



UNIVERSITAT POLITÈCNICA DE CATALUNYA  
BARCELONATECH

Escola Superior d'Enginyeries Industrial,  
Aeroespacial i Audiovisual de Terrassa

---

# STUDY ON ORBITAL PROPAGATORS

## CONSTELLATION ANALYSIS WITH NASA 42 AND MATLAB/SIMULINK

---

### REPORT

Author: IVÁN SERMANOUKIAN MOLINA

Director: DAVID GONZÁLEZ DIEZ

Co-director: MIQUEL SUREDA ANFRES

Bachelor's degree in Aerospace Technologies Engineering

Technical University of Catalonia

June 25, 2021

# Abstract

Since the beginning of the space age, satellite design philosophy was dominated by conservative designs built with highly reliable components to endure extreme environmental conditions. During the last two decades, the dawn of the CubeSats has changed this philosophy enabling a whole world of new possibilities.

The deployment of monumental CubeSat constellations in low Earth orbit is set to revolutionise the space sector by enabling faster and economical innovation cycles. However, CubeSat reliability is still considered an obstacle due to the sizeable fail rates among universities and companies, generally attributed to the dead-on-arrival cases and subsystem malfunctions.

This thesis is developed in the framework of the PLATHON research project that intends to develop a Hardware-in-the-loop emulation platform for nanosatellite constellations with optical inter-satellite communication and ground-to-satellite links. A crucial aspect of this project is to have a sufficiently precise orbital propagator with real-time manoeuvring control and graphical representation.

The available propagator programmes are analysed to select NASA's OpenSatKit, a multi-facet platform with an inbuilt propagator known as 42. The purpose of this dissertation is to analyse the implementation feasibility of the programme for the creation of a constellation testing bench compared to previously self-developed propagators based on MATLAB/Simulink.

The initial documentation is a scouting approach to examine 42's capabilities under distinct scenarios to adapt the PLATHON system to the programme's inner workings and constraints.

The programme modifications and simulations pave the way for the future development of the interconnected PLATHON network; specifically, the inter-process communication capabilities have been tested to imitate the inputs of spacecraft attitude control systems through bidirectional socket interfaces.

## **Keywords:**

Orbit Propagation, Nanosatellites, Constellations, Inter-Process Communication

# Abstract

Desde el comienzo de la era espacial, la filosofía de diseño de satélites estuvo dominada por diseños conservadores contruidos con componentes altamente duraderos para soportar condiciones ambientales extremas. Durante las últimas dos décadas, la aparición de los CubeSats ha cambiado esta filosofía permitiendo todo un mundo de nuevas posibilidades.

El despliegue de grandes constelaciones de CubeSats en órbita terrestre baja (LEO, en inglés) revolucionará el sector espacial al permitir ciclos de innovación más rápidos y económicos. Sin embargo, la confiabilidad de los CubeSats todavía se considera un obstáculo debido a las considerables tasas de fallo entre universidades y empresas, generalmente atribuidas a casos de pérdida completa de misión tras la eyección del desplegador orbital y al fallo de los subsistemas.

Esta tesis se desarrolla en el marco del proyecto de investigación PLATHON, que pretende desarrollar una plataforma de emulación Hardware-in-the-loop para constelaciones de nanosatélites con comunicación óptica entre satélites y enlaces tierra-satélite. Un aspecto crucial de este proyecto es tener un propagador orbital suficientemente preciso con control de maniobras y representación gráfica en tiempo real.

Los programas de propagadores disponibles se han analizado para seleccionar el sistema OpenSatKit de la NASA, una plataforma multifacética con un propagador incorporado conocido como 42. El propósito de esta disertación es analizar la viabilidad de implementación del programa para la creación de un banco de pruebas de constelaciones en comparación con un propagador previo desarrollado en MATLAB/Simulink.

La documentación inicial es un enfoque de exploración para examinar las capacidades del 42 en distintos escenarios con objeto de adaptar el sistema PLATHON al funcionamiento interno y las limitaciones del programa.

Las modificaciones y simulaciones del programa allanan el camino para el futuro desarrollo de la red interconectada PLATHON; específicamente, las comunicaciones entre procesos se han probado para imitar las entradas de los sistemas de control de actitud de las naves espaciales a través de interfaces de comunicación bidireccionales.

## **Palabras clave:**

Propagación orbital, Nanosatélites, Constelaciones, Comunicación entre procesos

# Acknowledgements

I want to highlight the support of my thesis director, David González, and all of the PLATHON project individuals who have contributed to the development of this thesis.

To my father, for transmitting his passion for technology since I was a child.

To my mother, for her unconditional support during all these years.

To my sister, for introducing me into the aerospace sector.

To my grandparents, for laying my foundations with their wisdom.

Especially to my grandfather, who I know would be proud of me.

To my friends, for morally supporting me and making these years a memorable experience.



# Declaration on Honour

I declare that,

the work in this Degree Thesis is completely my own work,

no part of this Degree Thesis is taken from other people's work without giving them credit,

all references have been clearly cited,

I'm authorised to make use of the research group TIEG and PLATHON related information I'm providing in this document.

I understand that an infringement of this declaration leaves me subject to the foreseen disciplinary actions by *The Technical University of Catalonia - BarcelonaTECH*.

Iván Sermanoukian Molina

25 June 2021

---

Name of the Student

---

Signature

---

Date

Thesis title:

**Study on orbital propagators: Constellation analysis with NASA 42 and Matlab/Simulink**

# Contents

<b>Abstract</b>	<b>I</b>
<b>Acknowledgements</b>	<b>I</b>
<b>Declaration on Honour</b>	<b>II</b>
<b>List of Figures</b>	<b>IX</b>
<b>List of Tables</b>	<b>X</b>
<b>List of Codes</b>	<b>XI</b>
<b>Nomenclature</b>	<b>XII</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Aim . . . . .	1
1.2 Scope . . . . .	1
1.3 Requirements . . . . .	2
1.4 Motivation . . . . .	2
<b>2 State of the art</b>	<b>3</b>
2.1 Orbit propagation . . . . .	3
2.2 Types of orbit propagation . . . . .	4
2.2.1 Analytic orbit propagation . . . . .	4
2.2.2 Semi-Analytic Propagation . . . . .	6
2.2.3 Numerical Orbit propagation . . . . .	7
2.2.4 Programmes . . . . .	9
2.3 PLATHON project . . . . .	15
2.3.1 OpenSatKit and NS3 . . . . .	18
2.3.2 Step-by-step PLATHON system . . . . .	19
2.4 Satellite classification . . . . .	21
2.4.1 The CubeSat standard . . . . .	21

2.5	Satellite constellations . . . . .	23
2.5.1	Walker Pattern . . . . .	24
2.6	Optical communications . . . . .	25
2.6.1	Quantum cryptography . . . . .	26
<b>3</b>	<b>Theoretical concepts</b>	<b>27</b>
3.1	Orbital perturbations . . . . .	27
3.1.1	Earth's Gravitational Potential . . . . .	28
3.1.2	Aerodynamic Drag . . . . .	28
3.1.3	Solar Radiation Pressure . . . . .	30
3.1.4	Spacecraft Manoeuvres . . . . .	31
3.2	Coordinate systems . . . . .	32
3.2.1	Heliocentric Coordinate System . . . . .	32
3.2.2	Inertial and rotating frames . . . . .	33
3.2.3	Local Vertical - Local Horizon Frame . . . . .	34
3.2.4	Body Frame . . . . .	34
3.2.5	Formation Frame . . . . .	34
3.3	Rotations . . . . .	35
3.3.1	Euler angles . . . . .	35
3.3.2	Quaternions . . . . .	36
3.4	Satellite dynamics . . . . .	38
3.4.1	The Tensor of Inertia . . . . .	38
3.4.2	Angular momentum . . . . .	38
3.5	Attitude Determination and Control System . . . . .	39
3.5.1	Attitude Determination . . . . .	39
3.5.2	Attitude Control actuators . . . . .	41
<b>4</b>	<b>42: Spacecraft Simulation</b>	<b>42</b>
4.1	Installation and testing . . . . .	42
4.1.1	Validation . . . . .	43
4.2	File distribution . . . . .	44
4.2.1	Algorithm folders . . . . .	45
4.3	Simulation procedure . . . . .	46
4.4	Input files . . . . .	48
4.4.1	Simulation Control . . . . .	49
4.4.2	Command Script . . . . .	51
4.4.3	Reference Orbits . . . . .	52
4.4.4	Spacecraft . . . . .	54

4.4.5	Regions . . . . .	60
4.4.6	TDRS . . . . .	61
4.4.7	Inter-Process Communications . . . . .	62
4.4.8	Graphics Configuration . . . . .	65
4.4.9	Field of View . . . . .	65
4.5	Output files . . . . .	66
<b>5</b>	<b>42: PLATHON project</b>	<b>67</b>
5.1	Report modifications . . . . .	68
5.2	MATLAB GUI . . . . .	69
5.2.1	Use of MATLAB GUI . . . . .	70
5.3	42 file creation and testing . . . . .	71
5.4	Constellation . . . . .	74
5.4.1	IRIDIUM NEXT constellation . . . . .	75
5.5	InterProcess Communication . . . . .	77
5.5.1	Architecture I . . . . .	77
5.5.2	Architecture II . . . . .	79
5.6	Satellite propagation . . . . .	82
5.6.1	CubeSat . . . . .	83
5.6.2	International Space Station . . . . .	85
5.7	ISS comparison with SPICE . . . . .	86
5.7.1	NAIF Integer ID codes . . . . .	86
5.8	Comparison with Matlab models . . . . .	88
<b>6</b>	<b>Project Planning</b>	<b>89</b>
6.1	Work Breakdown Structure . . . . .	89
6.1.1	Task identification . . . . .	89
6.2	Gantt diagram . . . . .	91
<b>7</b>	<b>Budget</b>	<b>93</b>
<b>8</b>	<b>Environmental impact</b>	<b>95</b>
<b>9</b>	<b>Conclusions and future prospects</b>	<b>96</b>
9.1	Conclusions . . . . .	96
9.2	Future prospects . . . . .	97
<b>A</b>	<b>Data appendix</b>	<b>103</b>
A.1	TLE data format . . . . .	104
A.1.1	TLE Line 1 . . . . .	104

A.1.2	TLE Line 2 . . . . .	105
A.2	Numerical methods for ODE systems . . . . .	106
A.2.1	Explicit Euler Method . . . . .	106
A.2.2	Implicit Euler Method . . . . .	106
A.2.3	Implicit Midpoint Rule . . . . .	107
A.2.4	Runge Kutta methods . . . . .	107
A.3	42 Images . . . . .	108
A.4	5th ESA CubeSat Industry Days . . . . .	119
<b>B</b>	<b>Algorithms</b>	<b>120</b>
B.1	42 Command Script . . . . .	121
B.2	42 Software . . . . .	123
B.2.1	Kit . . . . .	123
B.2.2	Include . . . . .	126
B.2.3	Source . . . . .	127
B.3	Algorithms . . . . .	129
B.3.1	Report function . . . . .	129
B.3.2	Constellation Input File . . . . .	133
B.3.3	Constellation Output File . . . . .	136
B.3.4	TLE acquisition . . . . .	138
B.3.5	IPC Architecture I . . . . .	139
B.3.6	IPC Architecture II . . . . .	143

# List of Figures

2.1	2D Propagation diagram . . . . .	3
2.2	Two body problem . . . . .	4
2.3	Free body diagram . . . . .	4
2.4	3D Comparison $45^\circ$ inclination orbit. . . . .	5
2.5	Distance between orbits with 42 . . . . .	5
2.6	N-body simulation with Runge-Kutta-Nystrom 10-12 developed under Dr. Arnau Miró supervision. Code available on GitHub [8] . . . . .	7
2.7	Orbit Propagation diagram . . . . .	8
2.8	GMAT Desktop, Windows version. Source: [9]. . . . .	9
2.9	42 complete demonstration with GUI . . . . .	10
2.10	OpenSatKit connection diagram. Source: [12]. . . . .	11
2.11	COSMOS Architecture. Source: [13]. . . . .	12
2.12	Top-level Orekit packages. Extracted from [16]. . . . .	13
2.13	ISS Orbit. Source: Ivan Parrot [18] . . . . .	14
2.14	IoT data transmission mission . . . . .	15
2.15	PLATHON constellation diagram. Source: [1]. . . . .	16
2.16	PLATHON system for emulation of communication missions between CubeSats and Earth	17
2.17	PLATHON project diagram . . . . .	18
2.18	<sup>3</sup> Cat-4, UPC's NanoSat Lab. Source: [20]. . . . .	21
2.19	CubeSat sizes and mass. Source: [21]. . . . .	22
2.20	Total Nanosats and CubeSats launched. Source: [22]. . . . .	22
2.21	Iridium constellation routing signals in orbit. Source: [26] . . . . .	24
2.22	Difference in data rates between RF and optical communications. Source [29]. . . . .	25
3.1	A comparison of the disturbing accelerations for the main sources of perturbation. Source: [32]. . . . .	27
3.2	NASA - US Standard Atmosphere 1976. Data extracted from [35]. . . . .	29
3.3	MSIS-86 Thermospheric Model. Data extracted from [33]. Plotted with [37]. . . . .	29
3.4	Global average thermospheric mass density at an altitude of 400 km. Extracted from [38].	30

3.5	ISS orbit desaturation with pyramidal configuration. Extracted from [18]. . . . .	31
3.6	Heliocentric Coordinate System. Extracted from [5]. . . . .	32
3.7	Earth Centred Inertial Frame. Extracted from [5]. . . . .	33
3.8	CubeSat Body frame with 42. . . . .	34
3.9	CubeSat LVLH frame with 42. . . . .	34
3.10	CubeSat Formation frame with 42. . . . .	34
3.11	Euler angles rotations . . . . .	35
3.12	Rotation of an angle $\theta$ around a unit vector $\vec{e}$ . . . . .	37
3.13	HRG. Extracted from [43]. . . . .	39
3.14	3-Axis Magnetometer. Extracted from [44]. . . . .	39
3.15	Coarse Sun Sensor. Extracted from [43]. . . . .	40
3.16	Fine Sun Sensor. Extracted from [44]. . . . .	40
3.17	Star image and its amplitude taken in LEO. Extracted from [47]. . . . .	40
3.18	PLATHON reaction wheel prototype . . . . .	41
3.19	PLATHON prototype . . . . .	41
3.20	PLATHON magnetorquer prototype . . . . .	41
3.21	PLATHON prototype . . . . .	41
4.1	Linux terminal running example programme without GUI . . . . .	43
4.2	Main folder connections . . . . .	44
4.3	42 Simulation loop . . . . .	47
4.4	Input files flowchart with dependencies . . . . .	48
4.5	Global Data Structure Relationships. Source: [50]. . . . .	57
4.6	TDRS configuration, March 2019. Source: [51]. . . . .	61
4.7	Standalone IPC Simulation . . . . .	63
4.8	Tx-Rx IPC Simulation . . . . .	65
4.9	FOV representation . . . . .	65
5.1	Simulink 3D Animation . . . . .	69
5.2	Two satellite 3D Animation . . . . .	70
5.3	Iridium NEXT simulation (2021/06/14). 3D Camera view. . . . .	75
5.4	Iridium NEXT simulation (2021/06/14). 2D Map view. . . . .	75
5.5	Iridium NEXT simulation (2021/06/14). Orrery view. . . . .	76
5.6	42 system architecture I . . . . .	77
5.7	IPC Architecture I - Terminal distribution . . . . .	78
5.8	42 system architecture II . . . . .	79
5.9	IPC Architecture II - Terminal distribution . . . . .	80
5.10	IPC Architecture II - Initial attitude . . . . .	81

5.11 IPC Architecture II - Modified attitude . . . . .	81
5.12 TLE update rate of CubeSats and PocketQubes. Extracted from [52]. . . . .	82
5.13 SOMP CubeSat. Extracted from [53] . . . . .	83
5.14 Forward propagation . . . . .	84
5.15 Forward propagation . . . . .	85
5.16 ISS orbit Start: (2012-04-19 T00:00:00) NDAYs = 0.1; . . . . .	86
5.17 ISS orbit difference between SPICE and NASA 42. [2005/04/19] . . . . .	87
5.18 ISS orbit difference between SPICE and NASA 42. [2012/04/19] . . . . .	87
6.1 Work Breakdown Structure . . . . .	90
6.2 Weekly time distribution . . . . .	92
A.1 $[a, b]$ interval . . . . .	106
A.2 42 Cam - Voyager mission . . . . .	108
A.3 42 2D map - Saturn with Voyager mission . . . . .	108
A.4 42 Cam - Rosetta space probe . . . . .	109
A.5 42 2D map - 67P/Churyumov–Gerasimenko comet . . . . .	109
A.6 42 Cam - 6U + 3U CubeSat formation . . . . .	110
A.7 42 Unit Sphere Viewer - Major celestial bodies . . . . .	110
A.8 42 Cam - Space Shuttle . . . . .	111
A.9 42 Unit Sphere Viewer - Constellations . . . . .	111
A.10 42 Cam - International Space Stations . . . . .	112
A.11 42 2D map - ISS and CubeSat orbits . . . . .	112
A.12 42 Cam - CubeSat axes . . . . .	113
A.13 42 2D map - Jupiter . . . . .	113
A.14 42 Cam - Truth vectors . . . . .	114
A.15 42 Cam - Flight Software vectors . . . . .	114
A.16 42 Cam - Fermi sky . . . . .	115
A.17 42 Unit Sphere Viewer - Spacecraft vectors . . . . .	115
A.18 42 Unit Sphere Viewer - Body frame . . . . .	116
A.19 42 Unit Sphere Viewer - Body and LHLV frames . . . . .	116
A.20 42 Unit Sphere Viewer - Inertial and LHLV frames . . . . .	116
A.21 42 Orrery - Solar System . . . . .	117
A.22 42 Orrery - Sun Earth Lagrange points . . . . .	118
A.23 42 Orrery - Earth orbits . . . . .	118



# List of Tables

2.1	Oblateness and second zonal harmonics. Source: [5].	5
2.2	42 GUI window description	10
2.3	Constellation design issues. Source: [25].	23
3.1	Radiation pressure coefficient	30
3.2	Quaternion multiplication table	36
5.1	Satellite information	82
5.2	Leap seconds according to date	87
6.1	Task identification with code	89
6.2	Gantt diagram	91
7.1	Costs Breakdown	93
7.2	Total costs	94
8.1	Environmental impact with carbon emissions	95
A.1	Line 0: Title line of Starlink-1919 TLE	104
A.2	Line 1: Starlink-1919 TLE	104
A.3	Line 1: Content description. Data from CelesTrak [60].	104
A.4	Line 2: Starlink-1919 TLE	105
A.5	Line 2: Content description. Data from CelesTrak [60].	105

# Algorithms

B.1	Report . . . . .	129
B.2	Constellation Input File . . . . .	133
B.3	Constellation Output File . . . . .	136
B.4	TLE acquisition . . . . .	138
B.5	AllocateAC function . . . . .	140
B.6	SetPoint function . . . . .	141
B.7	main function . . . . .	142
B.8	Makefile function modifications . . . . .	144

# Nomenclature

## Acronyms

AcApp	Attitude Control Application
ADCS	Attitude Determination and Control System
cFS	core Flight System
COMS	Communication System
ECEF	Earth-centred-Earth-fixed
ECI	Earth-centred inertial
EPS	Electric Power System
FSW	Flight Software
GEO	Geosynchronous Equatorial Orbit
GNC	Guidance, Navigation and Control
GUI	Graphical User Interface
HITL	Hardware-in-the-Loop
LEO	Low Earth Orbit
MDL	Mission Design Lab
NAIF	Navigation and Ancillary Information Facility
NASA	National Aeronautics and Space Administration
OBC	Onboard Computer
OS	Operating System
PPOD	Poly Picosatellite Orbiting Deployers
PU	Pattern Unit
RF	Radio frequency
SGP	Simplified General Perturbations
SITL	Software-in-the-Loop
TDRS	Tracking and Data Relay Satellite

TIEG Terrassa Industrial Electronics Group

TLE Two-Line Element

UPC Universitat Politècnica de Catalunya

VE Vernal Equinox

VM Virtual Machine

## Symbols

$\lambda$  Longitude

$\Omega$  Right Ascension of the Ascending Node

$\theta$  True anomaly

$w$  Argument of the periapsis

$i$  Inclination

$L$  Latitude

$m$  Mass

## Physical constants

$G$  Gravitational Constant  $6.67384 \times 10^{-11} \text{ N} \cdot \text{m}^2/\text{kg}^2$

# Chapter 1

## Introduction

### 1.1 Aim

This study aims to analyse the feasibility of NASA's 42 propagator in the PLATHON<sup>1</sup> project to simulate CubeSat constellations with laser-based inter-satellite communication and ground-to-satellite links.

### 1.2 Scope

The depth of this thesis is summarised as follows:

- Study orbital propagator types.
- Analysis of constellations models for the PLATHON project.
- Study CubeSat optical communications and connectivity restrictions.
- Comparison of available orbital propagators.
- Installation and implementation of NASA 42 on Linux and Windows OS.
- Analysis of 42 propagator capabilities.
- Analysis of 42 environment and spacecraft models.
- Development of a Matlab algorithm to modify the initial conditions and run 42.
- Development of a Matlab algorithm to read the output data from the programme.
- Development of a C algorithm to save data from all bodies.
- Development of a C algorithm to test Attitude Control.
- Study the feasibility of creating a Matlab Graphical User Interface.
- Analysis of the intercommunication between NASA's OpenSatKit modules for the entire GNC testing and verification.
- Interface of Virtual Machines and computers for resource sharing.
- Compare the results with previous software models.

On the contrary, any aspect not mentioned in this list is considered out of scope.

---

<sup>1</sup>Integrated Hardware in the loop simulation PLATform of Optical communications in Nanosatellites.  
(from Spanish: PLATaforma integrada de simulación Hardware-in-the loop para comunicaciones Ópticas en Nanosatélites)

## 1.3 Requirements

- CubeSat design
- Low Earth Orbit
- Open-Source software
- Laser-based inter-satellite communication
- Ground-to-satellite RF links
- Cooperation among PLATHON project students and professors

## 1.4 Motivation

The PLATHON project, promoted by UPC's Terrassa Industrial Electronics Group (TIEG), aims to simulate a constellation scenario with inter-satellite optical communications based on CubeSat technologies.

CubeSats appeared as a teaching tool in universities during the end of the last century but immediately shifted from university laboratories to commercial applications. They have become a mature standard in the aerospace industry, allowing the first CubeSat-based constellations to offer global communication coverage. However, TIEG considers that there are some absences in two areas related to constellations based missions [1]:

- **Simulation tools:** current communication networks simulators used by the scientific community (NS2, SNS3 and Opensand) do not have the features required to design communications among nodes stationed at LEO, between LEO and GEO and beyond. Furthermore, there is neither a feature to simulate ground stations nor the power availability of nodes to establish proper communications is considered.
- **Reliability:** CubeSat missions still have a sizeable fail rate (55% in CubeSat from Universities and 33% in those from Companies) [2]. Most of these failures are attributed to a subsystem malfunction due to little to no subsystem-level testing.

To successfully address nanosatellite constellation mission simulation, it is essential to combine orbital propagation tools with command and telemetry transmissions between satellites and ground stations; and the operational status of the satellites. In missions in which satellites are designed to act as communication nodes, it is also necessary to integrate communication network models.

The software applications can be validated by simulating hardware behaviour and emulating the flight conditions. This kind of software validation is referred to as Software-in-the-Loop (SITL).

The next step to guarantee the success of a mission is to include the satellite's hardware operation verification. This aspect is covered when the actual hardware is incorporated into the simulation and validation environment. This concept is referred to as Hardware-in-the-Loop (HITL).

The PLATHON project aims to develop a HITL integrated platform with network and orbital simulators that allow the verification of a CubeSat constellation communications network considering the operational restrictions of the nodes that form it. Specifically, it is intended to address the current need to design low Earth orbit missions aimed at establishing a communication network between satellites of the same and different orbits and ground stations.

Therefore, this bachelor's thesis is aimed to analyse the feasibility of NASA's OpenSatKit 42 orbital propagator module for the implementation of a HITL PLATHON system.

# Chapter 2

## State of the art

This chapter presents orbit propagation techniques and introduces the programmes that the PLATHON project could use along with the reasons to choose the 42 propagator. Then, satellites are classified to describe the CubeSat standard and constellations. Finally, laser communication milestones are included.

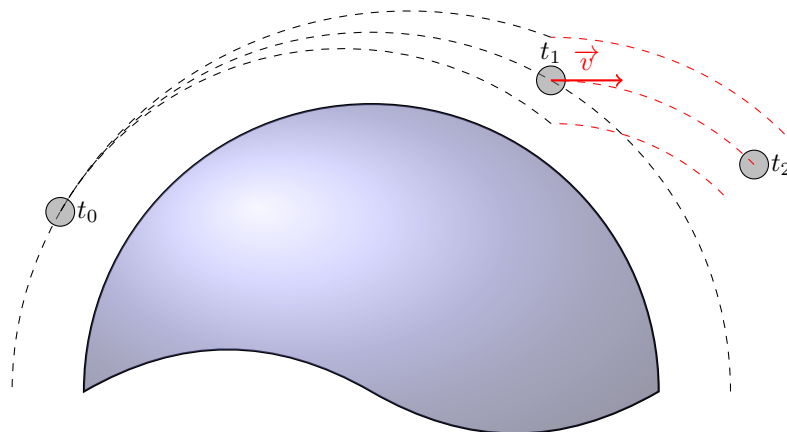
### 2.1 Orbit propagation

Orbit propagation consists of determining the orbital characteristics of an orbiting body such as a satellite at some future date, given the properties of an initial state [3].

Determined by the mission objectives and requirements, the set of equations describing the satellite motion is obliged to fulfil a minimum accuracy. At the same time, the models of the external perturbations depend on the available data. For instance, the satellite orbit altitude influences the selection of an atmosphere model for a circular orbit.

To preserve computational efficiency, external models can be typically simplified to solely consider the effects of dominant perturbations without incurring a significant error. If enough approximations are made, the problem can be solved analytically.

However, if some extra variables are included, there is no systematic solution. The complex modelling of these forces results in a highly nonlinear set of dynamical equations which can only be solved through numerical methods, the accuracy of which also influences propagation deviations [4]. For this reason, systems have been developed to suit specialised needs over the years while preserving speed and stability, purely focusing on the critical aspects of each mission.



**Figure 2.1** 2D Propagation diagram

## 2.2 Types of orbit propagation

Orbit propagation techniques are divided into two large groups, analytic and numerical methods, which can be combined into semi-analytic propagators. Following, each method is described with a set of examples:

### 2.2.1 Analytic orbit propagation

Fast, low-fidelity analytic propagators are used to design and test operations during the first development stages. However, they are not suitable for more advanced mission stages as well as to run current operations. This kind of propagation relies typically on a low number of bodies in which one body's mass is superlative concerning the others.

Analytic propagators require an approximation of the body motion to solve the system of equations. Also known as general perturbations, they replace the original equations with an analytical approach that captures the essential features of the problem. They can be used to model maintained orbits not requiring maintenance manoeuvres without incurring a significant inaccuracy.

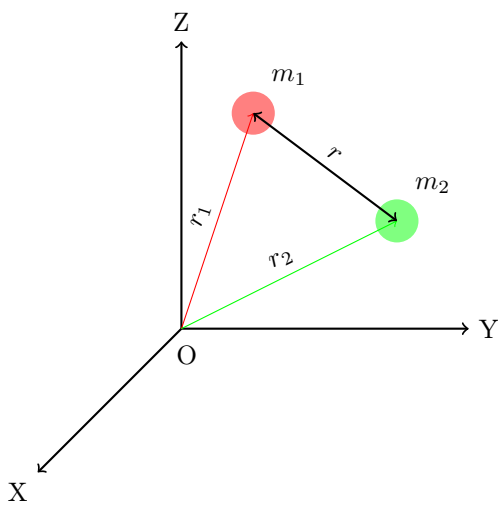
#### 2.2.1.1 Two-Body problem

The two-body equation of motion is handy for initial back-of-the-envelope calculations in the astrodynamics field. The two-body problem encompasses two objects abstractly considered point particles to generate ephemeris from the exact solution of the approximated equation considering only the effect of the other body. This assumption requires the bodies of the satellite and attracting body to be spherically symmetrical, with uniform density, incurring major error by neglecting the oblateness caused by rotational effects. Other perturbations are not considered.

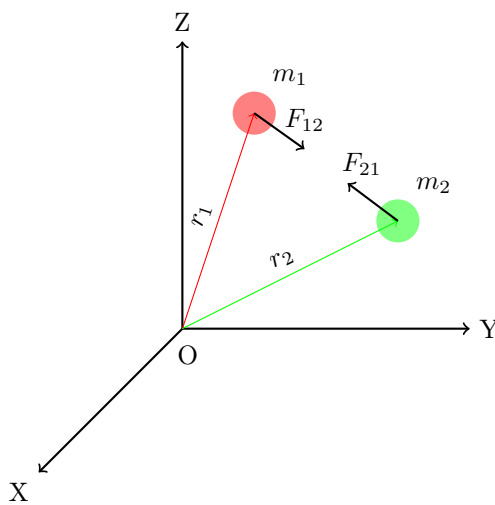
For two bodies of masses  $m_1$  and  $m_2$  at a distance  $r$ , combining Newton's laws of motion with his universal gravitation law, the resultant forces take on the form

$$\vec{F}_g = -\frac{Gm_1m_2}{r^2} \left( \frac{\vec{r}}{r} \right) \quad (2.2.1)$$

where  $G$  corresponds to the universal gravitational constant.



**Figure 2.2** Two body problem



**Figure 2.3** Free body diagram

By contrast, if more bodies with enough mass to exert minimal gravitational forces are brought into play, this case is transformed into an  $n$ -body problem, where  $n \geq 3$  can only be solved in analytical terms for exceptional cases. This method can be found on [2.2.3.1](#).



### 2.2.1.2 J2 Propagator

The J2 perturbation propagator is a first-order propagator which accounts for secular variations in the orbit elements due to a celestial body oblateness. In a planet such as Earth, lacking the perfect symmetry of a sphere, the force of gravity on an orbiting body is not directed towards its centre. Thus, oblateness causes a zonal variation which is quantified by the dimensionless parameter  $J_2$ , the second zonal harmonic [5]. The Earth's gravitational potential perturbation is explained in 3.1.1.

For rocky spheroidal planets, the oblateness can be obtained as

$$Oblateness = \frac{R_e - R_p}{R_e} \quad (2.2.2)$$

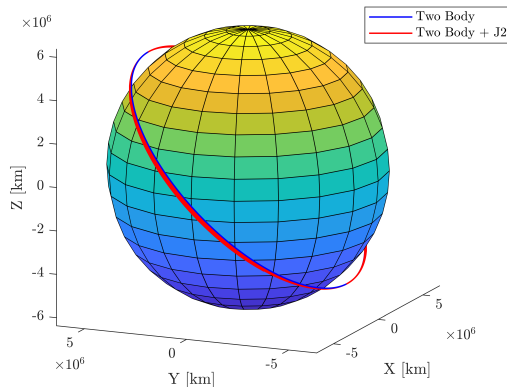
where  $R_e$  is the distance from the centre of the ellipsoid to the equator (semi-major axis), and  $R_p$  is the distance from the centre to the pole (semi-minor axis).

As it can be seen on the following table, which lists Solar System planets oblateness and  $J_2$  parameters, Earth's oblateness is caused by the polar radius being shorter than the equatorial radius by 21 km.

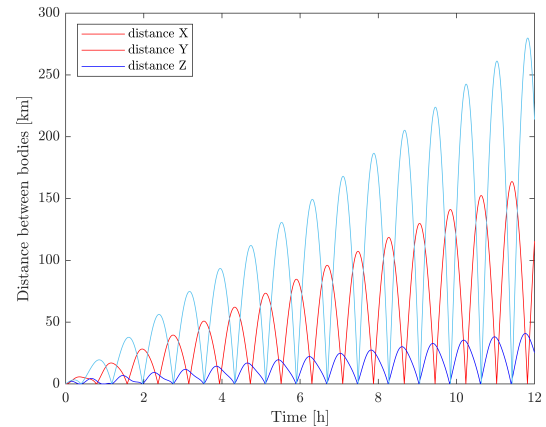
Planet	Oblateness	$J_2$ ( $10^{-6}$ )
Mercury	0.000	60
Venus	0.000	4.458
Earth	0.003353	1.08263
Mars	0.00648	1960.45
Jupiter	0.06487	14736
Saturn	0.09796	16298
Uranus	0.02293	3343.4
Neptune	0.01708	3411
Moon	0.0012	202.7

**Table 2.1** Oblateness and second zonal harmonics. Source: [5].

Following the same gravitational potential perturbation, the J4 perturbation propagator is a second-order propagator that includes the effects of the J2 propagator and the first-order effects of J4. However, as J4 is approximately a thousand times smaller than J2 (3.1), the variations are too small to be relevant for an analytic propagator. As a result, the J2 effect can still be viewed as a small but noticeable perturbation compared to the spherical Earth's attraction.



**Figure 2.4** 3D Comparison 45° inclination orbit.



**Figure 2.5** Distance between orbits with 42

## 2.2.2 Semi-Analytic Propagation

Semi-Analytic propagators are the middle ground between accuracy and computational speed. While they share some approximations with analytic propagators, some features have been enhanced to be obtained numerically by distinguishing between the perturbations' short- and long-period components.

### 2.2.2.1 Simplified general perturbation models

The simplified perturbation models are a set of mathematical models used to obtain orbital state vectors of a satellite or spaceship relative to the Earth-centred inertial coordinate system. The most commonly used model is SGP4 due to its combination with two-element data sets. Its results are substantially more precise than solely analytic methods because the model predicts the effects of Earth's shape, atmosphere drag and gravitational effects of major celestial bodies and stars such as the Moon and the Sun. The mean model error is approximately 1 km at epoch, increasing up to 1 to 3 km per day and is limited to near-Earth objects with an orbital period of fewer than 225 minutes or a 5,878 km radius, assuming a circular orbit.

#### 2.2.2.2 Two-Line Elements

A Two-Line Element (TLE) set is a data format that encodes orbital elements of an Earth-orbiting object for a specific epoch. An algorithm using TLE as a data source must implement simplified perturbation models to compute the state of an object during the time interval of interest. The TLE format is a de facto data standard for the distribution of Earth-orbiting object's orbital elements.

The North American Aerospace Defense Command and NASA are in charge of tracking all detectable objects in Earth orbit and create the corresponding TLE for each object. In recent years, TLEs have been used to characterise the orbital tracks of space debris.

An exemplary TLE of the Starlink constellation is presented below:

```
STARLINK-1919
1 46769U 20074AG 21128.33742380 .00000625 00000-0 60859-4 0 9994
2 46769 53.0569 346.0824 0001673 80.0957 280.0221 15.06396809 29963
```

#### Starlink-1919 TLE

NORAD Two-Line Element Sets Current Data for this thesis are obtained from CelesTrak [6] and Space Track [7].

The maximum number of Satellite Numbers which can be encoded in a TLE is rapidly being approached due to the recent commercialisation of space. Thus, it will result in adaptations of the TLE format in the foreseeable future.

The data format analysis of TLE can be found in Appendix [A.1].

## 2.2.3 Numerical Orbit propagation

The numerical approach provides a much more accurate result than analytical methods in exchange for computational time. The correct step size selection is crucial for both constant and variable step solvers. More information about numerical solvers can be found on the Numerical Methods Appendix [A.2].

### 2.2.3.1 N-body problem

The N-body problem is the extension of the Two-body problem for  $N \geq 3$ . While there are analytic solutions available for selected configurations, in general, the problem must be solved using numerical methods.

Given  $N$  bodies in an inertial reference frame in  $\mathbb{R}^3$  with an initial position  $\vec{x}_i$  and velocity  $\vec{v}_i$  for  $1 \leq i \leq N$ , the force vector  $\vec{f}_{ij}$  on body  $i$  caused by its gravitational attraction to body  $j$  is given by the following equation:

$$\vec{f}_{ij} = G \frac{m_i m_j}{|\vec{r}_{ij}|^2} \cdot \frac{\vec{r}_{ij}}{|\vec{r}_{ij}|} \quad (2.2.3)$$

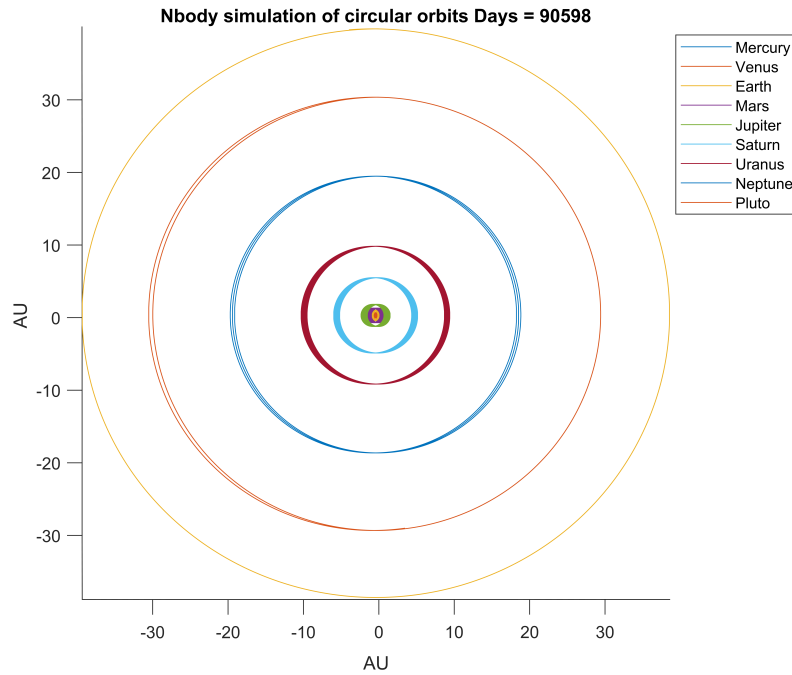
Where  $m_i$  and  $m_j$  are the masses of the bodies  $i$  and  $j$  respectively; and  $\vec{r}_{ij} = \vec{x}_j - \vec{x}_i$  is the distance between the position vectors.

Summing over all masses yields the  $N$ -body equations of motion:

$$\vec{F}_i = m_i \frac{d^2 \vec{x}_i}{dt^2} = \sum_{\substack{j=1 \\ j \neq i}}^n \vec{f}_{ij} = \sum_{\substack{j=1 \\ j \neq i}}^n \frac{G m_i m_j (\vec{r}_{ij})}{\|\vec{r}_{ij}\|^3} = - \frac{\partial U}{\partial \vec{x}_i} \quad (2.2.4)$$

where  $U$  is the self-potential energy

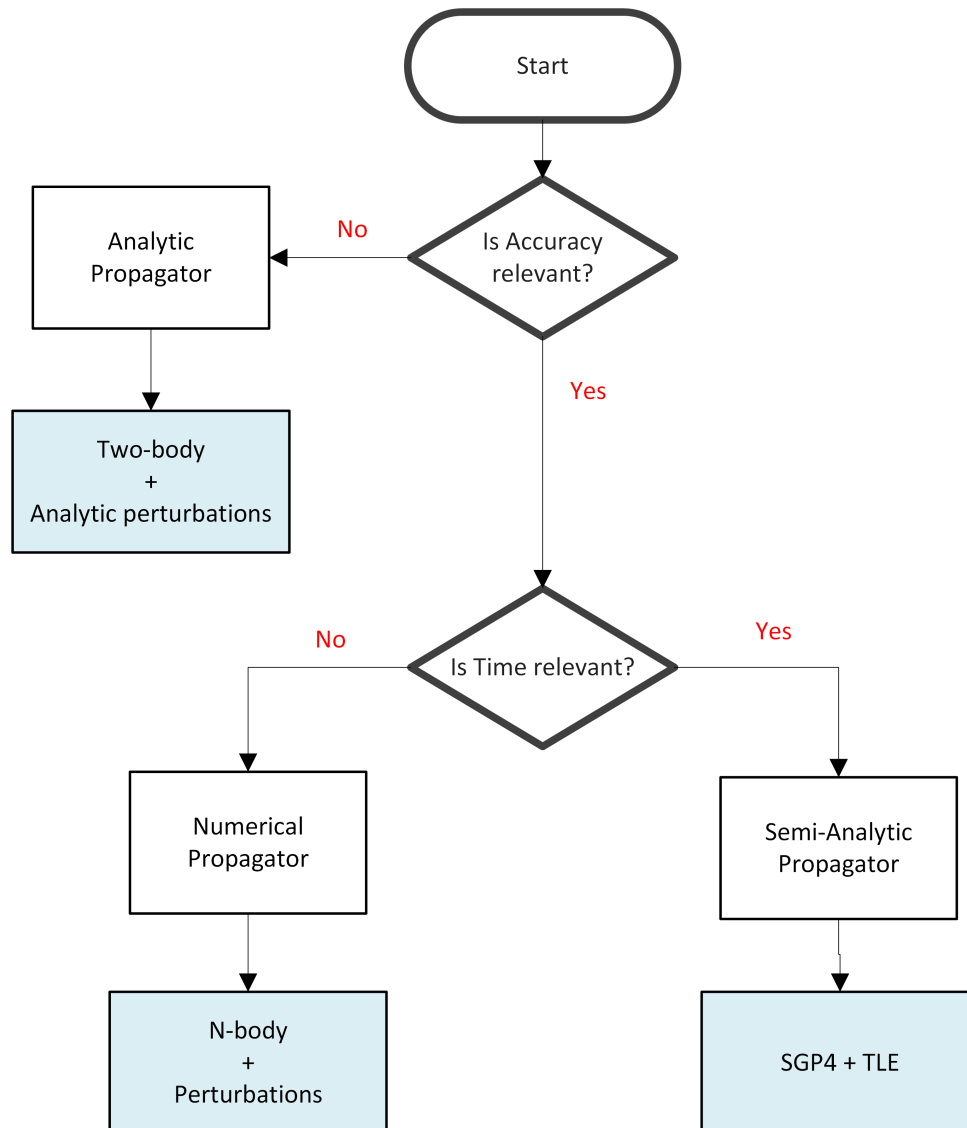
$$U = - \sum_{1 \leq i < j \leq n} \frac{G m_i m_j}{\|\vec{r}_{ij}\|} \quad (2.2.5)$$



**Figure 2.6** N-body simulation with Runge-Kutta-Nystrom 10-12 developed under Dr. Arnau Miró supervision. Code available on GitHub [8]

### 2.2.3.2 Orbit Propagation diagram

Once all the propagators have been explained, the following diagram determines when it is appropriate to select each one:



**Figure 2.7** Orbit Propagation diagram

## 2.2.4 Programmes

In order to propagate orbits, there are several programmes, both open-source and private, which can perform the simulations with more or less detail depending on the mission requirements.

Open-source programmes include national aerospace agencies such as NASA's GMAT and 42, and ESA's Orekit. On the other hand, Systems Tool Kit is a private programme property of Analytical Graphics, Inc..

These programmes have been briefly analysed for the PLATHON project and are presented below:

### 2.2.4.1 NASA General Mission Analysis Tool (GMAT)

The General Mission Analysis Tool, GMAT, is part of NASA's Open Source Software. It is designed for space mission design, optimisation and navigation from LEO to deep space missions [9].

GMAT is updated yearly, and the latest version can be downloaded from Sourceforge [10]. It is available for Linux, Windows and macOS.

An example indicating the core capabilities of the program is shown:

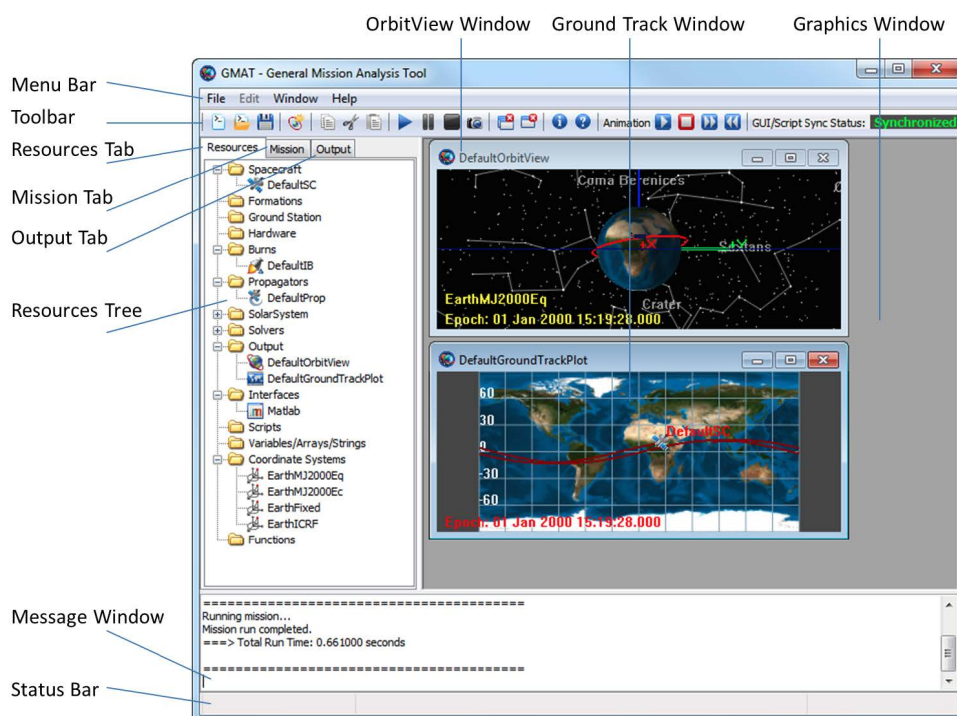


Figure 2.8 GMAT Desktop, Windows version. Source: [9].

Each resource can be easily customised to suit the mission requirements. The resources tree contains all GMAT capabilities and organises them into logical groups; the mission tree displays GMAT commands that control the time-ordered sequence of events in a mission, and the output tree contains the programme output such as report files and graphical displays.

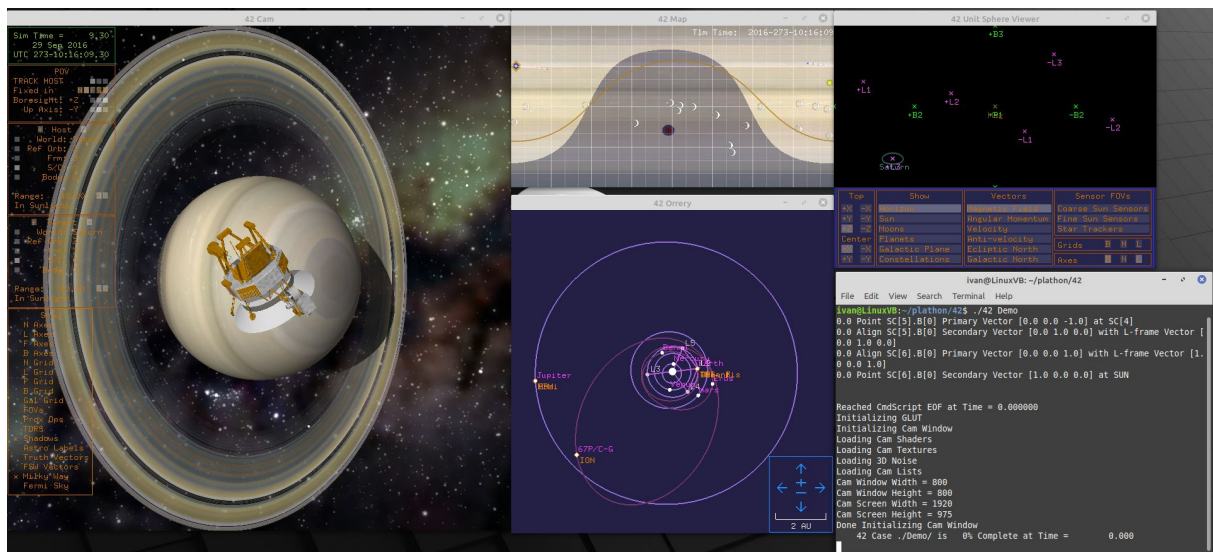
Although it is a powerful mission analysis tool, it does not allow real-time HITL connections.

### 2.2.4.2 NASA 42

NASA's open-source software named 42 is a comprehensive, general-purpose simulation of attitude and trajectory dynamics and control which can be applied to numerous spacecraft composed of multiple rigid or flexible bodies [11].

Spacecrafts can be assembled from diverse models, adjusting the general satellite properties and those from the available sensors and actuators. Environment perturbation models are only designed for the Earth, Moon and Mars. However, the programme includes ephemerides for all Solar System planets, major moons and a considerable number of asteroids.

42 supports geometrical visualisation through an OpenGL interface designed to support the entire GNC design cycle, enabling rapid prototyping for MDL research. High-fidelity dynamics handle flight code verification.



**Figure 2.9** 42 complete demonstration with GUI

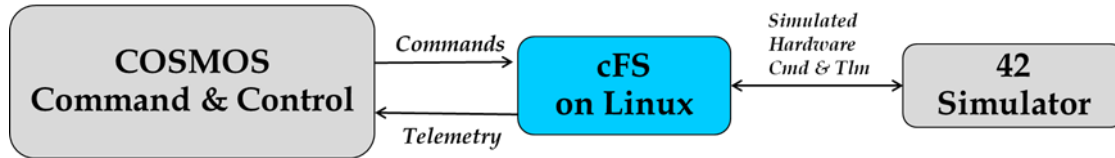
The programme's GUI is divided into four independent windows:

Window	Description
42 Cam	Leading visualisation window. It displays the simulation time and Point of View (POV) of the Host body and the target. The POV can be changed with the fixed frames or by rotating the camera with the mouse. The Host and Target bodies can also be changed during the simulation time alongside the camera range. At the bottom-left section, several options allow the user to activate different axes and additional options.
42 Map	2D map of the orbited celestial body. It displays the current trajectory and information about the ground stations.
42 Orrery	2D projection of the Solar System with all the simulated spacecraft positions and orbits. If zoomed, the 3D orbits can be seen around a celestial body. Lagrange points can be added.
42 Unit Sphere Viewer	Extra window in which the position of different vectors and celestial bodies is displayed with respect to the spacecraft.

**Table 2.2** 42 GUI window description

As can be seen on the image, 42 does not have a start menu and is executed through terminal commands. More images can be seen on Appendix [A.3].

The 42 simulator is part of NASA's OpenSatKit [12]. The kit combines 42 with two other open-source tools: Ball Aerospace Corporation's COSMOS [13] command and control platform for embedded systems, and NASA's cFS [14], a platform and project independent, reusable software framework.



**Figure 2.10** OpenSatKit connection diagram. Source: [12].

The OpenSatKit general programme is capable of running on different computers that have real-time communications. Thus, both SITL and HITL are available.

### cFS

Core Flight System takes advantage of a rich heritage of successful Goddard Space Flight Center flight software efforts captured by three critical aspects to the cFS architecture: a dynamic run-time environment, layered software, and a component-based design [14]. The combination of these critical aspects makes it suitable for reuse on any flight project or embedded software system at a significantly reduced cost. cFS enables the creator to run and test software applications on a developer's desktop and then deploy that same software without changes to the embedded system.

As a summary, the cFS architecture has been proven to:

- Reduce time to deploy high-quality flight software
- Reduce project schedule and cost uncertainty
- Facilitate formalised software reuse
- Enable collaboration across organisations
- Simplify flight software sustaining engineering
- Provide a platform for advanced concepts and prototyping



## COSMOS

COSMOS is a group of applications that can be used to control a set of embedded systems [13] in conjunction with the other OpenSatKit modules. It implements a client-server architecture with the Command and Telemetry Server and various other tools typically acting as clients to retrieve data. The Command and Telemetry Server connects to the Targets, depicted with green circles, and sends commands and receives telemetry, mainly system status data, from them. The GS is trying to control or get status from the targets. The arrows from the Server to the targets indicate Interfaces that can be over TCP/IP, serial, UDP/IP or a custom interface that has been previously defined. Yellow boxes indicate data items like configuration files, log files or reports.

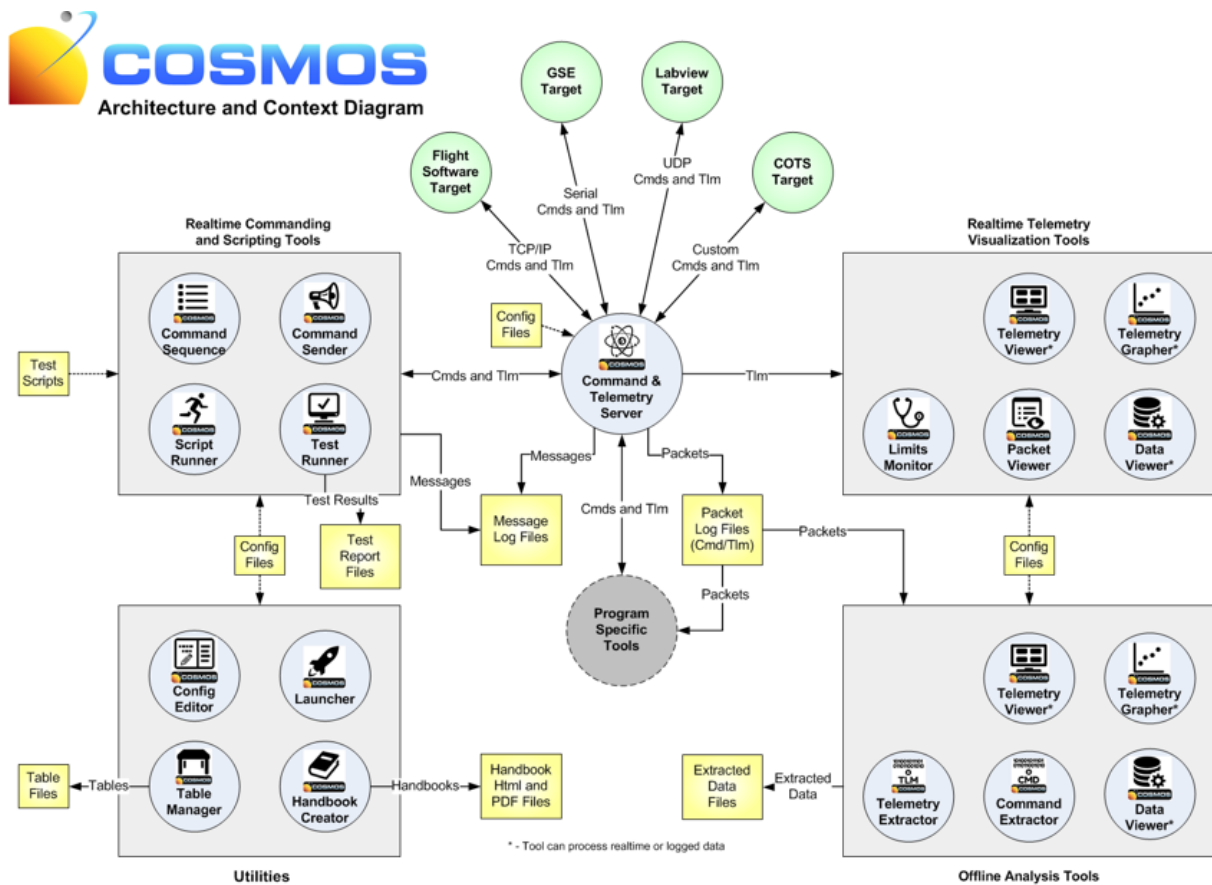


Figure 2.11 COSMOS Architecture. Source: [13].



### 2.2.4.3 Orekit

OREKIT (ORbits Extrapolation KIT) is a low-level space dynamics library written in Java. The Orekit project is driven according to an open-source model, involving representatives from different space fields actors [15]. Among some relevant users are Airbus Defence and Space, Thales Alenia Space, the Centre National d'Études Spatiales (CNES) and the European Space Agency (ESA).

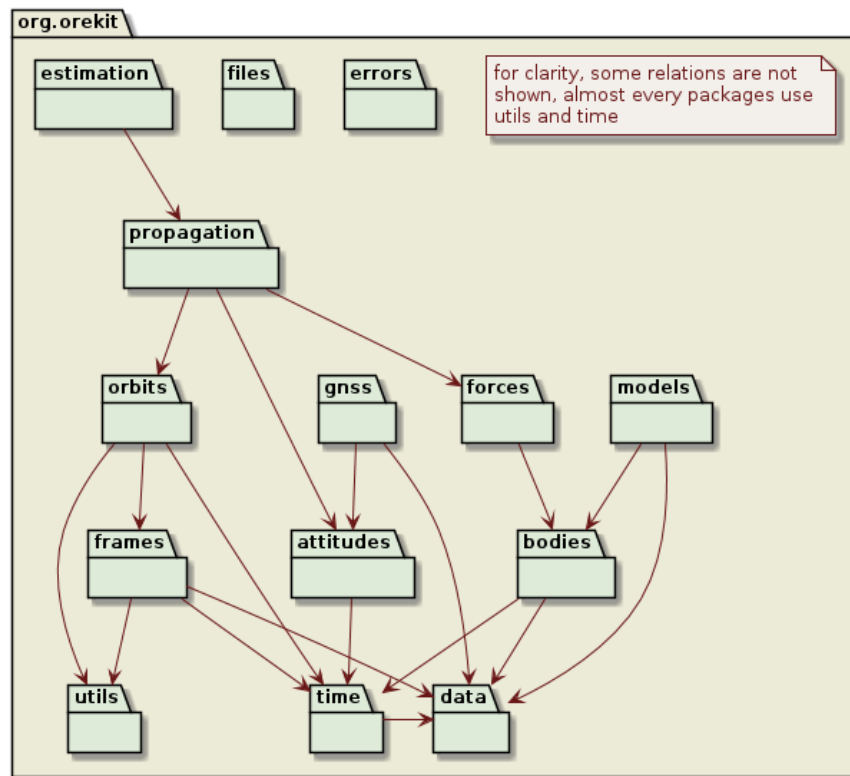


Figure 2.12 Top-level Orekit packages. Extracted from [16].

Starting from top to bottom, each package provides the following data [16]:

- **Estimation:** provides classes to manage orbit determination.
- **Files:** provides various file parsers, among which there is an interface for ephemeris file parsers and writers.
- **Errors:** provides classes to generate and handle errors, based on the classical mechanism of exceptions.
- **Propagation:** provides tools to propagate orbital states with different methods.
- **Orbits:** provides classes to represent orbits. It is the basis for all of the other space mechanics tools.
- **GNSS:** is an independent package providing classes to handle classical GNSS files and attitude providers for navigation satellites.
- **Forces:** provides the interface for the force models to be used by a numerical propagator.
- **Models:** provides physical models. Only Earth is available with the main download.
- **Frames:** provides classes to handle frames and transforms between them.

- **Attitudes:** provides classes to represent simple attitudes.
- **Bodies:** provides an interface to the representation of the position and geometry of space objects such as stars, planets or asteroids.
- **Utils:** provides methods for managing mathematical or geometrical objects.
- **Time:** is an independent package providing classes to handle epochs and time scales and to compare instants.
- **Data:** provides base classes for exploring the configured data directory tree and read external data that the library can use.

#### 2.2.4.4 Systems Tool Kit (STK)

Systems Tool Kit (STK) is a platform for analysing and visualising complex systems in the mission context. Although it was initially created to solve problems involving Earth-orbiting satellites, it is now used in the aerospace and defence communities. Among the most important clients are organisations such as NASA, ESA, Boeing and Airbus. However, it has been discarded due to price constraints.

#### 2.2.4.5 Matlab Tools

The TIEG team already has an operational Matlab and Simulink model developed by David Poza Hernández for his bachelor's thesis "*Study of magnetorquers control strategies for energy-efficient manoeuvres of a CubeSat*" [17] based on Matlab Aerospace Toolbox.

In addition, the research developed by Iván Parrot Martínez for his master's thesis *Study of energy-efficient manoeuvring strategies for CubeSats* encouraged the foundation of the PLATHON project with an analytic ADCS Simulink model [18]. However, spacecraft specifications have not been taken into consideration.

The models obtained from these thesis will be discussed on the comparison section 5.8.

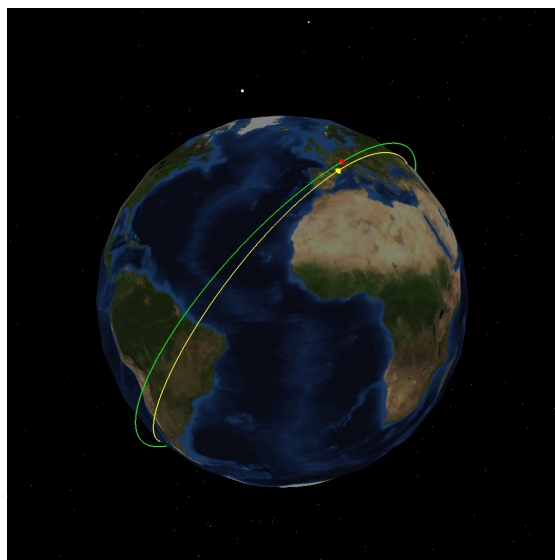


Figure 2.13 ISS Orbit. Source: Ivan Parrot [18]

## 2.3 PLATHON project

The PLATHON project has selected 42 as the main propagator due to its convergence with all the OpenSatKit modules. The cFS capabilities are not found on any of the other open-source programme. Similar propagation applications such as Orekit do not allow Software-In-The-Loop nor Hardware-In-The-Loop utilities and, therefore, are not suitable for the project.

Once the programme basis has been established, a mission example would be to collect data from an IoT network (sensors that communicate through the Internet of things protocol) located in planet areas inaccessible to the internet, and optimally send that data to a ground station connected to the network through a satellite cluster.

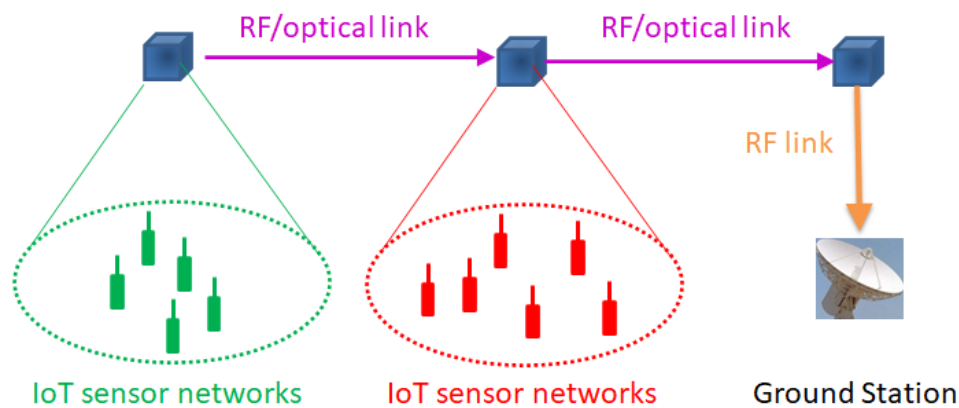


Figure 2.14 IoT data transmission mission

The mission planning must contemplate the calculation of the active communication satellites at all times. Thus, it must be taken into account that, at a certain moment, the satellite receiving a message might be inactive for any of these reasons:

- **Link not available:** when there is no direct line of sight between the sending and receiving satellites.
- **Battery depleted:** when the battery charge is depleted by prolonged exposure to a solar eclipse, preventing sufficiently recharging the batteries.
- **Saturated optical receiver:** when the optical receiver is saturated due to frontal incidence to the Sun or another light-emitting body.

CubeSats hardware prototypes will simulate the system floating without friction on air-bearing structures activated fuelled with compressed air. Each CubeSat contains four electronic boards:

- **Electrical Power System:** powering the CubeSat through solar panels.
- **Attitude Determination and Control System:** CubeSat attitude determination sensors and spacecraft actuators.
- **Communication System:** RF communication to exchange commands and telemetry from the CubeSat with the Ground Station (GS), and through RF or optical communication between CubeSats to send messages from the IoT sensor networks to the GS.
- **Onboard Computer:** Onboard computer controlling the mission operation of the satellite flight.

The following figure shows several GS or IoT transceivers (called GSi), CubeSats or Nanosatellites in LEO orbit, and stationary satellites in GEO orbit. The map is divided into the area illuminated by the Sun, in yellow, and the zone in Eclipse, in grey.

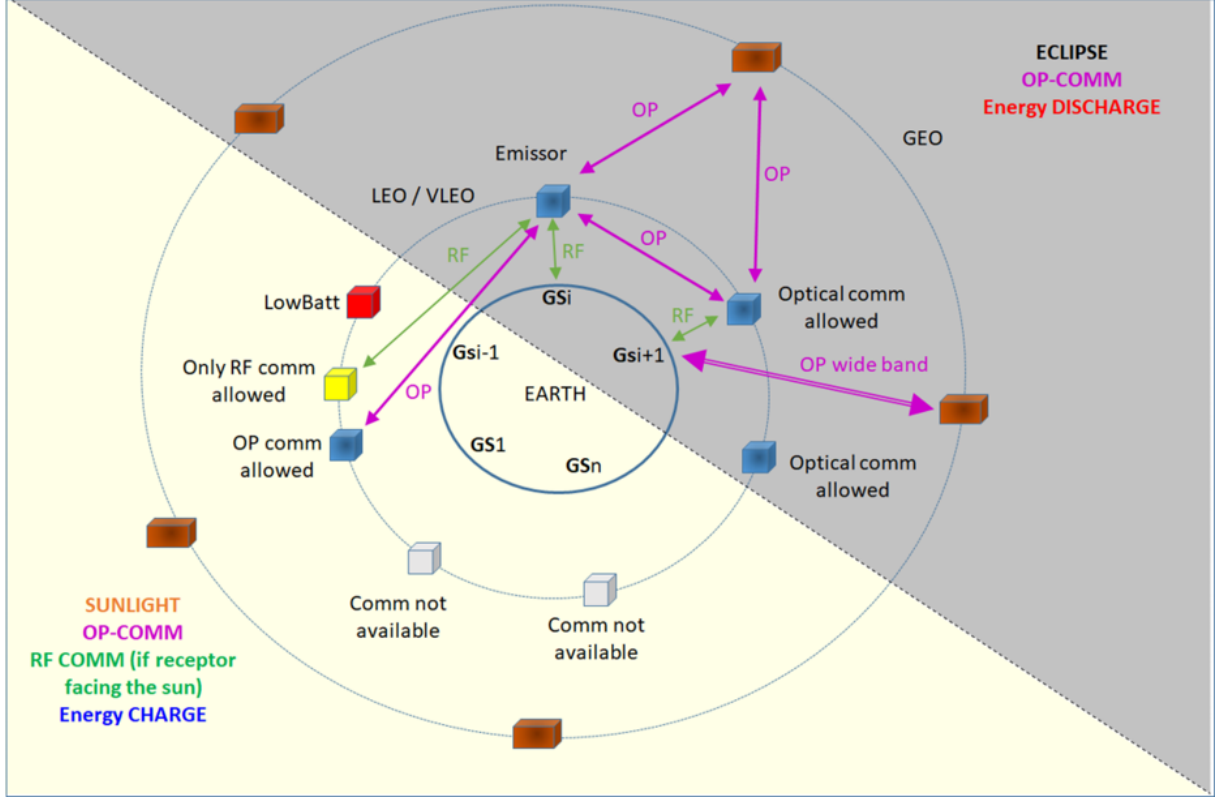


Figure 2.15 PLATHON constellation diagram. Source: [1].

CubeSats from the previous image are represented with the following colour code:

- **Grey:** no direct vision with the emitting CubeSat.
- **Red:** low satellite battery and, therefore, not available for communications.
- **Yellow:** CubeSat receiver with direct solar incidence on the photoreceptor not enabled for optical communication but capable of RF communication.
- **Blue:** enabled for communication, both optical and RF.

The RF (green) and optical (magenta) communications between CubeSats are represented by arrows. Communications between CubeSats and GEO satellites are only optical. Finally, the communication between CubeSats and ground stations (GSi) is in RF and between GEO and GSi in the broadband optical link.

An initial prototype is developed on this thesis with the inclusion of SiTL simulations. Future analysis will delve into emulations with HiTL when the hardware models are ready.

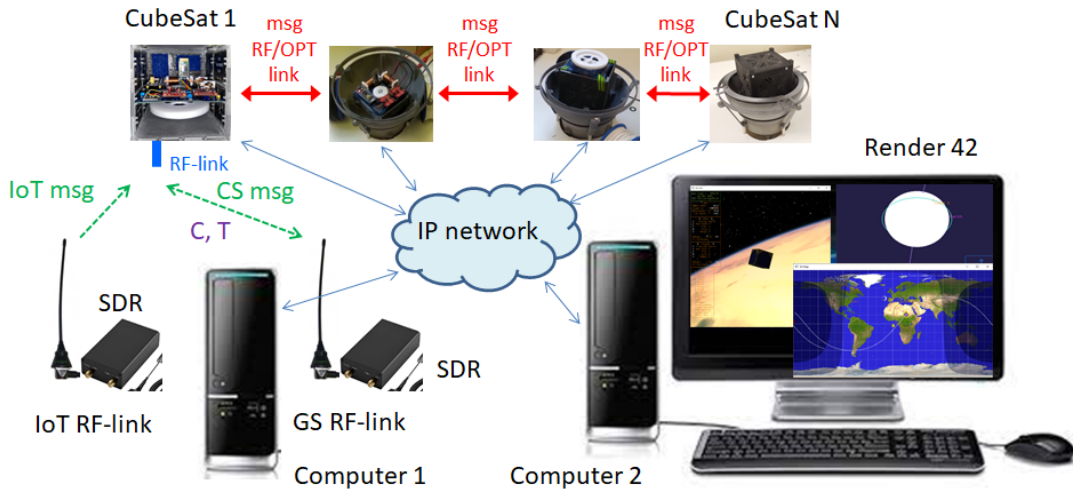
The following figure shows the hardware of the PLATHON system adapted to RF and optical communications between satellites. However, only RF communication with ground stations (IoT and GS) are represented. The required hardware devices are:

- **Computer 1:** contains the orbital propagator simulation platform and the communications simulator.
- **Computer 2:** only the orbital propagator viewer is executed with a high computing power graphics card.
- **CubeSat i:** One of N cubeSat prototypes floating in air-bearing structures without friction.
- **IoT RF-link:** RF and SDR antenna to emulate IoT messages, depicted as IoT msg. Messages from Earth to the corresponding satellite.
- **GS RF-link:** RF and SDR antenna to emulate the messages between GS and satellites.

GS RF-link messages can be classified as

- **Command (C):** message sent by the GS to the satellite. It is an order such as requesting sensor calibration.
- **Telemetry (T):** message sent by the satellite to the GS. The satellite informs the GS of the spacecraft status.
- **CubeSat message (CS msg):** message sent by the satellite to the GS, containing the messages initially sent by the IoT to the corresponding GS.

All hardware devices are connected through an IP network, either wired or with Wi-Fi links, to exchange essential information for the operation of the system.

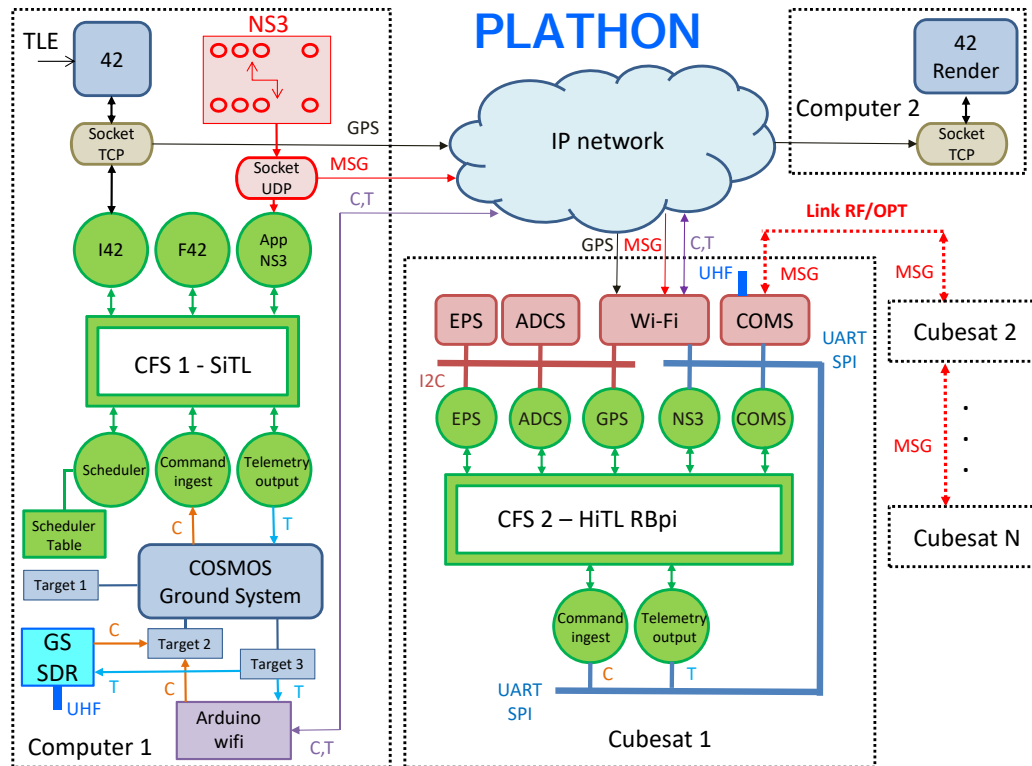


**Figure 2.16** PLATHON system for emulation of communication missions between CubeSats and Earth

The PLATHON system allows the entire mission to be emulated with Hardware-in-the-loop, except for the measurement of the GPS position, which will be supplied to each satellite from the orbital propagator through the IP network. The measurement and change of the attitude, the power supply of the battery system, and the delivery and reception of messages between satellites, or between Earth and satellites, will be performed in real-time in conjunction with the hardware devices. The distances between the elements will not be up to scale as they are subject to laboratory limitations.

### 2.3.1 OpenSatKit and NS3

The following diagram portraits the PLATHON software applications with the Hardware-in-the-loop connections. The OpenSatKit capabilities are fused with the NS3 programme.



**Figure 2.17** PLATHON project diagram

For the core Flight System, only the essential applications have been included to explain the operation of the PLATHON platform. These are:

- **I42:** information interface between 42 and the cFS software.
- **F42:** application that contains the same models and algorithms as 42. It can be used instead of 42 if both programmes are not to be running at the same time.
- **NS3 App:** interface with the NS3 communications simulator.
- **Scheduler:** table that contains the mission commands to be carried out by the satellite as sequential steps.
- **Command Ingest:** application that manages Command messages from the GS to the satellite.
- **Telemetry Output:** application that manages Telemetry messages from the satellite to the GS.

The cFS will be in the satellite OBC, but it is possible to execute it in Computer 1 and perform a simulation with Software-in-the-loop capabilities. The use of SiTL will be helpful during the first mission steps while the hardware devices are not ready or when required to evaluate the software and hardware of the satellite separately.

The Network Simulator 3 (NS3) is a communications simulator containing precise communication channel models between nodes which can calculate the optimal route between two points. It allows computing the latency and attenuation of each possible link, classifying each channel as an ON link or an OFF link.

Since the communication system nodes are mobile, and their communication state changes with the satellite's time, position and attitude, it is necessary to previously calculate the possible routes at each point of the orbit.

For this purpose, it is necessary to simulate the mission with the OpenSatKit System and transfer the results to the NS3 so that it establishes the optimal routes for each position, taking into account all satellite positions as well as those from the IoT and GS, which will also be mobile due to the Earth rotation movement.

### 2.3.2 Step-by-step PLATHON system

A complete PLATHON emulation can be described with the following stages:

1. Creation of satellite orbits in 42 from Matlab file creator or Matlab TLE data interface.
2. Mission design and translation into communication links to be carried out at each time step or orbit position, allowing the data to be sent from the original point (for instance, from an IoT sensor network) to the corresponding GS.
3. Connection to the NS3 programme to establish the optimal routes.
4. Design the Command and Telemetry messages sent between GS and satellite each time they have visibility. Only one GS is selected as the tracking station.
5. There is a simulated satellite as SiTL in Computer 1, and N hardware satellites as HiTL.
6. The communication of 42 and COSMOS with the cFS of the SiTL satellite inside Computer 1 is programmed through sockets or software communication ports since all the software is on the same machine.
7. If 42 manages the satellite attitude control, the attitude and position data are passed to the cFS through the TCP socket with the I42 application.
7. If the cFS performs the attitude control, the attitude and position values are provided by the F42 application.
8. NS3 will provide, in a synchronised way with 42, the information of the attitude command that must be followed at all times. Given that the mission has been previously planned, it is known for every time step when to rotate the antenna or point the optical link to focus on another satellite that will act as the next communication node, following the optimal route. That information is sent through the UDP socket to the NS3 application of the cFS.
9. When NS3 has estimated that the satellite has already performed the required movement, the programme designs the following message to send through the antenna or the optical link. It is assumed that the previous operation has been carried out correctly by the receiving satellite, and therefore, they are accurately oriented. When the message is sent, the receiving satellite must verify that the data has been received correctly.
10. Every time a satellite flies over the monitoring station, COSMOS activates the Command Ingest applications to send Command messages and Telemetry Output to receive the satellite sensors' status. This process is managed through the scheduler table.

11. The communication of 42 and COSMOS with the cFS of the HiTL satellite is through sockets or communication ports through the IP Network with Wi-Fi access points.
12. 42 sends the position data (GPS) through Wi-Fi to the satellite through the IP network. Furthermore, the Wi-Fi injects it through the I2C bus to the GPS application, transforming it into a cFS readable format sent through the satellite cFS bus software.
13. NS3, in a synchronised way with 42, will provide the information of the attitude command that must be followed and the message (MSG) to be sent through the Wi-Fi IP network. The satellite Wi-Fi will inject the attitude command data through the I2C bus to the ADCS application and the message through the SPI or UART bus to the NS3 application. These applications format the data and inject it into the software bus.
14. The ADCS board detects and changes the attitude of the satellite according to the data received, orienting it to the corresponding satellite, IoT or GS according to the mission design.
15. The data is sent by the COMS board, either in RF or by optical link, and is received by the next satellite, which will have also carried out the attitude command previous operation to point to the transmitter. Then, the values are either read by the GS or transmitted from the IoT, the hardware of which is also an SDR.
16. Communication with COSMOS to send the information to or from the monitoring station is also performed through cFS applications, the Command Ingest or the Telemetry Output of the satellite that sends the data by UART or SPI bus to the Wi-Fi boards or COMS.
17. The Command Ingest or Telemetry Output data is sent through Wi-Fi when there is no SDR hardware connected to Computer 1. In this case, the data is sent through the IP network to the Wi-Fi board of Computer 1.
17. If SDR is connected by USB to Computer 1, the Command Ingest or the Telemetry Output data is sent through the COMS RF antenna to the SDR RF antenna.
18. The Command Ingest or the Telemetry Output data that arrives either by Wi-Fi or by SDR, are injected to COSMOS through the corresponding Targets.
19. Finally, 42 position data and satellite attitude are sent through the IP network to Computer 2 to display the 42 render in real-time.



## 2.4 Satellite classification

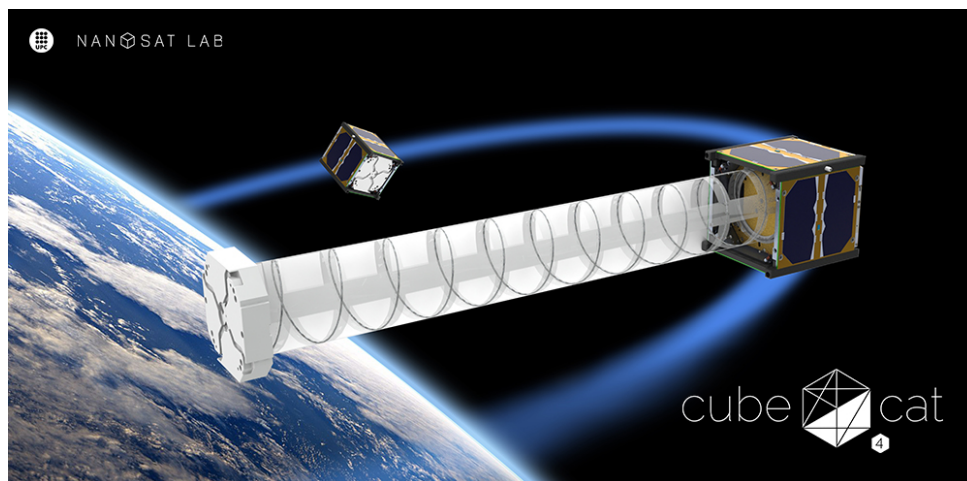
According to their mass, satellites can be classified as follows:

- Large satellites:  $> 1000$  kg
- Medium satellites: 500 to 1000 kg
- Small satellites:  $< 500$  kg
  - Minisatellites: 100 to 500 kg
  - Microsatellites: 10 to 100 kg
  - Nanosatellites: 1 to 10 kg
  - Picosatellites: 100 g – 1 kg
  - Femtosatellites: 10 g – 100 g
  - Attosatellites: 1 g – 10 g
  - Zeptosatellites: 0.1 g – 1 g

### 2.4.1 The CubeSat standard

The CubeSat is a miniaturised satellite for space research that originated in 1998 with a project at the Aeronautics and Astronautics Department that Stanford University had with DARPA, and the Aerospace Corporation [19].

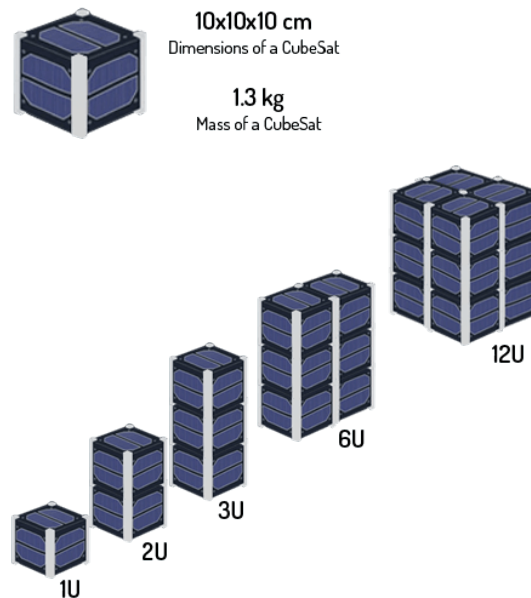
1-U CubeSat's are cubic modules of 10 x 10 x 10 cm with a maximum mass of 1.33 kg. The standard was designed to be compatible with Poly Picosatellite Orbiting Deployers (PPOD), a standardised launch interface with an available size of 10 x 10 x 33 cm or up to 3-U CubeSats. However, creative models make the most of these reduced dimensions:



**Figure 2.18** <sup>3</sup>Cat-4, UPC's NanoSat Lab. Source: [20].

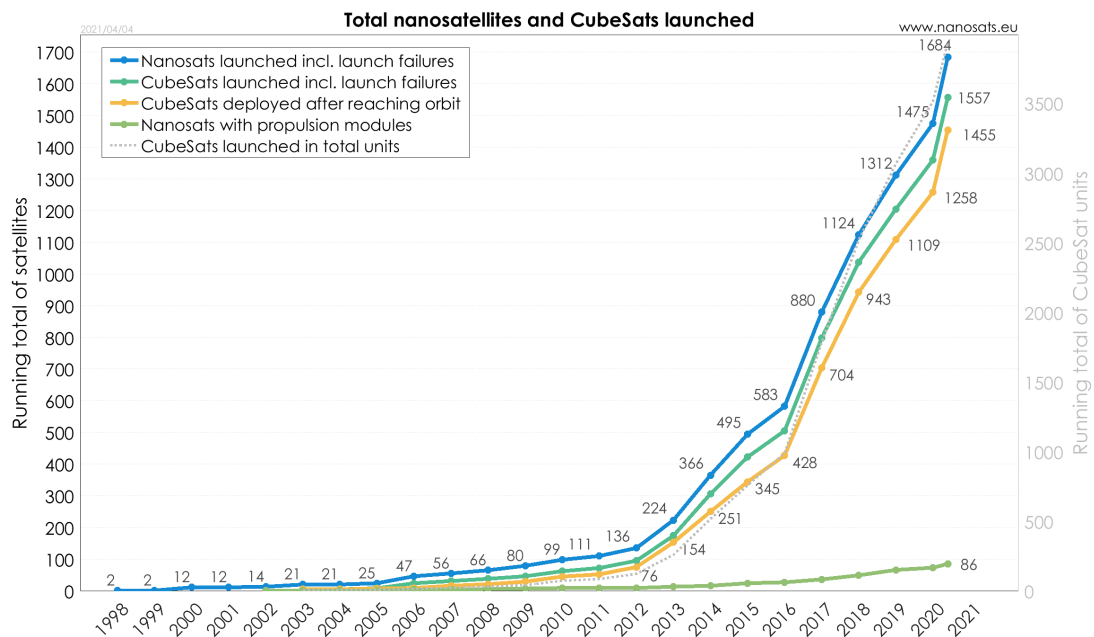
The CubeSat subdivisions can range between picosatellites and microsatellites depending on the number of units which can range between 0.25U and 27U while the weight ranges between 0.1 to 40 kg.

The following images represents standardised CubeSat unions:



**Figure 2.19** CubeSat sizes and mass. Source: [21].

The launch tendency of both Nanosats and CubeSats has been increasing since 2012. According to Nanosats Database, as of April 4, 2021, a total of 1557 CubeSats have been launched, and over 2500 nanosats have been predicted to be launched over the next six years; an exponential increase promoted by reduced costs that can yield substantial results.



**Figure 2.20** Total Nanosats and CubeSats launched. Source: [22].

## 2.5 Satellite constellations

A constellation is a set of satellites distributed over space intended to work together to achieve common objectives. If the satellites are close together, exchanging data or material with each other, then it is called a cluster or formation, rather than a constellation [23].

Compared with GEO satellites, LEO and MEO satellite constellations can provide low latency in conjunction with lower power requirements due to the shorter distance to Earth's surface, which allows for equipment miniaturisation. The main obstacle with these constellations is that there must be a sufficient number of operational in-orbit satellites, which generally translates into long distribution times [24].

The following table summarises the main constellation design issues:

Issue	Importance	Determination	Principal Issues or Options
Coverage	Principal performance parameter	Altitude, Minimum elevation angle, Inclination Constellation pattern	Discontinuous coverage - Gap times Continuous coverage - Simultaneous satellites in view
Number of satellites	Principal cost driver	Altitude, Minimum elevation angle, Inclination Constellation pattern	
Launch options	Major cost driver Major mission risk	Altitude, Inclination Spacecraft mass	Low altitude and low inclination orbits cost less
Environment	Radiation level - Lifetime - Shielding	Altitude	Below, in or above Van Allen radiation belts
Orbit perturbations (station keeping)	Constellation disassociates over time	Altitude, Inclination, Eccentricity	Keep all satellites at common altitude and inclination to avoid drifting apart
Collision avoidance	Principal mission risk	Constellation pattern, Orbit control	No option: must design entire system for collision avoidance
Constellation Build-up, Replenishment and End-of-Life	Determines level of service over time and impact of outages	Altitude, Constellation pattern, Build-up and sparing philosophy	Sparing: - On-orbit spares vs. launch on demand End-of-Life: - Deorbit vs. raise to a higher orbit
Number of orbit planes	Determines performance plateaus	Altitude, Inclination	Fewer planes means more growth plateaus and more graceful degradation

**Table 2.3** Constellation design issues. Source: [25].

### 2.5.1 Walker Pattern

A Walker pattern constellation is described by its main parameters in I: T/P/F, where I denotes the inclination of orbital planes, T is the total number of satellites, P is the number of evenly spaced orbital planes and F is the phasing parameter, which defines the relative position of the satellites in the adjacent planes with values ranging between 0 and P-1 [25].

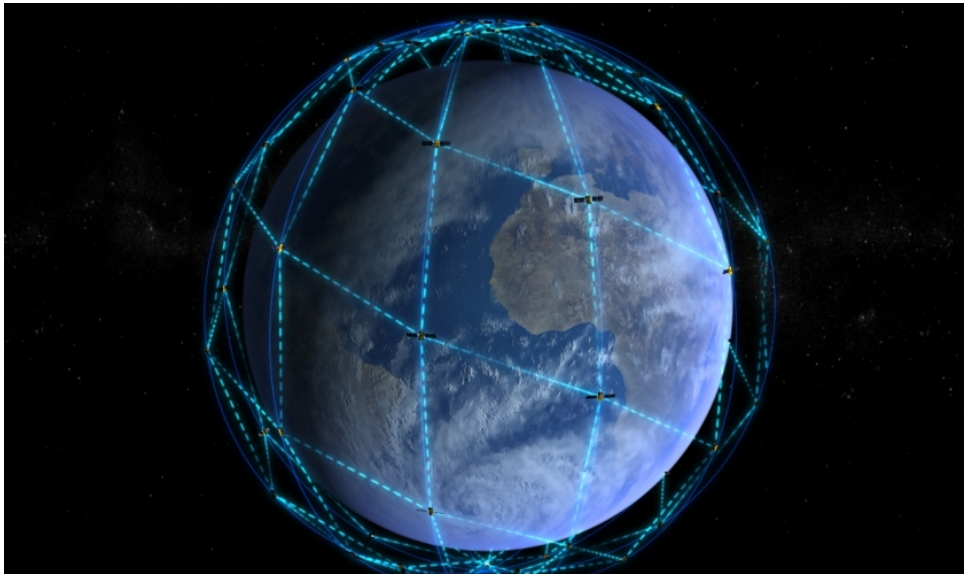
The number of satellites per plane is defined as  $S = T/P$ . Planes are spaced at intervals of  $S \cdot PU$  in node, where the Pattern Unit (PU) is defined as  $PU = 360/T$  [°]. Then, satellites are spaced at intervals of  $P \cdot PU$  within each plane; and a satellite at its ascending node has the next easterly satellite at  $F \cdot PU$  past the node.

For instance, the Iridium system corresponds to a Walker constellation.

#### 2.5.1.1 Iridium NEXT constellation

The Iridium NEXT constellation is the replacement of the original Iridium constellation. With 75 new satellites, the constellation covers the entire Earth with 66 active devices while nine remain backups. Another six remain on the ground as spares.

The satellites are orbiting in circular orbits at 780 km in altitude with an inclination of 86.4 degrees. The orbital period is 100.5 minutes. Each satellite is linked to the four closest satellites enabling an independent service from any local ground infrastructure.



**Figure 2.21** Iridium constellation routing signals in orbit. Source: [26]

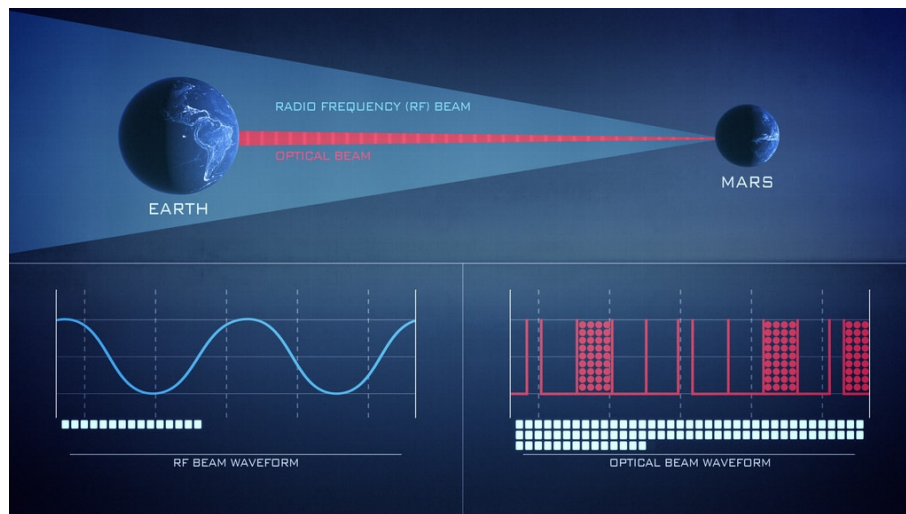
## 2.6 Optical communications

Currently, most space missions use radio frequency communications to send data to and from spacecraft. Although their use has a proven track record of success, as satellite technology capabilities are enhanced, space missions generate and require to collect an increasing data volume each year, which is constrained by the speed at which it can be sent to the appropriate receiver [27].

Optical communications have proven to increase data bandwidth between 10 to 100 times while reducing the instrument's size, weight, and power requirements. Considering a spacecraft with constant volume, a decreased communication subsystem size translates into more space for other subsystems. If the volume is diminished, the weight reduction signifies that a lower budget is allocated to the launch and can be transferred to further research or enhanced capabilities. As a result, the spacecraft's batteries would be less drained and able to fully recharge faster.

ESA's operated OPS-SAT is a CubeSat intended to demonstrate the improvements of more powerful onboard computers in mission control capabilities. The launch took place on 18 December 2019, when OPS-SAT was injected into a circular, polar orbit at 515 km altitude [28].

The latest NASA test with the Laser Communications Relay Demonstration (LCRD), which is scheduled to launch in Summer 2021, has been able to simulate downlink data over optical signals at a rate of 1.2 gigabits per second. Nonetheless, laser communications must be exact when broadcasting from thousands or millions of kilometres away. A deviation of even a fraction of a degree can result in the laser missing its target entirely, thus, losing all the gathered data.



**Figure 2.22** Difference in data rates between RF and optical communications. Source [29].

Optical communications between CubeSats have also been researched in the last years. Recently, the diverse constellation systems planned and launched in the following years are depleting the available frequency spectre. As an alternative, optical communications with wider band widths could revolutionise the space system architecture. It is necessary to highlight that optical space communications do not have regulatory restrictions yet [24]. As an example, SpaceX's Starlink constellation uses inter-satellite laser communication technology.

Recently, CubeSat experiments have proven the in-orbit laser communications performance, confirming the feasibility of space optical communications using CubeSat class satellites [30].

Some constellation and optical communication issues were discussed on the 5th ESA CubeSat Industry Days A.4.

### 2.6.1 Quantum cryptography

Communication connections are vulnerable to cyberattacks, and both inter-satellite communications and ground-to-satellite links are no exception.

The Quantum Key Distribution Satellite is a highly innovative ESA Partnership Project with the intention of demonstrating how to keep secure the exchange of sensitive information between several parties employing the laws of quantum mechanics for space-based infrastructure [31].

Quantum cryptography is based on taking advantage of quantum mechanical properties to perform cryptographic tasks. The aforementioned satellite, currently under development, is founded on quantum key distribution. This method enables two parties to create a shared random secret key. Therefore, as this key is known only to them, it can be used to encrypt and decrypt messages.

## Chapter 3

# Theoretical concepts

This chapter introduces the required theoretical concepts to understand the basis upon which this thesis is settled.

### 3.1 Orbital perturbations

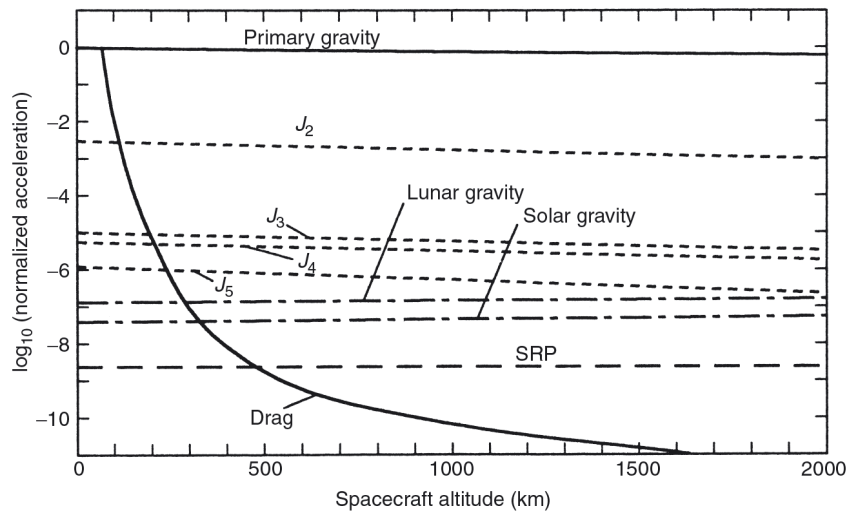
Any influence that causes a celestial body's idealised motion to deviate from a Keplerian trajectory is considered a perturbation.

The perturbations that influence a spacecraft can be divided into external forces caused by natural phenomena and those associated with satellite operations. Among the ordinary external stimulus on the Solar System are the asphericity of the central body, N body gravitation, solar radiation pressure, and atmospheric drag and lift if the orbit radius is close to the surface. Besides, the most common artificial perturbation is caused by propulsive thrust. For instance, cold gas is used for orbit maintenance and manoeuvring as well as attitude control.

The following relation can describe the motion of a perturbed body [5]:

$$\ddot{\vec{r}} + \mu \frac{\vec{r}}{r^3} = \frac{\sum \vec{F}_{perturbation}}{m} \quad (3.1.1)$$

where  $\mu$  is called the gravitational parameter and defined as  $\mu = G(m_1 + m_2)$



**Figure 3.1** A comparison of the disturbing accelerations for the main sources of perturbation.

Source: [32].

### 3.1.1 Earth's Gravitational Potential

The Keplerian conics are the result of perfectly spherical, homogeneous central bodies. However, Earth is an asymmetric spheroid bulged at the Equator and flattened at the Poles [3]. As a result, the expression for the gravitational potential can be written as:

$$U = \frac{\mu}{r} \cdot \left( 1 - \sum_{n=2}^{\infty} J_n \left( \frac{R}{r} \right)^n P_n(\sin L) + \sum_{n=2}^{\infty} \sum_{m=1}^n \left( \frac{R}{r} \right)^n P_n^m(\sin L) (C_{nm} \cos m\lambda + S_{nm} \sin m\lambda) \right) \quad (3.1.2)$$

where  $L$  is the latitude,  $\lambda$  is the longitude,  $r$  is the distance between the centres of mass and  $R$  is Earth's mean radius.

Additionally,  $P_n(x)$  are the Legendre functions defined by

$$P_n(x) = \frac{1}{2^n n!} \frac{d^n}{dx^n} \left[ (x^2 - 1)^n \right] \quad -1 \leq x \leq 1 \quad (3.1.3)$$

whereas  $P_n^m(x)$  correspond to another Legendre function defined by

$$P_n^m(x) = (1 - x^2)^{\frac{m}{2}} \cdot \frac{d^m}{dx^m} [p_n(x)] \quad -1 \leq x \leq 1; m \leq n \quad (3.1.4)$$

$J_n$  are constants that represent the zonal harmonics, harmonics which are only dependant on the mass distribution along the latitude, while  $C_{nm}$  and  $S_{nm}$  represent tesseral harmonics, which are dependant on both the longitude and latitude. On figure 3.1 the comparison of  $J_n$  magnitudes is depicted.

### 3.1.2 Aerodynamic Drag

Due to the non-zero density in the upper reaches of the atmosphere, an aerodynamic drag force opposes the direction of motion of an orbiting spacecraft. The formulation of atmospheric drag equations is plagued with atmospheric uncertainties such as the drag coefficient or the orientation-dependent cross-section.

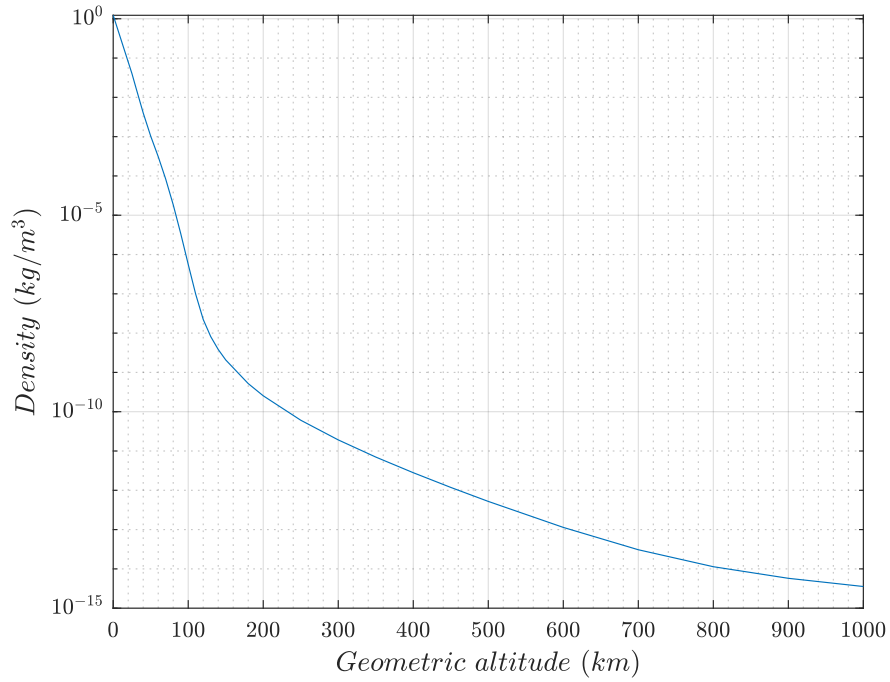
To discern the key parameters that affect the spacecraft, a simple aerodynamics theory equation can approximate the atmospheric deceleration effect:

$$\left. \begin{aligned} D &= -\frac{1}{2} \rho V^2 S C_D \\ D &= m \frac{\delta V}{\delta t} \end{aligned} \right\} \quad \frac{\delta V}{\delta t} = -\frac{1}{2} \rho V^2 \frac{S C_D}{m} \quad (3.1.5)$$

Where  $\rho$  is the density,  $V$  is the speed modulus,  $S$  is the cross-sectional area, and  $C_D$  is the drag coefficient which may vary in the interval  $2 \leq C_D \leq 4$  for LEO satellites [33]. As a result of complex analysis, it has been demonstrated that drag only secularly affects eccentricity and the semi-major axis [34].

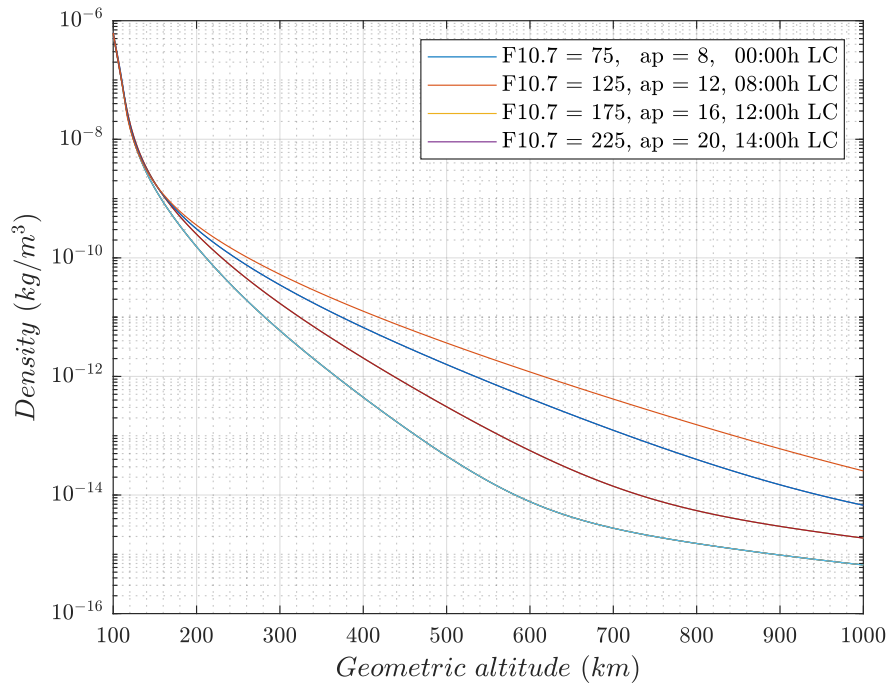
For advanced calculations, atmospheric models are required. For Earth, the Kármán line, the commonly accepted altitude at which space begins is 100 km. From that point on, several models describe the variation of atmospheric properties with altitude. Some of the most historically acclaimed are the US Standard Atmosphere [35], and the MSIS-86 Thermospheric Model [36], which can be found on the 42 propagator.





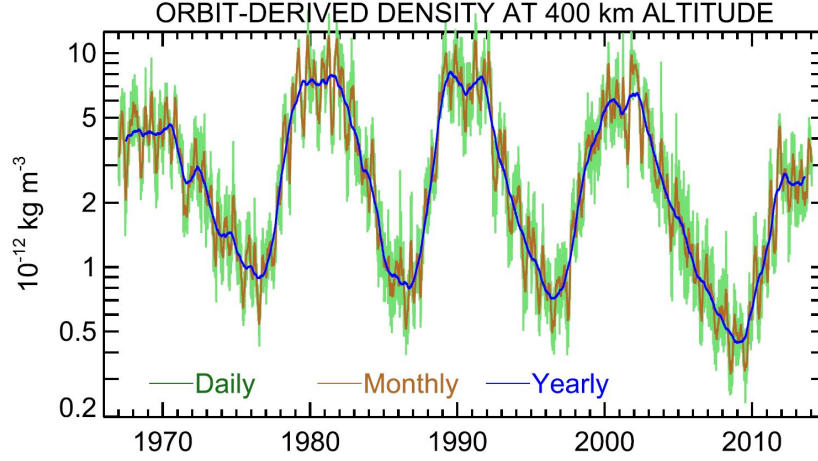
**Figure 3.2** NASA - US Standard Atmosphere 1976. Data extracted from [35].

An improvement comes from the MSIS-86 model, which considers the variation of orbit density with the fluctuations caused by the solar cycles. It considers more variables and depends on the accuracy of the input data. This model is used by the 42 propagator.



**Figure 3.3** MSIS-86 Thermospheric Model. Data extracted from [33]. Plotted with [37].

That way, the model tries to replicate the wavy empirical data due to the space weather effects shown in the following image:



**Figure 3.4** Global average thermospheric mass density at an altitude of 400 km. Extracted from [38].

### 3.1.3 Solar Radiation Pressure

Solar Radiation Pressure comprises photons, which are massless elementary particles travelling at the speed of light. An approximation for a spherical body has been obtained in [5]:

$$\mathbf{F} = -\nu \frac{S}{c} C_R A_{sc} \hat{\mathbf{u}} \quad (3.1.6)$$

where  $\nu$  is the shadowing function,  $P_{SR}$  is the power per unit area,  $C_R$  is the radiation pressure coefficient,  $A_s$  is the absorbing area and  $\hat{\mathbf{u}}$  is the unit vector pointing from the satellite towards the Sun.

$$v : \text{shadow function} \begin{cases} v = 0 & \text{in eclipse} \\ v = 1 & \text{not in eclipse} \end{cases}$$

$C_R$	Surface	Spacecraft
0	Transparent	Does not absorb any momentum
1	Black body absorption	Absorbs all of the momentum of the incident photon stream
2	Reflective	Doubles the force, as all incident photon momentum is reversed in direction

**Table 3.1** Radiation pressure coefficient

As explained during the PLATHON project description, laser communications are not possible during solar radiation influence due to the saturation of the optical receiver.

### 3.1.4 Spacecraft Manoeuvres

The PLATHON project considers CubeSats without thrusters. This constraint leads to manoeuvres with other actuators analysed on [18]:

#### Detumbling

To keep the proper orientation of the satellite is essential for most space missions. At a specific moment, such as after separation from the launcher, the CubeSat may be rotating at high rotational speeds. The detumbling manoeuvre is performed to dissipate all the momentum by reducing the velocity. Magnetorquers brake the satellite up to a point in which the reaction wheels can return to stable operation.

#### Pointing

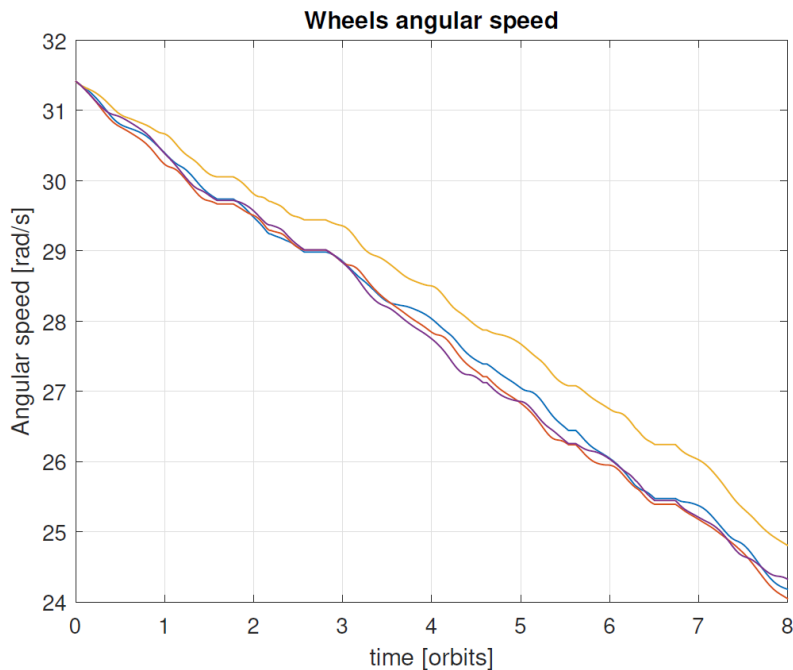
During the CubeSat mission, the spacecraft will need to point in a particular direction and maintain the action as long as required. This point could either be the nadir for Earth-orbiting satellites or a fixed reference such as the Sun.

In theory, for this purpose, only the reaction wheels are activated to keep the current attitude within high precision and without accumulating too much momentum.

If the reaction wheels surpass the angular velocity threshold during the operation, the detumbling manoeuvre would be activated.

#### Desaturation

Once the reaction wheels have stored a considerable amount of momentum, there is the risk of reaching saturation when the reaction wheels are about to reach their structural limit. In addition, when the power needed to resist the friction losses is too high that the battery cannot withstand it, this process might also occur.



**Figure 3.5** ISS orbit desaturation with pyramidal configuration. Extracted from [18].

## 3.2 Coordinate systems

Coordinates are expressed in the reference that best suits the spacecraft or constellation. Therefore, a distinction must be made between coordinate systems and frames.

A reference system is a theoretical definition that includes a set of models and conventions together for its implementation, while a reference frame is a practical implementation based on fiducial directions agreeing with the corresponding reference frame [39]. For instance, a set of fundamental stars are used as a reference for a space-fixed frame.

A rectangular coordinate system requires an origin, a fundamental plane, a preferred direction, and the sense or positive direction [40]. All of the following systems are used by the 42 propagator.

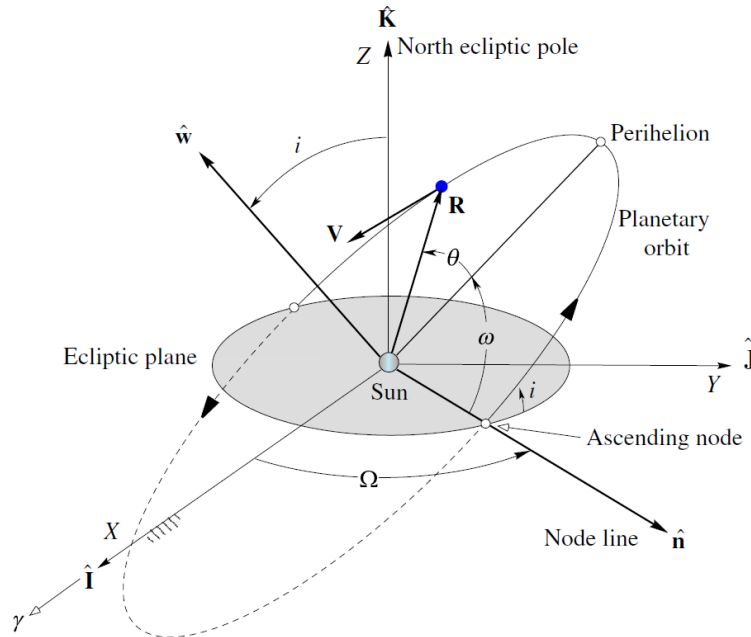
### 3.2.1 Heliocentric Coordinate System

The Heliocentric Coordinate System, also known as Ecliptic Coordinate System, has the Sun centre as the origin with a fundamental plane corresponding to the ecliptic plane. It rotates around the solar-system barycentre with the Z-axis normal to the plane and the X-axis pointing to the vernal equinox, often referred to as the first point of Aries ( $\Upsilon$ )<sup>1</sup>. It follows the right-hand side convention [5].

This system is based on a group of variables to describe the orbit of a celestial body. These are represented in Fig. 3.6 and listed as follows:

- $\Omega$  represents the Right Ascension of the Ascending Node (RAAN)
- $i$  symbolises the inclination of the orbit with respect to the ecliptic plane
- $w$  is the argument of the periapsis
- $\theta$  depicts the true anomaly of the orbiting body.

Oblateness causes  $\Omega$  and  $\omega$  angles to vary significantly with time.



**Figure 3.6** Heliocentric Coordinate System. Extracted from [5].

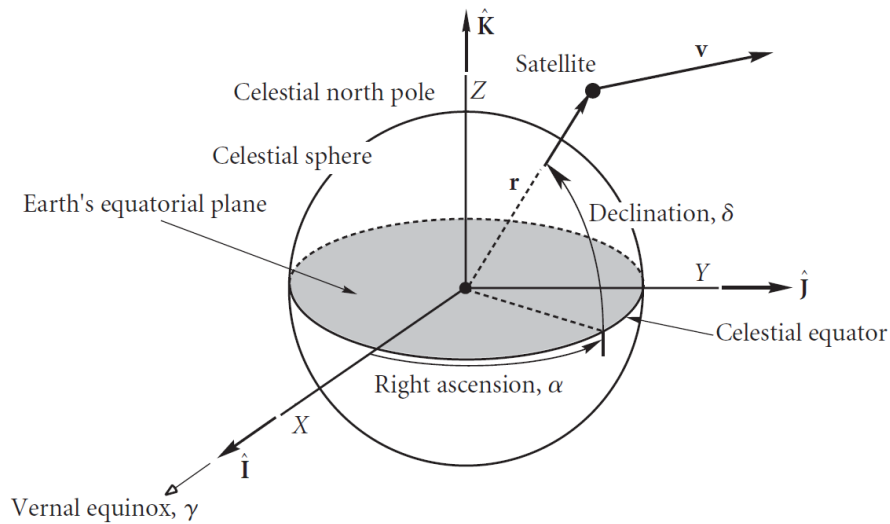
<sup>1</sup>VE pointed to the constellation Aries during Christ's lifetime.

### 3.2.2 Inertial and rotating frames

Each world can be modelled with an inertial and rotating frame. In this thesis, both Earth-centred inertial (ECI) and Earth-centred-Earth-fixed (ECEF) frames will be used alongside the 42 propagator.

ECI coordinate frames originate at Earth's centre of mass; the equinox and plane of the equator move so slowly over time that a "pseudo" Newtonian inertial system is considered referred to a particular epoch. Star vectors are provided in J2000 and converted to Heliocentric, while planet ephemerides are given at the J2000 epoch.

- **X-axis** oriented towards the vernal equinox.
- **Y-axis** oriented towards the direction resultant of a Cartesian right-hand between X and Z.
- **Z-axis** oriented towards the North Pole and Earth's axis of rotation.



**Figure 3.7** Earth Centred Inertial Frame. Extracted from [5].

ECEF coordinate frames also originate at the Earth's centre of mass. The X-axis is aligned with the International Reference Meridian; while the Z-axis is aligned with the International Reference Pole, both fixed about Earth's surface. The Y-axis completes the Cartesian right-hand rule.

### 3.2.3 Local Vertical - Local Horizon Frame

The LVLH frame originates in the centre of mass of the body. The  $z_{LVLH}$  axis points to nadir, the line which connects the centre of mass of the orbiting spacecraft and the celestial body pointing to the latter, while  $x_{LVLH}, y_{LVLH}$  create a plane where  $y_{LVLH}$  points to the negative orbit normal.

### 3.2.4 Body Frame

The body frame originates in the centre of mass of the body. The  $x_B$  axis points forward following the movement direction creating a plane with  $y_B$ , while the  $z_B$  axis is perpendicular to  $x_B$  and points downwards during cruise conditions. It follows the right-hand rule.

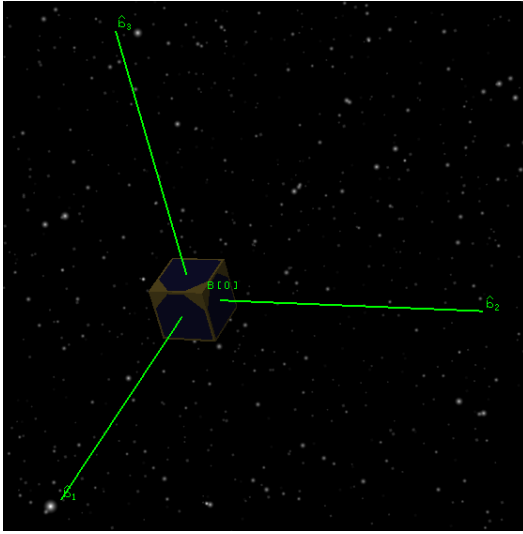


Figure 3.8 CubeSat Body frame with 42.

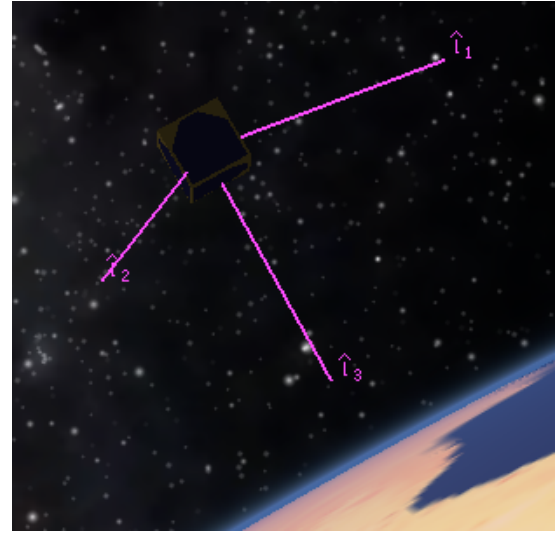


Figure 3.9 CubeSat LVLH frame with 42.

### 3.2.5 Formation Frame

Satellite formation flying is the coordination of multiple satellites into a cluster to accomplish the same capabilities of a larger and more expensive spacecraft. From this cluster, an initial position is considered the local origin from which the spacecraft are positioned. For instance, the example origin corresponds to the centre of mass of one CubeSat. The  $x_F$  axis points to nadir while  $x_F, y_F$  create a plane that follows the right-hand rule.

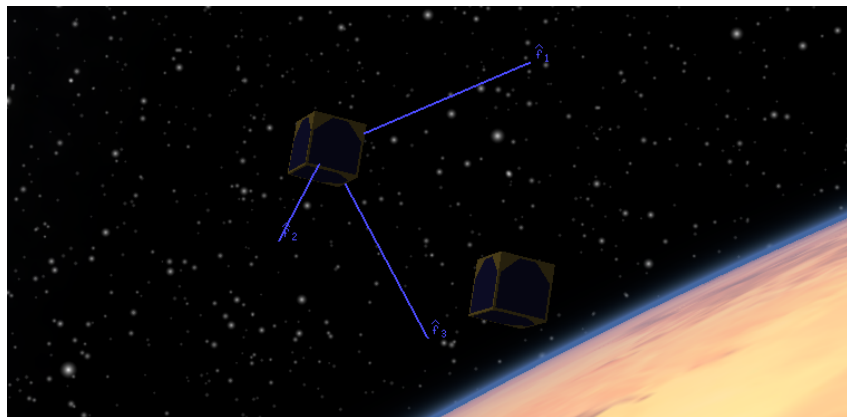


Figure 3.10 CubeSat Formation frame with 42.

### 3.3 Rotations

Once the reference frames are known, there is a need to change coordinates to simplify calculations.

NASA 42 is capable of understanding such rotations with different methods. Among the most useful are Euler angles and Quaternions.

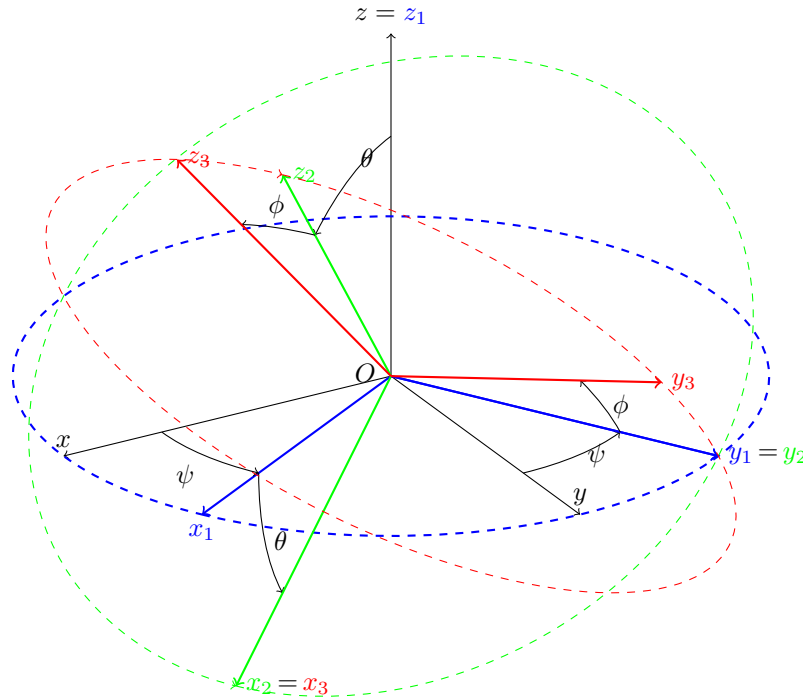
#### 3.3.1 Euler angles

The Euler angles represent three successive finite rotations around pre-established axes in an unalterable order.

For the orthonormal axes, it has been proven that a succession of three rotations results in a unique 3D space orientation. Classic Euler angles rotate around the first axis, then along the second axis, and complete the rotation around the first axis, while Tait–Bryan angles are a variation that does not repeat any previous axis.

The three Euler angles are designated Roll ( $\phi$ ), Pitch ( $\theta$ ) and Yaw ( $\psi$ ). They are defined as follows:

- $\phi$  : Rotation around the intrinsic x-axis.
- $\theta$  : Rotation around the intrinsic y-axis.
- $\psi$  : Rotation around the intrinsic z-axis.



**Figure 3.11** Euler angles rotations

The rotation  $R$  around  $\alpha$  axis with the  $\beta$  angle is written as  $R_\alpha(\beta)$ . The three independent rotations can be expressed with the following matrices:

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \phi & -\sin \phi \\ 0 & \sin \phi & \cos \phi \end{bmatrix} \quad (3.3.1)$$

$$R_y(\theta) = \begin{bmatrix} \cos \phi & 0 & \sin \phi \\ 0 & 1 & 0 \\ -\sin \phi & 0 & \cos \phi \end{bmatrix} \quad (3.3.2)$$

$$R_z(\psi) = \begin{bmatrix} \cos \phi & -\sin \phi & 0 \\ \sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3.3.3)$$

One typical order to combine the three rotations is ZYX, which can be expressed as

$$R_{zyx}(\psi, \theta, \phi) = R_z(\psi)R_y(\theta)R_x(\phi) \quad (3.3.4)$$

which results in

$$R_{zyx}(\psi, \theta, \phi) = \begin{bmatrix} \cos \theta \cos \psi & -\cos \phi \sin \psi + \sin \phi \sin \theta \cos \psi & \sin \phi \sin \psi + \cos \phi \sin \theta \cos \psi \\ \cos \theta \sin \psi & \cos \phi \cos \psi + \sin \phi \sin \theta \sin \psi & -\sin \phi \cos \psi + \cos \phi \sin \theta \sin \psi \\ -\sin \theta & \sin \phi \cos \theta & \cos \phi \cos \theta \end{bmatrix} \quad (3.3.5)$$

The major obstacle when using Euler angles is called *gimbal lock*. When the axes of two of the three gimbals are driven into a parallel configuration, one degree of freedom is lost.

### 3.3.2 Quaternions

Unit quaternions are used to represent spatial orientations and rotations of elements in 3D space. They do not represent intuitive rotations compared to Euler angles but are more efficient and numerically stable.

As an extension of complex numbers, unit quaternions can be expressed as

$$q = w + x \cdot \mathbf{i} + y \cdot \mathbf{j} + z \cdot \mathbf{k} \quad \text{where} \quad [a, b, c, d] \in \mathbb{R} \quad \text{and} \quad a^2 + b^2 + c^2 + d^2 = 1 \quad (3.3.6)$$

The basic quaternions are  $\mathbf{i}$ ,  $\mathbf{j}$  and  $\mathbf{k}$  and their multiplication can be seen on the following table:

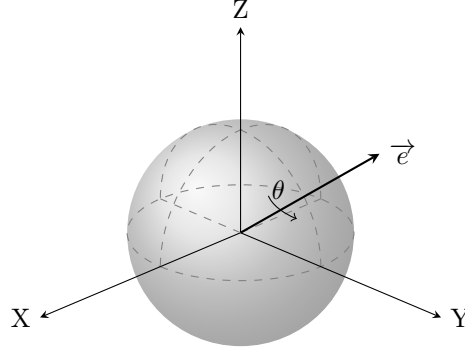
	<b>1</b>	<b>i</b>	<b>j</b>	<b>k</b>
<b>1</b>	1	i	j	k
<b>i</b>	i	-1	k	-j
<b>j</b>	j	-k	-1	i
<b>k</b>	k	j	-i	-1

**Table 3.2** Quaternion multiplication table



By using an extension of Euler's formula, it is commonly viewed as a rotation of an angle  $\theta$  around a unit vector  $\vec{e} = (e_1, e_2, e_3)$ .

$$q = \cos \frac{\theta}{2} + (x \cdot \mathbf{i} + y \cdot \mathbf{j} + z \cdot \mathbf{k}) \sin \frac{\theta}{2} \quad (3.3.7)$$



**Figure 3.12** Rotation of an angle  $\theta$  around a unit vector  $\vec{e}$

This rotation can also be written vectorially as

$$\vec{q} = \begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos(\theta/2) \\ \sin(\theta/2) \cdot e_1 \\ \sin(\theta/2) \cdot e_2 \\ \sin(\theta/2) \cdot e_3 \end{bmatrix} \quad (3.3.8)$$

In order to be able to use both Euler angles and quaternion, the transformation from Euler angles to quaternions can be expressed as

$$\begin{bmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{bmatrix} = \begin{bmatrix} \cos(\phi/2) \cos(\theta/2) \cos(\psi/2) + \sin(\phi/2) \sin(\theta/2) \sin(\psi/2) \\ \sin(\phi/2) \cos(\theta/2) \cos(\psi/2) - \sin(\phi/2) \cos(\theta/2) \sin(\psi/2) \\ \cos(\phi/2) \sin(\theta/2) \cos(\psi/2) + \sin(\phi/2) \cos(\theta/2) \sin(\psi/2) \\ \cos(\phi/2) \cos(\theta/2) \sin(\psi/2) - \sin(\phi/2) \sin(\theta/2) \cos(\psi/2) \end{bmatrix} \quad (3.3.9)$$

while the reverse operation, from quaternions to Euler angles, is expressed as

$$\begin{aligned} \phi &= \text{atan} \left( 2 (q_2 q_3 + q_0 q_1), (q_0^2 - q_1^2 - q_2^2 + q_3^2) \right) \\ \theta &= \text{asin} \left( -2 (q_1 q_3 - q_0 q_2) \right) \\ \psi &= \text{atan} \left( 2 (q_1 q_2 + q_0 q_3), (q_0^2 + q_1^2 - q_2^2 - q_3^2) \right) \end{aligned} \quad (3.3.10)$$

## 3.4 Satellite dynamics

Satellite dynamics is highly reliant on rotations to reach the desired attitude of the spacecraft. In order to easily understand this movement, some concepts are explained below:

### 3.4.1 The Tensor of Inertia

Analogous to the mass when dealing with rectilinear forces, it can be described as the body's resistance to change its angular velocity. The inertia tensor is referenced to a frame fixed to the body with an arbitrary origin that must be solidary to the body. The expression can be written in matrix form as,

$$\mathbf{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix} \quad (3.4.1)$$

where  $I_{ij}$  with  $i = j$  are called moments of inertia, while when  $i \neq j$  the quantities are called products of inertia. It can be seen as the resistance the object opposes to change its angular velocity in the coordinate  $j$  due to a moment of forces applied at the coordinate  $i$ .

It follows from the definition of the products of inertia that the tensors of inertia are always symmetric.

### 3.4.2 Angular momentum

The angular momentum is the angular analogous of the linear momentum ( $p = m \cdot v$ ) and gives us an idea about how the mass is distributed in a rigid body. The expression can be written in matrix form as,

$$\begin{pmatrix} H_{Gx} \\ H_{Gy} \\ H_{Gz} \end{pmatrix} = \begin{pmatrix} I_{xx} & -I_{xy} & -I_{xz} \\ -I_{yx} & I_{yy} & -I_{yz} \\ -I_{zx} & -I_{zy} & I_{zz} \end{pmatrix} \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix} \quad (3.4.2)$$

or,

$$\vec{H}_G = I_G \cdot \vec{\omega} \quad [\text{N} \cdot \text{m} \cdot \text{s}] \quad (3.4.3)$$

where  $I_G$  is the tensor of inertia about the centre of mass  $G$  and with respect to  $xyz$  axes. Analogously, the tensor of inertia can be defined about a point  $O$  with the proper moments and products of inertia about that point [41].

The derivative of the angular momentum describes the torque applied to the object. If  $\vec{\tau}$  is the vector of torque applied to the body, the following expression can be written as,

$$\frac{d}{dt}(\vec{H}) = \vec{\tau} \quad (3.4.4)$$

which, can be further developed for a constant value of the tensor as,

$$\vec{\tau} = \frac{d}{dt}(\vec{H}) = \frac{d}{dt}(I \cdot \vec{\omega}) = I \cdot \dot{\vec{\omega}} + I \cdot \vec{\dot{\omega}} = I \cdot \frac{d}{dt}(\vec{\omega}) \quad (3.4.5)$$

This expression is valid when using the inertial frame in conjunction with a rigid body.

## 3.5 Attitude Determination and Control System

The space environment is governed by the force of gravity and perturbation disturbances. An uncontrolled spacecraft in this medium will be affected by these forces and compelled to change the original orientation, either as a translation or a rotation caused by torque.

This orientation of a body is formally denominated attitude. The first step to change the current attitude vectors is determining their current position.

### 3.5.1 Attitude Determination

In order to obtain the satellite's attitude, a set of instruments can be used alone or in combination to reduce measurement errors:

#### Gyroscopes

Gyroscopes are internal devices able to measure three-dimensional space rotations. This device can be based on different operating principles depending on the precision required and the space available. If combined with magnetic compasses, they are called gyrocompasses. This combination is used to sense the direction toward the orbited body centre and obtain rotation about an axis normal to the orbit plane.

For instance, Hubble Space Telescope needed three of its six gyroscopes operating to ensure optimal efficiency [42]. Future telescopes such as the James Webb Space Telescope will use Hemispherical Resonator Gyroscopes (HRG).

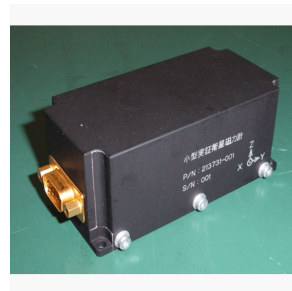
#### Magnetometers

Magnetometers measure magnetic field strengths and direction if a three-axis triad is assembled. These instruments were firstly used to map the Van Allen radiation belts around Earth and Earth's and other near planets magnetic field. With this data, the field strength and direction is compared to the onboard memory to determine the spacecraft's attitude.

Due to the distance dependence with the central body, this device can only measure the magnetic field for low altitude orbits.



**Figure 3.13** HRG.  
Extracted from [43].



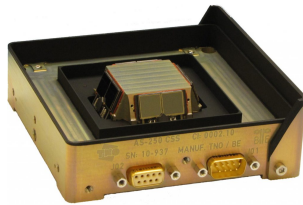
**Figure 3.14** 3-Axis Magnetometer.  
Extracted from [44].

## Sun Sensors

Sun sensors detect the direction of the nearest, brightest star, which is the Sun in the Solar System. They can update the initial gyroscope position and point the solar arrays perpendicular to the Sun for the highest efficiency.

Sun sensors can be classified into coarse and fine sun sensors. The coarse variant measures the current output as a value proportional to the cosine of the angle between the Sun and the normal to a photocell. Conversely, the fine variant uses an aperture to create a sunspot on either a position-sensitive device such as a four-quadrant photodiode [45].

Therefore, the first method is less accurate and considerably cheaper than the optimised, high accuracy fine variant. The difference is so that fine sun sensors typically cost more than 2000 times the coarse version [46].



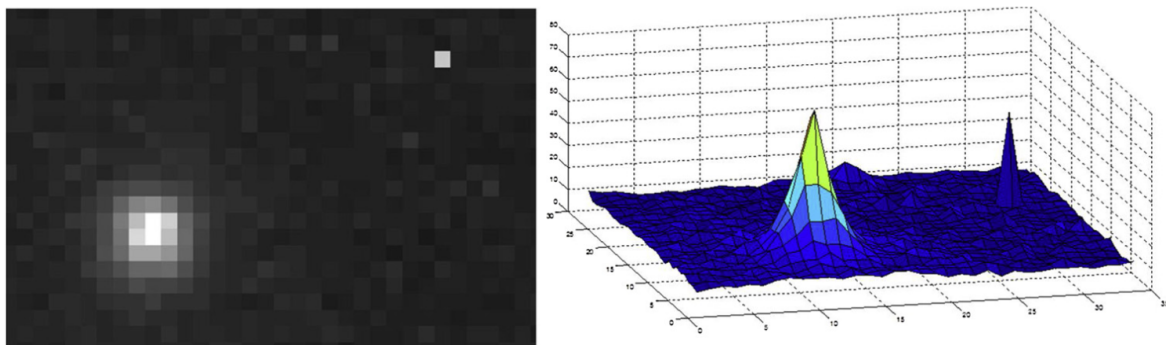
**Figure 3.15** Coarse Sun Sensor.  
Extracted from [43].



**Figure 3.16** Fine Sun Sensor.  
Extracted from [44]

## Star Trackers

A star tracker is a precise optical device that calculates the positions of stars using photocells or a mounted camera. Its operation is based on identifying and then obtaining the relative position to visible stars from the brightness and spectral type.



**Figure 3.17** Star image and its amplitude taken in LEO. Extracted from [47].

### 3.5.2 Attitude Control actuators

Once the attitude is known, the attitude control is in charge of maintaining the current vectors or changing them to the desired new attitude.

#### Reactions Wheels

Reactions wheels are equipped with an electric power source that can spin the wheel. Based on the conservation of angular momentum, the spacecraft will counter-rotate proportionally in the opposite direction to the motor's rotation.

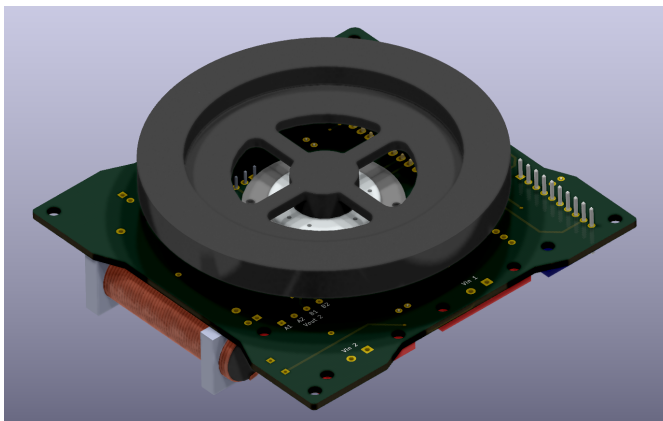


Figure 3.18 PLATHON reaction wheel prototype

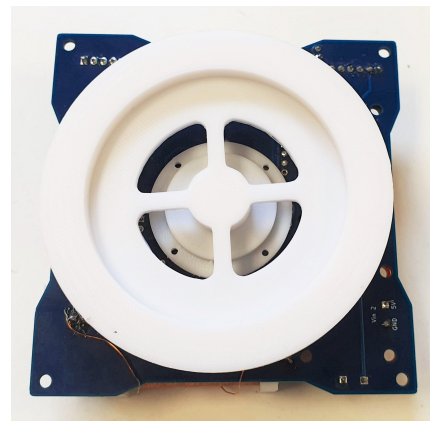


Figure 3.19 PLATHON prototype

#### Magnetorquers

Magnetorquers interact with the central body magnetic field, creating a magnetic dipole to provide a torque to the satellite. They are a superb choice for CubeSat orbiting LEO due to the reduced distance from Earth.

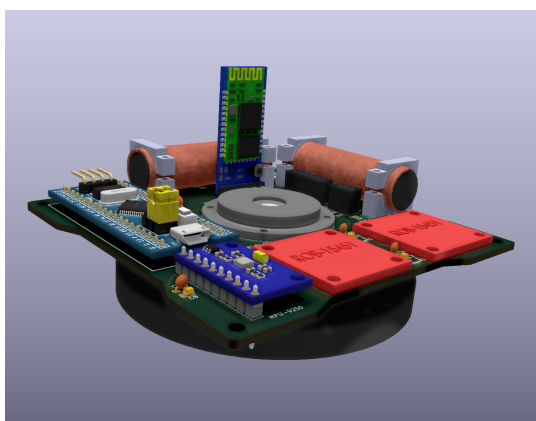


Figure 3.20 PLATHON magnetorquer prototype

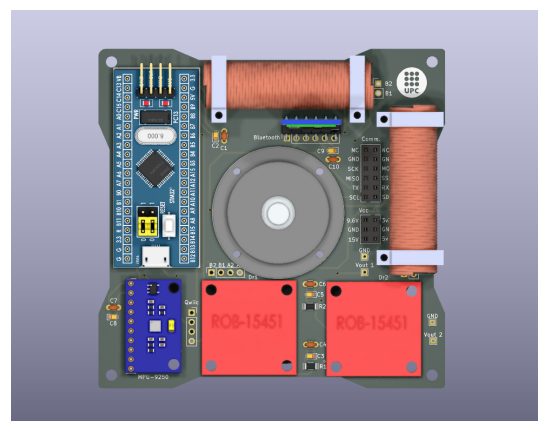


Figure 3.21 PLATHON prototype

#### Thrusters

Thrusters are based on the ejection of mass on each of the satellite axes to provide a torque to place the satellite into the desired attitude. For the PLATHON project, the use of thrusters has been discarded.

## Chapter 4

# 42: Spacecraft Simulation

### 4.1 Installation and testing

42 is a portable software that can be launched on Mac, Windows or Linux to minimise infrastructure requirements. However, Linux systems require less complexity and have been selected for this study. If Linux is not already installed, the reader can either create a new partition or install it in a VirtualMachine. Furthermore, consider that the Virtual Machine option will not use the graphical power from the computer, affecting the GUI speed.

In order to install 42 <sup>1</sup>, the following steps ought to be taken:

1. Create a directory to harbour 42.
2. Through Linux's terminal, access the directory and download the source code executing the command:

`git clone https://github.com/ericstoneking/42.git`

3. The libraries of OpenGL and other tools have to be installed into the system by executing the following commands as superuser:

- Enter `sudo apt-get update`
- Enter `sudo apt-get install freeglut3`
- Enter `sudo apt-get install freeglut3-dev`
- Enter `sudo apt-get install binutils-gold`
- Enter `sudo apt-get install g++ cmake`
- Enter `sudo apt-get install libglew-dev`
- Enter `sudo apt-get install mesa-common-dev`
- Enter `sudo apt-get install build-essential`
- Enter `sudo apt-get install libglew1.5-dev libglm-dev`

4. Access the directory where 42 is installed and execute the command: `make`

If all the necessary libraries are installed, the software should compile correctly, and inside the 42's directory, it will have created a new executable file.

---

<sup>1</sup>This tutorial has been tested with Ubuntu 20.04 and previous versions, as well as Linux Mint 20.1. Future versions may require additional steps.

Upon doing this thesis, the OpenSatKit integration is only possible with Ubuntu 18.04 LTS and previous versions.

The process of installing 42 on Windows requires MinGW and Msys and can be found on the documentation folder.

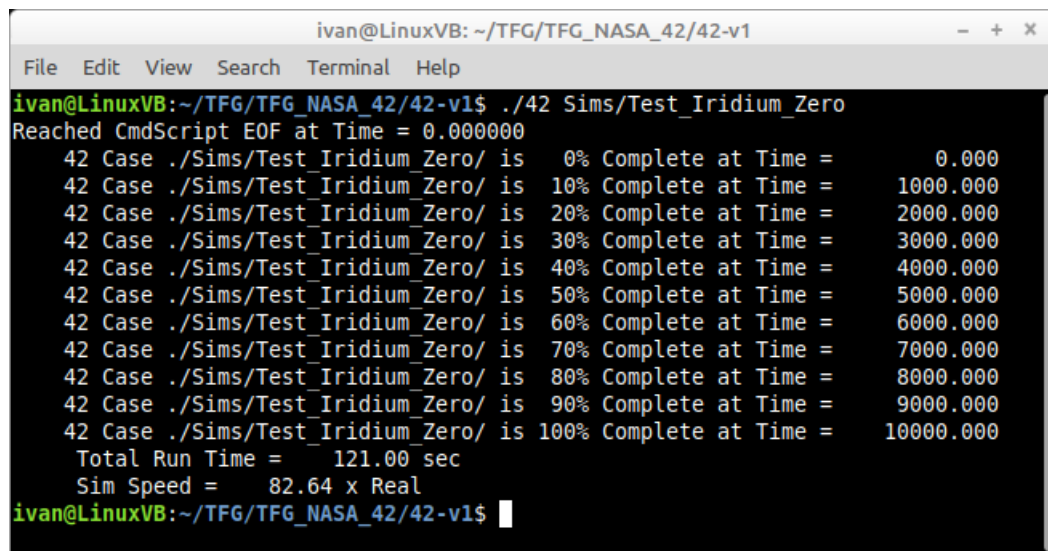
#### 4.1.1 Validation

Once the software is installed, the performance can be examined by accessing 42's directory and running the command:

`./42` or `./42 Demo`

In this case, `./42` opens the programme like a blank canvas while `./42 Demo` runs a predefined example mission. While entering these commands to the terminal, it is crucial to remember that it is case sensitive.<sup>2</sup>

If the GUI is disabled, the simulation will run and show the user the percentage of completion and the local time on the terminal:



```
ivan@LinuxVB: ~/TFG/TFG_NASA_42/42-v1
File Edit View Search Terminal Help
ivan@LinuxVB:~/TFG/TFG_NASA_42/42-v1$ ./42 Sims/Test_Iridium_Zero
Reached CmdScript EOF at Time = 0.000000
42 Case ./Sims/Test_Iridium_Zero/ is 0% Complete at Time = 0.000
42 Case ./Sims/Test_Iridium_Zero/ is 10% Complete at Time = 1000.000
42 Case ./Sims/Test_Iridium_Zero/ is 20% Complete at Time = 2000.000
42 Case ./Sims/Test_Iridium_Zero/ is 30% Complete at Time = 3000.000
42 Case ./Sims/Test_Iridium_Zero/ is 40% Complete at Time = 4000.000
42 Case ./Sims/Test_Iridium_Zero/ is 50% Complete at Time = 5000.000
42 Case ./Sims/Test_Iridium_Zero/ is 60% Complete at Time = 6000.000
42 Case ./Sims/Test_Iridium_Zero/ is 70% Complete at Time = 7000.000
42 Case ./Sims/Test_Iridium_Zero/ is 80% Complete at Time = 8000.000
42 Case ./Sims/Test_Iridium_Zero/ is 90% Complete at Time = 9000.000
42 Case ./Sims/Test_Iridium_Zero/ is 100% Complete at Time = 10000.000
Total Run Time = 121.00 sec
Sim Speed = 82.64 x Real
ivan@LinuxVB:~/TFG/TFG_NASA_42/42-v1$
```

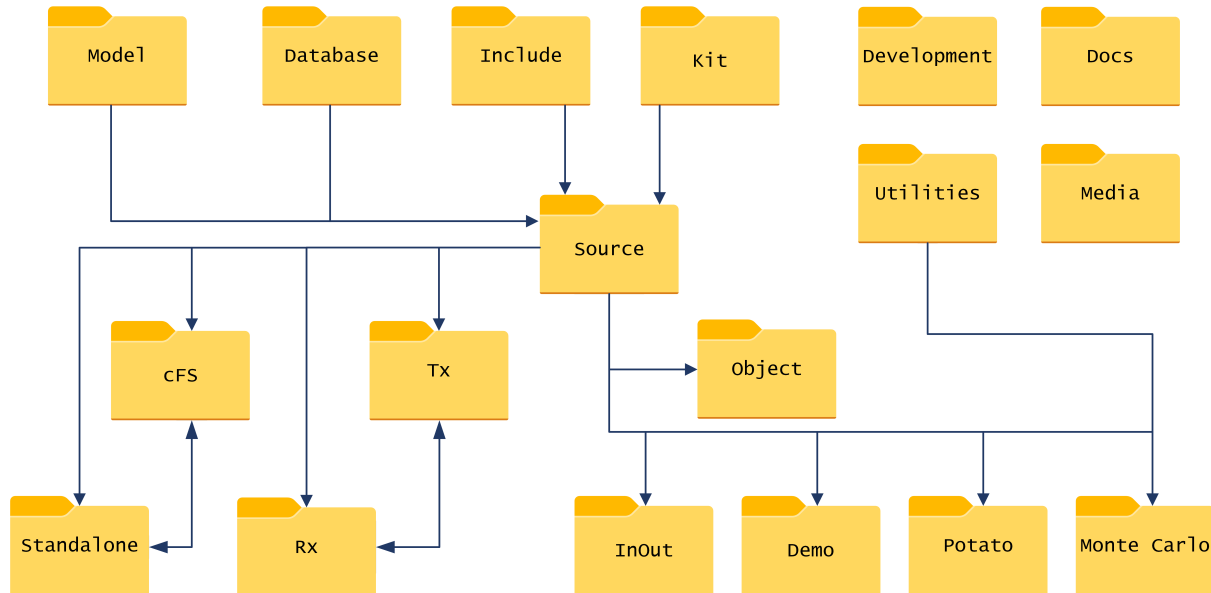
**Figure 4.1** Linux terminal running example programme without GUI

If an error appears after the execution, there can either be an error with the programme download process or some libraries have not been correctly installed.

<sup>2</sup>If the software is installed in a VM and the GUI is not displayed correctly, disabling 3D Acceleration might fix it.

## 4.2 File distribution

Once the files are downloaded from the GIT repository, after unzipping the *42-master* folder, the following folders are visible:



**Figure 4.2** Main folder connections

- **Docs:** contains all the official documentation. The *42 Overview* file summarises the programme capabilities.
- **Development:** contains C and Matlab functions to either add new capabilities or test the current programme.
- **Utilities:** contains C and Matlab secondary programmes. Among them, there are file transformations and parallel run examples.
- **Media (Screenshots):** contains screenshots from the demonstration project and the commands to record the simulation.
- **Model:** contains the image files for the textures and the object files for the spacecraft.
- **Database:** contains Python scripts to serve as the interface between 42 and some database functionality. The Python scripts build json dictionaries that can be read by the cFS or generate some Inter-Process Communication (IPC) C code.
- **Include:** contains the header files with 42 definitions and types.
- **Kit:** contains the headers and C codes for the 42 libraries.
- **Source:** contains all the C codes that comprise 42. It comprises the previous folders capabilities as shown in figure 4.2.
- **Objects:** contains the object files. The compiler creates an object file for each source file before linking them to the final executable with the make file.
- **InOut:** contains the default-executable files that run when the command `./42` is written.



- **Demo:** contains a demonstration of most of 42 capabilities with a set of bodies at different flight regimes.
- **Potato:** contains an example of an independent file simulation. Can be called using `./42 Potato`.
- **Monte Carlo:** contains a Monte Carlo simulation campaign built with Matlab that can be tailored to suit each project. Connected to the utilities folder.
- **Standalone:** contains a simulation case built to demonstrate running the controller code in `AcApp.c` either embedded within 42 or as a standalone app communicating with 42 via sockets.
- **Tx:** contains a simulation case to send data through a socket interface.
- **Rx:** contains a simulation case to receive data through a socket interface. The `.txt` files have to be the same as those from the Tx folder to prevent segmentation faults.
- **cFS:** contains the simulation files to test the Standalone capabilities mimicking the OpenSatKit cFS.

It is recommended to create a new folder for development purposes to concentrate all the simulations in one place. From now on, this folder will be referred to as the `Sims` folder.

#### 4.2.1 Algorithm folders

42 is written in C for speed and portability. The programme is divided into three layers of code:

Low-level toolkit functions are located in the `Kit` folder. These functions are subdivided into source code inside `Kit/Source` and function prototypes inside `Kit/Include`. Besides, there is a `shaders` folder for the GUI based on OpenGL Shading Language.

Mid-level functions are based on low-level definitions and have access to global variables to perform a particular need.

Top-level functions indicate the computation order and do minimal logic operations. They have access to all global variables.

Each code file is briefly commented in the Appendix [\[B.2\]](#).

## 4.3 Simulation procedure

The simulation algorithm starts loading data from the input files. The order of this initialisation can be seen on the Input files flowchart [4.4](#). Once all files have been read, the necessary variables are created to distribute all the celestial bodies, spacecraft and orbits through ephemeris models.

The ephemeris data is sent to both environment models and sensor models to consider the perturbation effects. The environment model also sends data to the sensor models as if the instruments were reading the real data.

Once the sensor models have obtained all the necessary data, it is sent to the Flight Software models, where control laws are processed to communicate with the actuator models. These control laws can either be initially stated inside 42 or sent as an external command. At that point, both the environmental and control forces and torques are obtained and sent to the next phase.

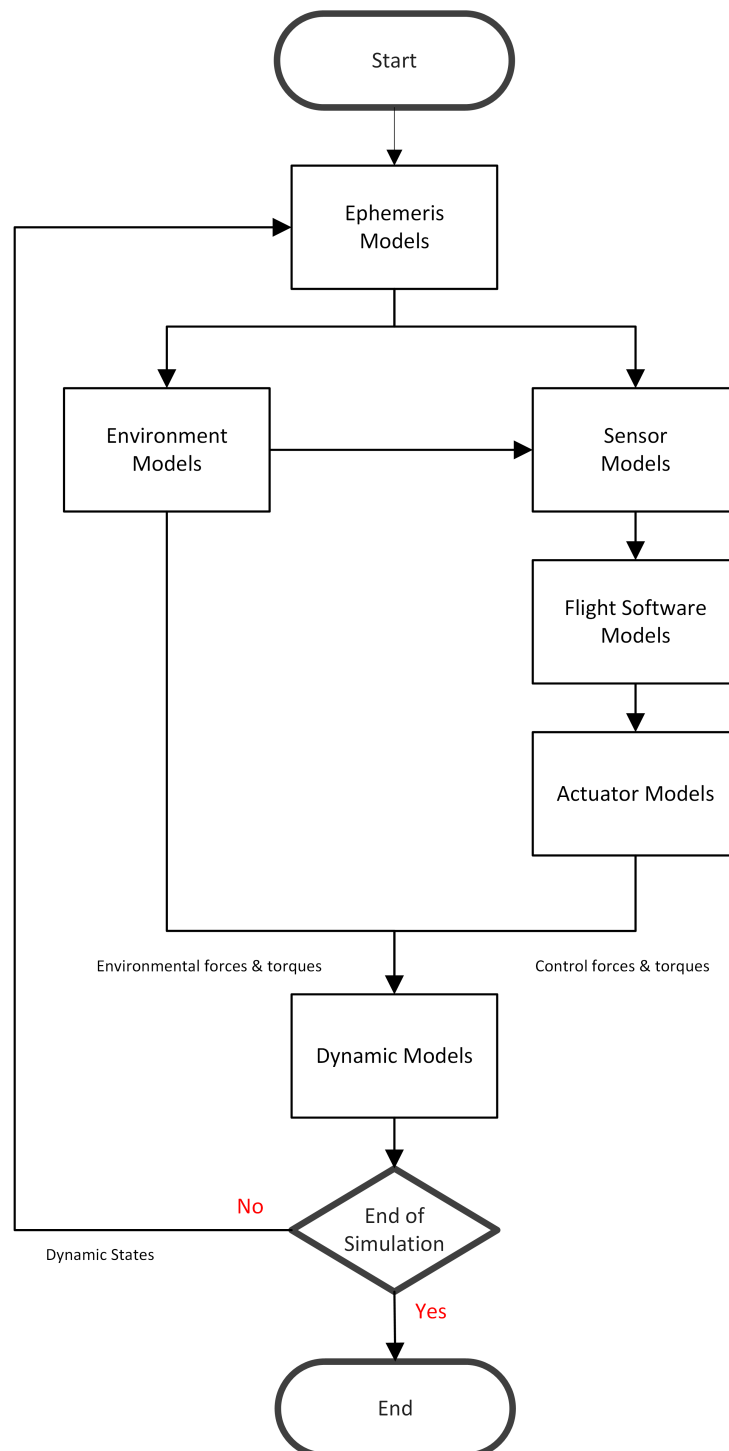
For this iteration, the dynamic equations of motion are integrated. If the current time is equal to the end of simulation time, the programme ends; if not, the obtained dynamic states are sent to the ephemeris models in a closed-loop, advancing to the next step.

If the simulation is completed, the programme creates output files to save relevant data.

To get a general idea of what each block contains:

- The ephemeris model gives the trajectory of natural bodies. Depending on the project range, the ephemeris type is more suitable for some planets or moons. 42 can use JPL DE430.
- The environment model takes into account the magnetic field, atmospheric properties and solar radiation. With this data, the environmental perturbations are obtained.
- The sensor model measures the simulated truth, obtained through position, quaternions and other variables, and includes noise and inaccuracies according to each system.
- The Flight Software model applies a particular control algorithm at a specific time. In most cases, a proportional-derivative controller is used.
- The actuator model receives the forces and torques from the Flight Software model and obtains the output forces and torques through simple models. If the actuators are disabled, the programme does not alter the initial values.
- The dynamic model solves the dynamics equations of motion numerically through the use of a constant step Runge-Kutta algorithm [A.2](#)

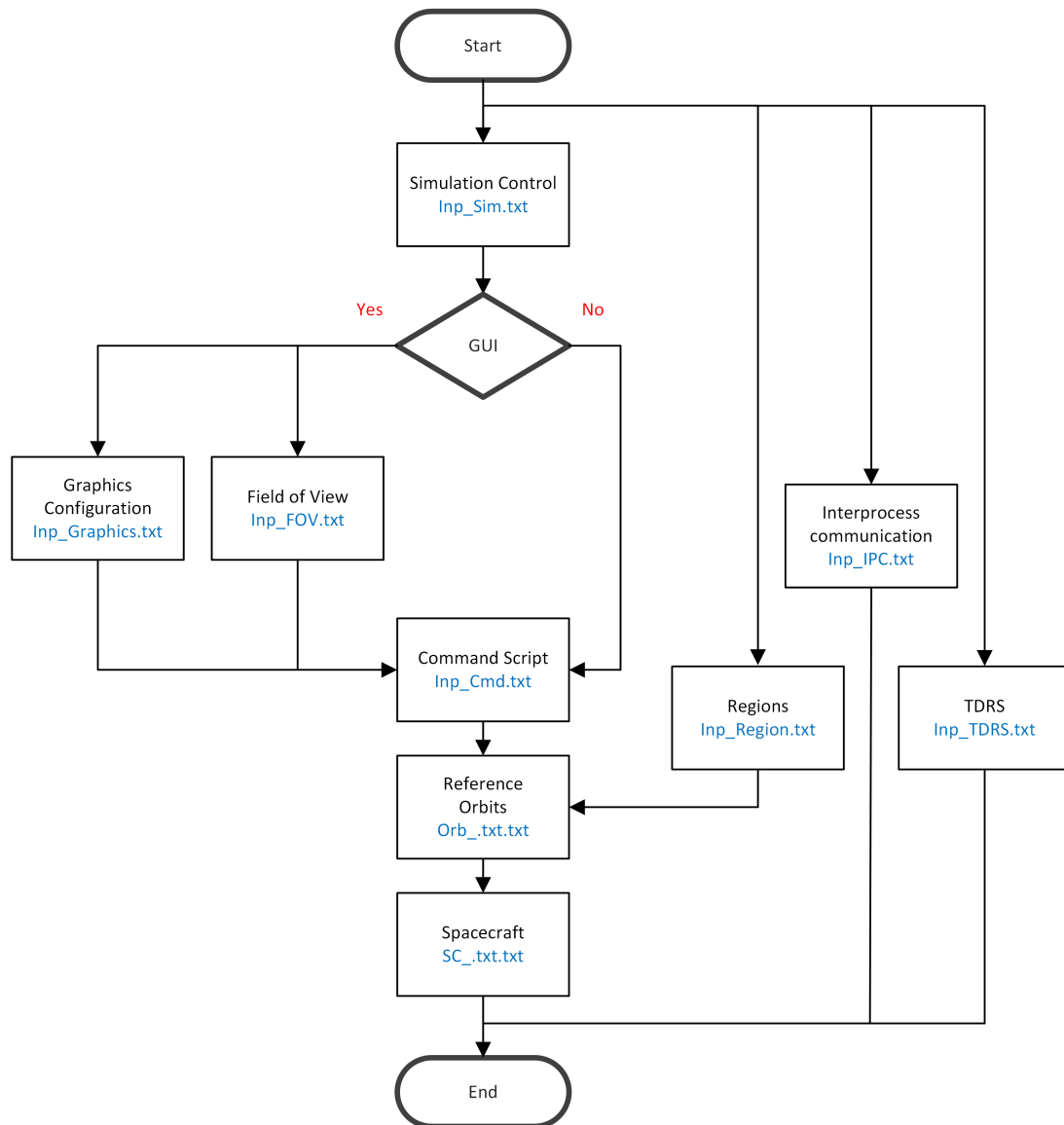
The aforementioned closed loop is described in the following flowchart:



**Figure 4.3** 42 Simulation loop

## 4.4 Input files

42 reads all initial input data from text documents located in the InOut folder when the `./42` command is executed. For reading other folders, the name has to be appended to the end. For instance, to execute the files inside the Demo folder, the command `./42 Demo` is executed as seen in the validation section. For other simulations, after creating the Sims folder, the command would be `./42 Sims/Example`. The following flowchart shows the mandatory input files and their dependencies:



**Figure 4.4** Input files flowchart with dependencies

If the Graphical User Interface is deactivated, the graphics configuration and field of view files are not required to initiate the simulation.

#### 4.4.1 Simulation Control

The simulation file is the simulation controller and is divided into the following parts:

[illegible]

## Simulation Control

The time mode determines the speed of the simulation. While FAST executes the program at maximum speed, the REAL mode mimics reality. The external mode is to get the mode from an external application such as cFS or NOS3.

The simulation duration is the total time, in seconds, during which the simulation will extract data. The step size refers to the computational time between two subsequent instants. Similarly, the file output interval is the frequency for which 42 saves the data into a text document or sends it through a socket.

The graphical user interface can be activated by writing the boolean values TRUE or FALSE. The initialisation of commands is performed at the Command Script with the provided name. This file is further explained at Section 4.4.2.

```

8 ***** Reference Orbits *****
9 2 ! Number of Reference Orbits
10 TRUE Orb_LEO.txt ! Input file name for Orb 0
11 FALSE Orb_GEO.txt ! Input file name for Orb 1

```

## Reference Orbits

The number of reference orbits determines the orbit pool from which a spacecraft can select a designated orbit, either if they are active or not. Each following line contains information about one distinguished reference orbit. The first value defines if it can be used, and the second value the number of the orbit file where all the data can be found. Orbit files are further explained at Section 4.4.3.

```

12 ***** Spacecraft *****
13 2 ! Number of Spacecraft
14 TRUE 0 SC_CfsSat0.txt ! Existence, RefOrb, Input file for SC 0
15 FALSE 1 SC_CfsSat1.txt ! Existence, RefOrb, Input file for SC 1

```

## Spacecraft

The number of spacecraft determines the number of bodies that are being generated and inserted into the simulation. The first value defines existence. The second value refers to the orbit at which the spacecraft navigates, and the third entry contains the name of the satellite file where all the data can be found. If the same satellite is used more than once, it must be paired each time with the corresponding reference orbit.

```

16 ***** Environment *****
17 03 21 2016          ! Date (Month, Day, Year)
18 12 00 00.00        ! Greenwich Mean Time (Hr,Min,Sec)
19 0.0                ! Time Offset (sec)
20 USER_DEFINED       ! Model Date Interpolation (TWO SIGMA_KP, NOMINAL or USER_DEFINED)
21 230.0              ! If USER_DEFINED, enter desired F10.7 value
22 100.0              ! If USER_DEFINED, enter desired AP value
23 IGRF               ! Magfield (NONE,DIPOLE,IGRF)
24 8 8                ! IGRF Degree and Order (<=10)
25 8 8                ! Earth Gravity Model N and M (<=18)
26 2 0                ! Mars Gravity Model N and M (<=18)
27 2 0                ! Luna Gravity Model N and M (<=18)
28 FALSE FALSE       ! Aerodynamic Forces & Torques (Shadows)
29 FALSE             ! Gravity Gradient Torques
30 FALSE FALSE       ! Solar Pressure Forces & Torques (Shadows)
31 FALSE             ! Gravity Perturbation Forces
32 FALSE             ! Passive Joint Forces & Torques
33 FALSE             ! Thruster Plume Forces & Torques
34 FALSE             ! RWA Imbalance Forces and Torques
35 FALSE             ! Contact Forces and Torques
36 FALSE             ! CFD SLOSH Forces and Torques
37 FALSE             ! Output Environmental Torques to Files

```

### Environment

The environment subsection sets up the initial date and time variables. Next, an optional time offset can be added to simulate a mission delay. Subsequently, the model data interpolation for the Solar Flux and AP values can be selected from TWO SIGMA\_KP, NOMINAL or USER DEFINED. Finally, for the last option, lines 21 and 22 are used to enter the values manually.

The magnetic field model can be disabled with NONE, simulated by a DIPOLE or use the International Geomagnetic Reference Field (IGRF). The following line refers to the Schmidt quasi-normalised associated Legendre functions of degree n and order m. The programme has been updated with values up to 10. In 2019, degree and order 13 were published by IAGA [48].

The gravity models represent an ellipsoidal harmonic model up to degree N and order M and are only available for Earth, Moon and Mars. From lines 28 to 37, different perturbations are defined and can be activated with the boolean values.

```

38 ***** Celestial Bodies of Interest *****
39 VSOP87             ! Ephem Option (VSOP87 or DE430)
40 FALSE              ! Mercury
41 FALSE              ! Venus
42 TRUE               ! Earth and Luna
43 FALSE              ! Mars and its moons
44 FALSE              ! Jupiter and its moons
45 FALSE              ! Saturn and its moons
46 FALSE              ! Uranus and its moons
47 FALSE              ! Neptune and its moons
48 FALSE              ! Pluto and its moons
49 FALSE              ! Asteroids and Comets

```

### Celestial Bodies of Interest

The Ephemeris can be described with the semi-analytic planetary theory VSOP (*French: Variations Séculaires des Orbites Planétaires*) or the Jet Propulsion Laboratory Development Ephemeris. Both are mathematical models of the Solar System. The boolean values determine if the planet and its moons are considered for the simulation and the subsequent view on the graphical interface.

## Lagrange Point System

## Ground Stations

51





Once the ellipse shape is known, the angle concerning the planet is inserted with the inclination. Next, the RAAN determines the node line with respect to the first point of Aries. The argument of the periapsis fixes the ellipse. Finally, the true anomaly determines the position of the spacecraft along the ellipse.

If the position and velocity option is selected, lines 21 and 22 read the 3-axis decomposition. For the file option, the format has to be selected between TLE and TRV, and the label serves as a search engine inside the file. Both the label and the file name have to be enclosed in double-quotes.

```

25 :::::::::::::: Use these lines if THREE_BODY ::::::::::::::
26 SUNEARTH                ! Lagrange system
27 LAGDOF_MODES            ! Propagate using LAGDOF_MODES or LAGDOF_COWELL or
    LAGDOF_SPLINE
28 MODES                   ! Initialize with MODES or XYZ or FILE
29 L2                      ! Libration point (L1, L2, L3, L4, L5)
30 800000.0                ! XY Semi-major axis, km
31 45.0                    ! Initial XY Phase, deg (CCW from -Y)
32 CW                      ! Sense (CW, CCW), viewed from +Z
33 0.0                     ! Second XY Mode Semi-major Axis, km (L4, L5 only)
34 0.0                     ! Second XY Mode Initial Phase, deg (L4, L5 only)
35 CW                      ! Sense (CW, CCW), viewed from +Z (L4, L5 only)
36 400000.0                ! Z Semi-axis, km
37 60.0                    ! Initial Z Phase, deg
38 1.05 0.5 0.0           ! Initial X, Y, Z (Non-dimensional)
39 0.0 0.0 0.0            ! Initial Xdot, Ydot, Zdot (Non-dimensional)
40 TRV "ORB_ID"           ! TLE, TRV or SPLINE format, Label to find in file
41 "TRV.txt"              ! File name

```

### Orbit Type: Three Body

For the three body orbit type, the available combinations are:

- EarthMoon
- SunEarth
- SunJupiter

```

42 ***** Formation Frame Parameters *****
43 L                      ! Formation Frame Fixed in [NL]
44 0.0 0.0 0.0 123       ! Euler Angles (deg) and Sequence
45 L                      ! Formation Origin expressed in [NL]
46 0.0 0.0 0.0           ! Formation Origin wrt Ref Orbit (m)

```

### Formation Frame

The formation frame can be referenced to the inertial or the LVLH frames. It is combined with a reference orbit to create satellite clusters. Euler angles can be expressed as in [3.3.1](#).



The dynamic flags are used to enable certain properties inside 42 code. The drag coefficient for the atmospheric drag is inserted in this section.

```

7 *****
8 ***** Body Parameters *****
9 *****
10 1 ! Number of Bodies
11 ===== Body 0 =====
12 100.0 ! Mass
13 100.0 200.0 300.0 ! Moments of Inertia (kg-m^2)
14 0.0 0.0 0.0 ! Products of Inertia (xy,xz,yz)
15 0.0 0.0 0.0 ! Location of mass center, m
16 0.0 0.0 0.0 ! Constant Embedded Momentum (Nms)
17 IonCruiser.obj ! Geometry Input File Name
18 NONE ! Flex File Name

```

### Body parameters

The main body parameters are inserted along the geometry file to define the 3D body of the spacecraft.

```

7 *****
8 ***** Joint Parameters *****
9 *****
10 (Number of Joints is Number of Bodies minus one)
11 ===== Joint 0 =====
12 0 1 ! Inner, outer body indices
13 1 213 GIMBAL ! RotDOF, Seq, GIMBAL or SPHERICAL
14 0 123 ! TrnDOF, Seq
15 FALSE FALSE FALSE ! RotDOF Locked
16 FALSE FALSE FALSE ! TrnDOF Locked
17 0.0 0.0 0.0 ! Initial Angles [deg]
18 0.0 0.0 0.0 ! Initial Rates, deg/sec
19 0.0 0.0 0.0 ! Initial Displacements [m]
20 0.0 0.0 0.0 ! Initial Displacement Rates, m/sec
21 0.0 0.0 0.0 312 ! Bi to Gi Static Angles [deg] & Seq
22 0.0 0.0 0.0 312 ! Go to Bo Static Angles [deg] & Seq
23 0.0 0.0 0.0 ! Position wrt inner body origin, m
24 0.0 0.0 0.0 ! Position wrt outer body origin, m
25 0.0 0.0 0.0 ! Rot Passive Spring Coefficients (Nm/rad)
26 0.0 0.0 0.0 ! Rot Passive Damping Coefficients (Nms/rad)
27 0.0 0.0 0.0 ! Trn Passive Spring Coefficients (N/m)
28 0.0 0.0 0.0 ! Trn Passive Damping Coefficients (Ns/m)

```

### Joint parameters

For spacecraft formed by several bodies, the joints have to be described. The locked degrees of freedom restrict the rotation of parts of the total body.

#### 4.4.4.1 ACDS

The spacecraft file contains all the information about the onboard ACDS systems.

Sensors are one interface between the truth model (environment, dynamics) and the FSW model. In general terms, a sensor model takes the truth, perhaps adds some noise or other errors, and repackages the result in the proper format for the FSW model.

The actuator models are another interface between the truth model and the FSW model. These actuator models accept commands from the FSW model and determine the forces and torques to apply to the spacecraft dynamical models.

#### Actuator modelling

```

1 ***** Wheel Parameters *****
2 0 ! Number of wheels
3 ===== Wheel 0 =====
4 0.0 ! Initial Momentum, N-m-sec
5 1.0 0.0 0.0 ! Wheel Axis Components, [X, Y, Z]
6 0.14 50.0 ! Max Torque (N-m), Momentum (N-m-sec)
7 0.012 ! Wheel Rotor Inertia, kg-m^2
8 0.48 ! Static Imbalance, g-cm
9 13.7 ! Dynamic Imbalance, g-cm^2
10 0 ! Flex Node Index

```

#### Wheel Parameters

Each satellite can select a number of wheels defined by the table parameters. The values for the PLATHON prototype wheels can be found on Yi Qiang Ji thesis [49].

The reaction wheel model is straightforward. It reads the required torque from the Flight Software function and verifies that the torque is not greater than the maximum torque the wheel can produce. Then, it also checks that the momentum of the wheel is not greater than the maximum value. The other parameters, such as the imbalance and wheel inertia, are accounted for as a perturbation force rather than inside the wheel modelling.

```

11 ***** MTB Parameters *****
12 0 ! Number of MTBs
13 ===== MTB 0 =====
14 180.0 ! Saturation (A-m^2)
15 1.0 0.0 0.0 ! MTB Axis Components, [X, Y, Z]
16 0 ! Flex Node Index

```

#### Magnetorquer Parameters

Magnetorquers are only defined by the saturation and the position inside the spacecraft. The model defines a torque limit equal to the maximum that the magnetorquer can produce and then applies this torque considering the axis of the magnetorquer and the direction of the magnetic field vector.

```

17 ***** Thruster Parameters *****
18 0 ! Number of Thrusters
19 ===== Thr 0 =====
20 1.0 ! Thrust Force (N)
21 0 -1.0 0.0 0.0 ! Body, Thrust Axis
22 1.0 1.0 1.0 ! Location in Body, m
23 0 ! Flex Node Index

```

### Thruster Parameters

The thrusters require the total thrust force that is ejected from a certain body along the 3D axes and the position of the thrusters with respect to the satellite.

The thruster model reads the required thrust and pulse width from the Flight Software function, verifies that it does not surpass the maximum thrust value and applies said thrust. When the resulting thrust force is not applied at the spacecraft centre of mass, a function computes the generated torque. Finally, it checks for the forces caused by the exhaust plume impacting other spacecraft surfaces if the function is enabled on the simulation file.

The following structure represents the links between data types:

## Global Data Structure Relationships

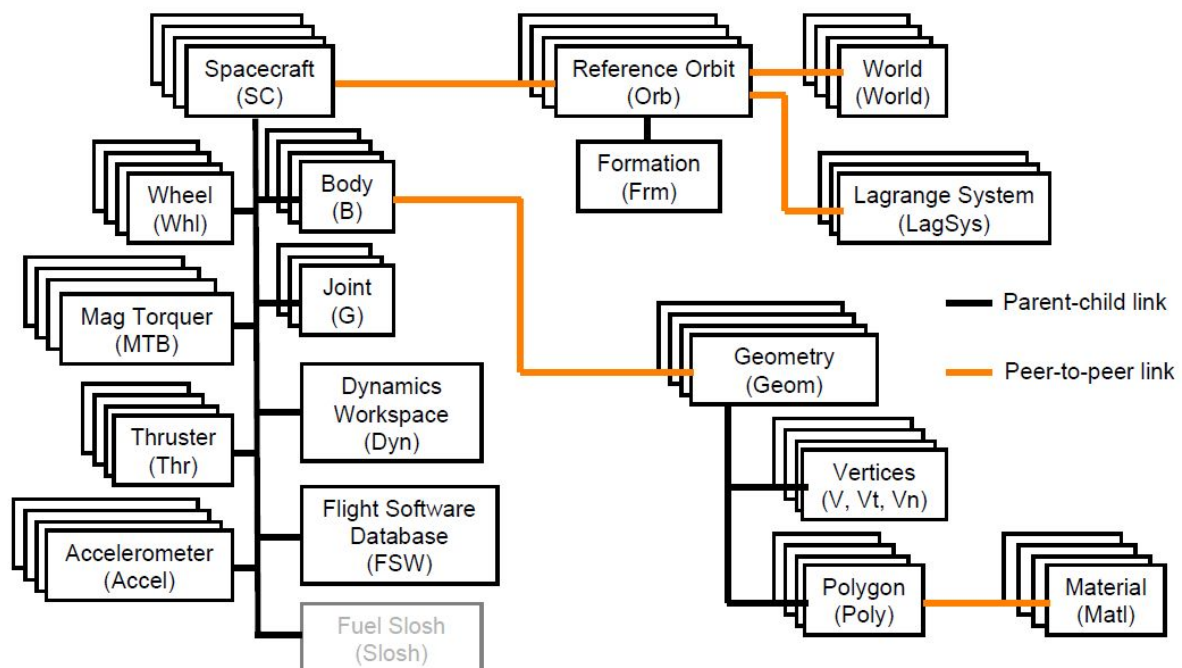


Figure 4.5 Global Data Structure Relationships. Source: [50].

## Sensor modelling

```

24 ***** Gyro *****
25 0 ! Number of Gyro Axes
26 ===== Axis 0 =====
27 0.1 ! Sample Time,sec
28 1.0 0.0 0.0 ! Axis expressed in Body Frame
29 1000.0 ! Max Rate, deg/sec
30 100.0 ! Scale Factor Error, ppm
31 1.0 ! Quantization, arcsec
32 0.07 ! Angle Random Walk (deg/rt-hr)
33 0.1 1.0 ! Bias Stability (deg/hr) over timespan (hr)
34 0.1 ! Angle Noise, arcsec RMS
35 0.1 ! Initial Bias (deg/hr)
36 1 ! Flex Node Index

```

### Gyroscope Parameters

The gyroscope only requires the axis in the body frame to obtain the angular velocity. As the spacecraft is considered a rigid solid, the position is not required. After that, the model reads the uncertainty sources from the scale error factor to the initial bias and computes a result that would approximate to the real world. The maximum rate is verified so that it is never surpassed.

```

37 ***** Magnetometer *****
38 0 ! Number of Magnetometer Axes
39 ===== Axis 0 =====
40 0.1 ! Sample Time,sec
41 1.0 0.0 0.0 ! Axis expressed in Body Frame
42 60.0E-6 ! Saturation, Tesla
43 0.0 ! Scale Factor Error, ppm
44 1.0E-6 ! Quantization, Tesla
45 1.0E-6 ! Noise, Tesla RMS
46 1 ! Flex Node Index

```

### Magnetometer Parameters

The magnetometer determines the value of the magnetic field in the determined axis expressed in the Body Frame. The saturation is taken into account, and then the model reads the uncertainty sources such as the scale error factor, the quantisation error and the noise to compute a result that would approximate to the real world.

```

47 ***** Coarse Sun Sensor *****
48 0 ! Number of Coarse Sun Sensors
49 ===== CSS 0 =====
50 0.1 ! Sample Time,sec
51 0 1.0 1.0 1.0 ! Body, Axis expressed in Body Frame
52 90.0 ! Half-cone Angle, deg
53 1.0 ! Scale Factor
54 0.001 ! Quantization
55 0 ! Flex Node Index

```

### Coarse Sun Sensor Parameters

The Coarse Sun Sensor determines if the sensor is illuminated or in eclipse. If light reaches the spacecraft, it computes the intensity and compares the angle of the actual Sun pointing vector with respect to the sensor defined axis with the half-cone angle. Noise and quantisation are added to the illumination value.

```

56 ***** Fine Sun Sensor *****
57 0 ! Number of Fine Sun Sensors
58 ===== FSS 0 =====
59 0.2 ! Sample Time,sec
60 70.0 0.0 0.0 231 ! Mounting Angles (deg), Seq in Body
61 32.0 32.0 ! X, Y FOV Size, deg
62 0.1 ! Noise Equivalent Angle, deg RMS
63 0.5 ! Quantization, deg
64 0 ! Flex Node Index

```

### Fine Sun Sensor Parameters

The Fine Sun Sensor determines the angles of the Sun pointing vector with respect to the sensor axis without the illumination parameter.

```

65 ***** Star Tracker *****
66 0 ! Number of Star Trackers
67 ===== ST 0 =====
68 0.25 ! Sample Time,sec
69 -90.0 90.0 00.0 321 ! Mounting Angles (deg), Seq in Body
70 8.0 8.0 ! X, Y FOV Size, deg
71 30.0 10.0 10.0 ! Sun, Earth, Moon Exclusion Angles, deg
72 2.0 2.0 20.0 ! Noise Equivalent Angle, arcsec RMS
73 1 ! Flex Node Index

```

### Star Tracker Parameters

The Star Trackers verify if the spacecraft sensor is being blinded, reading the values set in the input file for the mounting angles, the field of view and exclusion angles. The spacecraft can be blinded by the Sun, the central body, and the Moon if the body is orbiting the Earth. If the body is blinded, no value is returned. Then, the model computes the spacecraft's attitude, expressed in quaternions, with the measured value to which some errors are added similarly to the other sensors.

```

74 ***** GPS *****
75 0 ! Number of GPS Receivers
76 ===== GPSR 0 =====
77 0.25 ! Sample Time,sec
78 4.0 ! Position Noise, m RMS
79 0.02 ! Velocity Noise, m/sec RMS
80 20.0E-9 ! Time Noise, sec RMS
81 0 ! Flex Node Index

```

### GPS Parameters

The Global Positioning System modelling takes the truth values measured for the position and velocity of the spacecraft, each at a certain time, and adds some uncertainty. The results are measured in both ECI and ECEF reference frames for Earth.

```

82 ***** Accelerometer *****
83 0 ! Number of Accel Axes
84 ===== Axis 0 =====
85 0.1 ! Sample Time,sec
86 0.5 1.0 1.5 ! Position in B[0] (m)
87 1.0 0.0 0.0 ! Axis expressed in Body Frame
88 1.0 ! Max Acceleration (m/s^2)
89 0.0 ! Scale Factor Error, ppm
90 0.05 ! Quantization, m/s^2
91 0.0 ! DV Random Walk (m/s/rt-hr)
92 0.0 1.0 ! Bias Stability (m/s^2) over timespan (hr)
93 0.0 ! DV Noise, m/s
94 0.5 ! Initial Bias (m/s^2)
95 0 ! Flex Node Index

```

#### Accelerometer Parameters

The accelerometer model obtains the acceleration of the point in the body of the spacecraft where the accelerometer is located in the direction of the axis expressed in body frame. This value is saved as true acceleration. After that, the model reads the uncertainty sources from the scale error factor to the initial bias and computes a result that would approximate to the real world. The maximum acceleration is verified so that it is never surpassed.

#### 4.4.5 Regions

The regions file controls the definitions of certain areas of body surfaces. The position can either be defined with a world-centric inertial frame, or with a longitude-latitude-altitude map. Regions are useful for orbital flight and surface movements.

```

1 ***** Regions for 42 *****
2 2 ! Number of Regions
3 -----
4 TRUE ! Exists
5 "TAG" ! Name
6 MINORBODY_2 ! World
7 POSW ! POSW or LLA
8 2400.9966 -1074.41895 439.1271 ! Position in W, m
9 -24.120375966731 9.48979662 27.7155 ! Lng, Lat (deg), Alt (m)
10 1.0E6 1.0E4 0.1 ! Elasticity, Damping, Friction Coef
11 Rgn_TAG.obj ! Geometry File Name

```

#### Region definition





#### 4.4.7 Inter-Process Communications

The Inter-Process Communications files can be used to set up data transfer between nodes using sockets. In addition, 42 comes with two inbuilt models that can be used to create more advanced connections.

#### 4.4.7.1 Standalone AcApp

It is most convenient for early studies and algorithm development to call the controller directly as part of the 42 simulation. However, as the controller matures into Flight Software, or if the controller comes from an external source, it is more convenient to host the controller in a standalone app, communicating with 42 over a socket interface. The Standalone simulation case is an example that demonstrates both of these architectures [50].

[illegible]

## IPC Standalone

All IPC configuration files start with the number of sockets that will be initialised. The IPC mode sets the purpose of the communication. At the same time, only Standalone communication is available with the OFF mode. By changing this value to ACS, the attitude and control are enabled externally.

The socket role allocates the type of action that the socket will perform either as a server or client. NASA's Goddard Mission Services Evolution Center (GMSEC) is out of the scope of this project. The server hostname refers to the direction of communication through a specific port.

In order to test the Standalone example, the following steps have to be taken [50]:

1. In Standalone/Inp\_IPC.txt, make sure that the "IPC Mode" for both IPC channels is ACS.
2. In the makefile, comment (STANDALONEFLAG =) by adding # at the beginning.
3. In the makefile, uncomment (#STANDALONEFLAG = -D \_AC\_STANDALONE\_).
4. On the terminal, write `make clean` and then `make` to update 42.
5. On the terminal, write `make AcApp` to enable attitude and control external system.

Now, everything is set up and three terminal windows are needed at the directory where 42 and the AcApp have been compiled.

1. In the first window, run "42 Standalone". Wait until "Server is listening..." is shown.
2. In the second window, run "AcApp 0". Look back in the first window for another "Server is listening..." message.
3. In the third window, run "AcApp 1".

This simulation accounts for two orbiting spacecraft. For both, a socket is created to control the spacecraft with an AcApp. This method will be further explained on 5.5.

The results are sent in real-time and can be visualised on the terminals.

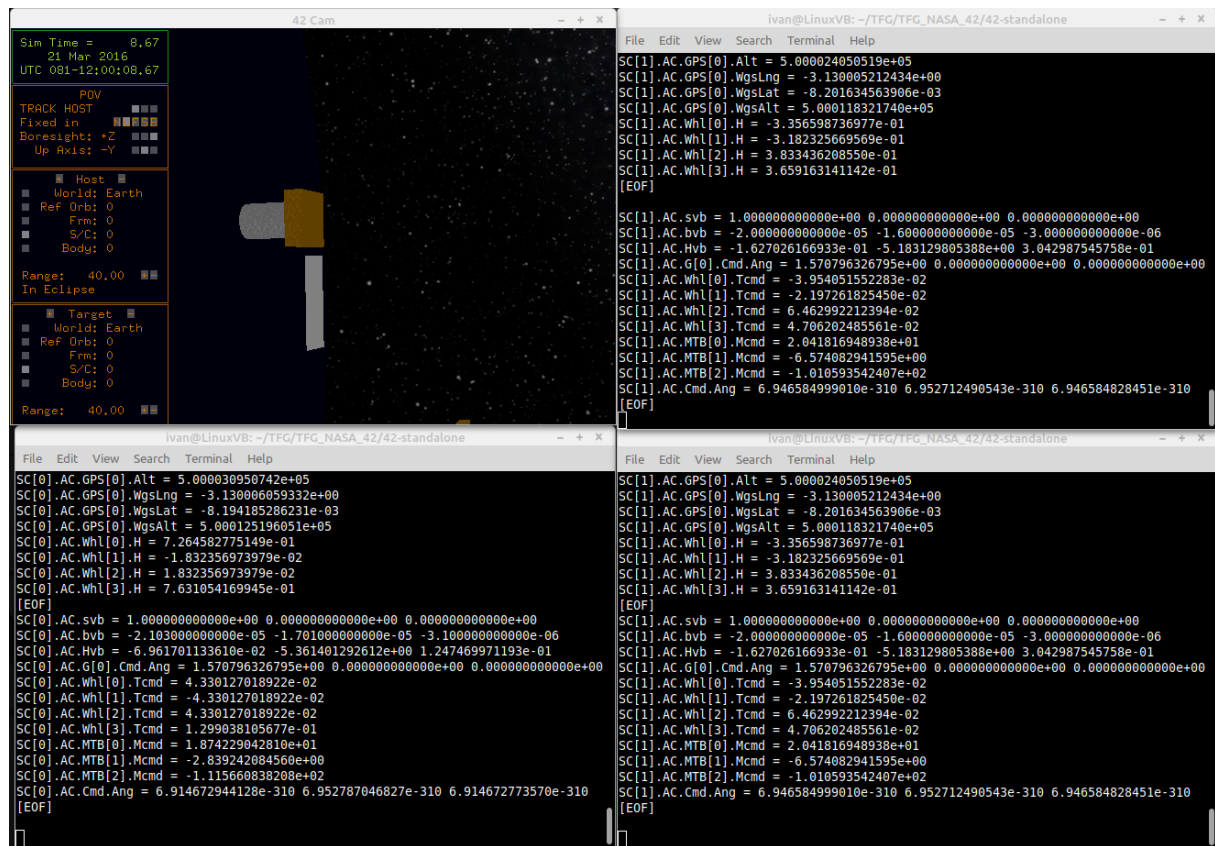


Figure 4.7 Standalone IPC Simulation

The top-right terminal runs as the first window, the bottom-left as the second and the bottom-right as the third.



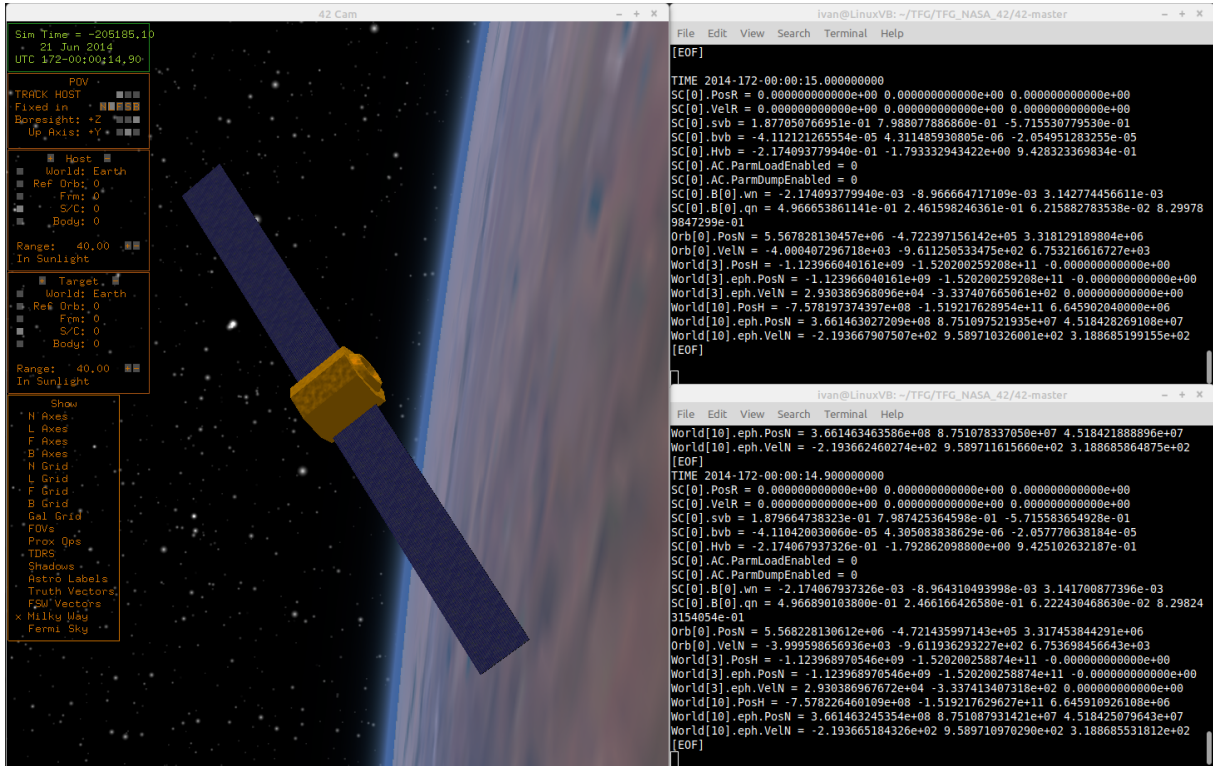


Figure 4.8 Tx-Rx IPC Simulation

On 5.5 the programme will be set to run on different machines.

#### 4.4.8 Graphics Configuration

The graphics configuration file controls the windows that will be opened when running the programme and the initial graphical properties. The modification of this file is outside of the scope of the project.

#### 4.4.9 Field of View

The FOV file describes the camera position during the simulation. Four predefined positions can be modified while new positions can be added. The modification of this file is outside of the scope of the project.

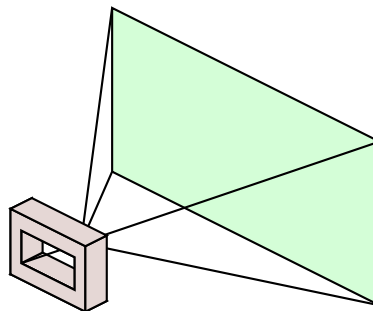


Figure 4.9 FOV representation

## 4.5 Output files

Once the simulation has reached an end, 42 saves certain variables in text files with the .42 extension for further analysis. The raw programme only outputs certain data for all spacecraft and all the available data about the first one. In order to obtain all data for every spacecraft, some changes are introduced in Section 5.1.

For the entire simulation, two time variables are obtained:

- **time.42** provides the local simulation time since the initial date and time in seconds.
- **DynTime.42** provides the simulation time with respect to the standard epoch J2000 in seconds.

For all spacecraft, where 00 indicates the body number, the following files are obtained:

- **Tree00.42** indicates the tree structure of the spacecraft. This structure contains the relations between bodies and joints.
- **u00.42** provides the dynamic states of the body. That is the angular velocity and the velocity relative to the spacecraft's centre of mass, both with respect to the inertial frame.
- **x00.42** provides the kinematic states of the body. That is the quaternion and the position relative to the spacecraft's centre of mass, both with respect to the inertial frame.

For the first spacecraft, the aforementioned files are accompanied by:

- **PosN00.42** provides the 3D position with respect to the inertial frame, for Earth it corresponds to ECI frame.
- **VelN00.42** provides the 3D velocity with respect to the inertial frame, for Earth it corresponds to ECI frame.
- **PosW00.42** provides the 3D position with respect to the rotating frame, for Earth it corresponds to ECEF frame.
- **VelW00.42** provides the 3D velocity with respect to the rotating frame, for Earth it corresponds to ECEF frame.
- **PosR00.42** provides the 3D position with respect to the reference frame. For two-body orbits, R moves on Keplerian orbit, while for three-body orbits, R propagates under the influence of both attracting centres seen as point masses. The spacecraft orbit perturbations are integrated with respect to R.
- **VelR00.42** provides the 3D velocity with respect to the reference frame.
- **qbn00.42** provides the quaternion of the body with respect to the inertial frame.
- **wbn00.42** provides the angular velocity of the body with respect to the inertial frame.
- **Hvn00.42** provides the angular momentum of the spacecraft with respect to the inertial frame.
- **Hwhl00.42** provides the angular momentum of all wheels.
- **svn00.42** provides the sun-pointing vector of the body with respect to the inertial frame.
- **svb00.42** provides the sun-pointing vector of the body with respect to the body frame.
- **KE00.42** provides the kinetic energy of the body.
- **RPY00.42** provides the spacecraft Euler angles in the LHLV frame, which correspond to the Roll, Pitch and Yaw angles of the body.
- **MTB00.42** provides information about the magnetorquers.
- **Acc00.42** provides the true and mean acceleration for all accelerometers of a certain body.

All output files are saved with the .42 extension which can be accessed with any file reader. For all files the 00 keeps changing in accordance to the spacecraft number.

## Chapter 5

# 42: PLATHON project

Once the 42 programme has been analysed, it is time to develop programme variations and examples to demonstrate the capabilities towards the PLATHON project. This chapter gathers all the programme simulation missions and the obtained results.

The main simulations are:

1. **Report modifications:** programme variation to save all data outputs for every spacecraft.
2. **Feasibility of MATLAB GUI:** creation of a MATLAB representation tool and analysis within the PLATHON project
3. **42 file creation and testing:** development of Matlab code creator and reader with some examples.
4. **Constellations:** development of a constellation reader based on the TLE format.
5. **InterProcess Communications:** basis of the PLATHON communication with the future SiTL and HiTL capabilities.
6. **Satellite variations:** analysis of medium-term propagation errors with TLE and databases.
7. **Comparison with SPICE data:** analysis of medium-term propagation errors with NASA SPICE data.

At the end of the chapter a brief combination with previous models is suggested.

All the codes developed during this thesis can be found on [Ivan-Sermanoukian/42PLATHON GitHub repository \[8\]](#).

## 5.1 Report modifications

The original Report file only saves the dynamic and kinematic states of all spacecraft. For all the remaining properties, they are only saved for the first body. As a result, the programme has to be able to save all the analysed data of all the spacecraft.

For this purpose, the `42report.c` file inside the Source folder has to be modified. The following pseudocode explains the Report function, which can be found on Software Appendix [B.3.1].

---

### Algorithm 1 Report modifications

---

```

1: FILE definitions
2: Memory allocation for each FILE
3: for Every Spacecraft do
4:     if Spacecraft exists then
5:         for Every variable do
6:             Create a FILE with write permission
7:         end for
8:     end if
9: end for

10: if Output writing is enabled then
11:     Save Time data
12:     for Every Spacecraft do
13:         if Spacecraft exists then
14:             for Every variable do
15:                 Save data
16:             end for
17:         end if
18:     end for
19: end if

```

---

With the current code, up to a 1000 satellites' data can be stored in text documents. If more were needed in the near future, it would only be necessary to change the memory allocation loop. For the following example, the format specifier `%03ld` would have to be changed accordingly.

```

sprintf(s, "PosN%03ld.42", Isc);
PosNfile[Isc] = FileOpen(InOutPath, s, "w");

```

File creation for new variables

As a vast number of text documents are going to be opened for constellations, it is common to encounter an error while executing, warning the user that "*Too many open files*" are needed. This problem is due to the OS limit of opening simultaneous files.

For knowing the current value, it can be typed on the terminal:

```
ulimit -n
```

Then, for changing that value the new maximum number has to be appended. For this example, 20000 is set:

```
ulimit -n 20000
```

As a result, two 42 variants will be used depending on whether the User wants to save all the simulated data for post-process analysis or for real-time communication purposes.

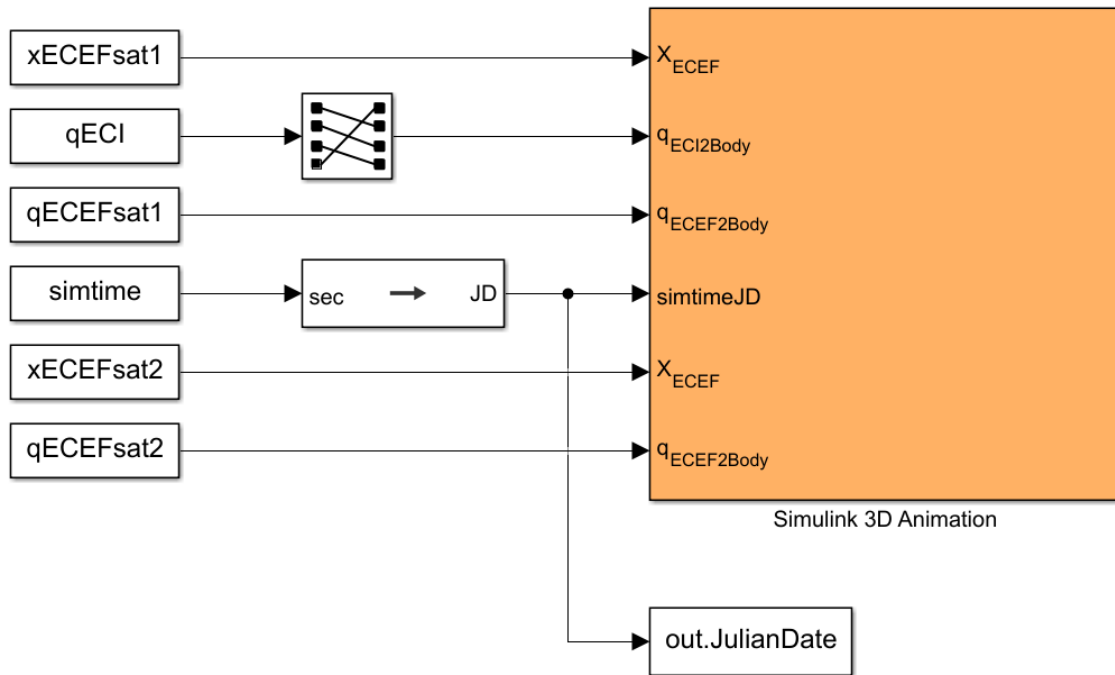


## 5.2 MATLAB GUI

Following the simulation models commented in the programme's introduction, an attempt to simulate the data from 42 to Matlab with Simulink has been made. The Aerospace Toolbox has a template to use Simulink 3D Animation that can be used for satellites and celestial bodies. Although the template satellite body is not the same as the one conceived by the PLATHON project, no error is added as it is only used as a graphical representation.

The template has been modified to represent the input of two spacecraft. All the required data comes from 42 output files except the quaternions in the ECEF frame, as 42 only saves them in the ECI frame. However, the input order of the quaternions is different as the scalar component corresponds to the first index of the array in Matlab, whereas it is assigned to the last digit by 42. To obtain the ECEF quaternion, a conversion is required from the ECI quaternion.

The Simulink 3D Animation is obtained from the following block:



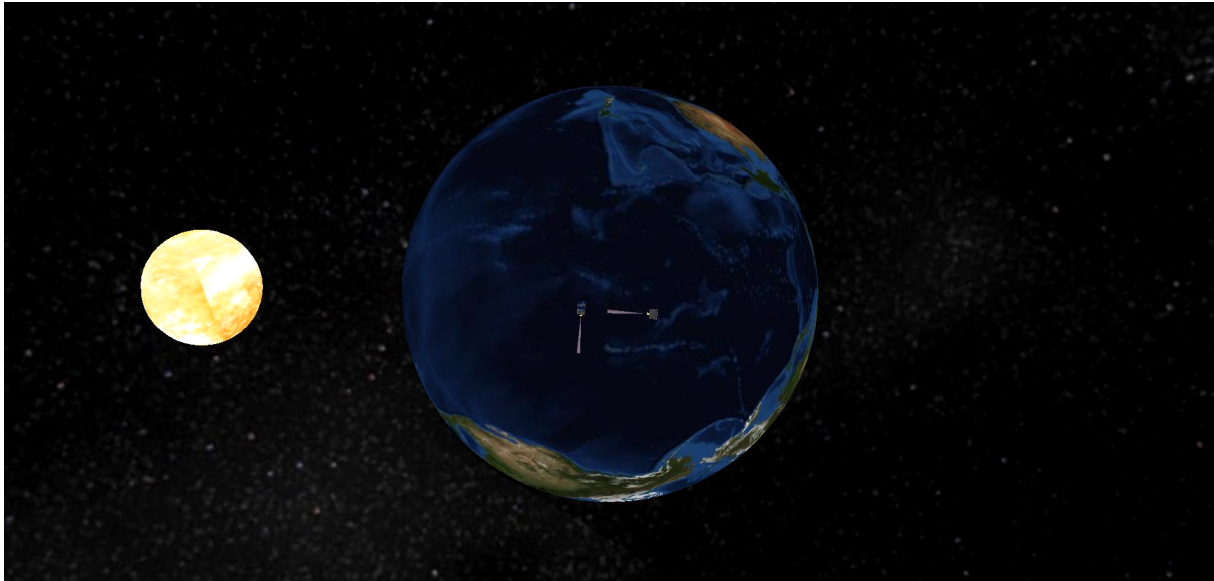
**Figure 5.1** Simulink 3D Animation

The inputs to the Simulink 3D Animation are:

- Position vector of the first satellite in ECEF frame
- Quaternion of the first satellite in ECI frame
- Quaternion of the first satellite in ECEF frame
- Julian date
- Position vector of the second satellite in ECEF frame
- Quaternion of the second satellite in ECEF frame

The Julian date is saved in case it is needed for post-process analysis.

The results obtained in the simulation can be saved into a video from which this frame has been taken:



**Figure 5.2** Two satellite 3D Animation

### 5.2.1 Use of MATLAB GUI

The implementation of the MATLAB GUI was encouraged by the slow simulation times on computers without graphical power and the need to visualising the results without having to rerun the programme. However, the idea has been discarded for the hardware-in-the-loop emulation due to the need for a real-time representation which Simulink cannot provide. Moreover, the Simulink 3D Animation was not designed for expandable numbers of bodies, constraining the visualisation of constellations to a low number of spacecraft.

Then, as the simulators were running on virtual machines with integrated graphic cards, it was decided to set up another computer with enough graphic power. At the same time, tests were performed on a desktop PC with an Nvidia GeForce GTX 1060 and the GUI did not slow down. Therefore, 42 GUI has been deemed the best option.

## 5.3 42 file creation and testing

42 input file system has been designed for a reduced number of bodies. However, in order to simulate a constellation, each spacecraft requires its input files to be described. For this purpose, a Matlab code has been developed to create all the necessary files to run the simulation. On Linux, running the code completes opening the 42 simulation folder and executing the necessary opening commands.

The following algorithm shows the structure that the main Matlab file creator function will have. Each mission will customise this template to suit the needs of the analysis.

---

**Algorithm 2** Input\_Programme.m

---

- 1: Choose simulation folder
  - 2: Select Windows or Linux path
  - 3: Input initial data
  - 4: Input orbit function
  - 5: Input spacecraft function
  - 6: Input ground stations
  - 7: Select Solar System variables
  - 8: Input command script
  - 9: **if** GUI activated **then**
  - 10:     Input graphics function
  - 11:     Input FOV function
  - 12: **end if**
  - 13: Input Tracking and Data Relay Satellite System function
  - 14: Input Region function
  - 15: Input InterProcess Communication function
  - 16: Input Simulation function
  - 17: **if** Linux OS **then**
  - 18:     Select simulation folder
  - 19:     Run 42
  - 20: **end if**
- 

The first step is to designate the name of the folder where the 42 input files will be created:

```
% Choose simulation folder
folder = 'Example_folder';
```

Then, the path where the Input functions are looked for is set inside the OS data path. In order to preserve the changes of the input functions for each analysis, it is recommended to create a brand-new folder. Windows OS will not be able to execute the 42 programme automatically.

```
% Add function paths
addpath(strcat(pwd,filesep,'Input_functions'));
```

The Input data section gathers the most used properties for rapid prototyping. Other variables can be directly modified on the Input files.

```
%% Input data

% Input Command Script

    % User manual
    User_manual = 'TRUE';           % T/F

% Input Simulation Control

    % Time variables
    time_mode     = 'FAST';          % FAST, REAL, EXTERNAL, or NOS3
    duration       = '10000';        % s
    step_size      = '0.1';          % s
    output_interval = '1';            % s

    % Initial time
    month          = '06';           % mm
    day            = '14';           % dd
    year           = '2021';         % YYYY
    hour           = '10';           % hh
    minute         = '00';           % mm
    second         = '00.00';        % ss.ss

    % Graphical User Interface
    GUI            = 'TRUE';         % T/F

    % Orbits
    reference_orbits = '75';
    [orbits] = Input_Orbit(folder,reference_orbits);

    % Spacecraft
    number_spacecraft = '75';
    [spacecraft] = Input_SC(folder,number_spacecraft);

    % Ground Stations
    number_ground_stations = '1';
    % Exists, World, Lng, Lat, Label
    ground_stations = ['TRUE','EARTH','-77.0','37.0','"GSFC"'];

    % Select Solar System variables
    % Me - V - E - Ma - J - S - U - N - P - Ast
    Solar_System = ['TRUE','TRUE','TRUE','TRUE','TRUE','TRUE','TRUE','TRUE','TRUE','TRUE'];
    % Earth-Moon / Sun-Earth / Sun-Jupiter
    Lagrange_System = ['FALSE','FALSE','FALSE'];
```

The User manual boolean variable defines if the command list from the Command Script is written commented at the end of the file. The time variables define the simulation temporal variables, while the initial time sets the simulation starting point.

The Graphical User Interface can be activated with the GUI variable. Then, the reference orbits and the number of spacecraft determine the number of each used in the simulation. The values have to agree with the data inside the functions.

Next, the ground stations are defined with the Solar System planets and the activation of the Lagrange Systems.

Then the remaining input functions are called to create all the necessary files. If the GUI is deactivated, the Graphics and FOV functions are not saved.

The complete file can be found for the following example about constellations [B.3.2].

Once the simulation has finished, and the .42 files are ready, it is time to post-process the data. Depending on the 42 version used, all or just the first spacecraft data will be saved entirely.

The following algorithm represents the structure of the code:

---

**Algorithm 3** Input\_Programme.m

---

- 1: Choose data folder
  - 2: Choose simulation folder
  - 3: Select number of spacecraft
  - 4: Read time variables
  - 5: **for** The first spacecraft to the last **do**
  - 6:     Read spacecraft variables
  - 7:     Save variables on satellite matrix
  - 8: **end for**
- 

It is important to outline that the reading process is performed in ASCII format. A complete file can be found with the Constellation analysis [\[B.3.3\]](#).

## 5.4 Constellation

The variables can be customised to create a unique constellation or read the latest data of a real one using TLE. The most up-to-day TLE are downloaded from the Celestrak web page when the programme is executed. As 42 is compatible with TLE data formats, the best method to simulate the constellation is to extract the label of each spacecraft instead of directly adding the Keplerian elements due to the fact that each satellite position has been obtained at a different timestamp. As a result, 42 propagates all satellites to the initial date from which the simulation is recorded. It is relevant to consider that if the satellite time stamp and the initial date are more than a day off, a considerable error might occur.

Acknowledge that for a more precise simulation, the Iridium spacecraft models should be created. This simulation has been carried out with 1U CubeSat.

The following algorithm explains the TLE download method:

---

**Algorithm 4** TLE\_to\_42.m

---

- 1: Select TLE data link
  - 2: Read data from TLE
  - 3: Save data into a matrix by lines
  - 4: Create TLE document file for 42
  - 5: Write TLE data into document
  - 6: Extract satellite ID (delete blank spaces)
- 

In order to assign the name of the Spacecraft from the TLE data to each body, the Input Spacecraft function has been changed accordingly. Option 1 takes into account the name variations, while Option 2 gives the same name to all spacecraft.

---

**Algorithm 5** Input\_SC.m

---

- 1: **if** Option 1 **then**
  - 2:     Get satellite IDs
  - 3:     Select spacecraft template
  - 4:     Read template data and save into a matrix
  - 5:     **for** Every active spacecraft **do**
  - 6:         Change satellite label
  - 7:         Create spacecraft text document
  - 8:         Add values to spacecraft matrix
  - 9:     **end for**
  - 10: **end if**
  - 11: **if** Option 2 **then**
  - 12:     **for** Every active spacecraft **do**
  - 13:         Assign satellite file
  - 14:         Add values to spacecraft matrix
  - 15:     **end for**
  - 16: **end if**
- 

The next section presents the results using the IRIDIUM NEXT constellation TLE. However, any constellation with available TLE can be represented.



### 5.4.1 IRIDIUM NEXT constellation

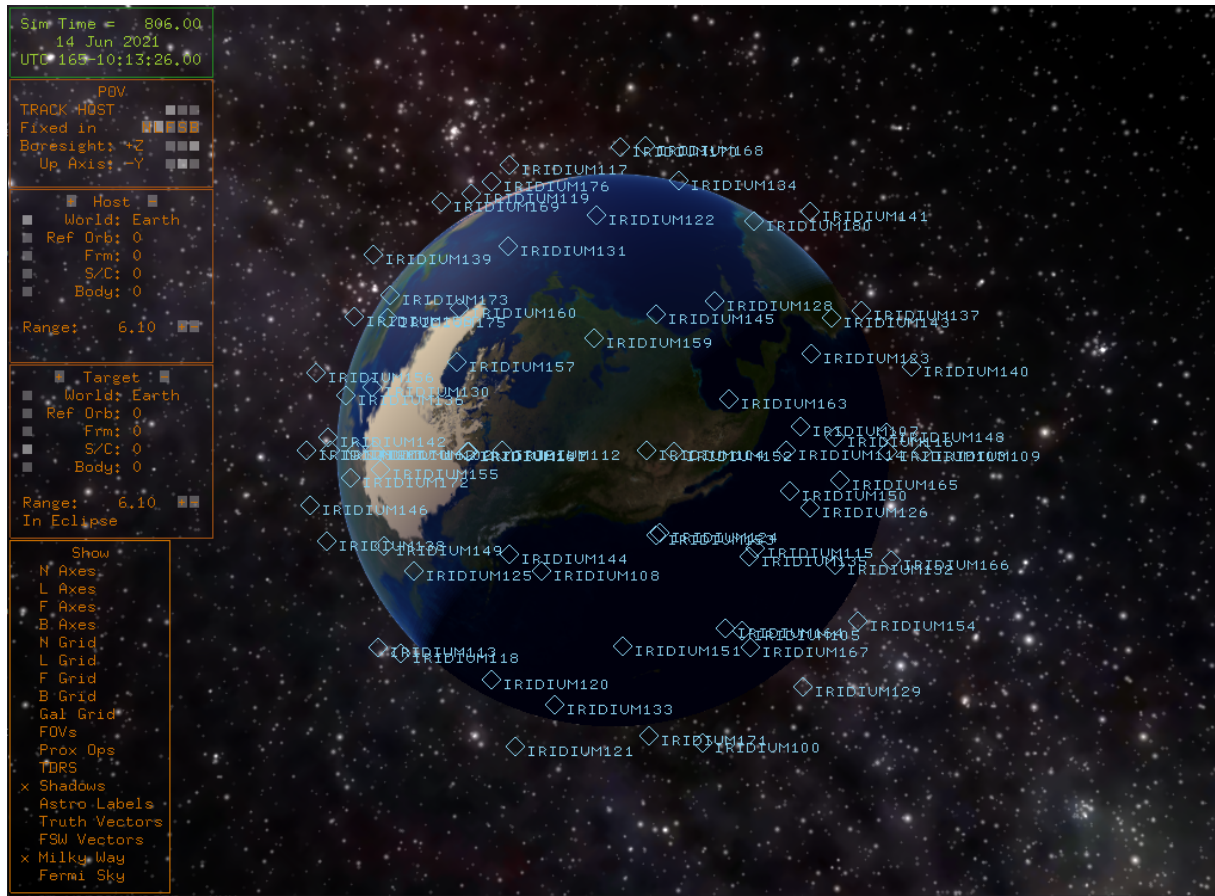


Figure 5.3 Iridium NEXT simulation (2021/06/14). 3D Camera view.

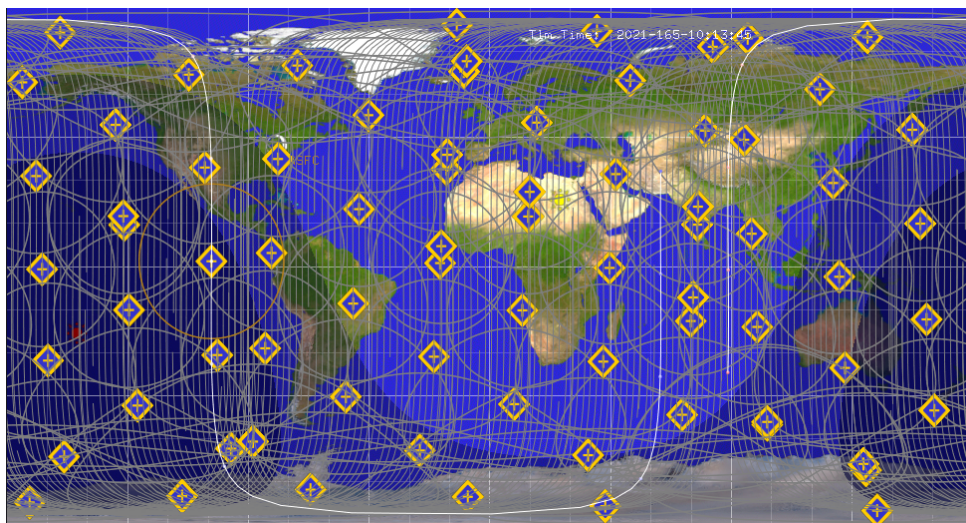
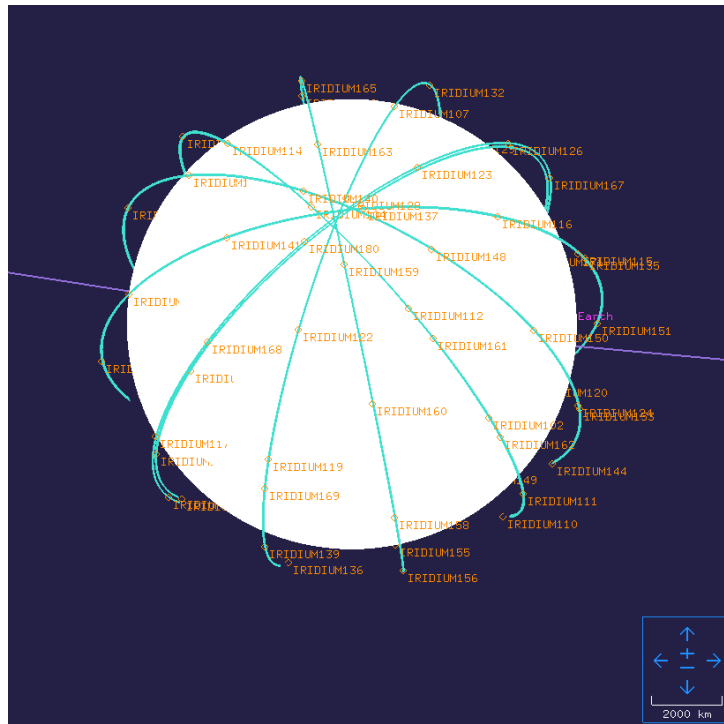


Figure 5.4 Iridium NEXT simulation (2021/06/14). 2D Map view.

The Iridium NEXT orbits are shown on the 42 Orrery.



**Figure 5.5** Iridium NEXT simulation (2021/06/14). Orrery view.

One problem when loading many spacecraft with different observation TLE time is that 42 requires to either propagate forward or backwards to the initial time, inducing an error that increases exponentially with time. This will be further analysed on the Satellite propagation section 5.6.



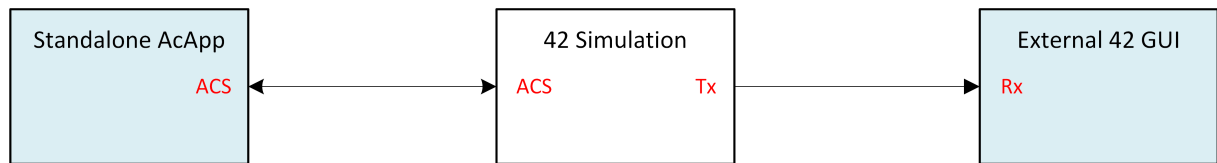
## 5.5 InterProcess Communication

The PLATHON project aims to create a hardware-in-the-loop simulation where the 42 simulator will only be responsible for providing the orbit simulation variables. All other tasks will be delegated to either external software for ACS and mission control, or hardware sensors and actuators connected with the inboard CubeSat computer.

### 5.5.1 Architecture I

The first system architecture is composed of the following:

- A main 42 simulator that interacts with all the dependencies.
- An external Standalone AcApp acting as an Attitude Control System.
- An external 42 simulator that receives data and acts as a GUI.



**Figure 5.6** 42 system architecture I

As it can be seen in the previous image, there is a bidirectional communication between the Standalone AcApp and the 42 Simulation whereas the External 42 GUI only receives data.

Each block can be located at a different PC or combining several features on one computer. For the PLATHON project, the 42 Simulation and the Standalone AcApp are based on the same server while the graphical representation is in an external machine. However, everything can be controlled remotely from a unique machine using ssh for convenience.

The IPC files for both machines and the code modifications of AcApp can be found on the IPC Architecture I appendix [B.3.5].

A significant remark for this simulation is that the receiver requires to have the same simulation files as the main simulator in order to allocate memory correctly.

The current AcApp modification allow to change the Euler angles at a certain time to simulate that one CubeSat rotates to point to another CubeSat and sends data through optical communications. The time calculation is performed based on the difference between the current time and the initial time as the AcApp does not receive simulation variables. This will be further enhanced by either reading terminal commands or information sent by the hardware simulation.

This analysis represents the first step towards the future hardware-in-the-loop simulation. Socket communication protocols would allow 42 to introduce inputs from real sensors and actuators directly through the send-receive mode, overwriting the simulated data through ACS messages.

With the command `netstat -antp`, the port status can be analysed.

In order to recreate the simulation, the following steps are to be taken:

1. In the first window (top-right), run the folder with the IPC Tx and ACS.  
For instance `./42 StandaloneRemoteViewer`.
2. In the second window (bottom-left), run the folder with the Rx.  
For instance `./42 RxPlathon102`.
3. In the third window (bottom-right), run the modified AcApp with the `./AcApp 0` command.<sup>1</sup>

The second action will initiate 42 GUI but the programme will wait until receives data from the AcApp.

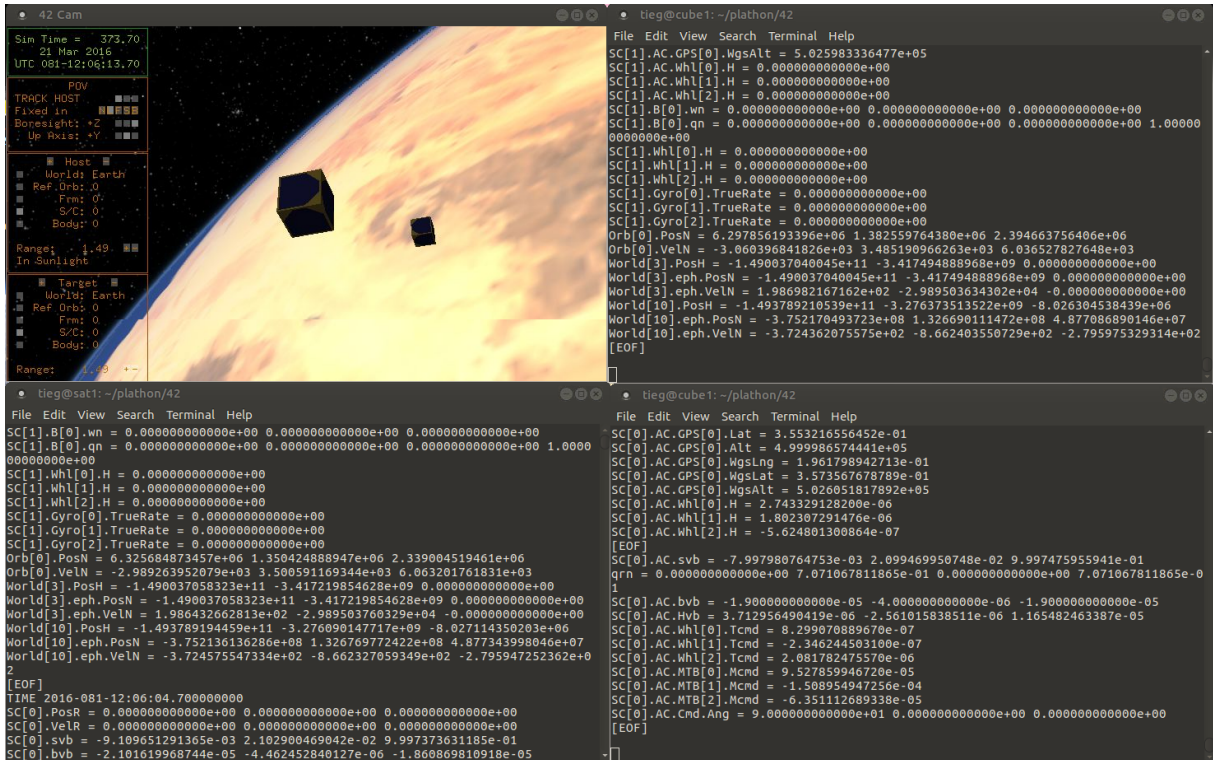


Figure 5.7 IPC Architecture I - Terminal distribution

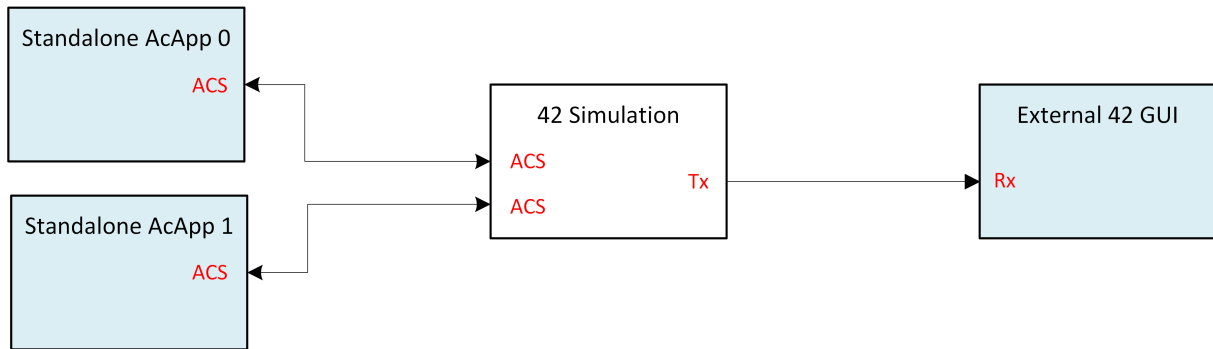
<sup>1</sup>If it is the first time, remember to modify the AcApp file, clean the compiled files and compile the AcApp.

### 5.5.2 Architecture II

For this architecture, the second spacecraft is also controlled by a second AcApp in order to transmit different commands to each satellite. This change translates into several modifications which are explained hereunder.

The second system architecture is composed of the following:

- A main 42 simulator that interacts with all the dependencies.
- An external Standalone AcApp 0 acting as an Attitude Control System for the first CubeSat.
- An external Standalone AcApp 1 acting as an Attitude Control System for the second CubeSat.
- An external 42 simulator that receives data and acts as a GUI..



**Figure 5.8** 42 system architecture II

The insertion of a second AcApp controller requires to modify the project makefile to compile both files independently. The AcApp file structure has not been changed from the previous architecture but has been duplicated with different function commands to simulate that both spacecraft point each other for communications.

This simulation serves as a demonstration of the IPC capabilities and as a proof of concept for the future PLATHON project architecture. Thus, it paves the way to more elaborate simulation to recreate a full-on constellation when assembled with the cFS and the node connectivity features currently under development.

The IPC files for both machines and the project makefile can be found on the IPC Architecture II appendix [B.3.6]. The complete files can be found on GitHub [8].

In order to recreate the simulation, the following steps are to be taken:

1. In the first window (top-left), run the folder with the IPC Tx and 2 ACS.  
For instance ./42 StandaloneRemoteViewer.
2. In the second window (top-right), run the folder with the Rx.  
For instance ./42 RxPlathon102.
3. In the third window (bottom-left), run the first modified AcApp with the ./AcApp 0 command.
4. In the fourth window (bottom-right), run the second modified AcApp with the ./AcApp 1 command.

The second action will initiate 42 GUI but the programme will wait until receives data from both AcApps.

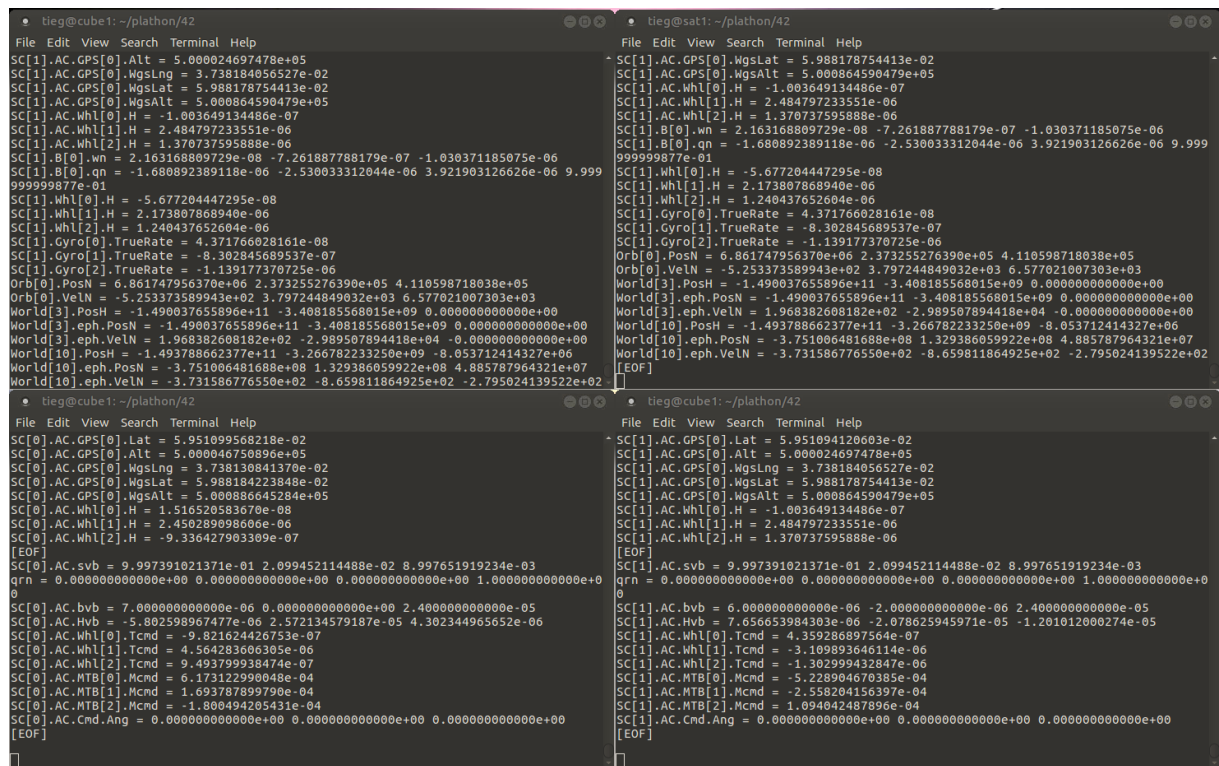
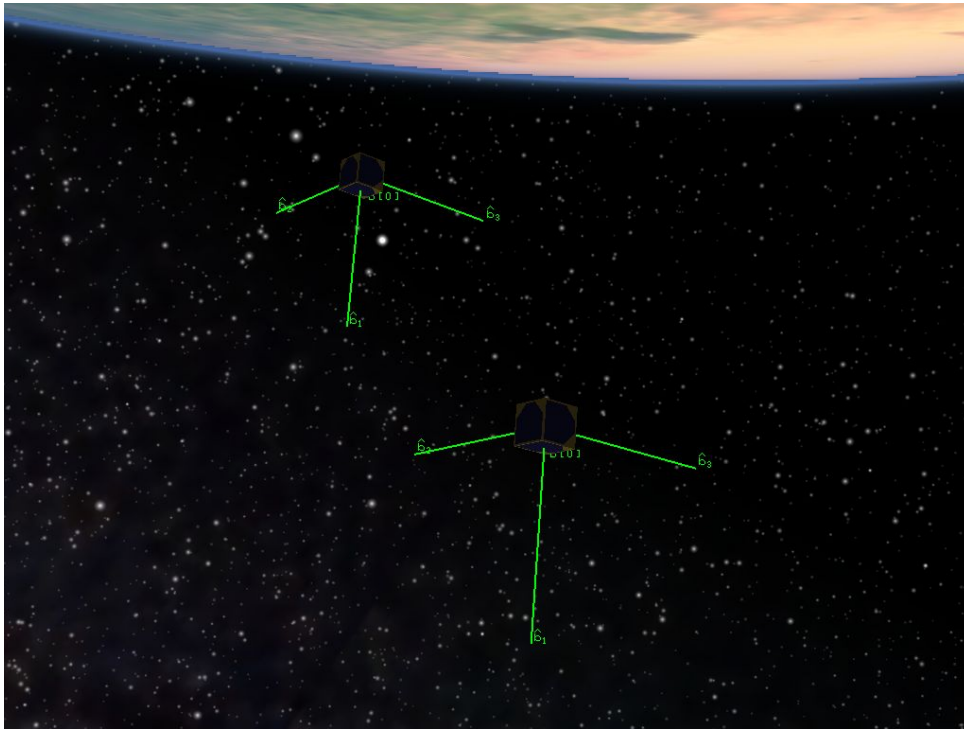


Figure 5.9 IPC Architecture II - Terminal distribution

The following images depict the attitude modifications:



**Figure 5.10** IPC Architecture II - Initial attitude

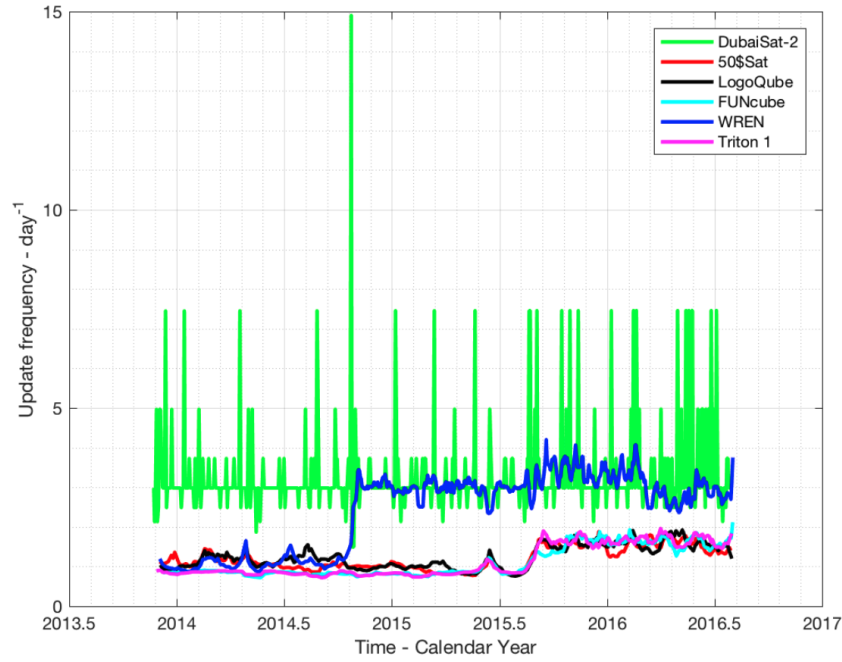


**Figure 5.11** IPC Architecture II - Modified attitude

## 5.6 Satellite propagation

To test the propagation accuracy in time, a CubeSat and the ISS have been taken as TLE samples of the study. According to TUDelft's 11th, IAA Symposium presentation [52], TLE are very practical but tend to have low accuracy with more than 1 km error. 2-way ranging methods have demonstrated that the error can be reduced on CubeSat measurements to approximately half a kilometre.

All objects get between one and three TLE per date. This update rate, although enough for most cases, can result in vast error propagation for semi-analytic and basic numerical propagation methods.



**Figure 5.12** TLE update rate of CubeSats and PocketQubes. Extracted from [52].

The aforementioned satellites have the following properties:

Satellite	Size	Type
DubaiSat2	1.5x1.5x1.95 m	-
50\$Sat	5x5x7.5 cm	1.5P PocketQube
LogoQube	5x5x12.5 cm	2.5P PocketQube
FUNcube	10x10x11.4 cm	1U CubeSat
WREN	5x5x5 cm	1P PocketQube
Triton1	10x10x34.2 cm	3U CubeSat

**Table 5.1** Satellite information

In order to test 42 propagation capabilities, no CubeSat from the database had more than 3 TLE per day. Other more massive satellites which receive more updates per day ought to be modelled and inserted into 42 to analyse the TLE propagation with shorter periods. Following, medium to long time periods have been studied.



### 5.6.1 CubeSat

From the list of available CubeSat TLE [6], SOMP (Students' Oxygen Measurement Project) CubeSat from the Dresden University of Technology has been chosen due to the available data.



**Figure 5.13** SOMP CubeSat. Extracted from [53]

For this analysis, two consecutive TLE with almost 11h of difference have been used. In order to extract the TLE of the same spacecraft during a specific period of time, the Space-Track website [7] is the best choice.

```
SOMP
1 39134U 13015E 21170.87477442 .00001682 00000-0 80789-4 0 9997
2 39134 64.8620 164.5918 0008865 313.4966 46.5425 15.22619016451583
```

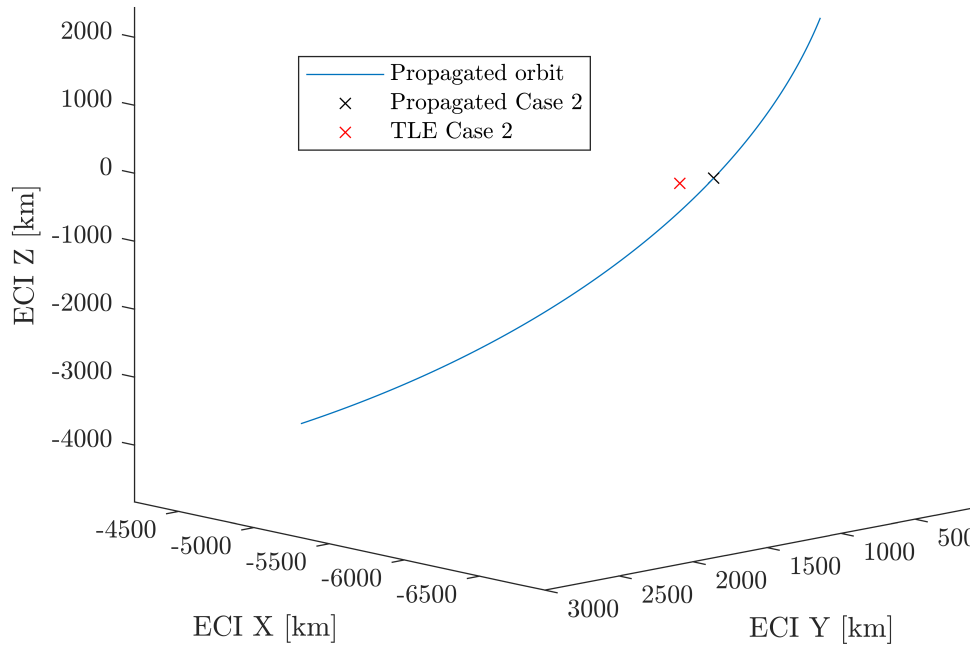
**Case 1 - SOMP TLE (19 June 20:59:40.51)**

```
SOMP
1 39134U 13015E 21171.33454014 .00001526 00000-0 74100-4 0 9996
2 39134 64.8620 163.0966 0008912 313.8408 46.1984 15.22620409452033
```

**Case 2 - SOMP TLE (20 June 8:01:44.27)**

The following simulations do not consider external perturbations to perceive the relevance they have for long period simulations. The correct adjustment of the perturbation for each mission will be crucial to preserve propagation accuracy.

The first study propagates the CubeSat from case 1 until the simulation time is equal to the case 2.



**Figure 5.14** Forward propagation

As it can be seen, more than 11 hours translates into an enormous position difference of 189.577 km.

The second study starts by propagating the CubeSat from case 2 to case 1 through backwards propagation. Then, starting at the time of case 1 with the new position, the CubeSat is propagated until the simulation time is equal to case 2.

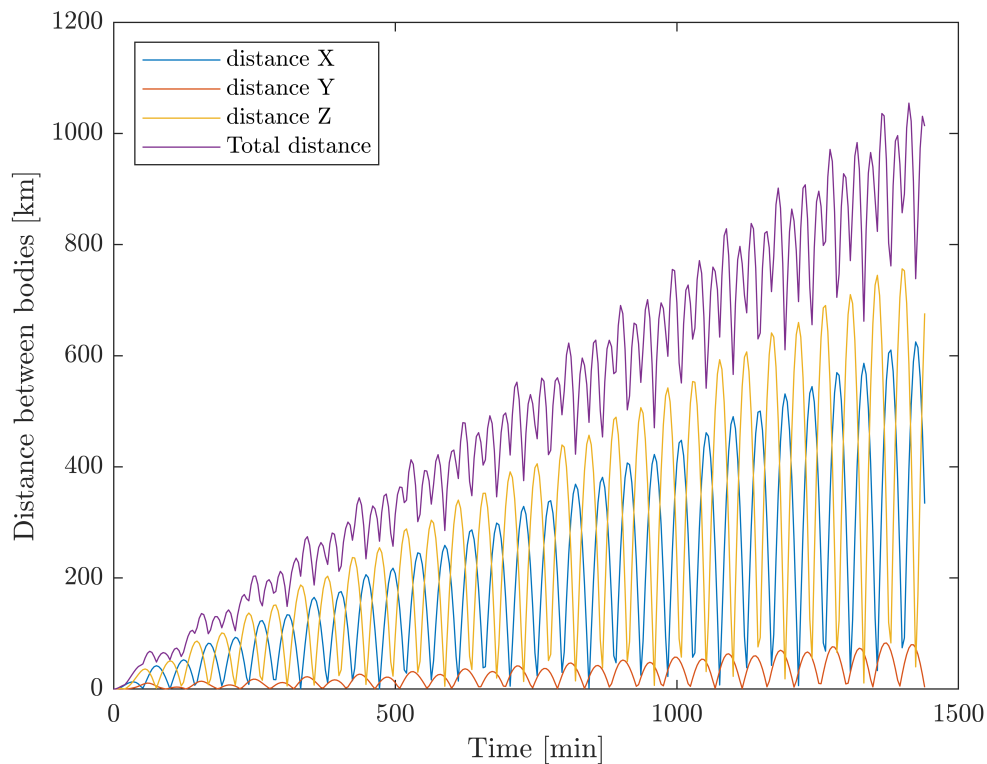
The result from the backwards propagation yields a similar error with a value of 188.808 km. Then, when forward propagation reaches the second case, the difference is reduced to 1.827 km. As it can be seen, there is an accumulated error as the last value ought to be null.



## 5.6.2 International Space Station

For larger bodies such as the ISS, the TLE can also be found on Celestrak. However, this time the propagation will be compared with updated ISS trajectory information made available as a CCSDS Orbital Ephemeris Message (OEM) in the J2000 reference system.

This format takes into account ISS manoeuvre details, predicted post-manoeuve state vectors, predicted coasting state vectors at 4-minute intervals, and ISS ballistic details [54]. The values for the comparison are obtained from the NASA database [55].



**Figure 5.15** Forward propagation

Compared with the previous case, the error is not that large for a time interval of a complete day, more than double the time analysed beforehand. This simulation has also been executed without external perturbations.

## 5.7 ISS comparison with SPICE

Spacecraft Planet Instrument C-matrix Events (SPICE) is described as an ancillary information system that provides the capability to include space geometry and event data into mission design, science observation planning, and science data analysis software.

Although it is primordiallly used for interplanetary missions, there are some data kernels for Earth-orbiting spacecraft such as the ISS based on early TLE measurements.

### 5.7.1 NAIF Integer ID codes

SPICE system kernels and routines refer to ephemeris objects, reference frames, and instruments by integer codes which are usually referred as ID [56].

Spacecraft ID codes are negative. The code assigned to interplanetary spacecraft is normally the negative of the code assigned to the same spacecraft by JPL's Deep Space Network (DSN) as determined the NASA control authority at Goddard Space Flight Center. Conversely, for Earth orbiting spacecraft that lack a DSN identification code, the NAIF ID is obtained from the spacecraft's NORAD ID available on TLEs via:

$$\text{NAIF ID} = -10000 - \text{NORAD ID code} \quad (5.7.1)$$

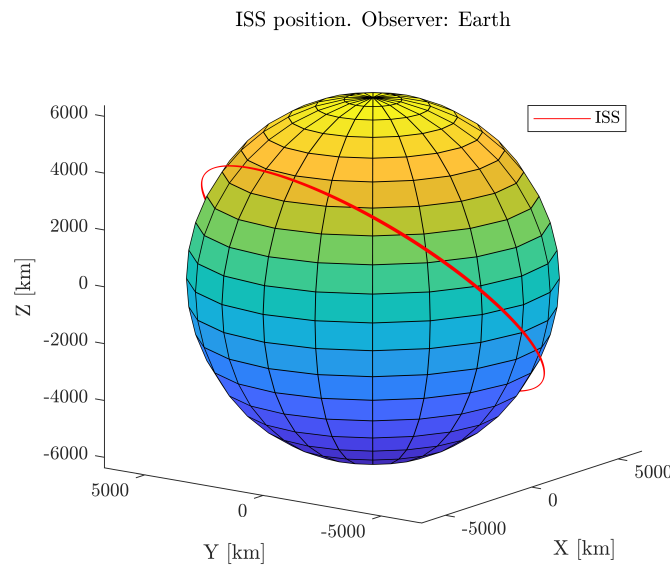
For example, ISS has an assigned NORAD ID code 25544. SPICE would use the NAIF ID -125544.

All the available ephemeris kernels for Earth orbiting missions can be found on the [MORE PROJECTS](#) folder.

For the ISS simulation, SPICE requires the leap seconds kernel file (.tls) in conjunction with the ephemeris data (.bsp). The leap seconds file should be the latest available which, by the time this thesis is being written, is [naif0012](#).

In order to produce type 10 SPK segments based upon the two-line element sets, the mkspk application needs the data in [geophysical.ker](#) kernel.

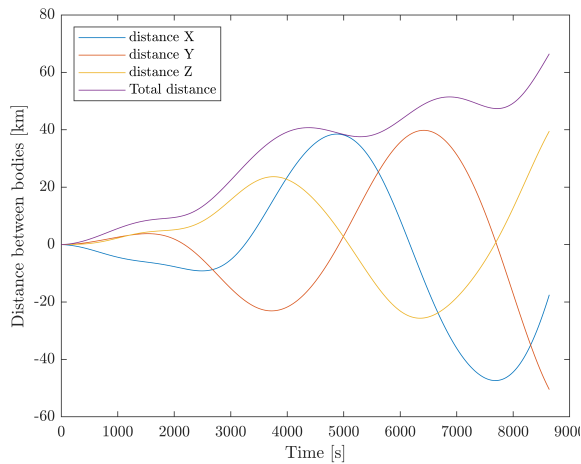
The ISS kernel contains the information from TLE data up to 2012.



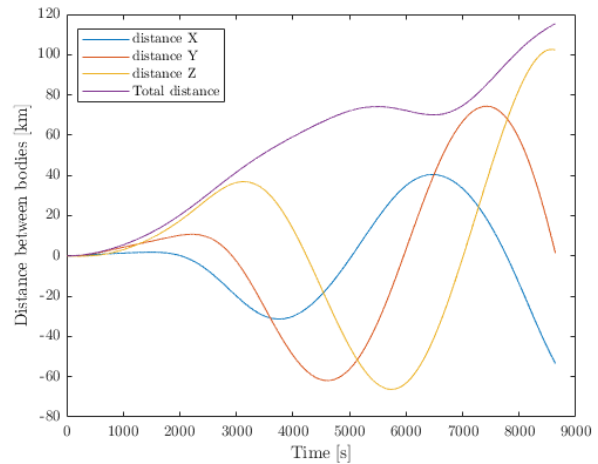
**Figure 5.16** ISS orbit Start: (2012-04-19 T00:00:00) NDAYS = 0.1;

To analyse the non-gravitational perturbation importance on massive bodies such as the ISS, two different dates have been chosen to be simulated with 42 and then compared to the available data with SPICE.

With the same initial conditions and the ISS spacecraft<sup>2</sup>, the following results are obtained:



**Figure 5.17** ISS orbit difference between SPICE and NASA 42. [2005/04/19]



**Figure 5.18** ISS orbit difference between SPICE and NASA 42. [2012/04/19]

As it can be seen for both cases, the accuracy changes drastically with a reduced time interval.

For this calculation the leap seconds have to be changed. The following table represents the insertion dates according to NAIF:

Leap Second Insertion Date	TAI-UTC	GPS-UTC Offset
December 31, 2016	37 seconds	18 seconds
June 30, 2015	36 seconds	17 seconds
June 30, 2012	35 seconds	16 seconds
December 31, 2008	34 seconds	15 seconds
December 31, 2005	33 seconds	14 seconds
December 31, 1998	32 seconds	13 seconds

**Table 5.2** Leap seconds according to date

SPICE capabilities could be used to extract textures from real images taken from planets and asteroids for a more realistic graphical representation.

<sup>2</sup>The ISS structure is considered to be at the same building phase.

## 5.8 Comparison with Matlab models

Once 42 propagator capabilities have been analysed, it is time to see how to merge the programme with previous TIEG MATLAB/Simulink models.

Compared to Hernández [17] and Parrot [18] thesis, 42 environment models entail a significant accuracy enhancement. Specifically, the gravity model, the Earth's magnetic field model and the disturbance torques take into account more data variables for each calculation. However, the energy balance of the CubeSat is not modelled by 42.

A significant property within the PLATHON HITL emulation is knowing whether or not a CubeSat is capable of communicating either with another CubeSat through inter-satellite optical communications, or through RF communications with Ground Stations. For this reason, the energy feasibility study should be added. Similarly to how the sun sensors works, a photo-voltaic module model ought to be added.

As the sun vector properties are already programmed, battery and battery charge models could be introduced. The battery status ought to be verified before any power-consuming action is performed. Then, the battery discharge model would have to be tailored for each sensor and actuator.

Another approach could take into consideration the energetic aspects directly within NOS3.

Furthermore, the 42 propagator actuator modelling could be enhanced with these studies and those being performed during the development of this thesis.

# Chapter 6

## Project Planning

### 6.1 Work Breakdown Structure

Below is presented a hierarchical decomposition of the project objectives where the following tasks will be crucial. The dependencies amongst tasks can be seen on Fig. 6.1.

#### 6.1.1 Task identification

This section includes the main tasks that have been performed in terms of documentation. Programme comprehension, testing and bureaucratic tasks are included in the Gantt diagram 6.2.

Study on orbital propagators: Constellation analysis with NASA 42 and Matlab/Simulink			
Task code	Task identification	Task code	Task identification
1.1	Orbital propagators	1.3	Satellites
1.1.1	Types	1.3.1	Cubesats
1.1.2	Orbital perturbations	1.4	Constellations
1.1.3	Coordinate Systems	1.4.1	Types
1.1.4	Programmes	1.4.2	Orbit visualisation
1.2	OpenSatKit	1.4.3	Ground stations
1.2.1	42	1.4.4	Node connectivity
1.2.1.1	Installation and implementation	1.4.5	Simulation
1.2.1.2	Input files	1.5	Comparison with previous models
1.2.1.3	Development of algorithms		
1.2.1.4	Output files		
1.2.1.5	Graphical User Interface		
1.2.1.6	InterProcess Communication		
1.2.1.7	Code analysis		
1.2.2	cFS & COSMOS		

**Table 6.1** Task identification with code

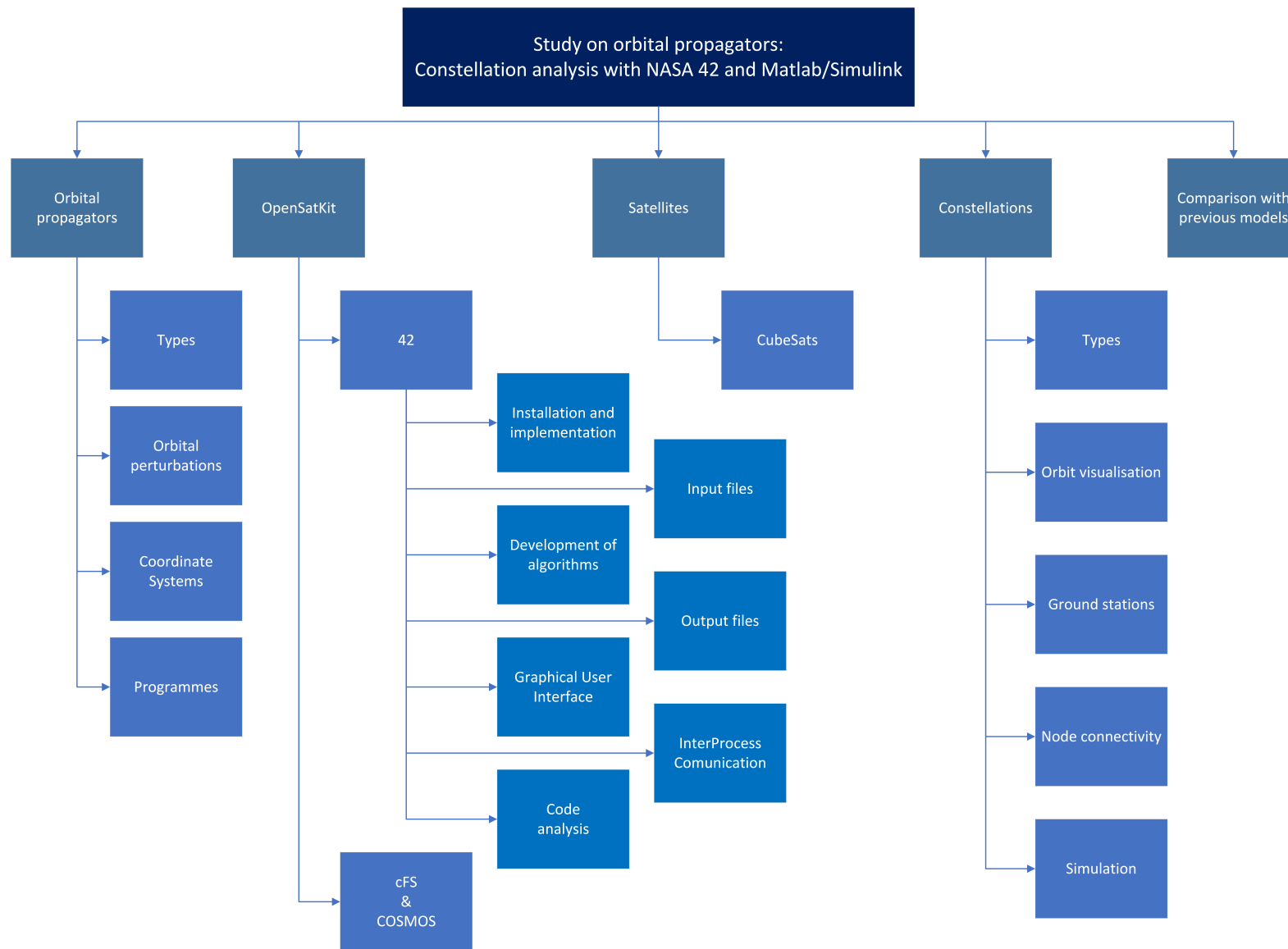


Figure 6.1 Work Breakdown Structure

## 6.2 Gantt diagram

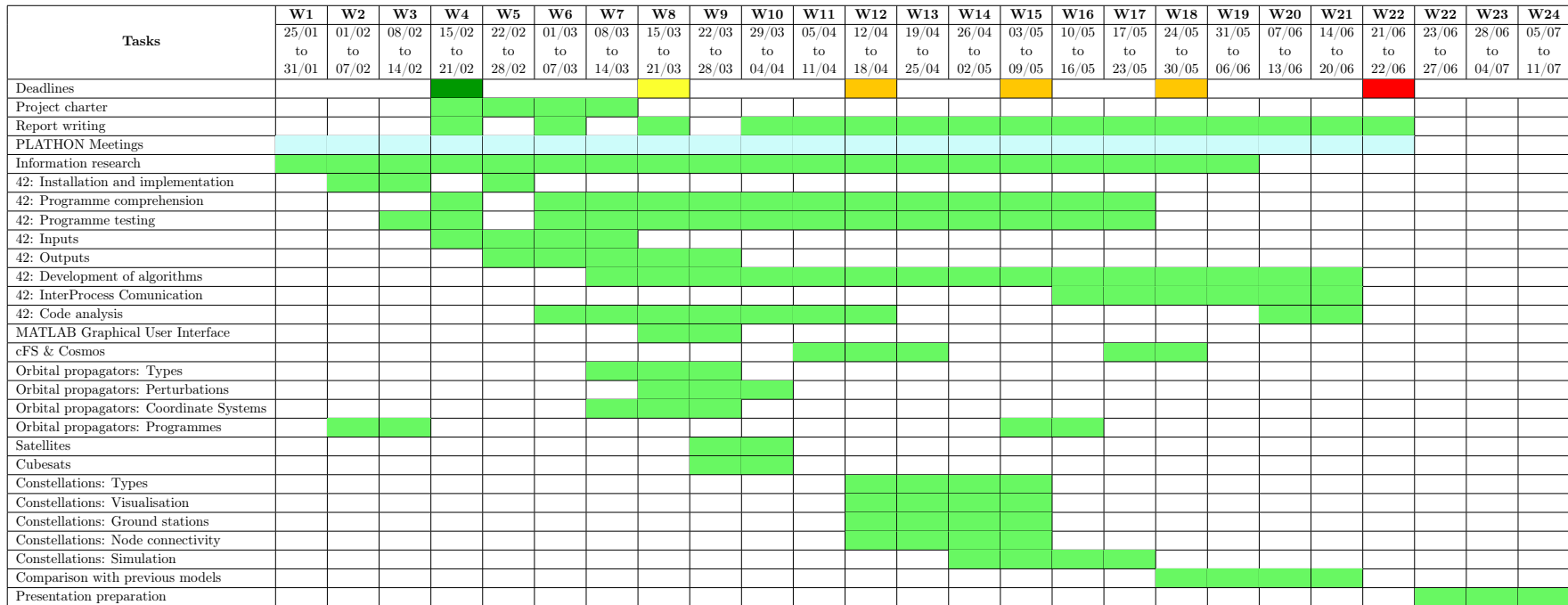
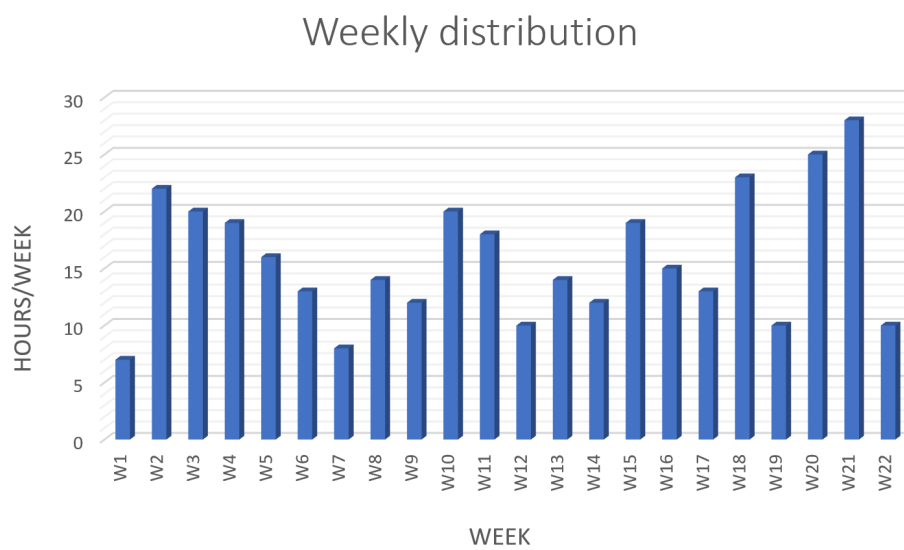


Table 6.2 Gantt diagram

The weekly work density is represented below:



**Figure 6.2** Weekly time distribution

As stated in the course guide for Bachelor's degree thesis, this project ought to oscillate around the 300 hours, which corresponds to the 12 ECTS. Adding up every week contribution, 348 hours have been devoted to the development of this thesis.



# Chapter 7

## Budget

This section contains the expenses related to this project development.

Human resources			
Concept	Quantity (h)	Unitary cost (€/h)	Total cost (€)
Income	336	20	6720
Tutoring	12	35	420
		<b>Subtotal</b>	<b>7140</b>

Equipment			
Concept	Quantity (units)	Unitary cost (€/h)	Total cost (€)
Laptop MSI GS63 Stealth	1	1200	1200
MATLAB Standard Annual license	1	800	800
Office material	1	20	20
		<b>Subtotal</b>	<b>2020</b>

Power consumption				
Concept	Quantity (h)	Electricity cost (€/KWh)	Power consumption (W)	Total cost (€)
Desktop computer	348	0.12	400	16.7
Main virtual machine (active hours)	120	0.12	350	5.0
Main virtual machine (passive hours)	2760	0.12	5	1.7
Secondary virtual machine (active hours)	40	0.12	250	1.2
Secondary virtual machine (passive hours)	1040	0.12	5	0.6
			<b>Subtotal</b>	<b>25.2</b>

**Table 7.1** Costs Breakdown

The average salaries from technical and scientific professionals have been approximated from the values extracted from IDESCAT [57].

The equipment expenses comes from all the necessary devices used for the development of this thesis. The laptop cost corresponds to the retail price while the MATLAB Standard Annual license cost has been obtained from MathWorks official website [58].

The office material expenditure is estimated, comprising basic tools.

The power consumption has been divided into the desktop computer, which has been connected for the entirety of the project and also accounts for the laptop use, and both virtual machines on active and passive modes. The main virtual machine has been used for 4 months while the second one was introduced for 45 days. The active hours correspond to the virtual machine in a computationally demanding environment while the passive hours correspond to idle mode.

The power consumption of each device has been calculated with an approximation of each computer components. The electricity cost has been obtained calculating the mean value of different companies.

As a result, the following total values are obtained:

<b>TAX BASE</b>	<b>7140.0 €</b>
<b>OTHER COSTS</b>	<b>2045.2 €</b>
VAT 21%	1499.4 €
<b>TOTAL</b>	<b>10684.6 €</b>

**Table 7.2** Total costs

## Chapter 8

# Environmental impact

The environmental impact of this project has been calculated based on the carbon footprint caused by electricity usage.

As the project majorly deals with programming concepts, the sporadic trips to the university lab to verify the virtual machine correct operation or to meet the director have not been taken into account.

The average  $CO_2$  emissions per kWh are extracted from the value of the 17th of June, 2021, according to Red Eléctrica de España [59].

Environmental impact					
Concept	Quantity (h)	Power consumption (W)	Total consumption (kWh)	Emissions (kg $CO_2$ /kWh)	Total carbon emissions (kg $CO_2$ )
Desktop computer	348	400	139.2	0.19	26.4
Main virtual machine (active hours)	120	350	42	0.19	8.0
Main virtual machine (passive hours)	2760	5	13.8	0.19	2.6
Secondary virtual machine (active hours)	40	250	10	0.19	1.9
Secondary virtual machine (passive hours)	1040	5	5.2	0.19	1.0
				<b>Total</b>	<b>39.9</b>

**Table 8.1** Environmental impact with carbon emissions

The implementation of a LEO constellation would be less pollutant in terms of space debris due to the natural atmosphere entry, which would completely obliterate small spacecraft.

## Chapter 9

# Conclusions and future prospects

In this section the project conclusions are presented along some recommended future studies that ought to be pursued for the continuation of the PLATHON project.

### 9.1 Conclusions

The PLATHON project has selected 42 as the main propagator due to its convergence with all the Open-SatKit modules. After analysing the whole system architecture, the connectivity between the programmes has been deemed possible to emulate CubeSat constellation missions with laser-based inter-satellite communication and ground-to-satellite links.

This thesis has paved the way for the initial Software-in-the-Loop simulations. Therefore, it has been designed as a basis for the following software connections and hardware implementations performed in the following years to reach the next mission validation level with a Hardware-in-the-Loop complete emulation.

Starting with the installation and testing efforts, all the process has been documented for facilitating future interface users into the programme depths. Once 42 is up and running on any of the available OS, the file distribution has been figured out to examine the outer layer of the programme. Finally, to assimilate the inner workings without reading the code, 42 comes with prepared examples of almost all the basic capabilities.

To lay the foundation of the programme use, the next reasonable step is to analyse the simulation procedure. As a result, the necessity to comprehend the input files, which are the only outer layer variables that can be modified, has resulted in a guide to use when creating new missions. After running the exemplary missions, the output files have also been analysed to understand each acronym. Next, code functions have been analysed in the dissertation appendix with brief comments to know where each code functionality can be found.

Therefore, due to the general project future mission, a series of modifications and comparisons have been made to both the base code and architecture.

The modification of the report function has enabled to save all spacecraft data if deemed necessary for post-process reason as only the first spacecraft of each simulation could write the data to a document file.

In order to pursue the analysis of future constellations, a Matlab file creator has been programmed to create the necessary files for each simulation. Once the results are obtained, the same Matlab Interface can read the aforementioned file to post-process the results.

The use of virtual machine supposed some Graphical User Interface that arose the idea of developing an independent Matlab GUI. However, after some Simulink representations, the non-easily scalable properties of this method ended up cancelling the viability of a Matlab real-time rendering. Therefore, the 42 GUI problems were addressed to affirm that the computer's graphical power bottlenecked the simulation speed.

For the purpose of testing prominent currently orbiting constellations, the TLE format was introduced and tested with the latest data of a specific constellation. The example portraits the Iridium NEXT constellation inside the 42 system. Once this was achieved, the renderisation of prominent constellations with 42 was considered suitable for the PLATHON needs.

The use of the TLE format required an analysis of the accuracy of the data and the method. As a result, it can be affirmed that their propagation values are only sufficiently accurate for short periods, with the distance between the actual satellite and the propagated one growing exponentially with time.

The last mission was to simulate the Inter-Process Communication for the SiTL and HiTL future capabilities. This task was performed with the use of several computers. One would act as the programme simulator while another PC only received the GUI visualisation. Furthermore, the first computer has 2 AcApps which can control the attitude of two separate spacecraft. However, these attitude controllers could be anywhere. The communications are via socket network connections.

All the thesis objectives have been satisfactorily accomplished, pulling forward both the Software and Hardware architecture of the PLATHON project, establishing the basis for future work enhancing the current features and integrating 42 into the whole system.

## 9.2 Future prospects

The PLATHON project still requires extensive efforts to reach a fully functional, stable status. During the development of the thesis, some aspects that will require further studies have been identified and could be the basis of the new bachelor's and master's thesis.

The 42 propagator perturbation models, although precise for these analyses, have been outdated for some years, and more recent, updated versions have arisen, resulting in better approximations and higher accuracy. For instance, the IGRF model should be updated to take into account geomagnetic Earth variations.

Another 42 property that could be analysed is the numerical ODE system method that is currently being used. The Runge-Kutta with constant step size could be enhanced to a higher-order iterative method with variable time steps for computational efficiency. However, for post-process analyses, there might be problems with uneven time variations among data measurements.

Furthermore, the current 42 models for both sensors and actuators might be considered too simple to portrait the fluctuating space environment. Thus, if more detailed models were added to the system, the simulation results would be closer to reality. Considering that the GPS position will be the only 42-dependant asset during the HiTL emulations, as the other sensors and actuators will be included with Hardware connections, that model should be enhanced the most.

Based on this thesis, the modified AcApp control applications will need to reach the same level of attitude control as the 42 independent simulator to use all the available command range. This task could be pursued with the same structure of the 42 Command file, with time-dependant commands, or through the input of real-time command on a connected terminal.

At the same time, the integration of all the OpenSatKit modules through the studied communications would be the logical step forward. Principally, combining the cFS with 42 and then with COSMOS.

Once the Hardware prototypes are finished, the spacecraft model should be inserted as a spacecraft file along with the geometry definition and the CubeSat internal parameters.

Once all the systems are independently ready, it will be time to analyse the communication between them and between software simulations and Hardware devices.

# Bibliography

- [1] TIEG. (2020). Plathon constellation scenario with optical comms. *SPANISH SMALL SATELLITES INTERNATIONAL FORUM* (cit. on pp. 2, 16).
- [2] Venturini, C. C. (2017). *Improving Mission Success of CubeSats* (tech. rep.). (Cit. on p. 2).
- [3] Pocha, J. J. (2012). *An introduction to mission design for geostationary satellites* (Vol. 1). Springer Science & Business Media. (Cit. on pp. 3, 28).
- [4] Shou, H. N. (2014). Orbit propagation and determination of low earth orbit satellites. *International Journal of Antennas and Propagation*, 2014. <https://doi.org/10.1155/2014/903026> (cit. on p. 3)
- [5] Curtis, H. D. (2019). *Orbital mechanics for engineering students*. Butterworth-Heinemann. (Cit. on pp. 5, 27, 30, 32, 33).
- [6] Kelso, T. (2021). *Norad two-line element set format*. <https://celestrak.com/NORAD/elements/>. (Cit. on pp. 6, 83)
- [7] SAIC. (n.d.). Retrieved June 21, 2021, from <https://www.space-track.org>. (Cit. on pp. 6, 83)
- [8] Sermanoukian, I. (n.d.). *42plathon*. Retrieved June 21, 2021, from <https://github.com/Ivan-Sermanoukian/42PLATHON>. (Cit. on pp. 7, 67, 79, 120, 129)
- [9] GMAT Development Team (NASA). (2020a). *Gmat user guide r2020a*. <http://gmat.sourceforge.net/docs/R2020a/help-letter.pdf>. (Cit. on p. 9)
- [10] GMAT Development Team (NASA). (2020b). *General mission analysis tool download page*. <https://sourceforge.net/projects/gmat/>. (Cit. on p. 9)
- [11] Goddard Space Flight Center. (n.d.[a]). *42: A comprehensive general-purpose simulation of attitude and trajectory dynamics and control of multiple spacecraft composed of multiple rigid or flexible bodies*. <https://software.nasa.gov/software/GSC-16720-1>. (Cit. on p. 10)
- [12] Goddard Space Flight Center. (n.d.[b]). *Opensatkit*. <https://github.com/OpenSatKit/OpenSatKit>. (Cit. on p. 11)
- [13] Ball Aerospace COSMOS. (n.d.). Retrieved April 3, 2021, from <https://cosmosrb.com/>. (Cit. on pp. 11, 12)
- [14] Core Flight System. (n.d.). Retrieved April 3, 2021, from <https://cfs.gsfc.nasa.gov/>. (Cit. on p. 11)
- [15] CS GROUP. (2021). *Orekit: An accurate and efficient core layer for space flight dynamics applications*. <https://www.orekit.org/>. (Cit. on p. 13)
- [16] CS GROUP. (2020). *Orekit overview*. <https://www.orekit.org/site-orekit-10.1/index.html>. (Cit. on p. 13)
- [17] Poza Hernández, D. (2020). *Study of magnetorquers control strategies for energy efficient manoeuvres of a CubeSat REPORT* (tech. rep.). Universitat Politècnica de Catalunya. <https://upcommons.upc.edu/handle/2117/329827>. (Cit. on pp. 14, 88)

- [18] Parrot Martínez, I. (2021). *Study of energy efficient manoeuvring strategies for CubeSats* (tech. rep.). Universitat Politècnica de Catalunya. (Cit. on pp. 14, 31, 88).
- [19] Cappelletti, C., Battistini, S., & Malphrus, B. K. (Eds.). (2021). Introduction: The history of the cubesat by bob twiggs and jordi puig-suari. In *Cubesat handbook* (pp. xxi–xxviii). Academic Press. <https://doi.org/https://doi.org/10.1016/B978-0-12-817884-3.09983-5>. (Cit. on p. 21)
- [20] NanoSat Lab - Universitat Politècnica de Catalunya. (2021). *Cubecat-4*. <https://nanosatlab.upc.edu/en/missions-and-projects/3cat-4>. (Cit. on p. 21)
- [21] Alen Space. (2021). *A basic guide to nanosatellites*. <https://alen.space/basic-guide-nanosatellites/>. (Cit. on p. 22)
- [22] Kulu, E. (2021). *Nanosats database*. <https://www.nanosats.eu/>. (Cit. on p. 22)
- [23] Wertz, J. R. (2009). *Orbit & constellations design & management* (1st ed.). Space Technology Library. (Cit. on p. 23).
- [24] Toyoshima, M. (2019). Recent trends in space laser communications for small satellites and constellations. *2019 IEEE International Conference on Space Optical Systems and Applications (ICSOS)*, 1–5. <https://doi.org/10.1109/ICSOS45490.2019.8978972> (cit. on pp. 23, 25)
- [25] Rovira Garcia, Dr. Adrià. (2020). *Module 1 Geocentric Orbits*. (Cit. on pp. 23, 24).
- [26] Thales Alenia Space. (n.d.). Iridium® NEXT constellation, built by Thales Alenia Space, now completely deployed in orbit | Thales Group. Retrieved May 14, 2021, from <https://www.thalesgroup.com/en/worldwide/space/press-release/iridiumr-next-constellation-built-thales-alenia-space-now-completely>. (Cit. on p. 24)
- [27] Mohon, L. (2021). *Laser communications relay demonstration (lcrd)*. [https://www.nasa.gov/mission\\_pages/tdm/lcrd/index.html](https://www.nasa.gov/mission_pages/tdm/lcrd/index.html). (Cit. on p. 25)
- [28] European Space Agency. (2021). *Ops-sat*. [http://www.esa.int/Enabling\\_Support/Operations/OPS-SAT](http://www.esa.int/Enabling_Support/Operations/OPS-SAT). (Cit. on p. 25)
- [29] Baird, D. (2021). *Laser communications: Empowering more data than ever before*. <https://www.nasa.gov/feature/goddard/2021/laser-communications-empowering-more-data-than-ever-before>. (Cit. on p. 25)
- [30] Knapek, M., Al-Mudhafar, A., Müncheberg, S., Shortt, K., & Soutullo, M. (2017). Development of a laser communication terminal for large leo constellations. *2017 IEEE International Conference on Space Optical Systems and Applications (ICSOS)*, 53–58. <https://doi.org/10.1109/ICSOS.2017.8357211> (cit. on p. 25)
- [31] European Space Agency. (n.d.[a]). Secure communication via quantum cryptography. Retrieved June 15, 2021, from [https://www.esa.int/Applications/Telecommunications\\_Integrated\\_Applications/Secure\\_communication\\_via\\_quantum\\_cryptography](https://www.esa.int/Applications/Telecommunications_Integrated_Applications/Secure_communication_via_quantum_cryptography). (Cit. on p. 26)
- [32] Fortescue, Peter, & Swinerd, Graham. (2011). *Spacecraft systems engineering* (4th ed.). WILEY. (Cit. on p. 27).
- [33] Larson, W. J., & Wertz, J. R. (1992). *Space mission analysis and design* (tech. rep.). Torrance, CA (United States); Microcosm, Inc. (Cit. on pp. 28, 29).
- [34] Walter, U. (2019). *Astronautics* (1st ed.). Springer. (Cit. on p. 28).
- [35] NASA. (1976). *U.s. standard atmosphere, 1976* (tech. rep. No. 1) [Document ID 19770009539]. NASA/NOAA. The address of the publisher. <https://ntrs.nasa.gov/citations/19770009539>. (Cit. on pp. 28, 29)
- [36] Hedin, A. E. (1987). MSIS-86 Thermospheric Model. *Journal of Geophysical Research*, 92, 4649–4662. <https://doi.org/10.1029/JA092iA05p04649> (cit. on p. 28)



- [37] Mahooti, M. (2019). Msis-86 atmosphere model (matlab code). <https://doi.org/10.13140/RG.2.2.20926.23363>. (Cit. on p. 29)
- [38] Emmert, J. T. (2015). Thermospheric mass density: A review. <https://doi.org/10.1016/j.asr.2015.05.038>. (Cit. on p. 30)
- [39] Feissel, M., & Mignard, F. (1998). *Letter to the Editor The adoption of ICRS on 1 January 1998: meaning and consequences* (tech. rep.). <https://ui.adsabs.harvard.edu/abs/1998A&A...331L..33F/abstract>. (Cit. on p. 32)
- [40] Vallado, D. A. (2013). *Fundamentals of astrodynamics and applications* (4th ed.). Microcosm Press. (Cit. on p. 32).
- [41] Peraire, J., & Widnall, J. (2008). *3D Rigid Body Dynamics: The Inertia Tensor* (tech. rep.). MIT. [https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-07-dynamics-fall-2009/lecture-notes/MIT16\\_07F09\\_Lec26.pdf](https://ocw.mit.edu/courses/aeronautics-and-astronautics/16-07-dynamics-fall-2009/lecture-notes/MIT16_07F09_Lec26.pdf). (Cit. on p. 38)
- [42] Goddard Space Flight Center (NASA). (2007). *Hubble space telescope servicing mission 4 gyroscopes*. [https://www.nasa.gov/pdf/206045main\\_GYROS\\_FS-2006-11-087\\_12\\_4.pdf](https://www.nasa.gov/pdf/206045main_GYROS_FS-2006-11-087_12_4.pdf). (Cit. on p. 39)
- [43] Cain, F. (2019). *Spacecraft gyroscopes and reaction wheels*. <https://www.universetoday.com/143152/spacecraft-gyroscopes-and-reaction-wheels-you-can-never-have-enough/>. (Cit. on pp. 39, 40)
- [44] Makesat. (2021). *3-axis magnetometer for small satellites*. <https://makesat.com/en/products/magnetometer-for-satellites>. (Cit. on pp. 39, 40)
- [45] Curtis, H. (2020). *Sun sensors on the global marketplace for space*. <https://blog.satsearch.co/2020-02-12-sun-sensors-an-overview-of-systems-available-on-the-global-marketplace-for-space>. (Cit. on p. 40)
- [46] Saborio, R. J. (2017). *Characterizing performance and errors of coarse sun sensors*. <https://conservancy.umn.edu/bitstream/handle/11299/191874/Characterizing%20Performance%20and%20Errors%20of%20Coarse%20Sun%20Sensors.pdf?sequence=1&isAllowed=y>. (Cit. on p. 40)
- [47] Sarvi, M., Abbasi, D., Abolghasemi, M., & Hoseini, H. (2020). Design and implementation of a star-tracker for leo satellite. *Optik*, 208. <https://doi.org/10.1016/j.ijleo.2020.164343> (cit. on p. 40)
- [48] IAGA. (2019). *International geomagnetic reference field*. <https://www.ngdc.noaa.gov/IAGA/vmod/igrf.html>. (Cit. on p. 50)
- [49] Ji Zhang, Yi Qiang. (2021). Project of 1 DoF Attitude Control System of 1U Cubesat Based On Reaction Wheel. UPC. (Cit. on p. 56).
- [50] Stoneking, E. (2010–2019). 42: A General-Purpose Spacecraft Simulation. Retrieved April 3, 2021, from <https://sourceforge.net/projects/fortytwospacecraftsimulation,%20https://github.com/ericstoneking/42>. (Cit. on pp. 57, 62)
- [51] Wallace, M. (2019). Tracking and Data Relay Satellite (TDRS) Fleet. [http://www.nasa.gov/directorates/heo/scan/services/networks/tdrs\\_fleet](http://www.nasa.gov/directorates/heo/scan/services/networks/tdrs_fleet) (cit. on p. 61)
- [52] Speretta, S., Sundaramoorthy, P., & Gil, E. (n.d.). *Long-term performance analysis of norad two-line elements for cubesats and pocketqubes*. Retrieved June 20, 2021, from <https://core.ac.uk/download/pdf/144878002.pdf>. (Cit. on p. 82)
- [53] European Space Agency. (n.d.[b]). *Somp (students' oxygen measurement project)*. Retrieved June 21, 2021, from <https://earth.esa.int/web/eoportal/satellite-missions/s/somp>. (Cit. on p. 83)
- [54] Dunbar, B. (n.d.). *Spaceflight.nasa.gov has been retired*. Retrieved June 21, 2021, from <https://www.nasa.gov/feature/spaceflightnasagov-has-been-retired/>. (Cit. on p. 85)

- [55] NASA. (n.d.). *Data.nasa.gov: A catalog of publicly available nasa datasets*. Retrieved June 21, 2021, from <https://data.nasa.gov/>. (Cit. on p. 85)
- [56] NASA. (2021). *Naif integer id codes*. [https://naif.jpl.nasa.gov/pub/naif/toolkit\\_docs/FORTRAN/req/naif\\_ids.html](https://naif.jpl.nasa.gov/pub/naif/toolkit_docs/FORTRAN/req/naif_ids.html). (Cit. on p. 86)
- [57] d'Estadística de Catalunya, I. (n.d.). Indicadors anuals. Salari brut anual i guany per hora. Per sexe i tipus d'ocupació. Retrieved June 18, 2021, from <https://www.idescat.cat/indicadors/?id=anuals&n=10403>. (Cit. on p. 94)
- [58] MathWorks. (n.d.). Pricing and licensing. Retrieved June 18, 2021, from <https://www.mathworks.com/pricing-licensing.html>. (Cit. on p. 94)
- [59] Red Eléctrica de España. (n.d.). *No renovables detalle emisiones co2 | red eléctrica de españa*. Retrieved June 18, 2021, from <https://www.ree.es/es/datos/generacion/no-renovables-detalle-emisiones-CO2>. (Cit. on p. 95)
- [60] Kelso, T. (2019). *Norad two-line element set format*. <https://www.celestrak.com/NORAD/documentation/tle-fmt.php>. (Cit. on pp. 104, 105)
- [61] Carles Bonet Reves, Angel Jorba Monte, M. Teresa Martínez-Seara Alonso, Joaquín Masdemont Soler, Mercè Ollé Torner, Antoni Susin Sánchez, & Marta València Guitart. (2012). *Càlcul numèric*. Edicions UPC. (Cit. on pp. 106, 107).
- [62] Hall, N. (n.d.). Mars atmosphere model. glenn research center. Retrieved April 18, 2021, from <https://www.grc.nasa.gov/WWW/K-12/airplane/atmosmrm.html>. (Cit. on p. 123)



# Appendix A

## Data appendix

This appendix chapter introduces the reader to some concepts and extends relevant data mentioned throughout the thesis.

The following sections can be found:

1. TLE data format
2. Numerical methods for ODE systems
3. 42 Propagator images
4. 5th ESA CubeSat Industry Days

## A.1 TLE data format

The data available for each satellite consists of three lines with the following format:

Line 0 is an optional twenty-four character name preceding the element data. The title is not required, as each data line includes a unique object identifier code.

01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
S	T	A	R	L	I	N	K	-	1	9	1	9											
1																							

**Table A.1** Line 0: Title line of Starlink-1919 TLE

### A.1.1 TLE Line 1

01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
1		4	6	7	6	9	U		2	0	0	7	4	A	G			2	1
1		2					3		4		5			6				7	

21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
1	2	8	.	3	3	7	4	2	3	8	0			.	0	0	0	0	0	6	2	5
8													9									

44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	
		0	0	0	0	0	-	0			6	0	8	5	9	-	4		0			9	9	9	4	
10										11										12		13				14

**Table A.2** Line 1: Starlink-1919 TLE

Line 1			
Field	Column	Description	Comments
1	01	Line Number of Element Data	
2	03-07	Satellite Number	NORAD ID
3	08	Classification	U=Unclassified, C=Classified, S=Secret
4	10-11	International Designator	Last two digits of launch year
5	12-14	International Designator	Launch number of the year
6	15-17	International Designator	Piece of the launch
7	19-20	Epoch Year	Last two digits of year
8	21-32	Epoch	Day of the year and fractional portion of the day
9	34-43	First Time Derivative of the Mean Motion	C34 can be (-) or blank for (+)
10	45-52	Second Time Derivative of Mean Motion	Leading decimal point assumed; C45 can be (-) or blank for (+)
11	54-61	BSTAR drag term	Leading decimal point assumed; C54 can be (-) or blank for (+)
12	63	Ephemeris type	Internal use only; Always zero in distributed TLE data
13	65-68	Element number	Increases one for each newly generated TLE
14	69	Checksum (Modulo 10)	Letters, blanks, periods, plus signs = 0; minus signs = 1

**Table A.3** Line 1: Content description. Data from CelesTrak [60].

### A.1.2 TLE Line 2

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	
2		4	6	7	6	9			5	3	.	0	5	6	9		3	4	6	.	0	8	2	4		
1		2					3										4									
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	
0	0	0	1	6	7	3			8	0	.	0	9	5	7		2	8	0	.	0	2	2	1		
5							6										7									
53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69										
1	5	.	0	6	3	9	6	8	0	9		2	9	9	6	3										
8											9					10										

Table A.4 Line 2: Starlink-1919 TLE

Line 2			
Field	Column	Description	Comments
1	01	Line Number of Element Data	
2	03-07	Satellite Number	NORAD ID
3	09-16	Inclination	Degrees [°]
4	18-25	Right Ascension of the Ascending Node	Degrees [°]
5	27-33	Eccentricity	Leading decimal point assumed
6	35-42	Argument of Perigee	Degrees [°]
7	44-51	Mean Anomaly	Degrees [°]
8	53-63	Mean Motion	Revolutions per day
9	64-68	Revolution number at epoch	Total revolution count
10	69	Checksum (Modulo 10)	Letters, blanks, periods, plus signs = 0; minus signs = 1

Table A.5 Line 2: Content description. Data from CelesTrak [60].

## A.2 Numerical methods for ODE systems

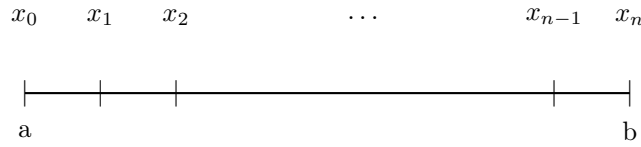
Numerical methods can be divided into explicit methods, which calculate the state of a system at a later time directly from the state of the system at the current time; and implicit methods, which find a solution by solving an equation involving both the current state of the system and the later one, becoming iterative in the nonlinear case.

An initial value problem is considered:

$$\begin{cases} \frac{dy}{dx} = f(x, y(x)) \\ y(a) = y_0 \end{cases} \quad \begin{matrix} \text{(A.2.1)} \\ \text{(A.2.2)} \end{matrix}$$

with  $x \in [a, b]$ ,  $y(x) \in \mathbb{R}$  i  $f: [a, b] \times \mathbb{R} \rightarrow \mathbb{R}$ . In general, most of the differential equations posed do not have an explicit solution  $y(x)$ . However, given points  $x_0, \dots, x_n \in [a, b]$ , we can obtain the value that takes  $y(x)$  in these [61].

The process consists first of all in subdividing the interval  $[a, b]$  into  $N$  parts which can either be equal or variable depending on the problem morphology. For simplicity, this example will consider equal divisions:



**Figure A.1**  $[a, b]$  interval

Defining  $h$  as the amplitude of this partition, it turns out that  $h = (b - a)/N$  and that  $x_n = a + nh$  for  $n = 0 \dots N$ . Henceforth,  $y(x_n)$  will denote the value of the exact solution at the point  $x_n$  and  $y_n$ , the approximation generated by the numerical method under consideration.

For this thesis, only single-step methods have been used.

### A.2.1 Explicit Euler Method

$$y_{n+1} = y_n + hf(y_n) \quad \text{(A.2.3)}$$

It uses a constant step size  $h$  to compute, one after the other, approximations  $y_1, y_2, y_3, \dots$  to the values  $y(h), y(2h), y(3h), \dots$  of the solution starting from a given initial value  $y(0) = y_0$ . The method is called the explicit Euler method, because the approximation  $y_{n+1}$  is computed using an explicit evaluation of  $f$  at the already known value  $y_n$ .

### A.2.2 Implicit Euler Method

$$y_{n+1} = y_n + hf(y_{n+1}), \quad \text{(A.2.4)}$$

is known for its all-damping stability properties. In contrast to (1.5), the approximation  $y_{n+1}$  is defined implicitly by (1.6), and the implementation requires the numerical solution of a nonlinear system of equations.

### A.2.3 Implicit Midpoint Rule

Taking the mean of  $y_n$  and  $y_{n+1}$  in the argument of  $f$ , we get the implicit midpoint rule

$$y_{n+1} = y_n + hf \left( \frac{y_n + y_{n+1}}{2} \right). \quad (\text{A.2.5})$$

It is a symmetric method, which means that the formula is left unaltered after exchanging  $y_n \leftrightarrow y_{n+1}$  and  $h \leftrightarrow -h$

### A.2.4 Runge Kutta methods

Runke-Kutta algorithms are given by [61]:

$$\begin{cases} y(a) = y_0 \\ y_{n+1} = y_n + h \sum_{i=1}^k c_i k_i^n, \quad 0 \leq n \leq N-1 \end{cases} \quad (\text{A.2.6})$$

$$\quad \quad \quad (\text{A.2.7})$$

with  $k \in \mathbb{N}$  previously defined, and the coefficients  $k_i^n$  are functions defined by:

$$\begin{cases} k_1^n = f(x_n, y_n) \\ k_i^n = f \left( x_n + a_i h, y_n + h \sum_{j=1}^{i-1} b_{ij} k_j^n \right), \quad 2 \leq i \leq k \end{cases} \quad (\text{A.2.8})$$

$$\quad \quad \quad (\text{A.2.9})$$

The RK4 algorithm is the Runge-Kutta algorithm obtained for  $k = 4$ . This is written as:

$$\begin{cases} y(a) = y_0 \\ y_{n+1} = y_n + \frac{h}{6} (k_1^n + 2k_2^n + 2k_3^n + k_4^n), \quad 0 \leq n \leq N-1 \end{cases} \quad (\text{A.2.10})$$

$$\quad \quad \quad (\text{A.2.11})$$

where the coefficients are defined by:

$$\begin{cases} k_1^n = f(x_n, y_n) \\ k_2^n = f \left( x_n + \frac{h}{2}, y_n + \frac{h}{2} k_1^n \right) \end{cases} \quad (\text{A.2.12})$$

$$\quad \quad \quad (\text{A.2.13})$$

$$\begin{cases} k_3^n = f \left( x_n + \frac{h}{2}, y_n + \frac{h}{2} k_2^n \right) \\ k_4^n = f(x_n + h, y_n + h k_3^n) \end{cases} \quad (\text{A.2.14})$$

$$\quad \quad \quad (\text{A.2.15})$$

Intuitively, Runke-Kutta methods discretise the domain defined by the independent variable  $x$  in steps defined by  $h$ , and for each point  $x_n$  of the discretisation, they calculate the value of  $y'$  in points around  $x_n$ , i.e., the coefficients  $k_i^n$ . From the current value  $y_n$  of the function and the coefficients, the following value of the function is calculated,  $y_{n+1}$ .

Numerical methods are, in general, approximations to the analytical solution of the problem. The accuracy of the approximation depends on  $h$ . However, the RK4 method is of global order 4, that is, the error is of the order of  $\mathcal{O}(h^4)$ . This implies that, if  $h' = h/2$  is defined and the RK4 algorithm is applied again, the error will be reduced 16 times.

Runge-Kutta methods, and in particular the RK4 method, are easily generalisable to systems of ordinary differential equations.



### A.3 42 Images

This section contains supplementary images of 42 orbital propagator.

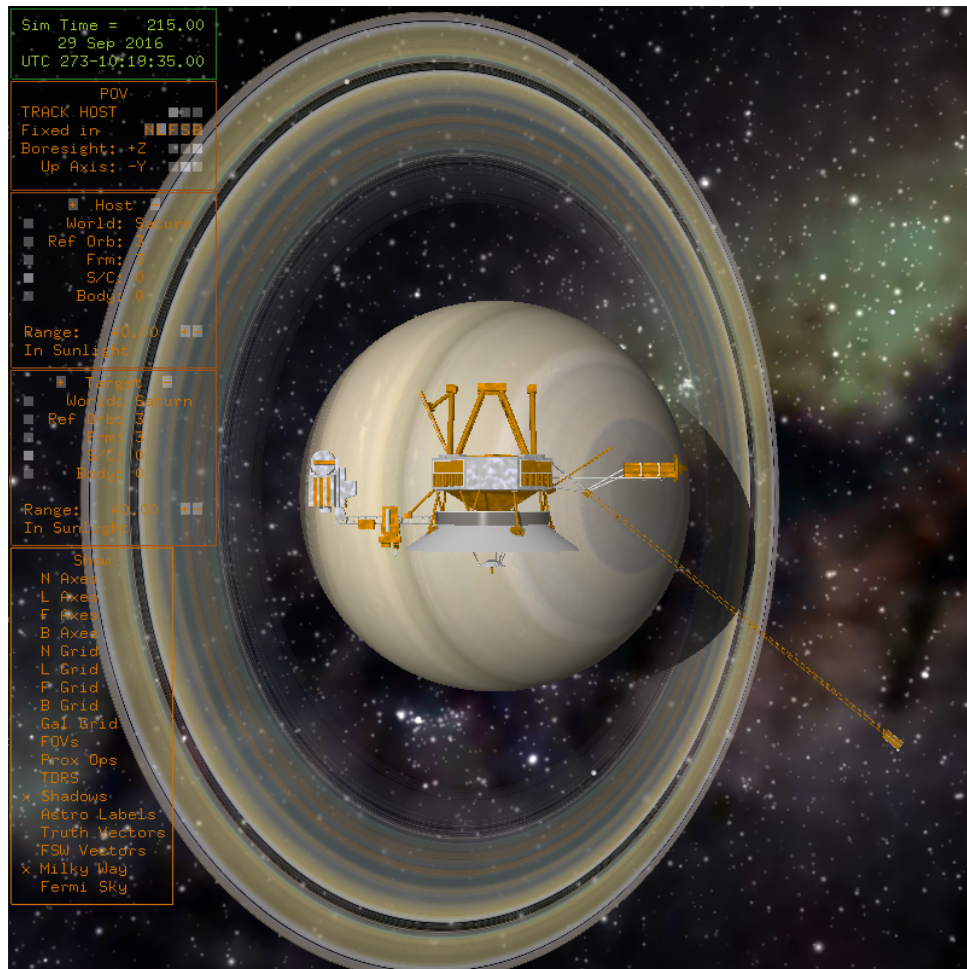


Figure A.2 42 Cam - Voyager mission

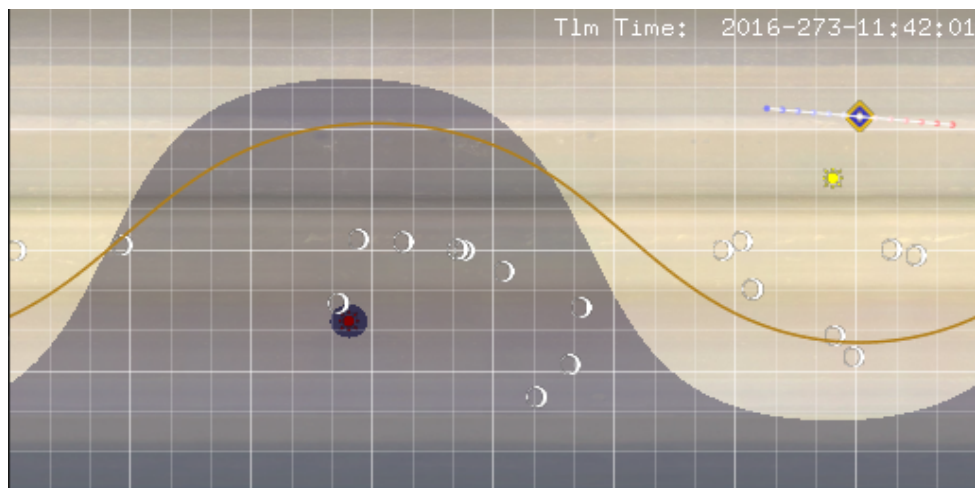


Figure A.3 42 2D map - Saturn with Voyager mission

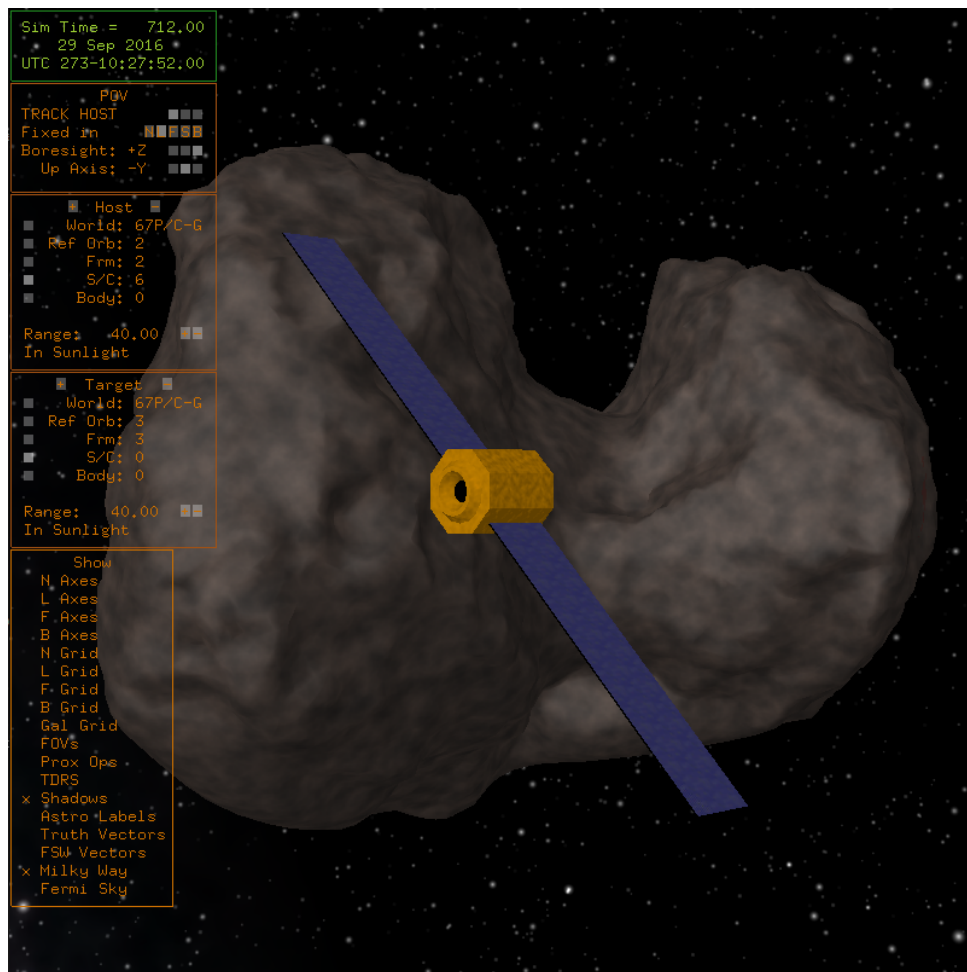


Figure A.4 42 Cam - Rosetta space probe

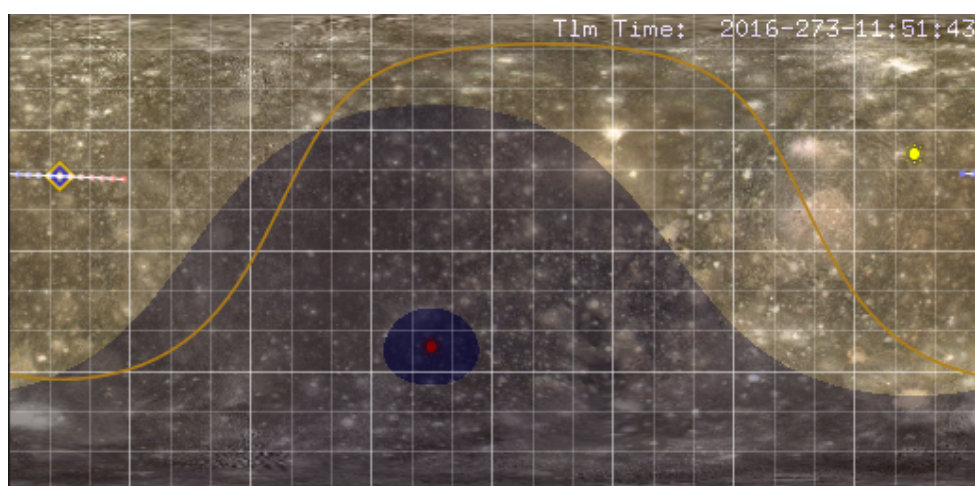


Figure A.5 42 2D map - 67P/Churyumov-Gerasimenko comet

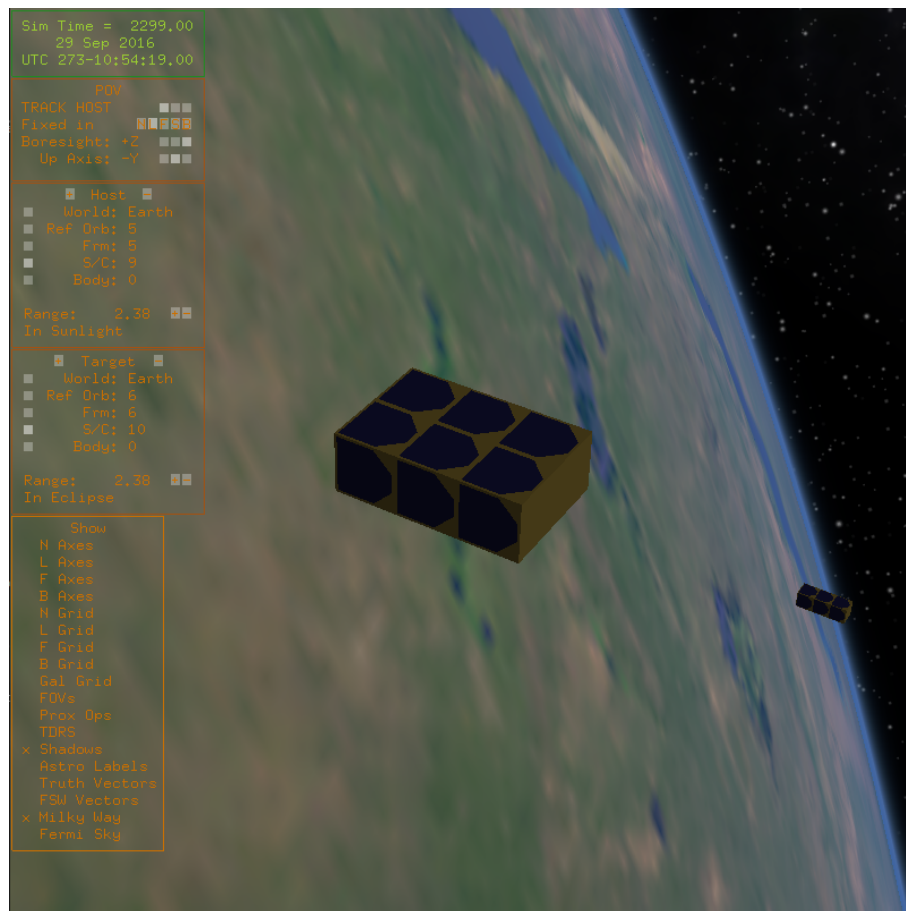


Figure A.6 42 Cam - 6U + 3U CubeSat formation

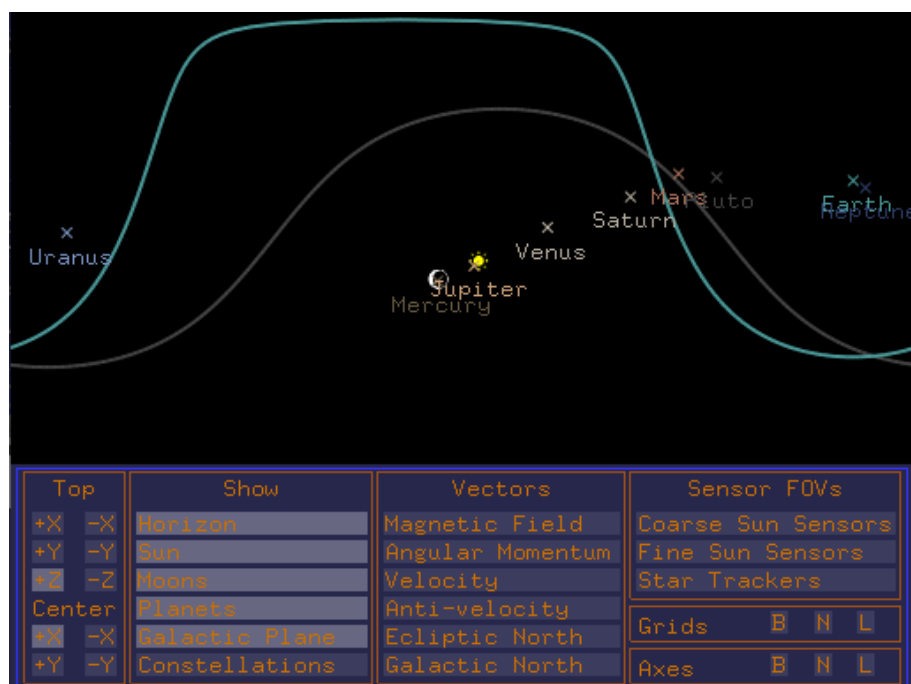


Figure A.7 42 Unit Sphere Viewer - Major celestial bodies

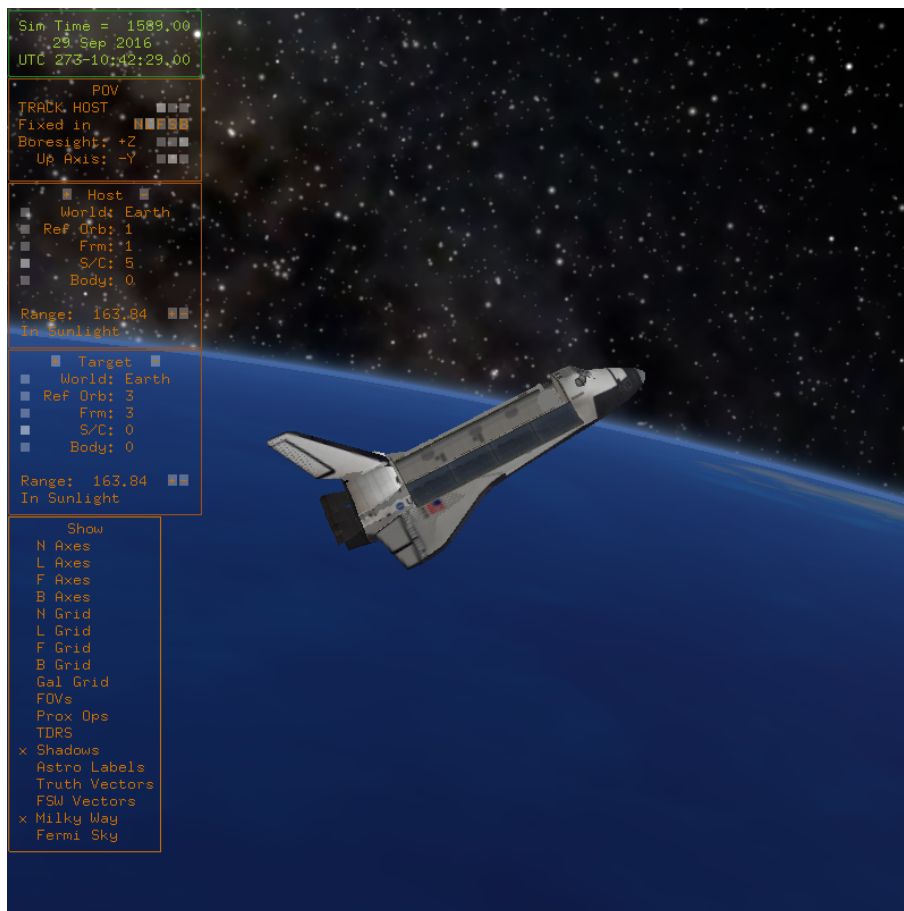


Figure A.8 42 Cam - Space Shuttle

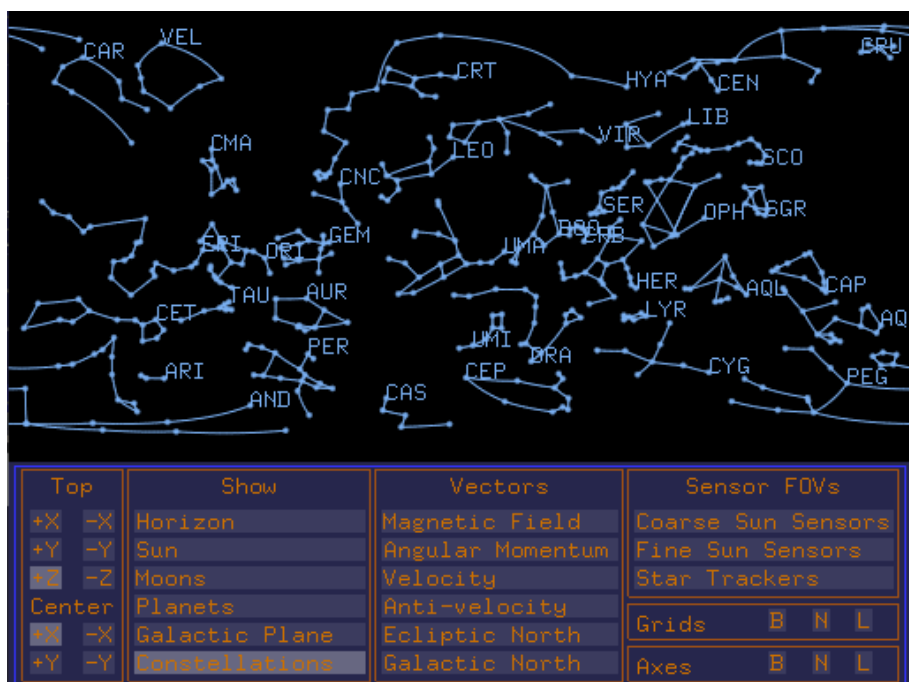


Figure A.9 42 Unit Sphere Viewer - Constellations





Figure A.10 42 Cam - International Space Stations

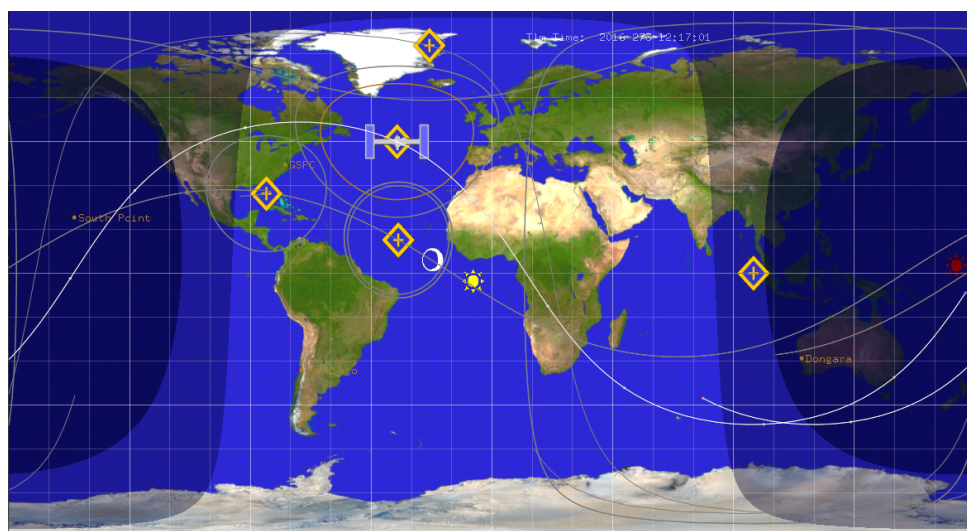


Figure A.11 42 2D map - ISS and CubeSat orbits

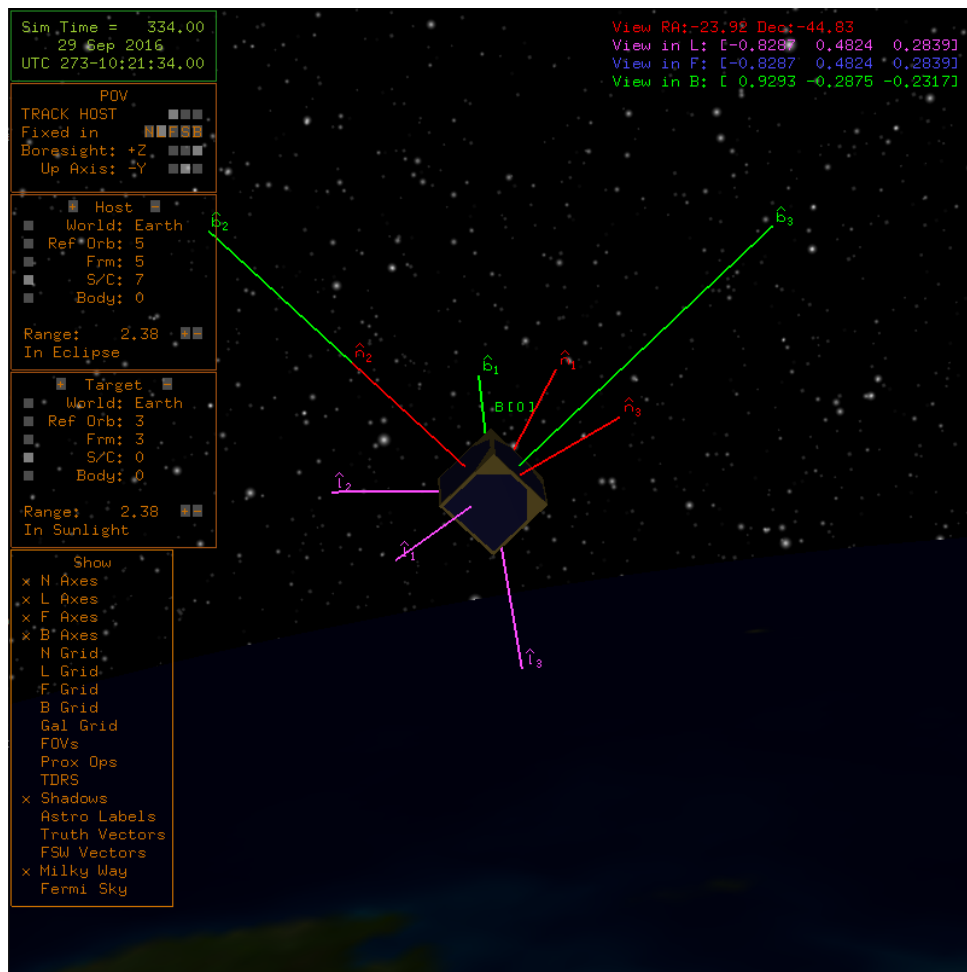


Figure A.12 42 Cam - CubeSat axes

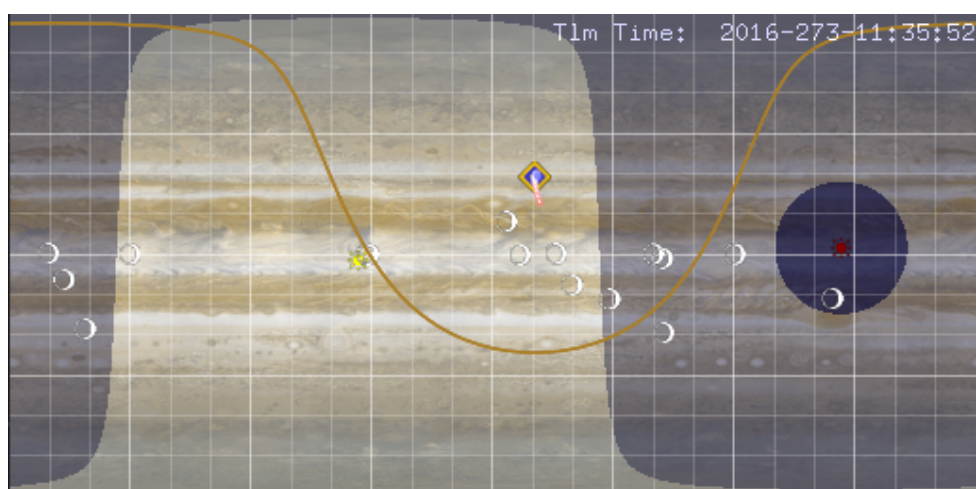


Figure A.13 42 2D map - Jupiter

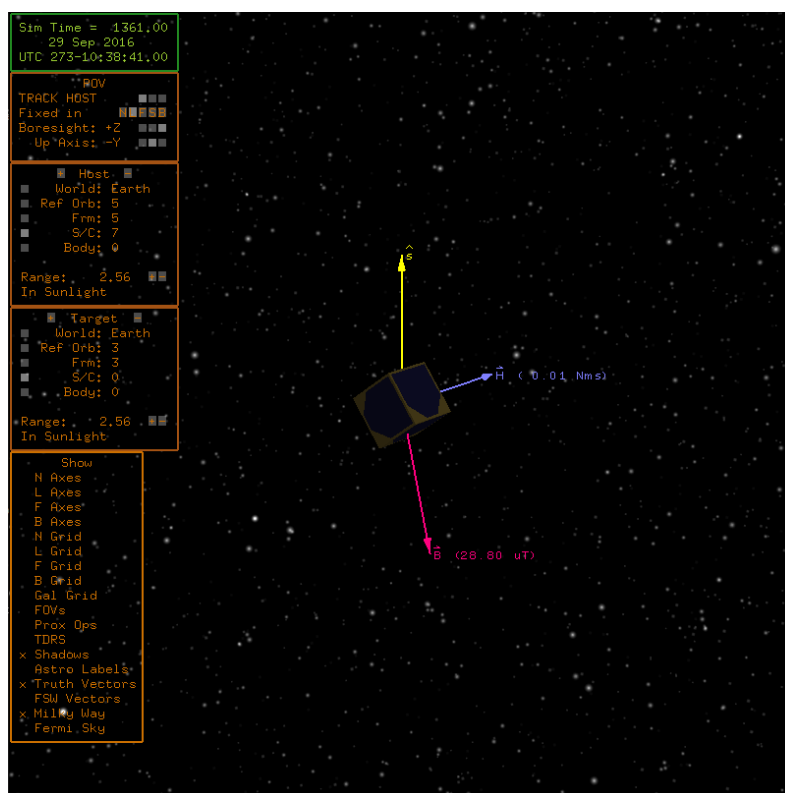


Figure A.14 42 Cam - Truth vectors

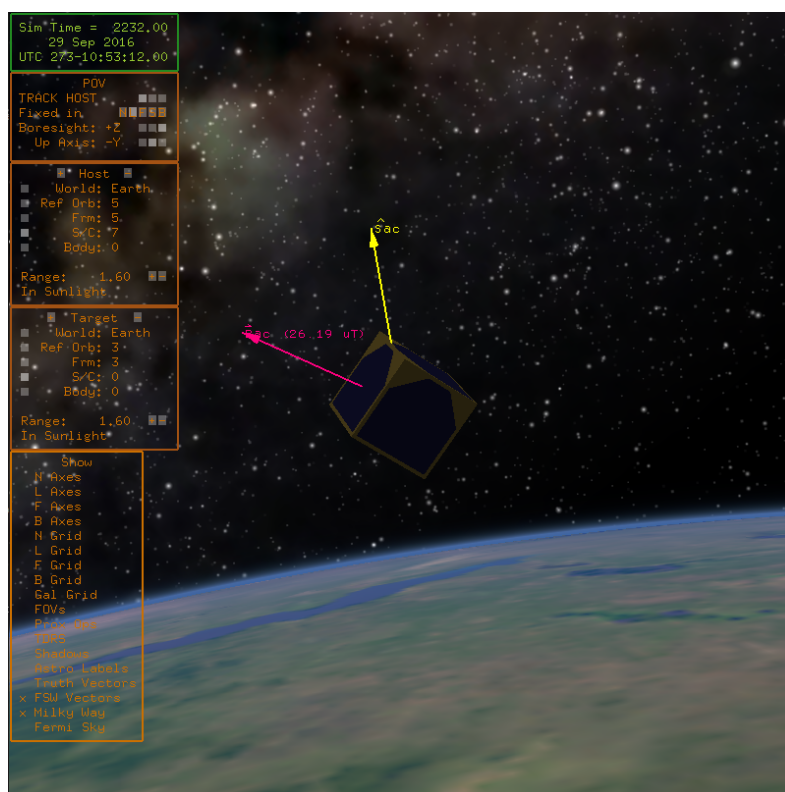


Figure A.15 42 Cam - Flight Software vectors



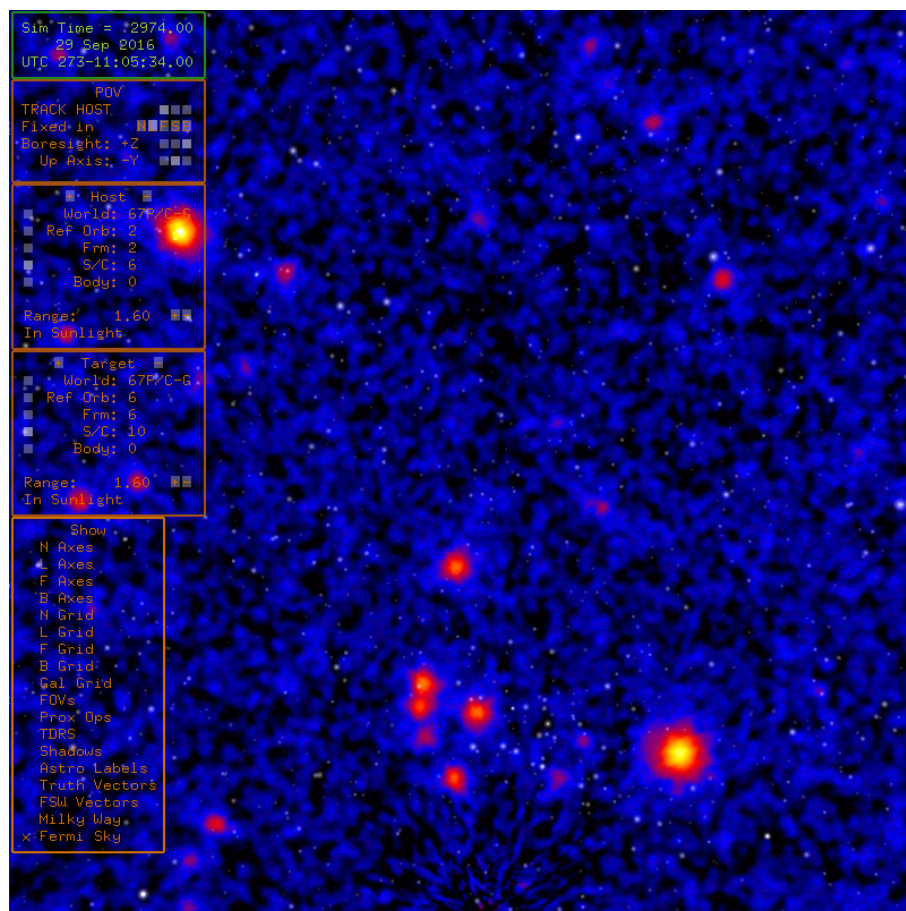


Figure A.16 42 Cam - Fermi sky



Figure A.17 42 Unit Sphere Viewer - Spacecraft vectors



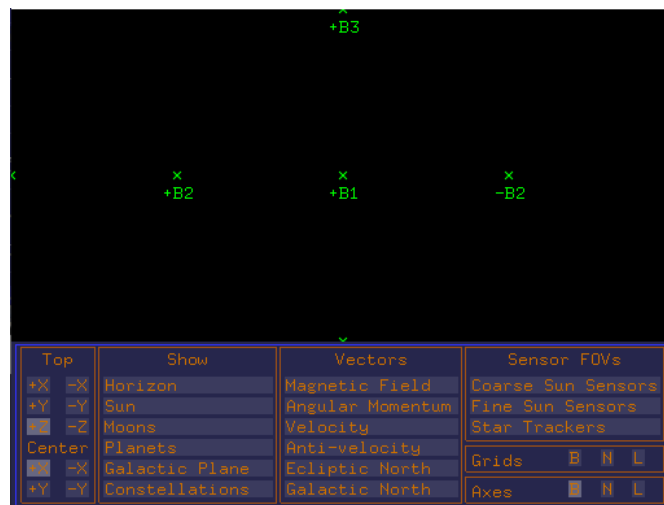


Figure A.18 42 Unit Sphere Viewer - Body frame

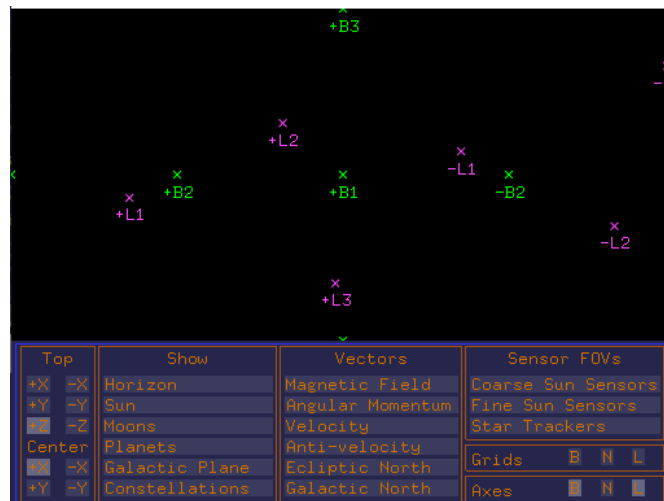


Figure A.19 42 Unit Sphere Viewer - Body and LHLV frames

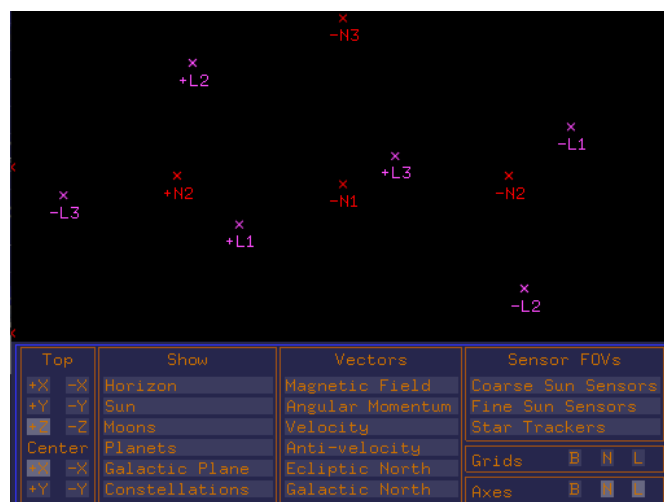


Figure A.20 42 Unit Sphere Viewer - Inertial and LHLV frames

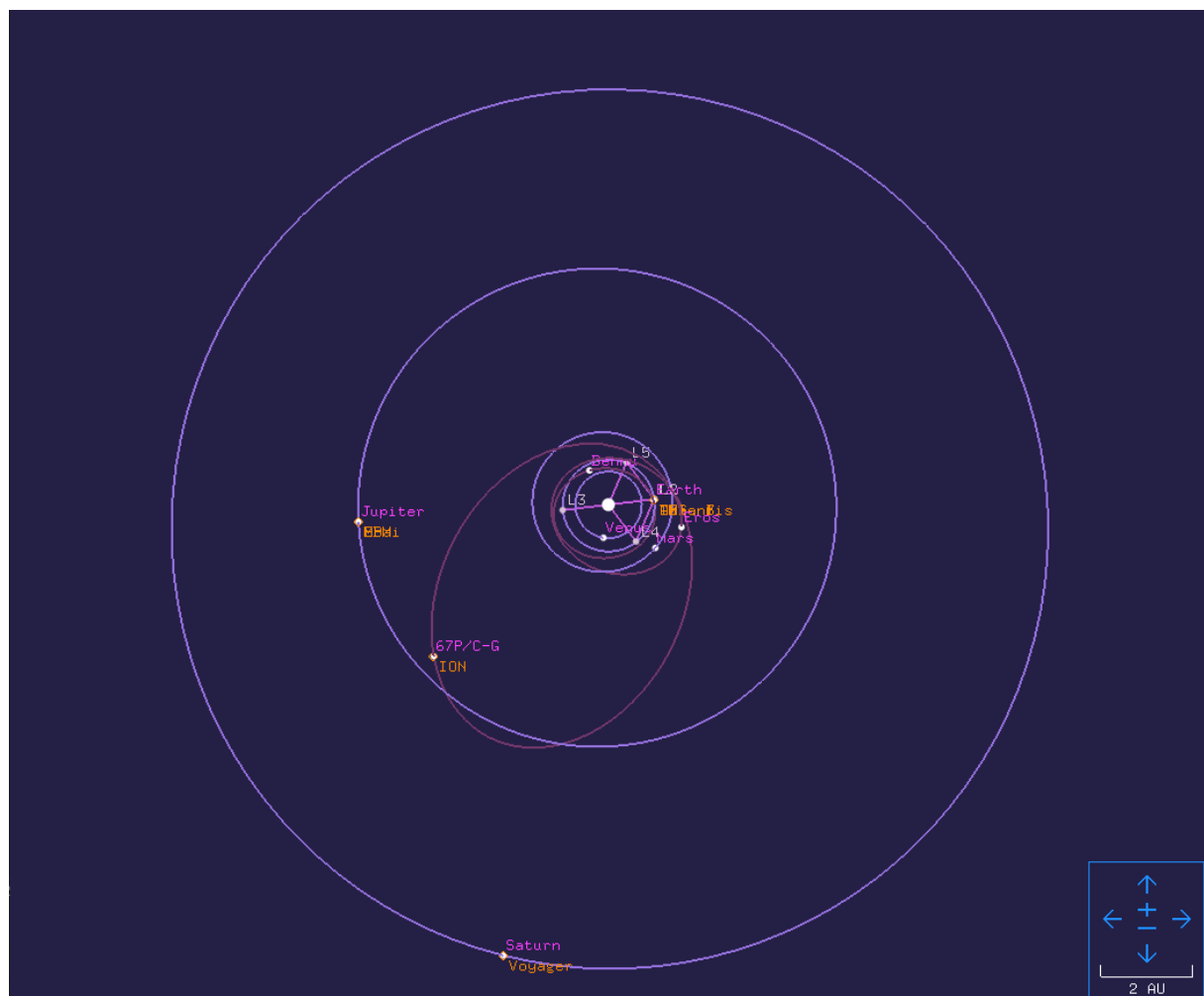
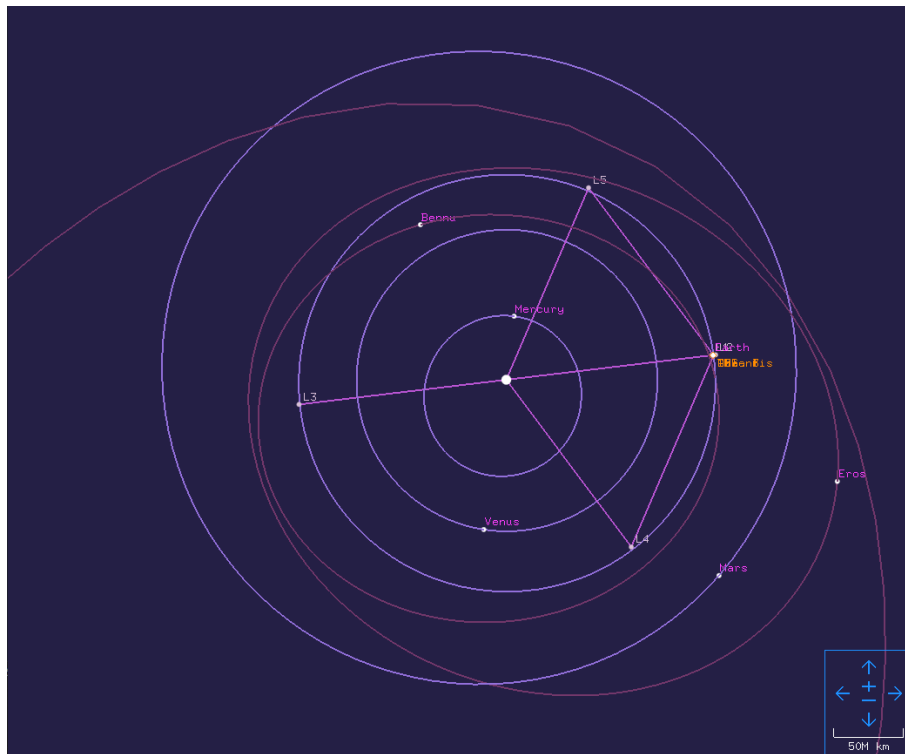
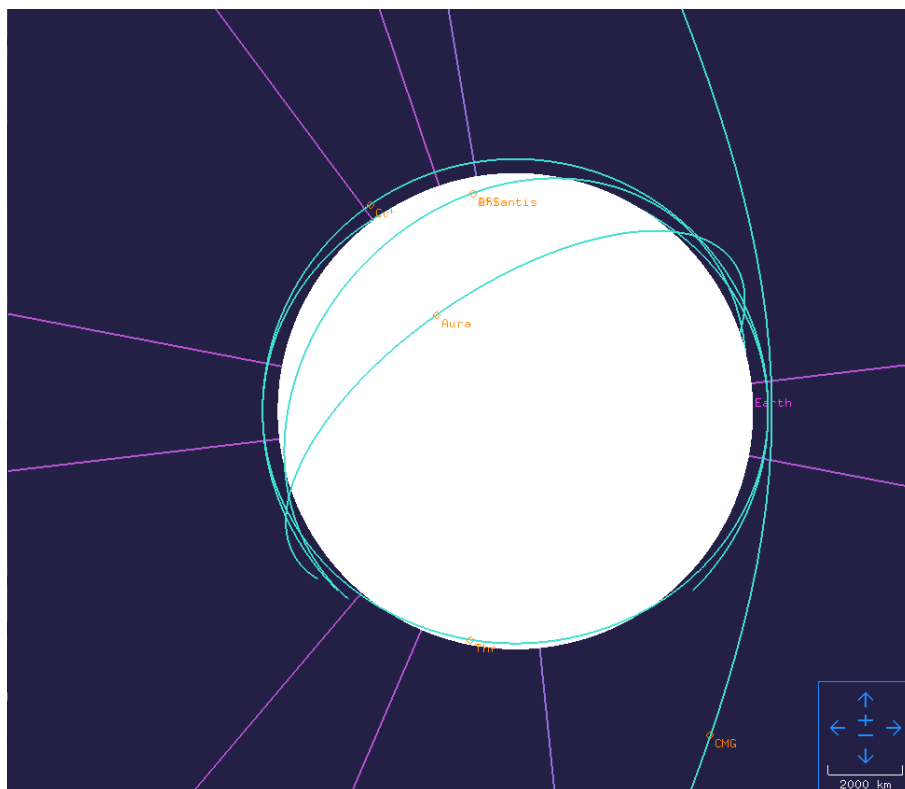


Figure A.21 42 Orrery - Solar System



**Figure A.22** 42 Orrery - Sun Earth Lagrange points



**Figure A.23** 42 Orrery - Earth orbits

## A.4 5th ESA CubeSat Industry Days

During the development of the thesis, I attended the 5th ESA CubeSat Industry Days in which Constellation and distributed system operations issues were discussed among space experts from all ESA countries.

Among the technologies that needed to mature for the deployment of large CubeSat constellations, experts highlighted

- Inter-satellite links, either RF or optical
- On-board autonomy for large satellite clusters
- Constellation simulator on ground (incorporating dynamic networks and unstructured constellations)
- Precise pointing

Then, the working group continued with a discussion about the possibility of CubeSats constellations sharing resources such as downlink bandwidth or computational time and the benefits that would have. As a result, it was stated that missions could be divided or distributed among nodes for specific functions. As most constellations of this type would be in LEO orbit, there would be a low latency for tasking and downloading payload data from any node to the ground network.

Later on, compared with huge institutional satellites, these constellations would be able to provide whole world coverage with low revisit time. An example was commented about CubeSats being used as scouting observation satellites.

As CubeSat development is more agile, there would be faster innovation cycles allowing higher capacity payloads to be introduced. In terms of Earth observation, these constellations could provide cheaper coverage with worse resolution for large scale quickly changing phenomena such as tornadoes.

Finally, CubeSat constellations would enable risk sharing at mission level and payload updating while requiring less time to develop in case of time-critical applications.

## Appendix B

# Algorithms

This appendix chapter includes code sections from the programmes mentioned throughout the thesis. The complete codes can be found on `Ivan-Sermanoukian/42PLATHON` GitHub repository [8].

The following sections can be found:

1. 42 Command Script
2. 42 Software (function analysis)
3. Algorithms

```

1 <<<<<<<<<<<<<<<< 42: Command Script File >>>>>>>>>>>>>>>>
2
3 # Commands are inserted here
4 # Commands are inserted here
5 # Commands are inserted here
6
7 EOF
8
9 #####
10 # All lines after EOF are ignored
11 # Comment lines begin with #, %, or //
12 # Blank lines are permitted
13
14 # Here are recognized command formats.
15 # %lf means that a floating-point number is expected
16 # %ld means that an integer is expected
17 # %s means that a string is expected
18 # %c means that a character is expected
19 # Look in functions SimCmdInterpreter, GuiCmdInterpreter,
20 #     and FswCmdInterpreter for strings and characters that
21 #     are meaningful in a particular context
22 # The first %lf is always the SimTime of command execution.
23
24 # Sim-related commands
25 %lf DTSIM = %lf
26 %lf SC[%ld].RotDOF %s
27 %lf SC[%ld].G[%ld].RotLocked[%ld] %s
28 %lf SC[%ld].G[%ld].TrnLocked[%ld] %s
29 %lf Impart Impulsive Delta-V of [%lf %lf %lf] m/s in Frame %c to Orb[%ld]
30     %c can be N or L
31 %lf SC[%ld].LoopGain = %lf
32 %lf SC[%ld].LoopDelay = %lf
33 %lf SC[%ld].GainAndDelayActive = %s
34
35 # GUI-related commands
36 %lf POV.Host.SC %ld
37 %lf CaptureCam %s
38 %lf CamSnap %s
39 %lf MapSnap %s
40 %lf Banner = "Banner in Quotes"
41 %lf GL Output Step = %lf
42 %lf POV CmdRange = %lf
43 %lf POV CmdSeq = %ld
44 %lf POV CmdAngle = [%lf %lf %lf] deg
45 %lf POV CmdPermute = [%lf %lf %lf; %lf %lf %lf; %lf %lf %lf]
46 %lf POV TimeToGo = %lf
47 %lf POV Frame = %c
48 %lf ShowHUD %s
49 %lf ShowWatermark %s
50 %lf ShowShadows %s
51 %lf ShowProxOps %s
52 %lf ShowFOV %s
53 %lf FOV[%ld].NearExists = %s
54 %lf FOV[%ld].FarExists = %s
55
56 # FSW-related commands
57 %lf SC[%ld] FswTag = %s
58     # %s is PASSIVE_FSW, PROTOTYPE_FSW, etc.
59 %lf SC[%ld] grn = [%lf %lf %lf %lf]
60 %lf SC[%ld] qrl = [%lf %lf %lf %lf]

```

```

61 %lf SC[%ld] Cmd Angles = [%lf %lf %lf] deg, Seq = %ld wrt %c Frame
62 # %c is either N or L
63 %lf SC[%ld].G[%ld] Cmd Angles = [%lf %lf %lf] deg
64 # In the following, the (first) %s is either "Primary" or "Secondary"
65 %lf Point SC[%ld].B[%ld] %s Vector [%lf %lf %lf] at RA = %lf deg, Dec = %lf deg
66 %lf Point SC[%ld].B[%ld] %s Vector [%lf %lf %lf] at World[%ld] Lng = %lf deg, Lat = %lf deg,
    Alt = %lf km
67 %lf Point SC[%ld].B[%ld] %s Vector [%lf %lf %lf] at World[%ld]
68 %lf Point SC[%ld].B[%ld] %s Vector [%lf %lf %lf] at GroundStation[%ld]
69 %lf Point SC[%ld].B[%ld] %s Vector [%lf %lf %lf] at %s
70 # Last %s is SUN, MOON, any planet, VELOCITY, or MAGFIELD
71 %lf Point SC[%ld].B[%ld] %s Vector [%lf %lf %lf] at SC[%ld]
72 %lf Point SC[%ld].B[%ld] %s Vector [%lf %lf %lf] at SC[%ld].B[%ld] point [%lf %lf %lf]
73 %lf Align SC[%ld].B[%ld] %s Vector [%lf %lf %lf] with %c-frame Vector [%lf %lf %lf]
74 # %c-frame can be H, N, or L
75 %lf Align SC[%ld].B[%ld] %s Vector [%lf %lf %lf] with SC[%ld].B[%ld] vector [%lf %lf %lf]
76 %lf SC[%ld].Thr[%ld] %s
77 %s is OFF or ON
78 Event Eclipse Entry SC[%ld] qrl = [%lf %lf %lf %lf]
79 Event Eclipse Exit SC[%ld] qrl = [%lf %lf %lf %lf]
80 Event Eclipse Entry SC[%ld] Cmd Angles = [%lf %lf %lf] deg, Seq = %ld wrt %c Frame
81 # %c is either N or L
82 Event Eclipse Exit SC[%ld] Cmd Angles = [%lf %lf %lf] deg, Seq = %ld wrt %c Frame
83 # %c is either N or L
84 %lf Set SC[%ld] RampCoastGlide wc = %lf Hz, amax = %lf, vmax = %lf
85 %lf Spin SC[%ld] about Primary Vector at %lf deg/sec

```

## B.2 42 Software

All low-level source codes are used as prototype libraries that will be called by upper-level functions. The library blocks can be split up into Kit and 42 libraries.

### B.2.1 Kit

Low-level toolkit functions are located in the Kit folder. These functions are subdivided into source code inside Kit/Source and function prototypes inside Kit/Include. Besides, there is a shaders' folder for the GUI based on OpenGL Shading Language.

Everything comes in and goes out through function arguments and return values without using global variables. Each source code has an intended purpose:

#### **dcmkit.c (Source) & dcmkit.h (Prototype)**

This function kit manages conversions between direction cosine matrices, quaternions and Euler angles. It computes rotations. For instance, there is a function that enables simple rotation of theta radians about a unit vector as explained in the quaternions section. Quaternion derivatives can be obtained with both Kane and Zimmerman methods.

In addition, this kit uses Huygens-Steiner theorem to determine the moment of inertia or the second moment of area of rigid bodies. For bodies with joint parts, there are also functions to obtain translations and rotations of selected pieces.

#### **envkit.c (Source) & envkit.h (Prototype)**

The environment kit contains gravity models and Polyhedron gravity estimates.

The current gravity models are only available for Earth, Mars and the Moon and are the following:

- Earth Gravity Model 96, published by NIMA as part of the WGS84
- Goddard Mars Model 2B. Truncated to 18x18
- Goddard Luna Gravity Model 2. Truncated to 18x18

It also includes complex magnetic field models such as the International Geomagnetic Reference Field (IGRF) or simple planetary dipole approximations.

Geomagnetic disturbances can be monitored by ground-based magnetic observatories which record the three magnetic field components. The global Kp index (*planetarische Kennziffer in German*) is obtained as the mean value of the disturbance levels in the two horizontal field components observed at 13 selected subauroral stations. The Kp index is in the scale  $[0_0, 0_+, 1_-, 1_0, 1_+, \dots, 8_+, 9_-, 9_0]$  but is mapped in the scale  $[0.0, 0.33, 0.67, 1.0, 1.33, \dots, 8.33, 8.67, 9.0]$  for ease of table lookup. A function is used to transform this index into the Ap index.

In addition, this kit includes atmospheric and geodetic models for Earth and Mars.

For Earth the JacchiaRoberts model is used within a range of validity from 120 km to 1000 km altitude. Another model called MSIS-86 is programmed with tabulated data and can be seen on the environmental introduction.

For Mars a simple model from Glenn Research Center is available [62].

There is a function to calculate Earth precession and nutation as well as one to transform geodetic to geocentric latitudes and vice versa.



### **fswkit.c (Source) & fswkit.h (Prototype)**

The Flight Software kit introduces a PD controller that is used by the CubeSat actuators.

The spin functions find the required spin rate, precession and nutation gain for an inertial spinner.

The TRIAD Attitude Determination function, given components of two vectors (V and W) defined in A frame and in B frame, find the direction cosine matrix CBA.

The QUEST attitude determination algorithm, given N reference vectors Ref(i) and the corresponding measurement vectors Meas(i) in the body, along with weights Weight(i), this routine finds the optimal estimate of q, the quaternion expressing the rotation from the reference frame (where References are given) to the body frame (where Measurements are taken). The FILTER-QUEST attitude determination algorithm is an enhanced version.

The PointGimbalToTarget function finds the Euler Angles to point a given boresight vector fixed in a joint's outer body parallel to a target vector fixed in the joint's inner body.

The SolarBeta function provides the angle between the Sun vector and the orbit plane. It is positive toward the positive orbit normal.

Then the tableau functions create the Runge-Kutta tableau and proceed with Gauss elimination.

A set of functions implement a "sequential" Kalman Filter, with measurement updates having 1, 2, or 3 dimensions.

### **geomkit.c (Source) & geomkit.h (Prototype)**

The geometry kit contains information about Polyhedron volumes, surfaces and materials. There are functions to load the spacecraft tree structure and write the geometry to an object file.

Solar pressure forces are computed with the direction of the incident rays.

### **glkit.c (Source) & glkit.h (Prototype)**

The OpenGL Utility Toolkit (GLUT) is a library of utilities for OpenGL programmes, which primarily performs system-level I/O with the host operating system.

### **gmseckit.c (Source) & gmseckit.h (Prototype)**

The Goddard Mission Services Evolution Center (GMSEC) kit provides the connectivity with the core features of the programme with standardised messages and formats. The implementation of this programme is out of the scope of the project.

### **iokit.c (Source) & iokit.h (Prototype)**

The Input/Output kit creates several functions to read input data from a file and convert file data to strings with formatted text.

The functions to initiate socket server and client communication are declared inside this kit.

### **mathkit.c (Source) & mathkit.h (Prototype)**

The mathematics kit creates function for general mathematics calculations that are consistently called throughout the code compilation.

### **msis86kit.c (Source) & msys86kit.h (Prototype)**

This kit includes the MSIS-86/CIRA 1986 Neutral Thermosphere Model which describes the neutral temperature and densities in the upper atmosphere above the Kármán line. The model has been plotted on the theoretical concepts chapter.

### **nrlmsise00kit.c (Source)**

This kit includes the NRLMSISE-00 Atmosphere Model, an empirical, global reference atmospheric model of the Earth from ground to space. More information can be found on the development folder.

### **orbkit.c (Source) & orbkit.h (Prototype)**

The orbital kit defines the orbit types and parameters. As the input file allows different kinds of orbit initialisation, transformations between position and velocity vectors, TLE and ephemerides among other can be found on this file.

The function to read TLE files and save the pertinent data is `LoadTleFromFile`.

Planet and Moons ephemerides, their radius of influence and Lagrange system values are obtained from this file. Then, the Lambert problem and a two-impulse rendezvous are also available. The J2 drift of an orbit is computed here.

### **radbeltkit.c (Source)**

This kit contains NASA's main radiation modelling program. However, it is still under testing validation.

### **sigkit.c (Source) & sigkit.h (Prototype)**

This kit contains a random number generator in conjunction with filter creation and destruction functions.

### **texkit.c (Source) & texkit.h (Prototype)**

The texture kit contains procedural Texturing functions for the graphical representation.

### **timekit.c (Source) & timekit.h (Prototype)**

The time kit contains all the time functions and conversions between real system time, real run time and J2000 as a reference epoch.

## B.2.2 Include

The Include folder contains the 42 propagator libraries defined as function prototypes:

### **42.h**

The 42 main library calls almost all Kit prototypes as well as 42 types and definitions. Then, defines the EXTERN variables and function prototypes that will be used by the Source code.

### **42defines.h**

This prototype contains all the programme definitions. Strings are transformed into numbers for property selection.

### **42fsw.h**

This prototype contains the Flight Software function.

### **42glfwgui.h & 42GluGui**

These prototypes contain EXTERN variables and function prototypes for the creation of the Graphical User Interface.

### **42types.h**

This prototype contains all the struct types created for 42.

### **Ac.h**

This prototype contains the external Flight Software attitude control function.

### **AcTypes.h**

This prototype contains all the struct types created for the external Flight Software attitude control function.

### **fswdefines.h**

This prototype contains all the Flight Software definitions.

### **fswtypes.h**

This prototype contains all the struct types created for the external Flight Software function.

### B.2.3 Source

The Source folder contains the upper-level functions which use the aforementioned kits and libraries.

The IPC subfolder contains the Inter-Process Communication functions. It defines the simulation reading and writing through socket interfaces from the configuration on the IPC file.

#### **42actuators.c**

This file contains all the actuator model functions.

#### **42cmd.c**

This file contains the functions in charge of reading and interpreting the orders initially stated on the Input Command file.

#### **42dynamics.c**

This file contains the function to obtain the dynamic and kinematic properties of the spacecraft. ODE integration functions are also defined here.

#### **42environs.c**

This file contains a function that summarises all the environment effects on a specific spacecraft.

#### **42ephem.c**

This file contains the functions that define the orbit motion and ephemerides. Spacecraft are assigned to their orbits.

#### **42exec.c**

This file contains the execution of 42 propagator in conjunction with time advancement, flag management and report progress functions.

The execution function starts reading data from the Input Simulation text file. Once all the items are set up, the following step is to read the Cmd commands and initiate the Inter-Process Communication, if available.

Once the initial conditions and commands are verified, the program derives the data to the GUI if it is activated. Finally, a while loop keeps advancing the simulation step until it reaches the pre-established waypoint.

#### **42fsw.c**

This file contains all the flight software functions.

#### **42glfwgui.c & 42GlutGui.c**

These files contain all the required function for the creation of the Graphical User Interface.

#### **42init.c**

This file reads the input files, decodes the strings and initiates the orbits and spacecraft. Then, the functions load the celestial bodies. The `InitSim` function set the reading, initiating and loading order.

#### **42ipc.c**

This file contains all the Inter-Process Communication functions. It is in charge of the IPC initiation and bidirectional communication.

#### **42main.c**

This file contains a minimal main function to facilitate compiling 42 as a library to be called from other applications. This main serves as the entry point for the standalone application. It calls 42's top-level executive function, `exec`.

#### **42nos3.c**

This file contains the function to communicate with NASA's Operational Simulator for Small Satellites (NOS3).

#### **42perturb.c**

This file contains all the perturbations that 42 can activate.

#### **42report.c**

This file is in charge of saving the variables on text files for post-processing.

#### **42sensors.c**

This file contains all the sensor model functions.

#### **AcApp.c**

This file contains all the functions that define the external Attitude Control application.

## B.3 Algorithms

The complete codes can be found on Ivan-Sermanoukian/42PLATHON GitHub repository [8]. The following codes correspond to functions and code extracts that have been changed.

### B.3.1 Report function

```
1 void Report(void)
2 {
3     static FILE *timefile,*DynTimeFile;
4     static FILE **xfile, **ufile, **xffile, **uffile;
5     static FILE **ConstraintFile;
6     static FILE **PosNfile,**VelNfile,**qbnfile,**wbnfile;
7     static FILE **PosWfile,**VelWfile;
8     static FILE **PosRfile,**VelRfile;
9     static FILE **Hvnfile,**KEfile;
10    static FILE **svnfile,**svbfile;
11    static FILE **RPYfile;
12    static FILE **Hwhlfile;
13    static FILE **MTBfile;
14    // static FILE **ProjAreaFile;
15    static FILE **AccFile;
16    static char First = TRUE;
17    long Isc,i;
18    struct DynType *D;
19    double CBL[3][3],Roll,Pitch,Yaw;
20    double PosW[3],VelW[3],PosR[3],VelR[3];
21    char s[40];
22    //double ZAxis[3] = {0.0,0.0,1.0};
23
24    if (First) {
25        First = FALSE;
26        timefile = FileOpen(InOutPath, "time.42", "w");
27        DynTimeFile = FileOpen(InOutPath, "DynTime.42", "w");
28
29        ufile = (FILE **) calloc(Nsc,sizeof(FILE *));
30        xfile = (FILE **) calloc(Nsc,sizeof(FILE *));
31        uffile = (FILE **) calloc(Nsc,sizeof(FILE *));
32        xffile = (FILE **) calloc(Nsc,sizeof(FILE *));
33        ConstraintFile = (FILE **) calloc(Nsc,sizeof(FILE *));
34
35        // Memory allocation for new variables
36        PosNfile = (FILE **) calloc(Nsc,sizeof(FILE *));
37        VelNfile = (FILE **) calloc(Nsc,sizeof(FILE *));
38        PosWfile = (FILE **) calloc(Nsc,sizeof(FILE *));
39        VelWfile = (FILE **) calloc(Nsc,sizeof(FILE *));
40        PosRfile = (FILE **) calloc(Nsc,sizeof(FILE *));
41        VelRfile = (FILE **) calloc(Nsc,sizeof(FILE *));
42        qbnfile = (FILE **) calloc(Nsc,sizeof(FILE *));
43        wbnfile = (FILE **) calloc(Nsc,sizeof(FILE *));
44        Hvnfile = (FILE **) calloc(Nsc,sizeof(FILE *));
45        svnfile = (FILE **) calloc(Nsc,sizeof(FILE *));
46        svbfile = (FILE **) calloc(Nsc,sizeof(FILE *));
47        KEfile = (FILE **) calloc(Nsc,sizeof(FILE *));
48        RPYfile = (FILE **) calloc(Nsc,sizeof(FILE *));
49        Hwhlfile = (FILE **) calloc(Nsc,sizeof(FILE *));
50        MTBfile = (FILE **) calloc(Nsc,sizeof(FILE *));
51        // ProjAreaFile = (FILE **) calloc(Nsc,sizeof(FILE *));
52        AccFile = (FILE **) calloc(Nsc,sizeof(FILE *));
53    }
```

```

54     for(Isc=0;Isc<Nsc;Isc++) {
55         if (SC[Isc].Exists) {
56             sprintf(s,"u%021d.42",Isc);
57             ufile[Isc] = FileOpen(InOutPath,s,"w");
58             sprintf(s,"x%021d.42",Isc);
59             xfile[Isc] = FileOpen(InOutPath,s,"w");
60             if (SC[Isc].FlexActive) {
61                 sprintf(s,"uf%021d.42",Isc);
62                 ufile[Isc] = FileOpen(InOutPath,s,"w");
63                 sprintf(s,"xf%021d.42",Isc);
64                 xfile[Isc] = FileOpen(InOutPath,s,"w");
65             }
66             if (SC[Isc].ConstraintsRequested) {
67                 sprintf(s,"Constraint%021d.42",Isc);
68                 ConstraintFile[Isc] = FileOpen(InOutPath,s,"w");
69             }
70         }
71     }
72     // Create files for new data variables
73     for(Isc=0;Isc<Nsc;Isc++) {
74         sprintf(s,"PosN%021d.42",Isc);
75         PosNfile[Isc] = FileOpen(InOutPath,s,"w");
76         sprintf(s,"VelN%021d.42",Isc);
77         VelNfile[Isc] = FileOpen(InOutPath,s,"w");
78         sprintf(s,"PosW%021d.42",Isc);
79         PosWfile[Isc] = FileOpen(InOutPath,s,"w");
80         sprintf(s,"VelW%021d.42",Isc);
81         VelWfile[Isc] = FileOpen(InOutPath,s,"w");
82         sprintf(s,"PosR%021d.42",Isc);
83         PosRfile[Isc] = FileOpen(InOutPath,s,"w");
84         sprintf(s,"VelR%021d.42",Isc);
85         VelRfile[Isc] = FileOpen(InOutPath,s,"w");
86         sprintf(s,"qbn%021d.42",Isc);
87         qbnfile[Isc] = FileOpen(InOutPath,s,"w");
88         sprintf(s,"wbn%021d.42",Isc);
89         wbnfile[Isc] = FileOpen(InOutPath,s,"w");
90         sprintf(s,"Hvn%021d.42",Isc);
91         Hvnfile[Isc] = FileOpen(InOutPath,s,"w");
92         sprintf(s,"svn%021d.42",Isc);
93         svnfile[Isc] = FileOpen(InOutPath,s,"w");
94         sprintf(s,"svb%021d.42",Isc);
95         svbfile[Isc] = FileOpen(InOutPath,s,"w");
96         sprintf(s,"KE%021d.42",Isc);
97         KEfile[Isc] = FileOpen(InOutPath,s,"w");
98         sprintf(s,"RPY%021d.42",Isc);
99         RPYfile[Isc] = FileOpen(InOutPath,s,"w");
100        sprintf(s,"Hwhl%021d.42",Isc);
101        Hwhlfile[Isc] = FileOpen(InOutPath,s,"w");
102        sprintf(s,"MTB%021d.42",Isc);
103        MTBfile[Isc] = FileOpen(InOutPath,s,"w");
104        // sprintf(s,"ProjArea%021d.42",Isc);
105        // ProjAreaFile = FileOpen(InOutPath,s,"w");
106        sprintf(s,"Acc%021d.42",Isc);
107        AccFile[Isc] = FileOpen(InOutPath,s,"w");
108    }
109
110 }
111
112 if (OutFlag) {
113     fprintf(timefile,"%1f\n",SimTime);
114     fprintf(DynTimeFile,"%1f\n",DynTime);
115     for(Isc=0;Isc<Nsc;Isc++) {
116         if (SC[Isc].Exists) {
117             D = &SC[Isc].Dyn;

```

```

118     for(i=0;i<D->Nu;i++) fprintf(ufile[Isc], "%1e ", D->u[i]);
119     fprintf(ufile[Isc], "\n");
120     for(i=0;i<D->Nx;i++) fprintf(xfile[Isc], "%1e ", D->x[i]);
121     fprintf(xfile[Isc], "\n");
122     if (SC[Isc].FlexActive) {
123         for(i=0;i<D->Nf;i++) fprintf(ufile[Isc], "%1e ", D->uf[i]);
124         fprintf(ufile[Isc], "\n");
125         for(i=0;i<D->Nf;i++) fprintf(xffile[Isc], "%1e ", D->xf[i]);
126         fprintf(xffile[Isc], "\n");
127     }
128     if (SC[Isc].ConstraintsRequested) {
129         for(i=0;i<D->Nc;i++)
130             fprintf(ConstraintFile[Isc], "%1e ",
131                 D->GenConstraintFrc[i]);
132         fprintf(ConstraintFile[Isc], "\n");
133     }
134
135     //Save data inside new variables
136     fprintf(PosNfile[Isc], "%1e %1e %1e\n",
137         SC[Isc].PosN[0], SC[Isc].PosN[1], SC[Isc].PosN[2]);
138     fprintf(VelNfile[Isc], "%1e %1e %1e\n",
139         SC[Isc].VelN[0], SC[Isc].VelN[1], SC[Isc].VelN[2]);
140     MxV(World[EARTH].CWN, SC[Isc].PosN, PosW);
141     MxV(World[EARTH].CWN, SC[Isc].VelN, VelW);
142     fprintf(PosWfile[Isc], "%18.121e %18.121e %18.121e\n",
143         PosW[0], PosW[1], PosW[2]);
144     fprintf(VelWfile[Isc], "%18.121e %18.121e %18.121e\n",
145         VelW[0], VelW[1], VelW[2]);
146
147     if (Orb[SC[Isc].RefOrb].Regime == ORB_FLIGHT) {
148         MxV(Rgn[Orb[SC[Isc].RefOrb].Region].CN, SC[Isc].PosR, PosR);
149         MxV(Rgn[Orb[SC[Isc].RefOrb].Region].CN, SC[Isc].VelR, VelR);
150         fprintf(PosRfile[Isc], "%1e %1e %1e\n",
151             PosR[0], PosR[1], PosR[2]);
152         fprintf(VelRfile[Isc], "%1e %1e %1e\n",
153             VelR[0], VelR[1], VelR[2]);
154     }
155     else {
156         fprintf(PosRfile[Isc], "%1e %1e %1e\n",
157             SC[Isc].PosR[0], SC[Isc].PosR[1], SC[Isc].PosR[2]);
158         fprintf(VelRfile[Isc], "%1e %1e %1e\n",
159             SC[Isc].VelR[0], SC[Isc].VelR[1], SC[Isc].VelR[2]);
160     }
161
162     fprintf(qbnfile[Isc], "%1e %1e %1e %1e\n",
163         SC[Isc].B[0].qn[0], SC[Isc].B[0].qn[1], SC[Isc].B[0].qn[2], SC[Isc].B[0].qn[3])
164 ;
165     fprintf(wbnfile[Isc], "%1e %1e %1e\n",
166         SC[Isc].B[0].wn[0], SC[Isc].B[0].wn[1], SC[Isc].B[0].wn[2]);
167     fprintf(Hvnfile[Isc], "%18.121e %18.121e %18.121e\n",
168         SC[Isc].Hvn[0], SC[Isc].Hvn[1], SC[Isc].Hvn[2]);
169     fprintf(svnfile[Isc], "%18.121e %18.121e %18.121e\n",
170         SC[Isc].svn[0], SC[Isc].svn[1], SC[Isc].svn[2]);
171     fprintf(svbfile[Isc], "%18.121e %18.121e %18.121e\n",
172         SC[Isc].svb[0], SC[Isc].svb[1], SC[Isc].svb[2]);
173     fprintf(KEfile[Isc], "%18.121e\n", FindTotalKineticEnergy(&SC[Isc]));
174     MxMT(SC[Isc].B[0].CN, SC[Isc].CLN, CBL);
175     C2A(123, CBL, &Roll, &Pitch, &Yaw);
176     fprintf(RPYfile[Isc], "%1f %1f %1f\n", Roll*R2D, Pitch*R2D, Yaw*R2D);
177     for(i=0;i<SC[Isc].Nw;i++) fprintf(Hwhlfile[Isc], "%1f ", SC[Isc].Whl[i].H);
178     fprintf(Hwhlfile[Isc], "\n");
179     if (SC[Isc].Nmtb > 0) {
180         for(i=0;i<SC[Isc].Nmtb;i++) fprintf(MTBfile[Isc], "%1f ", SC[Isc].MTB[i].M);
181         fprintf(MTBfile[Isc], "\n");

```



```

181         }
182         //fprintf(ProjAreaFile[Isc], "%18.12le %18.12le\n",
183         //    FindTotalProjectedArea(&SC[Isc], ZAxis),
184         //    FindTotalUnshadedProjectedArea(&SC[Isc], ZAxis));
185         if (SC[Isc].Nacc > 0) {
186             for(i=0; i<SC[Isc].Nacc; i++)
187                 fprintf(AccFile[Isc], "%le %le ", SC[Isc].Accel[i].TrueAcc, SC[Isc].Accel[i
188         ].MeasAcc);
189             fprintf(AccFile[Isc], "\n");
190         }
191     }
192     // if (SC[0].Exists) {
193     //
194     //     //MagReport();
195     //     //GyroReport();
196     //
197     // }
198
199 }
200
201 /* An example how to call specialized reporting based on sim case */
202 /* if (!strcmp(InOutPath, "./Potato/")) PotatoReport(); */
203
204
205 if (CleanUpFlag) {
206     fclose(timefile);
207 }
208
209 }

```

**Algorithm B.1** Report

### B.3.2 Constellation Input File

```
1 %-----%
2 % Input_main: Creation of 42's constellation input files (.txt)
3 %-----%
4
5 %{
6     Date: 13/06/2021
7     Author: Ivan Sermanoukian Molina
8     Title: Study on orbital propagators: Constellation analysis with NASA 42
9           and Matlab/Simulink
10    Director: David Gonzalez Diez
11 %}
12
13 % Clear workspace, command window and close windows
14 clc
15 clear all
16 close all
17
18 %% Operating System selection
19
20 % Choose simulation folder
21 folder = "Test_Iridium_Zero";
22
23 % Windows data path
24 if ispc
25     display("Windows Path");
26     linux = false;
27     % Add function paths
28     addpath(strcat(pwd, filesep, 'Input_functions_Constellation'));
29
30     % Create folder
31     if not(isfolder(folder))
32         mkdir(folder);
33     else
34         rmdir(folder, 's')
35         mkdir(folder);
36     end
37
38 % Linux data path
39 else
40     display("Linux Path");
41     linux = true;
42     % Add function paths
43     addpath(strcat(pwd, filesep, 'Input_functions'));
44
45     % Create folder
46     if not(isfolder(folder))
47         mkdir(folder);
48     else
49         rmdir(folder, 's')
```

```

50         mkdir(folder);
51     end
52 end
53
54
55 %% Input data
56
57 % Input Command Script
58
59     % User manual
60     User_manual = "TRUE";           % T/F
61
62 % Input Simulation Control
63
64     % Time variables
65     time_mode      = "FAST";         % FAST, REAL, EXTERNAL, or NOS3
66     duration       = "10000";        % s
67     step_size      = "0.1";          % s
68     output_interval = "1";           % s
69
70     % Initial time
71     month          = "06";           % mm
72     day            = "14";           % dd
73     year           = "2021";         % YYYY
74     hour           = "10";           % hh
75     minute         = "00";           % mm
76     second         = "00.00";        % ss.ss
77
78     % Graphical User Interface
79     GUI            = "TRUE";         % T/F
80
81     % Orbits
82     reference_orbits = "75";
83     [orbits] = Input_Orbit(folder,reference_orbits);
84
85     % Spacecraft
86     number_spacecraft = "75";
87     [spacecraft] = Input_SC(folder,number_spacecraft);
88
89     % Ground Stations
90     number_ground_stations = "1";
91     % Exists, World, Lng, Lat, Label
92     ground_stations = ["TRUE","EARTH","-77.0","37.0","""GSFC"""];
93
94     % Select Solar System variables
95     % Me - V - E - Ma - J - S - U - N - P - Ast
96     Solar_System = ["TRUE","TRUE","TRUE","TRUE","TRUE","TRUE","TRUE","TRUE",
97     "","TRUE","TRUE"];
98     % Earth-Moon / Sun-Earth / Sun-Jupiter
99     Lagrange_System = ["FALSE","FALSE","FALSE"];

```

```
100 %% Create files
101
102 % List of commands
103 Input_Command_Script(folder,User_manual);
104
105 % Graphical User Interface and FOV seetings
106 if (GUI == "TRUE")
107     Input_Graphics(folder);
108     Input_FOV(folder);
109 end
110
111 % Tracking and Data Relay Satellite System
112 Input_TDRS(folder);
113 % World Regions
114 Input_Region(folder);
115 % Intercommunications
116 Input_IPC(folder)
117
118 % Simulation settings
119 Input_Simulation(folder,time_mode,duration,step_size,output_interval,GUI
    ,...
120     month,day,year,hour,minute,second,reference_orbits,orbits,
    ...
121     number_spacecraft,spacecraft,number_ground_stations,
    ground_stations,...
122     Solar_System,Lagrange_System);
123
124
125 if (linux)
126
127     % Run 42
128     cd('/home/ivan/plathon/42');
129     system('./42 Sims/Test_Iridium_Zero');
130
131 end
```

**Algorithm B.2** Constellation Input File

### B.3.3 Constellation Output File

The commented lines correspond to results that require code activation to be saved and are generally not used.

```
1 %-----%
2 % Output_main: Analysis of 42's output data
3 %-----%
4
5 %{
6     Date: 14/04/2021
7     Author: Ivan Sermanoukian Molina
8     Title: Study on orbital propagators: Constellation analysis with NASA 42
9           and Matlab/Simulink
10    Director: David Gonzalez Diez
11 %}
12
13 % Clear workspace, command window and close windows
14 clc
15 clear all
16 close all
17
18 % LaTeX configuration
19 set(groot,'defaultAxesTickLabelInterpreter','latex');
20 set(groot,'defaultTextInterpreter','latex');
21 set(groot,'defaultLegendInterpreter','latex');
22
23 %% Operating System selection
24
25 % Choose simulation folder
26 data_folder = "data";
27 mission_folder = "Test_Iridium_Zero";
28 folder = strcat(data_folder,filesep,mission_folder);
29
30 % Windows data path
31 if ispc
32     display("Windows Path");
33     linux = false;
34     % Add simulation paths
35     addpath(strcat(pwd,filesep,folder));
36     addpath(genpath(strcat(pwd,filesep,'Output_results')));
37
38 % Linux data path
39 else
40     display("Linux Path");
41     linux = true;
42     % Add simulation paths
43     addpath(strcat(pwd,filesep,folder));
44     addpath(genpath(strcat(pwd,filesep,'Output_results')));
45
46 end
47
```

```

48 %% Initial conditions
49
50 Nsc = 75;
51
52 %% Output data
53
54 % Simulation time [s]
55 sim_time = load(strcat(folder,filesep,'time.42'),' -ascii');
56 % Simulation time since J2000 [s]
57 sim_time_J2000 = load(strcat(folder,filesep,'DynTime.42'),' -ascii');
58
59 for Isc = 0:1:(Nsc-1)
60
61     str = sprintf("u%02ld.42",Isc);
62     u(:, :, Isc+1) = load(strcat(folder,filesep,str),' -ascii');
63     str = sprintf("x%02ld.42",Isc);
64     x(:, :, Isc+1) = load(strcat(folder,filesep,str),' -ascii');
65     % str = sprintf("uf%02ld.42",Isc);
66     % uf(:, :, Isc+1) = load(strcat(folder,filesep,str),' -ascii');
67     % str = sprintf("xf%02ld.42",Isc);
68     % xf(:, :, Isc+1) = load(strcat(folder,filesep,str),' -ascii');
69     % str = sprintf("Constraint%02ld.42",Isc);
70     % Constraint(:, :, Isc+1) = load(strcat(folder,filesep,str),' -ascii');
71
72     str = sprintf("PosN%02ld.42",Isc);
73     PosN(:, :, Isc+1) = load(strcat(folder,filesep,str),' -ascii');
74     str = sprintf("VelN%02ld.42",Isc);
75     VelN(:, :, Isc+1) = load(strcat(folder,filesep,str),' -ascii');
76     str = sprintf("PosW%02ld.42",Isc);
77     PosW(:, :, Isc+1) = load(strcat(folder,filesep,str),' -ascii');
78     str = sprintf("VelW%02ld.42",Isc);
79     VelW(:, :, Isc+1) = load(strcat(folder,filesep,str),' -ascii');
80     str = sprintf("PosR%02ld.42",Isc);
81     PosR(:, :, Isc+1) = load(strcat(folder,filesep,str),' -ascii');
82     str = sprintf("VelR%02ld.42",Isc);
83     VelR(:, :, Isc+1) = load(strcat(folder,filesep,str),' -ascii');
84     str = sprintf("qbn%02ld.42",Isc);
85     qbn(:, :, Isc+1) = load(strcat(folder,filesep,str),' -ascii');
86     str = sprintf("wbn%02ld.42",Isc);
87     wbn(:, :, Isc+1) = load(strcat(folder,filesep,str),' -ascii');
88     str = sprintf("Hvn%02ld.42",Isc);
89     Hvn(:, :, Isc+1) = load(strcat(folder,filesep,str),' -ascii');
90     str = sprintf("svn%02ld.42",Isc);
91     svn(:, :, Isc+1) = load(strcat(folder,filesep,str),' -ascii');
92     str = sprintf("svb%02ld.42",Isc);
93     svb(:, :, Isc+1) = load(strcat(folder,filesep,str),' -ascii');
94     str = sprintf("KE%02ld.42",Isc);
95     KE(:, :, Isc+1) = load(strcat(folder,filesep,str),' -ascii');
96     str = sprintf("RPY%02ld.42",Isc);
97     RPY(:, :, Isc+1) = load(strcat(folder,filesep,str),' -ascii');
98     str = sprintf("Hwhl%02ld.42",Isc);

```

```

99     Hwhl(:, :, Isc+1) = load(strcat(folder, filesep, str), '-ascii');
100     str = sprintf("MTB%02ld.42", Isc);
101     MTB(:, :, Isc+1) = load(strcat(folder, filesep, str), '-ascii');
102 %     str = sprintf("ProjArea%02ld.42", Isc);
103 %     ProjArea(:, :, Isc+1) = load(strcat(folder, filesep, str), '-ascii');
104     str = sprintf("Acc%02ld.42", Isc);
105     Acc(:, :, Isc+1) = load(strcat(folder, filesep, str), '-ascii');
106 end

```

**Algorithm B.3** Constellation Output File

### B.3.4 TLE acquisition

```

1 function [sat_id] = TLE_to_42(folder)
2
3 % TLE Example
4
5 % ISS (ZARYA)
6 % 1 25544U 98067A 08264.51782528 -.00002182 00000-0 -11606-4 0 2927
7 % 2 25544 51.6416 247.4627 0006703 130.5360 325.0288 15.72125391563537
8
9 url = 'https://www.celestrak.com/NORAD/elements/iridium-NEXT.txt';
10 data_IRIDIUM_NEXT = webread(url);
11 data = splitlines(data_IRIDIUM_NEXT);
12
13 name = "TLE_IRIDIUM_NEXT";
14 name_txt = strcat(name, ".txt");
15 directory = strcat(folder, filesep, name_txt);
16 % directory = strcat(name_txt);
17 writecell(data, directory)
18
19 sat_id = strings(1, (length(data)-1)/3);
20 % epoch(1) = data()
21
22 for i = 1:1:((length(data)-1)/3)
23     sat_id(i) = deblank(string(data(3*i-2)));
24 end
25 end

```

**Algorithm B.4** TLE acquisition





### B.3.5.3 AcApp modifications

The modified AcApp.c file functions are the following:

```
void AllocateAC(struct AcType *AC)
{
    /* Bodies */
    AC->Nb = 1;
    if (AC->Nb > 0) {
        AC->B = (struct AcBodyType *) calloc(AC->Nb, sizeof(struct AcBodyType));
    }

    /* Joints */
    AC->Ng = 1;
    if (AC->Ng > 0) {
        AC->G = (struct AcJointType *) calloc(AC->Ng, sizeof(struct AcJointType));
    }

    /* Wheels */
    AC->Nwhl = 3;
    if (AC->Nwhl > 0) {
        AC->Whl = (struct AcWhlType *) calloc(AC->Nwhl, sizeof(struct AcWhlType));
    }

    /* Magnetic Torquer Bars */
    AC->Nmtb = 3;
    if (AC->Nmtb > 0) {
        AC->MTB = (struct AcMtbType *) calloc(AC->Nmtb, sizeof(struct AcMtbType));
    }

    /* Thrusters */
    AC->Nthr = 0;
    if (AC->Nthr > 0) {
        AC->Thr = (struct AcThrType *) calloc(AC->Nthr, sizeof(struct AcThrType));
    }

    /* Control Moment Gyros */

    /* Gyro Axes */
    AC->Ngyro = 3;
    if (AC->Ngyro > 0) {
        AC->Gyro = (struct AcGyroType *) calloc(AC->Ngyro, sizeof(struct AcGyroType));
    }

    /* Magnetometer Axes */
    AC->Nmag = 3;
    if (AC->Nmag > 0) {
        AC->MAG = (struct AcMagnetometerType *) calloc(AC->Nmag, sizeof(struct
AcMagnetometerType));
    }

    /* Coarse Sun Sensors */
    AC->Ncss = 3;
    if (AC->Ncss > 0) {
        AC->CSS = (struct AcCssType *) calloc(AC->Ncss, sizeof(struct AcCssType));
    }

    /* Fine Sun Sensors */
    AC->Nfss = 0;
    if (AC->Nfss > 0) {
        AC->FSS = (struct AcFssType *) calloc(AC->Nfss, sizeof(struct AcFssType));
    }
}
```

```

/* Star Trackers */
AC->Nst = 3;
if (AC->Nst > 0) {
    AC->ST = (struct AcStarTrackerType *) calloc(AC->Nst, sizeof(struct AcStarTrackerType)
);
}

/* GPS */
AC->Ngps = 1;
if (AC->Ngps > 0) {
    AC->GPS = (struct AcGpsType *) calloc(AC->Ngps, sizeof(struct AcGpsType));
}

/* Accelerometer Axes */
}

```

**Algorithm B.5** AllocateAC function

```

void SetPoint(struct AcType *AC) {

    double C[3][3];

    if (AC->FirstTime == 0) {
        AC->FirstTime = 1;
        AC->initTime = AC->Time;
    }

    if ((AC->Time - AC->initTime) < 100) {
        AC->Cmd.Ang[0] = 0;
        AC->Cmd.Ang[1] = 0;
        AC->Cmd.Ang[2] = 0;
        AC->Cmd.RotSeq = 321;
    }
    // At t= 100 change change Euler angles
    else {
        AC->Cmd.Ang[0] = 0;
        AC->Cmd.Ang[1] = 0;
        AC->Cmd.Ang[2] = 0;
        AC->Cmd.RotSeq = 321;
    }

    A2C(AC->Cmd.RotSeq, AC->Cmd.Ang[0]*D2R, AC->Cmd.Ang[1]*D2R, AC->Cmd.Ang[2]*D2R, C);
    C2Q(C, AC->Cmd.qrn);

    AC->Cmd.wrn[0] = 0;
    AC->Cmd.wrn[1] = 0;
    AC->Cmd.wrn[2] = 0;
}

```

**Algorithm B.6** SetPoint function

```
int main(int argc, char **argv)
{
    FILE *ParmDumpFile;
    char FileName[120];
    struct AcType AC;
    SOCKET Socket;
    char hostname[20] = "localhost";
    int Port = 10201;

    if (argc > 1) {
        AC.ID = atoi(argv[1]);
        Port = Port + AC.ID;
    }

    AllocateAC(&AC);

    Socket = InitSocketClient(hostname, Port, 1);

    /* Load parms */
    AC.EchoEnabled = 1;
    ReadFromSocket(Socket, &AC);
    AC.FirstTime = 0;
    SetPoint(&AC);

    InitAC(&AC);
    AcFsw(&AC);

    sprintf(FileName, "./Database/AcParmDump%021d.txt", AC.ID);
    ParmDumpFile = fopen(FileName, "wt");
    WriteToFile(ParmDumpFile, &AC);
    fclose(ParmDumpFile);
    WriteToSocket(Socket, &AC);

    while(1) {
        ReadFromSocket(Socket, &AC);
        SetPoint(&AC);
        AcFsw(&AC);
        WriteToSocket(Socket, &AC);
    }

    return(0);
}
```

**Algorithm B.7** main function



### B.3.6.3 Makefile modifications

```
##### Macro Definitions #####

# If not _AC_STANDALONE_, link AcApp.c in with the rest of 42
ifneq ($(strip $(STANDALONEFLAG)),)
    ACOBJ =
else
    ACOBJ = $(OBJ)AcApp.o
    ACOBJ2 = $(OBJ)AcApp2.o
endif

##### Rules to link 42 #####

42 : $(42OBJ) $(GUIOBJ) $(SIMIPCOBJ) $(FFTBOBJ) $(SLOSHOBJ) $(KITOBJ) $(ACOBJ) $(GMSECOBJ)
    $(CC) $(LFLAGS) $(GMSECBIN) -o $(EXENAME) $(42OBJ) $(GUIOBJ) $(FFTBOBJ) $(SLOSHOBJ) $(KITOBJ)
    ) $(ACOBJ) $(GMSECOBJ) $(SIMIPCOBJ) $(LIBS) $(GMSECLIB)

AcApp : $(OBJ)AcApp.o $(ACKITOBJ) $(ACIPCOBJ) $(GMSECOBJ)
    $(CC) $(LFLAGS) -o AcApp $(OBJ)AcApp.o $(ACKITOBJ) $(ACIPCOBJ) $(GMSECOBJ) $(LIBS)

AcApp2 : $(OBJ)AcApp2.o $(ACKITOBJ) $(ACIPCOBJ) $(GMSECOBJ)
    $(CC) $(LFLAGS) -o AcApp2 $(OBJ)AcApp2.o $(ACKITOBJ) $(ACIPCOBJ) $(GMSECOBJ) $(LIBS)

##### Rules to compile objects #####

$(OBJ)AcApp.o      : $(SRC)AcApp.c $(INC)Ac.h $(INC)AcTypes.h
    $(CC) $(CFLAGS) -c $(SRC)AcApp.c -o $(OBJ)AcApp.o

$(OBJ)AcApp2.o     : $(SRC)AcApp2.c $(INC)Ac.h $(INC)AcTypes.h
    $(CC) $(CFLAGS) -c $(SRC)AcApp2.c -o $(OBJ)AcApp2.o

##### Miscellaneous Rules #####

clean :
ifeq ($(42PLATFORM),_WIN32)
    del .\Object\*.o .\$(EXENAME) .\InOut\*.42
else ifeq ($(42PLATFORM),_WIN64)
    del .\Object\*.o .\$(EXENAME) .\InOut\*.42
else
    rm -f $(OBJ)*.o ./$(EXENAME) ./AcApp ./AcApp2 $(INOUT)*.42 ./Standalone/*.42 ./Demo/*.42 ./
    Rx/*.42 ./Tx/*.42
endif
```

**Algorithm B.8** Makefile function modifications