'A forz 'e l'ambizione è chell c'avvicin 'e stell a terr

"The power of ambition is what brings the stars closer to the ground"

- Co'Sang

Abstract

perché

Acknowledgments

Thanksss

Contents

List of Figures

List of Tables

List of Abbreviations and Acronyms

API	Application Programming interface
GMAT	General Mission Analysis Tool
JIT	just-in-time
LEO	Low Earth Orbit
MIT	Massachusetts Institute of Technology
NASA	National Aeronautics and Space Administration
RAAN	Right Ascension of the Ascending Node
SSO	Sun-Synchronous Orbit
TLE	Two-Line Element

Chapter 1

Introduction

1.1 Earth Observation and Remote Sensing

1.1.1 Hyperspectral imaging

1.2 Kuva Space

1.2.1 Company Vision

1.2.2 Infrastructure

1.2.3 Distribution

1.2.4 Applications

1.3 Thesis Purpose

1.4 Organization

Chapter 2

Background

This chapter aims to provide a theoretical overview on the fundamentals of Space Flight Dynamics that constitute the foundations of this thesis, with a specific focus on Earth Observation applications in Low Earth Orbit (LEO), as well as a literature review on orbit management methods addressed by this work.

2.1 Space Flight Dynamics Overview

This paragraph is intended as an introductory overview in space flight dynamics. Orbital Mechanics, Astrodynamics, Astronautics and Space Flight Dynamics are all titles of university courses whose principal topic is two-body orbital motion, that involves orbit determination, orbital flight time, and orbital maneuvers [5]. In this context, a proper definition of the subject can be the following: the study of the motion of man-made objects in space, subject to both natural and artificially induced forces [4].

2.1.1 Satellite State Representations

To define the *state* of a satellite in space six quantities are required, and they may take on many equivalent forms. Whatever the form, the collection is called either

a *state vector*, usually associated with position and velocity vectors, or an *element set*, typically used with scalar magnitude and angular representations of the orbit called *orbital elements*. Either set of quantities completely specify the two-body orbit and provide a complete set of initial conditions for solving an initial value problem class of differential equations. Time is always associated with a state vector and is often considered a seventh component. State vectors and element sets are referenced to a particular coordinate frame [11].

This thesis will use both state vectors and a specific element set. The latter, which is also the most common one in this field, is represented by the ***classical orbital elements***:

- Semimajor axis, a: the orbit size is defined by one half of the major axis dimension

- Eccentricity, e: the ratio of minor to major dimensions of an orbit defines the shape

- Inclination, i: the angle between the orbit plane and the reference plane or the angle between the normals to the two planes

- Longitude of the ascending node or Right Ascension of the Ascending Node (RAAN), Ω: the angle between the vernal equinox vector and the ascending node measured in the reference plane in a couterclockwise direction as viewed from the northern hemisphere

- Argument of periapsis, ω: the angle from the ascending node to the periapsis, measured in the orbital plane in the direction of spacecraft motion. The ascending node is the point where the spacecraft crosses the reference plane headed from south to north. the line of nodes is the line formed by the intersection of the orbit plane and the reference plane

- True anomaly, ν: the sixth element locates the spacecraft position on the orbit

[1]

This thesis work makes also use of the *argument of latitude, u*, which is the angle measured between the ascending node and the satellite's position vector in the direction of satellite motion. A relation that is always valid exists between classical orbital elements and the argument of latitude [11]:

$$u = \omega + \nu \tag{2.1}$$

One more element that is often used is the *mean anomaly, M*, which is the angle between the periapsis of an orbit and the position of an imaginary body that orbits in the same period as the real one but at a constant angular speed (circular orbit). The angular speed assigned to the imaginary body is the satellite's average angular velocity over one orbit, and is called *mean motion, n* [7].

This research utilizes also another element set which results necessary to address case studies of past missions: the *Two-line element set* (TLE). A two-line element consists of a satellite identifier, an epoch, six orbital elements and a B^* term related to the ballistic coefficient [8]. These elsets are available to the general public through Air Force Space Command (AFSPC) [11].

2.1.2 Reference Frames

lllllllllllll

Chapter 3

Methodology

Exclusively open-source resources have been used to carry out the thesis work. In particular, the orbital scenarios under examination have been simulated through dynamic propagation models created in Python, taking advantage of existing free Python libraries. In order to validate the results, the General Mission Analysis Tool (GMAT) has been chosen as the reference software to make comparisons. In the following paragraphs a brief description of the tools mentioned previously is presented.

3.1 Python for Astrodynamics Applications

Due to the computationally intensive nature of astrodynamics tasks, astrodynamicists have relied on compiled programming languages such as Fortran for the development of astrodynamics software. Interpreted languages like Python on the other hand offer higher flexibility and development speed thereby increasing the productivity of the programmer. While interpreted languages are generally slower than compiled languages, recent developments such as just-in-time (JIT) compilers or transpilers have been able to close this speed gap significantly. Another important factor for the usefulness of a programming language is its wider ecosys-

tem which consists of the available open-source packages and development tools including integrated development environments and debuggers.

In light of the above, Python can be considered as a suitable language for scientific computing due to the existence of well established libraries like NumPy and SciPy for mathematical calculations. Not only do these libraries make use of compiled code from existing accelerated libraries, but it is also possible in Python to interface with compiled code to speed up specific algorithms where required. This allows a user to define a problem in Python, while benefiting from the speed of compiled languages for computationally expensive algorithms. Indeed, JIT-compiled dynamic languages such as Python with Numba have reached a competitive level of performance while still offering the advantages of lower complexity and better programmer productivity [3].

And not for nothing, nowadays, by a wide margin, Python is the most popular interpreted language in astronomy [6].

3.1.1 Numba

Arguably, the most important library in the scientific Python stack is NumPy, which implements n-dimensional arrays and its related methods in C programming language, and wraps them using the CPython API (Application Programming interface). It is a fundamental piece of software that powers most numerical codes written in Python nowadays. However, while it is possible to vectorize certain kinds of numerical operations, there might be other cases where this may not be feasible and where the dynamic nature of Python leads to a performance penalty, especially when the algorithm involves several levels of nested looping. To overcome these limitations it is possible to use Numba, an open-source library which can infer types for array-oriented and math-heavy Python code and generate optimized machine instructions using the LLVM compiler infrastructure [9].

In brief, Numba is a JIT compiler for scientific Python, which allows to optimize the running time.

3.1.2 poliastro

poliastro is an open-source Python library for astrodynamics and orbital mechanics released under the MIT (Massachusetts Institute of Technology) license. It features algorithms which are written in pure Python and compiled using Numba. It is dedicated to astrodynamics applications, such as orbit propagation, resolution of the Kepler and Lambert problems, conversion between position and velocity vectors and classical orbital elements and orbit plotting [9]. In addition, thanks to Astropy, poliastro can perform seamless coordinate frame conversions and use proper pysical units and timescales. At the moment, poliastro is the longest-lived Python library for astrodynamics, has contributors from all around the world, and several New Space companies and people in academia use it [10].

3.2 General Mission Analysis Tool

The General Mission Analysis Tool is a software system for trajectory optimization, mission analysis, trajectory estimation, and prediction developed by NASA (National Aeronautics and Space Administration), the Air Force Research Lab, and private industry. GMAT is designed to model, optimize, and estimate spacecraft trajectories in flight regimes ranging from LEO to lunar applications, interplanetary trajectories, and other deep space missions. GMAT's design and implementation are based on four basic principles: open-source visibility for both the source code and design documentation; platform independence; modular design; and user extensibility [2].

Chapter 4

Satellite Constellation Management Tools

4.1 Orbit Propagators

4.1.1 Undisturbed Motion

4.1.2 Perturbations

4.1.3 Atmospheric Models

4.1.4 Mean Orbital Elements Converter

4.1.5 Sun Synchronous Orbits Functions

4.1.6 Satellite Constellation Propagator

4.2 Revisit Time Collector

4.3 Station-Keeping Simulator

4.4 Differential Drag Algorithm

Chapter 5

Case Studies

5.1 Reaktor Hello World

5.2 Hyperfield Next Generation

5.3 Planet CubeSat Constellation

5.4 Future Kuva Constellation

Chapter 6

Analysis and Results

6.1 Reaktor Hello World Life Data

6.2 Hyperfield Orbit Maintenance Design

6.3 Planet Constellation Differential Drag Results

6.4 Kuva Constellation Management Results

Chapter 7

Conclusion

Donec et nisl id sapien blandit mattis. Aenean dictum odio sit amet risus. Morbi purus. Nulla a est sit amet purus venenatis iaculis. Vivamus viverra purus vel magna. Donec in justo sed odio malesuada dapibus. Nunc ultrices aliquam nunc. Vivamus facilisis pellentesque velit. Nulla nunc velit, vulputate dapibus, vulputate id, mattis ac, justo. Nam mattis elit dapibus purus. Quisque enim risus, congue non, elementum ut, mattis quis, sem. Quisque elit.

Bibliography

[1] Brown, C. D. *Spacecraft mission design.* AIAA, 1998.

[2] Conway, D. J., and Hughes, S. P. The general mission analysis tool (gmat): Current features and adding custom functionality. In *International Conference on Astrodynamics Tools and Techniques (ICATT)* (2010), no. LEGNEW-OLDGSFC-GSFC-LN-1107.

[3] Eichhorn, H., Cano, J. L., McLean, F., and Anderl, R. A comparative study of programming languages for next-generation astrodynamics systems. *CEAS Space Journal 10* (2018), 115–123.

[4] Griffin, M. D. *Space vehicle design.* AIAA, 2004.

[5] Kluever, C. A. *Space flight dynamics.* John Wiley & Sons, 2018.

[6] Momcheva, I., and Tollerud, E. Software use in astronomy: an informal survey. *arXiv preprint arXiv:1507.03989* (2015).

[7] Ridpath, I. *A dictionary of astronomy.* Oxford Quick Reference, 2012.

[8] Riesing, K. Orbit determination from two line element sets of iss-deployed cubesats.

[9] Rodríguez, J. L. C., Eichhorn, H., and McLean, F. Poliastro: an astrodynamics library written in python with fortran performance. In *6th International Conference on Astrodynamics Tools and Techniques* (2016).

[10] Rodríguez, J. L. C., and Garrido, J. M. poliastro: a python library for interactive astrodynamics.

[11] Vallado, D. A. *Fundamentals of Astrodynamics and Applications*, fourth ed. Microcosm Press, 2013.