed information regarding functions, see the attached Doxygen documentation. File `constants.h` with various physics-related constants is also used here.

Euler numerical method

Euler's method for solving ordinary differential equations (ODE) is functionally related to the problems of kinematics, as shown in fig. 2.1. Note, that the position vector is propagated in a straight line by the velocity vector. This simple method leaves room for inaccuracies, even with small time steps. Figure 3.4 shows the function $y(t)$, approximated by the dotted line, where position of the next point is estimated by the position of previous one and a length of step multiplied by function derivative of previous point.

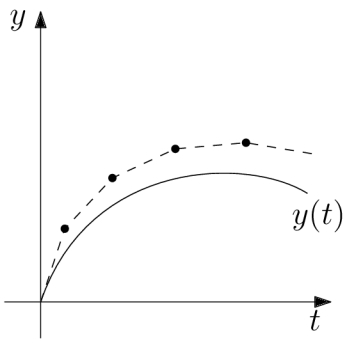

Figure 3.4: The principle of Euler's method, as indicated by dotted line.

Mathematically, this can be expressed as

$$y(t+h) = y(t) + h \cdot f(t, y(t)) \tag{3.1}$$

where h represents the time step and function $f(t, y(t))$ derivation of $y(t)$ in the point in time t.

Runge-Kutta numerical method

The Runge-Kutta method is similar to the classic Euler method for numerical integration, but the accumulated errors are much smaller. Runge-Kutta methods have various implementations, with the most common being 4th order Runge-Kutta method (RK4). The position in delta time is not determined with a single derivation as in Euler's method, but with a combination of 4 separate derivations in inter-steps. Since position is determined by velocity, which is derived from acceleration, the 4 separate measurements refer to calculating acceleration in 4 specific points.

Formally, RK4 method can be mathematically expressed via eq. (3.2), eq. (3.3), eq. (3.4) and eq. (3.5),

$$k_1 = h \cdot f(t, y(t)) \tag{3.2}$$

$$k_2 = h \cdot f\left(t + \frac{h}{2}, y(t) + \frac{k_1}{2}\right) \tag{3.3}$$

$$k_3 = h \cdot f\left(t + \frac{h}{2}, y(t) + \frac{k_2}{2}\right) \tag{3.4}$$

$$k_4 = h \cdot f(t + h, y(t) + k_3) \tag{3.5}$$

$$y(t+h) = y(t) + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6} \tag{3.6}$$

where k_1, k_2, k_3 and k_4 represent four intersteps, then t is point in time and h fixed time step.

Implementation of RK4 algorithm can be found in the `simFunctions.cpp` file. Integrating the method with the principles of kinematics is tricky, since position and velocity vectors require solving 2nd order ODE equations. Function f is represented by acceleration, which can be calculated for any position vector, omitting time variable t completely.

At the first step k_1, state vector containing position vector $\mathbf{r}$ and velocity vector $\mathbf{v}$ in their initial time t_0 is used to calculate the acceleration $\mathbf{a}$ (recall eq. (2.48)). Computing intersteps k for both vectors also happens separately, as seen in eq. (3.7). Interstep's subscript V stands for velocity and R for position. Function f from eq. (3.2) representing derivation in initial time t_0 is replaced by the derivatives of position and velocity (recall

eq. (2.29)). Note, that $\mathbf{a}(\,)$ shows acceleration vector $\mathbf{a}$ as a function with its respective input variables (perturbation elements such as B^* and others omitted for simplicity).

$$\begin{aligned} k_{1_V} &= h \cdot \mathbf{a}(\mathbf{r}_{t_0}, \mathbf{v}_{t_0}) \\ k_{1_R} &= h \cdot \mathbf{v}_{t_0} \end{aligned} \tag{3.7}$$

Similar principles are applied for steps k_2 and k_3 in eq. (3.8) and eq. (3.9), but as observed in eq. (3.3) and eq. (3.4), derivations are calculated with initial values in t_0 plus half the results from the first step k_1.

$$\begin{aligned} k_{2_V} &= h \cdot \mathbf{a}\left(\mathbf{r}_{t_0} + \frac{k_{1_R}}{2}, \mathbf{v}_{t_0} + \frac{k_{1_V}}{2}\right) \\ k_{2_R} &= h \cdot \left(\mathbf{v}_{t_0} + \frac{k_{1_V}}{2}\right) \end{aligned} \tag{3.8}$$

$$\begin{aligned} k_{3_V} &= h \cdot \mathbf{a}\left(\mathbf{r}_{t_0} + \frac{k_{2_R}}{2}, \mathbf{v}_{t_0} + \frac{k_{2_V}}{2}\right) \\ k_{3_R} &= h \cdot \left(\mathbf{v}_{t_0} + \frac{k_{2_V}}{2}\right) \end{aligned} \tag{3.9}$$

The last step k_4 in eq. (3.10) simply adds values of step k_3 to the initial vectors.

$$\begin{aligned} k_{4_V} &= h \cdot \mathbf{a}\left(\mathbf{r}_{t_0} + k_{3_R}, \mathbf{v}_{t_0} + k_{3_V}\right) \\ k_{4_R} &= h \cdot \left(\mathbf{v}_{t_0} + k_{3_V}\right) \end{aligned} \tag{3.10}$$

With initial vectors in t_0 and with eq. (3.7), eq. (3.8), eq. (3.9) and eq. (3.10), the new state vector in time t_1 is found with eq. (3.11).

$$\begin{aligned} \mathbf{v}_{t_1} &= \mathbf{v}_{t_0} + \frac{k_{1_V}}{6} + \frac{k_{2_V}}{3} + \frac{k_{3_V}}{3} + \frac{k_{4_V}}{6} \\ \mathbf{r}_{t_1} &= \mathbf{r}_{t_0} + \frac{k_{1_R}}{6} + \frac{k_{2_R}}{3} + \frac{k_{3_R}}{3} + \frac{k_{4_R}}{6} \end{aligned} \tag{3.11}$$

As a sidenote, the orbit propagation is stopped when the spacecraft descends to a preset height, 90 km by default. At such heights, the drag becomes significant and proper simulation of reentry effects is beyond the scope of this work. Needles to say, predicting where any object reenters the atmosphere is hard even with modern simulators, as reentry date is estimated in the 10% of satellite's remaining lifetime. Satellite designed for 10 years on-orbit will fall sometime in the 10th year. Analogically, prediction that object is to reenter within 24 hours is accurate only to 2 hours [17].

Analytical propagation of orbital elements

Third and final method is substantially different from procedures described in previous chapters. This method simply calculates the changes of satellite's orbital elements over time as a closed-form solution. Although the notable effects of oblate Earth are accounted for, other perturbation forces are in this case omitted, resulting in greater inaccuracies over time. This method is however fast, inexpensive [10] and useful for large quantities of satellites or where CPU power is limited [25]. Estimation of collision risks with large number of satellites might be one of the use cases. If any collisions might occur, trajectories of both satellites are propagated again, through one of the more precise numerical methods.

As mentioned in section 2.4, only the effects of Earth's obliquity are included, meaning, that only RAAN Ω, argument of periapsis ω and mean anomaly M change with time and others stay the same. Their propagation to new time step $t + 1$ is trivial, as implemented in eq. (3.12), eq. (3.13) and eq. (3.14),

$$\Omega_{t+1} = \Omega_t + \dot{\Omega}_{J_2} \cdot t \tag{3.12}$$

$$\omega_{t+1} = \omega_t + \dot{\omega}_{J_2} \cdot t \tag{3.13}$$

$$M_{et+1} = M_{et} + \dot{M}_{eJ_2} \cdot t \tag{3.14}$$

where Ω_t, ω_t and M_{et} are the original values at time t, to which $\dot{\Omega}_{J_2}$, $\dot{\omega}_{J_2}$ and $\dot{M}_{eJ_2}$ from eq. (2.45), eq. (2.46) and eq. (2.47) are added respectively. The time step t can be of any desired size, without any impact on precision. Setting it to the total simulation time is recommended as it is the fastest approach.

Naturally, propagated orbital elements can be further converted into other coordinate frames freely and as described in section 2.3.

3.4 Plotting the results

As seen in fig. 3.2, user interface of LEOSIM includes an interactive part, where satellites and their orbits are displayed. The interface is implemented in PyQtGraph, a popular GUI library for scientific graphics in Python.

When any satellite is selected from the table of objects, its orbital (TLE) values are passed to the C++ code, which can return either satellite's current position as an ECEF vector, or list of such vectors, which form a simulated orbit. When C++ handles the simulation, only a fraction (1 in 100) of ECI positions are converted first to ECEF spherical coordinates of latitude and longitude, and finally to ECEF position vector, as indicated in fig. 3.3, which is used to plot the results. The list doesn't include every ECI position for the sake of efficiency. Satellite's current (or simulated) position is shown as a large green dot and its future orbit is generally plotted as a red line. When satellite is selected, only its next full orbit is shown. Each individual satellite can be checked, in which case its orbit changes colour to yellow and stays drawn even after said satellite is deselected. In special cases of

decayed satellites, plotted orbit may end prematurely, as the spacecraft descended below 90 km of altitude, a low-boundary threshold of the model.

Working with ECEF vectors instead of ECI vectors has its drawbacks and advantages. The Earth's model in PyQtGraph is a modified spherical object, that is fixed in place and does not rotate. This is ideal for ECEF vectors, that represent spacecraft's position relative to fixed Earth, meaning, that positions are plotted directly over the locations of their ground tracks. Some orbits that start at the poles may however plot as slightly twisted, thanks to the Earth's rotation. Using ECI vectors directly would be convenient for visualizations with rotating Earth and real-time animations.

For implementing planet Earth, LEOSIM utilizes modified source code of class `GLImageItem` from PyQtGraph, using it in class `GLTexturedSphericalItem`. Plotted objects are implemented as `SatelliteVisuals` class, containing both the green dot representing satellite and a list of lines, that form the orbit. Complete visuals are then handled by `SimulatorView` class, which automatically shows selected and checked satellites and hides all, that were unselected. LEOSIM's graphical environment is implemented in the `simView.py` file.

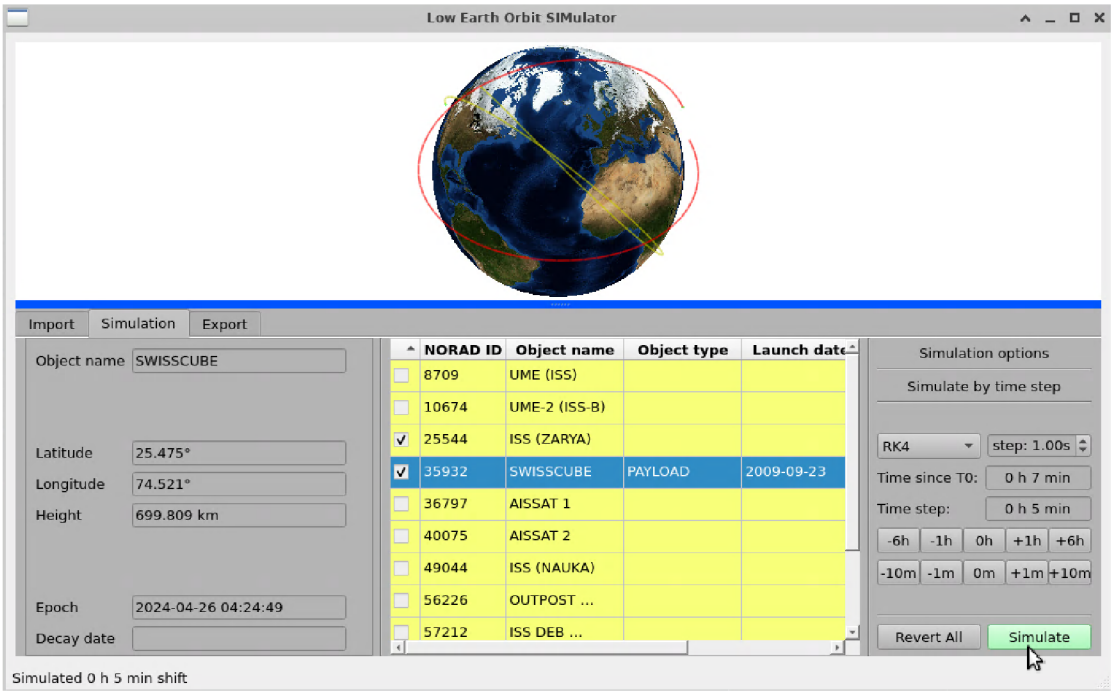

Figure 3.5: UI of simulation environment.

Chapter 4

Simulating Experiments

So far, we have covered the theoretical principles and implementation of the LEOSIM. This chapter focuses on validating and testing the numerical model through comparison with reference positions, comparison with future TLE measurements and with the reference model SGP4. Additionally, outputs of the analytical approach from section 3.3 will be also compared with reference measurements. It is important to mention, that simulating few days of satellite's movements pushes both LEOSIM and SGP4 numerical models to their limits. In real-world scenarios, common usage case consist of simulating TLE values only to point in time, when new and precise TLE measurement is made. The reliable positioning of satellites worldwide for amateurs is ensured through quantity – frequent updates to their position, rather than quality – a precise simulation [1].

The subject of comparison will be the ECEF spherical coordinates of latitude, longitude and height only.[1] No orbital elements will be compared in this chapter, as LEOSIM would have to create just the format of orbital parameters, contained within TLEs, a discipline, that is strongly advised not to do, as such parameters are not created through simple conversions, nor with primitive equations [15]. Although SGP4 model is commonly used for their creation, its public python implementation[2], does not implement this behaviour at all.

Examples in this chapter are monitored space-debris in LEO, which are not expected to perform maneuvers and their trajectory is easily predictable.

4.1 Precision of coordinate conversion

The basic assumption for correct behaviour in subsequent examples is accurate conversion from orbital elements to ECEF coordinates of latitude, longitude and height. Recall section 2.3, that this is done via converting orbital elements to two-dimensional perifocal frame, then to ECI state vector around the Earth using zxz Euler transformation matrices and finally to ECEF frame, with the use of reference epoch, that rotates the planet. Since this approach consists of closed-form equations, the errors are generally low.

[1] The used distance calculator taken from NOAA, avaliable at: https://www.nhc.noaa.gov/gccalc.shtml.
[2] Available from https://pypi.org/project/sgp4/.

The table 4.1 shows the results of five tests, that check the accuracy of computed position. Error values E are formally obtained via eq. (4.1),

$$E = |R - C| \tag{4.1}$$

where the rows containing reference values R are subtracted from the calculated values C. Error rows are the absolute values of these computations.

All reference values are obtained from two separate and independent public sources[3][4].

Satellite 25544 (ISS ZARYA)					
Test	1	2	3	4	5
Ref. latitude	49.878°	−0.029°	51.68°	−0.008°	42.567°
Calc. latitude	49.895°	0.078°	51.702°	0.078°	42.6037°
Error [deg]	0.017°	0.107°	0.0217°	0.086°	0.0357°
Error [km]	2 km	12 km	2 km	10 km	4 km
Ref. longitude	159.674°	55.448°	3.024°	−109.765°	−67.373°
Calc. longitude	159.6717°	55.5356°	2.9819°	−109.6804°	−67.2998°
Error [deg]	0.0023°	0.0876°	0.0421°	0.0846°	0.0732°
Error [km]	0 km	10 km	5 km	9 km	8 km
Ref. height	424.82 km	416.1 km	424.52 km	416.05 km	420.61 km
Calc. height	431.64 km	415.327 km	431.192 km	415.204 km	426.516 km
Error [km]	6.82 km	0.7728 km	6.6723 km	0.8462 km	5.906 km
Total Error [km]	8.82 km	22.77 km	13.67 km	19.84 km	17.9 km

Table 4.1: Example No.1: Precision of coordinate conversion.

Total error is simply calculated as a sum of all three errors of the three coordinates. Given, that timestamp of reference measurements was rounded to the nearest second and that ISS travels at approximately $8\,\mathrm{km} \cdot s^{-1}$, the total error is well within the limits of TLE error margins.

[3]Tracking program created for Smithsonian Air & Space Museum and the Boston Museum of Science, available from: http://www.isstracker.com/home.

[4]Tracking program from Dominic Ford, Institute of Astronomy, Cambridge UK; available from: https://in-the-sky.org/satmap_worldmap.php

4.2 Comparison with reference measurements

This section compares the outputs of numerical and analytical approaches to the reference measurements. The precision of both approaches is tested with satellite's position above the Earth in spherical coordinates of latitude, longitude and height.

Numerical approach

In this approach, first, an initial ECI position is calculated from the original TLE, which is furthermore propagated and compared to the positions of subsequently measured TLEs. Since measurements take place on non-regular basis (recall section 3.2), the time interval between measured reference values can differ highly from few hours to few days. The testing process is illustrated in fig. 4.1.

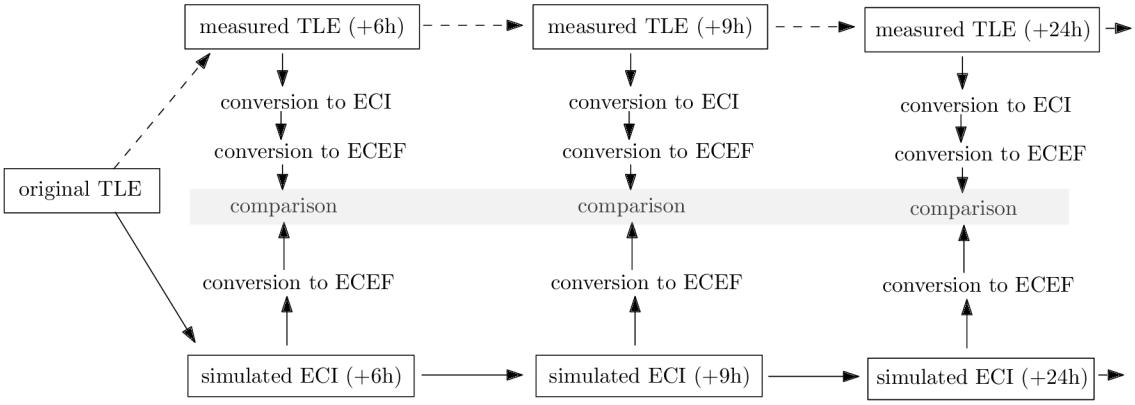

Figure 4.1: Scheme of the tests with reference values.

Since the measured TLE describes the satellite "as is", it is important to validate the model against an object, which does not change its trajectory during the course of measurement. Space debris or retired satellites are ideal for this role. All tests are written in C++ in the `test.cpp` file, using a public BOOST test library [24].

The object used for the following experiments has NORAD ID 58229, with the name "ISS DEB". It is probably some type of debree from the ISS, that does not have maneuvering capabilities and is frequently monitored. For reference, it was measured on 2024-04-13 at 10:50:30.

Errors are obtained as a net values, meaning, that calculated values are just subtracted from reference without absolute values. Comparison with the reference TLEs using RK4 method with 1 s step yields the results from table 4.2.

As can be observed, simulated orbital parameters deviated rather quickly. This is believed to be a consequence of custom TLE propagation methods. As mentioned in section 3.2, TLE stores mean orbital elements, whose periodic variations were removed by SGP4 in a specific way and need to be introduced similarly, as they were removed. It is strongly advised to further propagate the elements with SGP family of perturbation models only [1][12][30], else degraded predictions are to be expected.

Epoch	—	Latitude	Longitude	Height
	Reference	0.0787°	−122.6578°	355.35 km
13.4.24 12:22:06	Calculated	0.4519°	−122.3611°	355.47 km
+1 h 31 min	Error [degree]	−0.3735°	−0.2967°	—
	Error [km]	−42 km	−33 km	−0.14 km
	Reference	0.0787°	167.4833°	355.13 km
13.4.24 16:56:53	Calculated	1.6271°	168.7042°	355.63 km
+6 h 6 min	Error [degree]	−1.5485°	−1.2208°	—
	Error [km]	−172 km	−136 km	−0.57 km
	Reference	0.0787°	178.2222°	354.01 km
14.4.24 15:50:39	Calculated	5.9483°	−177.1326°	356.75 km
+29 h 0 min	Error [degree]	−5.8695°	−4.6451°	—
	Error [km]	−652 km	−516 km	−2.73 km
	Reference	0.0787°	−124.4275°	353.07 km
15.4.24 11:41:04	Calculated	7.7138°	−118.3599°	357.46 km
+48 h 50 min	Error [degree]	−7.6350°	−6.0676°	—
	Error [km]	−848 km	−674 km	−4.38 km

Table 4.2: Example No.2: Precision of comparison with TLEs.

Analytical approach

In this part, the trajectory of orbit, as defined by orbital elements in the reference TLE, will be propagated analytically, as described in section 3.3. As mentioned before, propagating perturbation forces is limited and will not reach the desired accuracy of the SGP4 model. However, where the RK4 simulation takes tens of seconds to complete, analytical propagation is computed instantaneously and in a single step.

The propagated TLEs are first converted to ECI and then to familiar ECEF coordinate system of latitude, longitude and height. Results from these calculations are compared to the reference values from table 4.2. Errors are again taken as net values, meaning, that numbers are not absolute.

As can be seen in table 4.3, this method is paradoxically much more precise, than the numerical approach from previous section. Although it only covers the effect of Earth's oblateness, propagation of orbital elements in TLEs is much simpler, as it does not require specific methods introduced in a specific order. More surprisingly, this approach is actually more precise for the first two measurements (first 6 hours), than outputs of the SGP4 model in table 4.4. The dramatically increasing error could be additionally lowered by implementing other perturbative effects.

Epoch	—	Latitude	Longitude	Height
13.4.24 12:22:06 +1 h 31 min	Reference	0.0787°	−122.6578°	355.35 km
	Calculated	0.043°	−122.6831°	355.41 km
	Error [degree]	0.0357°	0.0254°	—
	Error [km]	4 km	3 km	−0.06 km
13.4.24 16:56:53 +6 h 6 min	Reference	0.0787°	167.4833°	355.13 km
	Calculated	0.0393°	167.4541°	355.38 km
	Error [degree]	0.0394°	0.0292°	—
	Error [km]	4 km	3 km	−0.2470 km
14.4.24 15:50:39 +29 h 0 min	Reference	0.0787°	178.2222°	354.01 km
	Calculated	−0.4445°	177.8123°	355.23 km
	Error [degree]	0.5232°	0.4099°	—
	Error [km]	58 km	46 km	−1.2102 km
15.4.24 11:41:04 +48 h 50 min	Reference	0.0787°	−124.4275°	353.07 km
	Calculated	−1.4256°	−125.6067°	355.08 km
	Error [degree]	1.5044°	1.1793°	—
	Error [km]	167 km	131 km	−2 km

Table 4.3: Example No.3: Precision of analytical propagation of TLEs.

4.3 Comparison with SGP4 model

This section briefly outlines the history behind the reference SGP4 model, its usage cases and drawbacks. The numerical trajectories from LEOSIM are then compared to this model.

Simplified General Perturbation model (SGP4)

The SGP4 belongs to a simulation family of Simplified perturbation models, that are developed since 1960s and have been released for public use in 2006. Originally created for the now-called United States Space Surveillance Network, the model utilizes official TLE sets to numerically simulate satellite's movement in LEO with orbital periods lower than 225 minutes (5 877.5 km in altitude) [20]. For reference, Medium Earth orbits (MEO) typically start at 2 000 km, up until Geostationary orbits (GEO) at 35 786 km. For orbital periods above 225 minutes, Simplified Deep Space Perturbation (SDP) models are typically used. None of the models simulate reentry effects.

The model is nowadays widely used, as TLEs are the only large-scale public source of satellite's positions, that is regularly updated and the SGP family of models offers the only proper way to utilize it. There are however many technical limitations and general issues, that stem from model's historic usage (as described in [30]):

- TLEs do not show which SGP model was used to create them. This is problematic, since the TEME coordinate frame, which is used here, was redefined multiple times

in history. To propagate historic data correctly, user has to decide between different versions of SGP4, or otherwise risk errors in hundreds of kilometers [30].

- Although TLEs are mostly standardized, the model cannot handle its historical variations with omitted parameters, in cases when they were equal to zero.
- The model uses TEME ECI coordinate frame (recall section 3.2), which presents difficulties in replicating the model, as its exact operational definition is difficult to find.
- Analysing data of multiple satellites can be complicated by the concurrent usage of UT1 and UTC.

Overall, not SGP4 nor TLEs are suitable for a precise orbit estimation [1], but rather for less demanding usages, such as satellite radiocommunication, where antennas have certain targeting and tracking tolerances. The model takes into account the effects of atmospheric drag through B* (recall section 2.4) and the effects of zonal harmonics J_2, J_3 and J_4 for oblate Earth (recall section 2.4). Other effects, such as third-body gravitational effects, SRP or ERP are not directly implemented.

Approximate accuracy of the model is as follows: for an orbit of approximately 400 km, the error ranged between 4 to 10 km after one day, then 10 and 40 km after 3 days, 60 to 300 km after 7 days and 300 to 1000 km after full two weeks [8]. The accuracy often depends on various factors, such as Sun's activity effecting SRP, that are not perturbed.

Comparing results

The object, that is used for direct comparisons is the same as in section 4.2. For table 4.4, the ECI positions were propagated by the SGP4 model and converted into ECEF frame of latitude, longitude and height with the help of LEOSIM. Reference positions represent those of reference TLE measurements, calculated positions are the outputs of the SGP4 model.

Comparison of SGP4 error margins with results from LEOSIM's model in table 4.2 seem a bit off. Again, this is believed to be caused by the unpredictability of TLE data in foreign and custom models, other than from the SGP family [1][12][30]. In the grand overview, satellite in LEO travelling at $7.8\,\mathrm{km}\cdot\mathrm{s}^{-1}$ travels 1,375,920 km in 49 hours and errors in hundreds of kilometers after this time interval could be summarized as still acceptable for many applications.

Epoch	—	Latitude	Longitude	Height
	Reference	0.0787°	−122.6578°	355.35 km
13.4.24 12:22:06	Calculated	−0.0250°	−122.7394°	356.12 km
+1 h 31 min	Error [degree]	0.1037°	0.0817°	—
	Error [km]	12 km	9 km	−0.77 km
	Reference	0.0787°	167.4833°	355.14 km
13.4.24 16:56:53	Calculated	−0.0107°	167.4131°	355.97 km
+6 h 6 min	Error [degree]	0.0894°	0.0702°	—
	Error [km]	10 km	8 km	−0.84 km
	Reference	0.0787°	178.2222°	354.02 km
14.4.24 15:50:39	Calculated	−0.0572°	178.1129°	355.24 km
+29 h 0 min	Error [degree]	0.1359°	0.1093°	—
	Error [km]	15 km	12 km	−1.22 km
	Reference	0.0787°	−124.4275°	353.08 km
15.4.24 11:41:04	Calculated	−0.1886°	−124.6424°	354.61 km
+48 h 50 min	Error [degree]	0.2674°	0.2149°	—
	Error [km]	30 km	24 km	−1.53 km
	Reference	0.788°	2.7943°	352.59 km
16.4.24 2:56:39	Calculated	−0.3679°	2.4368°	354.14 km
+64 h 6 min	Error [degree]	0.4467°	0.3575°	—
	Error [km]	50 km	40 km	−1.55 km
	Reference	0.0787°	−160.1342°	352.03 km
16.4.24 13:37:30	Calculated	−0.5185°	−160.6104°	353.80 km
+74 h 46 min	Error [degree]	0.5972°	0.4762°	—
	Error [km]	66 km	53 km	−1.77 km
	Reference	0.0787°	176.5916°	351.95 km
16.4.24 15:09:03	Calculated	−0.5265°	176.1069°	353.76 km
+76 h 18 min	Error [degree]	0.6082°	0.4847°	—
	Error [km]	68 km	54 km	−1.81 km

Table 4.4: Example No.4: Precision of the SGP4, expanded.

Chapter 5

Conclusion

The goal of this thesis was to implement an user-friendly tool with graphical interface, which retrieves positions of satellites from various public databases and visually propagates their orbit in time. Secondly, this work successfully serves as an educational and theoretical material for any space enthusiast, interested in the problematic of coordinate transformation, trajectory propagation or orbital mechanics in general. Since the Two-line element sets (TLEs) used here require specific methods and approach, this work also serves as a reference on how to partially process them. The final section highlights the accuracy of coordinate transformations and further compares propagated positions of the numerical and analytical approaches to reference measurements and to the reference model SGP4.

This work successfully explained theory and summarized program's implementation. Precision of converting TLEs to latitude, longitude and height was excellent with an average error of 16.6 km. Comparison on numerical approach with reference measurements and reference model was satisfactory and with greater errors, as expected. Ultimately, this model did not surpass SGP4's performance. Additionally, analytical propagation of orbital elements yielded good results and surpassed SGP4's performance in the first 6 to 7 hours.

The author has learned, that working with TLEs requires unconventional approach and that their precise numerical propagation is beyond the scope of this bachelor's thesis. Many different implementation paths were explored and tried, which ultimately led to a dead-end. However, the program serves as a solid foundation for further expansion and the author is thrilled to incorporate various interesting ideas: expanding analytical orbit propagation, reentry prediction, orbital maneuvers, garbage-collecting satellites, debree collision analysis and more, with the intent to create a handy universal program and to provide comprehensive overview of these procedures to an average space nerd.

Bibliography

[1] Aida, S. and Kirschner, M. Accuracy assessment of SGP4 orbit information conversion into osculating elements. In: *6th European Conference on Space Debris, Darmstadt, Germany*. ESA, April 2013, p. 1–8. Available at: https://conference.sdo.esoc.esa.int/proceedings/sdc6/paper/41/SDC6-paper41.pdf.

[2] Astronomical Applications Department. *Julian Date Converter* [online]. U.S. Naval Observatory [cit. 2024-04-09]. Available at: https://aa.usno.navy.mil/data/JulianDate.

[3] Beggs, J. S. *Kinematics*. 1st ed. CRC Press, 1983, p. 1. DOI: https://doi.org/10.1137/1027026. ISBN 978-0891163558. Available at: https://www.esa.int/Space_Safety/Space_Debris/Space_debris_by_the_numbers.

[4] Capitaine, N., Wallace, P. T. and McCarthy, D. D. Expressions to implement the IAU 2000 definition of UT1. *Astronomy & Astrophysics* [online]. EDP Sciences. 2003, vol. 406, no. 3, p. 1149, [cit. 2024-04-09]. DOI: https://doi.org/10.1051/0004-6361:20030817. Available at: https://www.aanda.org/articles/aa/abs/2003/30/aa3487/aa3487.html.

[5] Chao, C.-C. G. and Campbell, S. Estimating Solar Radiation Pressure for GEO Debris. In: Lacoste, H., ed. *Fifth European Conference on Space Debris*. ESA, March 2009, vol. 672, p. 1–5. ISBN 978-92-9221-236-0. Available at: https://conference.sdo.esoc.esa.int/proceedings/sdc5/paper/115/SDC5-paper115.pdf.

[6] Croitoru, E.-I. and Oancea, G. Satellite tracking using norad two-line element set format. *Scientific Research and Education in the Air Force-AFASES*. 2016, vol. 1, p. 423–431. DOI: https://doi.org/10.19062/2247-3173.2016.18.1.58. Available at: https://www.afahc.ro/ro/afases/2016/MATH&IT/CROITORU_OANCEA.pdf.

[7] Curtis, H. D. Chapter 12 - Introduction to Orbital Perturbations. In: Curtis, H. D., ed. *Orbital Mechanics for Engineering Students (Third Edition)*. 3rd ed. Boston: Butterworth-Heinemann, 2014, p. 651–720. DOI: https://doi.org/10.1016/B978-0-08-097747-8.00012-8. ISBN 978-0-08-097747-8. Available at: https://www.sciencedirect.com/science/article/pii/B9780080977478000128.

[8] Dong, W. and Chang yin, Z. An Accuracy Analysis of the SGP4/SDP4 Model. *Chinese Astronomy and Astrophysics*. 2010, vol. 34, no. 1, p. 69–76. DOI: https://doi.org/10.1016/j.chinastron.2009.12.009. ISSN 0275-1062. Available at: https://www.sciencedirect.com/science/article/pii/S0275106209001404.

[9] Farres, A., Puig, A. and Zardaín, L. High-fidelity Modeling and Visualizing of Solar Radiation Pressure: A Framework for High-fidelity Analysis. In: *20202 AAS/AIAA Astrodynamics Specialist Conference*. August 2020. Available at: http://www.ub.edu/wai/wp-content/uploads/2021/07/AAS-20-481.pdf.

[10] Flores, R., Burhani, B. M. and Fantino, E. A method for accurate and efficient propagation of satellite orbits: A case study for a Molniya orbit. *Alexandria Engineering Journal* [online]. 2021, vol. 60, no. 2, p. 2661–2676. DOI: https://doi.org/10.1016/j.aej.2020.12.056. ISSN 1110-0168. Available at: https://www.sciencedirect.com/science/article/pii/S1110016821000016.

[11] Hakima, H., Bazzocchi, M. C. and Almstrom, B. Analysis of Satellite Drag Coefficients Based on Physical and Orbital Specifications. In: *2022 IEEE Aerospace Conference (AERO)*. IEEE, 2022, p. 1–2. DOI: https://doi.org/10.1109/AERO53065.2022.9843417. ISSN 1095-323X. Available at: https://ieeexplore.ieee.org/document/9843417.

[12] Hoots, F. R. and Roehrich, R. L. *Spacetrack Report No. 3: Models for Propagation of NORAD Element Sets*. U.S. Department of Defense, december 1988. 34 p. Available at: https://celestrak.org/NORAD/documentation/spacetrk.pdf.

[13] Jo, J.-H., Park, I.-K., Choe, N.-M. and Choi, M.-S. The Comparison of the Classical Keplerian Orbit Elements, Non-Singular Orbital Elements (Equinoctial Elements), and the Cartesian State Variables in Lagrange Planetary Equations with J2 Perturbation: Part I. In: *Journal of Astronomy and Space Sciences*. The Korean Space Science Society, March 2011, vol. 28, p. 37–54. DOI: https://doi.org/10.5140/JASS.2011.28.1.037. ISSN 2093-1409. Available at: https://koreascience.kr/article/JAKO201110441050581.page.

[14] Kelso, T. S. Orbital Coordinate Systems, Part III. *Satellite Times* [online]. Bob Grove, WA4PYQ. 1996, vol. 2, no. 3, [cit. 2024-04-09]. ISSN 1077-2278. Available at: https://www.worldradiohistory.com/Archive-Satellite-Times/Satellite-Times-1996-01-02.pdf.

[15] Kelso, T. S. More Frequently Asked Questions. *Satellite Times* [online]. Bob Grove, WA4PYQ. 1998, vol. 4, no. 5, [cit. 2024-04-09]. ISSN 1077-2278. Available at: https://www.worldradiohistory.com/Archive-Satellite-Times/Satellite-Times-1998-03.pdf.

[16] Kelso, T. A New Way to Obtain GP Data (aka TLEs). [online]. CelesTrak. may 2020, [cit. 2024-04-09]. Available at: https://celestrak.org/NORAD/documentation/gp-data-formats.php.

[17] Kennewell, J. Satellite Orbital Decay Calculations. [online]. IPS Radio & Space Services, Sydney, Australia. 1999, p. 2, [cit. 2024-04-09]. Available at: https://www.sws.bom.gov.au/Category/Educational/Space%20Weather/Space%20Weather%20Effects/Satellite%20Orbital%20Decay%20Calculations.pdf.

[18] NASA. *Time Zones and Universal Time* [online]. NASA, 2010 [cit. 2024-04-09]. Available at: https://eclipse.gsfc.nasa.gov/SEhelp/TimeZone.html.

[19] NOAA. *U.S. standard atmosphere, 1976*. National Oceanic and Atmospheric Administration, 1976, p. 65–88. Available at: https://www.ngdc.noaa.gov/stp/space-weather/online-publications/miscellaneous/us-standard-atmosphere-1976/.

[20] Payne, T., Hoots, F., Butkus, A., Slatton, Z. and Nguyen, D. Improvements to the SGP4 propagator (SGP4-XP). In: *2022 AMOS Technical Conference*. Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS), September 2022, p. 1–10. Available at: https://amostech.com/TechnicalPapers/2022/Astrodynamics/Payne_2.pdf.

[21] Peale, S. J. Kepler's laws of planetary motion. [online]. Britannica. [cit. 2024-04-09]. Available at: https://www.britannica.com/science/celestial-mechanics-physics/Keplers-laws-of-planetary-motion.

[22] Peet, M. M. *Lecture 12: Orbital Perturbations* [online]. Arizona State University, march 2023 [cit. 2024-04-09]. Available at: https://control.asu.edu/Classes/MAE462/462Lecture12.pdf.

[23] Rees, J. *Natural Resources: Allocation, Economics and Policy*. 1st ed. Routledge, London, 1985. DOI: https://doi.org/10.4324/9781315112770. ISBN 9781315112770. Available at: https://www.taylorfrancis.com/books/mono/10.4324/9781315112770.

[24] Rozental, G. *Boost Test Library* [online]. October 2004 [cit. 2024-04-09]. Available at: https://www.boost.org/doc/libs/1_32_0/libs/test/doc/index.html.

[25] San Juan, J., Perez, I., Vergara, E., Martín, M., López, R. et al. Hybrid SGP4 propagator based on machine-learning techniques applied to GALILEO-type orbits. In:. 69th International Astronautical Congress, January 2018, p. 2–6. Available at: https://www.esa.int/gsp/ACT/doc/MAD/pub/ACT-RPR-MAD-2018-HybridPropagation.pdf.

[26] Schaub, H. and Junkins, J. L. Gravitational Potential Field Models. In: *Analytical Mechanics of Space Systems, Fourth Edition*. 4th ed. AIAA, April 2018, p. 621–646. DOI: https://doi.org/10.2514/5.9781624105210.0621.0646. ISBN 978-1-62410-657-6. Available at: https://www.researchgate.net/publication/322605303_Analytical_Mechanics_of_Space_Systems_Fourth_Edition.

[27] Slater, J. A. and Malys, S. WGS 84 — Past, Present and Future. In: Brunner, F. K., ed. *Advances in Positioning and Reference Frames*. Springer, Berlin, Heidelberg, 1998, vol. 118, p. 4 [cit. 2024-04-09]. DOI: https://doi.org/10.1007/978-3-662-03714-0. ISBN 978-3-662-03714-0. Available at: https://link.springer.com/chapter/10.1007/978-3-662-03714-0_1.

[28] Space Track. *Introduction to the API* [online]. [cit. 2024-04-09]. Available at: https://www.space-track.org/documentation#/api.

[29] Vallado, D. A. 1 - Perturbed Motion. In: Gurfil, P., ed. Butterworth-Heinemann, 2006, vol. 1, p. 7 [cit. 2024-04-09]. Elsevier Astrodynamics Series. DOI: https://doi.org/10.1016/S1874-9305(07)80003-3. ISSN 1874-9305. Available at: https://www.sciencedirect.com/science/article/abs/pii/S1874930507800033.

[30] Vallado, D. A., Crawford, P., Hujsak, R. and Kelso, T. S. Revisiting Spacetrack Report #3: rev 2. In: *AIAA-2006-6753-Rev2. AIAA/AAS Astrodynamics*

Specialists Conference and Exhibit .... 2006, p. 5–32. Available at: https://celestrak.org/publications/aiaa/2006-6753/AIAA-2006-6753-Rev2.pdf.

[31] Zhao, J.-H. *Astronomical Times* [online]. Center for Astrophysics, Harward Smithsonian, 1996 [cit. 2024-04-09]. Available at: https://lweb.cfa.harvard.edu/~jzhao/times.html.

[32] Zheng, Y. *Generation of network-based differential corrections for regional GNSS services*. 2007. 78-80 p. Dissertation. Queensland University of Technology. Available at: https://eprints.qut.edu.au/16359/1/Yi_Zheng_Thesis.pdf.

Appendix A

Two-line element format

The table A.1 and table A.2 illustrate the format of the TLE sample from section 3.2.

Position:	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
Data:	**1**		**2**	**5**	**5**	**4**	**4**	**U**		**9**	**8**	**0**	**6**	**7**	**A**				**2**	**4**	**0**	**8**	**4**	**.**	**8**	**4**	**5**	**3**	**6**	**4**	**2**	**2**
Section:	1		2					3		4		5			6				7				8										

Position:	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61
Data:			**.**	**0**	**0**	**0**	**3**	**4**	**3**	**2**	**7**			**0**	**0**	**0**	**0**	**0**	–	**0**			**6**	**1**	**9**	**2**	**3**	–	**3**
Section:			9											10									11						

Position:	62	63	64	65	66	67	68	69
Data:		**0**			**9**	**9**	**9**	**5**
Section:		12			13			14

Table A.1: 1st line of TLE

Section 1	Line number of Element Data
Section 2	Satellite catalog number (unique number assigned by NORAD)
Section 3	Classification (U – unclassified; C – classified; S – secret)
Section 4	International designation (last two digits of launch year)
Section 5	International designation (launch number of the year)
Section 6	International designation (payload part)
Section 7	Last two digits of Epoch year
Section 8	Epoch day and its fraction
Section 9	First time derivative of Mean motion; also the Ballistic coefficient
Section 10	Second derivative of Mean motion
Section 11	B* drag value (for orbit propagation)
Section 12	Ephemeris type (orbital model that generated the data, nowadays unused)
Section 13	Element set number (increases with each new TLE for this object)
Section 14	Modulo check (adding values and dividing by 10 yields this number)

The second line of TLE mainly displays various orbital elements. Note, that true anomaly θ is replaced by the more convenient mean anomaly M.

Position:	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
Data:	**2**		**2**	**5**	**5**	**4**	**4**			**5**	**1**	**.**	**6**	**4**	**0**	**9**			**1**	**3**	**.**	**4**	**9**	**9**	**8**	
Section:	1		2							3									4							

Position:	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52
Data:	**0**	**0**	**0**	**4**	**4**	**5**	**0**				**2**	**.**	**7**	**9**	**5**	**5**			**5**	**6**	**.**	**5**	**1**	**5**	**0**	
Section:	5									6									7							

Position:	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69
Data:	**1**	**5**	**.**	**4**	**9**	**4**	**1**	**8**	**3**	**0**	**0**	**4**	**4**	**5**	**4**	**6**	**2**
Section:	8											9					10

Table A.2: 2nd line of TLE

Section 1	Line number of Element Data
Section 2	Satellite catalog number (unique number assigned by NORAD)
Section 3	Inclination i (in degrees)
Section 4	Right ascension of the ascending node; RAAN Ω (in degrees)
Section 5	Eccentricity e
Section 6	Argument of periapsis ω (in degrees)
Section 7	Mean anomaly of elliptical orbit M_e (in degrees)
Section 8	Mean motion n (in orbits per day)
Section 9	Number of elapsed orbits
Section 10	Modulo check (adding values and dividing by 10 yields this number)

Appendix B

XML response format

This XML response represents alternative response format to the TLE in appendix A.

```
<ndm xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="https://sanaregistry.org/r/
 ndmxml_unqualified/ndmxml-3.0.0-master-3.0.xsd">
<script/>
  <omm id="CCSDS_OMM_VERS" version="2.0">
    <header>
      <COMMENT>GENERATED VIA SPACE-TRACK.ORG API</COMMENT>
      <CREATION_DATE>2024-03-24T21:06:15</CREATION_DATE>
      <ORIGINATOR>18 SPCS</ORIGINATOR>
    </header>
    <body>
      <segment>
        <metadata>
          <OBJECT_NAME>ISS (ZARYA)</OBJECT_NAME>
          <OBJECT_ID>1998-067A</OBJECT_ID>
          <CENTER_NAME>EARTH</CENTER_NAME>
          <REF_FRAME>TEME</REF_FRAME>
          <TIME_SYSTEM>UTC</TIME_SYSTEM>
          <MEAN_ELEMENT_THEORY>SGP4</MEAN_ELEMENT_THEORY>
        </metadata>
        <data>
          <meanElements>
            <EPOCH>2024-03-24T20:17:19.468608</EPOCH>
            <MEAN_MOTION>15.49418300</MEAN_MOTION>
            <ECCENTRICITY>0.00044500</ECCENTRICITY>
            <INCLINATION>51.6409</INCLINATION>
            <RA_OF_ASC_NODE>13.4998</RA_OF_ASC_NODE>
            <ARG_OF_PERICENTER>2.7955</ARG_OF_PERICENTER>
            <MEAN_ANOMALY>56.5150</MEAN_ANOMALY>
          </meanElements>
          <tleParameters>
            <EPHEMERIS_TYPE>0</EPHEMERIS_TYPE>
            <CLASSIFICATION_TYPE>U</CLASSIFICATION_TYPE>
```

```
            <NORAD_CAT_ID>25544</NORAD_CAT_ID>
            <ELEMENT_SET_NO>999</ELEMENT_SET_NO>
            <REV_AT_EPOCH>44546</REV_AT_EPOCH>
            <BSTAR>0.00061923000000</BSTAR>
            <MEAN_MOTION_DOT>0.00034327</MEAN_MOTION_DOT>
            <MEAN_MOTION_DDOT>0.0000000000000</MEAN_MOTION_DDOT>
          </tleParameters>
          <userDefinedParameters>
            <USER_DEFINED parameter="SEMIMAJOR_AXIS">6796.564</USER_DEFINED>
            <USER_DEFINED parameter="PERIOD">92.938</USER_DEFINED>
            <USER_DEFINED parameter="APOAPSIS">421.453</USER_DEFINED>
            <USER_DEFINED parameter="PERIAPSIS">415.404</USER_DEFINED>
            <USER_DEFINED parameter="OBJECT_TYPE">PAYLOAD</USER_DEFINED>
            <USER_DEFINED parameter="RCS_SIZE">LARGE</USER_DEFINED>
            <USER_DEFINED parameter="COUNTRY_CODE">ISS</USER_DEFINED>
            <USER_DEFINED parameter="LAUNCH_DATE">1998-11-20</USER_DEFINED>
            <USER_DEFINED parameter="SITE">TTMTR</USER_DEFINED>
            <USER_DEFINED parameter="DECAY_DATE"/>
            <USER_DEFINED parameter="FILE">4242354</USER_DEFINED>
            <USER_DEFINED parameter="GP_ID">252609161</USER_DEFINED>
          </userDefinedParameters>
        </data>
      </segment>
    </body>
  </omm>
</ndm>
```

Appendix C

Contents of the included storage media

This directory tree describes the structure of LEOSIM program on the attached CD disc.

```
/
├── program ..... included program files
│   ├── img ..... images used by the program
│   │   └── earth.jpg ..... image of the surface of Earth
│   ├── src ..... source codes of the program (see table 3.1)
│   ├── ui ..... internal file with layout of the ui
│   └── 3LE_input_data.txt ..... demo file with sample inputs
├── thesis ..... source codes of the LaTeX thesis
├── doxygen ..... folder with Doxygen utility files
├── DOXYGEN.html ..... Doxygen documentation of the source code
├── README.md ..... README manual for program installation
└── xpospi0k-simulation_environment_for_LEO.pdf ..... text of the thesis
```