UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

Study of orbital propagator 42 for missions based on constelations of nanosatellites

Contents:
REPORT

Master's thesis
MUEA

Lluís Montilla Rodríguez

Supervisor: David González Díez — Delivery date: 22/06/2021

MUEA - ESEIAAT
Universitat Politècnica de Catalunya (UPC)

Contents

NOMENCLATURE **7**

1 INTRODUCTION **8**
1.1 Abstract . . . 8
1.2 Plathon project . . . 8
1.3 Scope . . . 9
1.4 Requirements . . . 9
1.5 Objectives . . . 10

2 PROGRAM OPERATION **11**
2.1 Simulation procedure . . . 11
2.2 Input definition . . . 13
2.3 Output files . . . 15
2.4 Sensor modelling . . . 17
2.5 Actuator modelling . . . 18

3 CONSTELLATION GENERATION **20**
3.1 Constellations and nanosatellites . . . 20
3.2 Input generator . . . 22
3.3 Iridium constellation . . . 25

4 ATTITUDE CONTROL SUBROUTINE **29**
4.1 Default routines . . . 29
4.2 Cubesat attitude control definition . . . 30
4.3 Subroutine testing . . . 30

5 POST-PROCESSING **35**
5.1 42's GUI . . . 35
5.2 Post-processing function . . . 39
5.3 3D simulation . . . 40
5.4 Final solution . . . 43

6 INTER PROCESS COMMUNICATION **44**
6.1 IPC configurations . . . 44
6.2 Standalone AcApp . . . 45
6.3 Integration with Plathon . . . 45
6.4 IPC simulation . . . 47

7 SCHEDULE, BUDGET AND ENVIRONMENTAL IMPACT **52**
7.1 Schedule . . . 52
7.2 Budget . . . 55
7.3 Environmental impact . . . 55

8 CONCLUSIONS **57**
8.1 Conclusions . . . 57
8.2 Recommendations and future studies . . . 59

Appendix A INPUT FILE EXAMPLES **5**

Appendix B CONSTELLATION INPUT GENERATOR **24**

Appendix C PROGRAMMING A FSW FUNCTION IN 42 **30**

Appendix D ADDED SIMULATION COMMANDS **34**

Appendix E MATLAB POSTPROCESSING FUNCTIONS **36**

Appendix F SIMULINK 3D MODEL **40**

Appendix G IRIDIUM SIMULATION **51**

Appendix H MODIFIED ACAPP **67**

Appendix I MODIFIED MAKEFILE **81**

Appendix J IPC SIMULATION INPUTS **91**

List of Figures

1 Plathon project schematic . . . 9
2 42 simulation procedure [1] . . . 13
3 Total nanosatellites launched [2] . . . 21
4 Nanosatellites current status [3] . . . 22
5 Example constellation ground track . . . 24
6 Example constellation orrery view . . . 24
7 Iridium simulation results . . . 27
8 Iridium simulation ground track . . . 27
9 Iridium simulation orbits . . . 28
10 Eclipse status during simulation . . . 31
11 Quaternion body with respect to ECI during simulation . . . 32
12 Sun vector in body during simulation . . . 32
13 Angular velocity of body in ECI during simulation . . . 33
14 42 GUI cam viewer . . . 35
15 42 GUI map viewer . . . 36
16 42 GUI orrery viewer . . . 37
17 42 GUI unit sphere viewer . . . 38
18 Position and velocity in ECI . . . 39
19 Position and velocity in ECEF . . . 39
20 Simulink 3D simulation . . . 41
21 Simulink 3D animation initial moment . . . 42
22 Simulink 3D animation satellites communicating . . . 42
23 42 system architecture . . . 46
24 IPC simulation architecture . . . 48
25 IPC simulation cam view . . . 49
26 IPC simulation map view . . . 49
27 IPC simulation spacecraft attitude change . . . 50
28 IPC simulation data traffic between applications . . . 51
29 Pie chart: hours spent per activity . . . 53
30 Gantt chart of the project . . . 54

List of Tables

1 TLE format description (line 1) 25
2 TLE format description (line 2) 26
3 Summary of fsw functions 30
4 42 GUI view description 38
5 Hours spent per activity 53
6 Project's budget 55
7 Project's environmental impact 56

ACKNOWLEDGEMENTS

First, I would like to acknowledge the help provided by the *Plathon* group, especially my thesis supervisor David González, as they provided some important aid regarding the communication of the different machines.

I would also like to thank my family for all the support provided during these years.

NOMENCLATURE

Δv Increment of velocity

ACS Attitude Control System

CSS Coarse Sun Sensor

FSS Fine Sun Sensor

$GDOP$ Geometric Dilution Of Precision

$GNSS$ Global Navigation Satellite System

GPS Global Positioning System

GUI Graphical User Interface

IPC Inter Process Communication

JD Julian Date

LEO Low Earth Orbit

$RAAN$ Right Ascension of the Ascending Node

$TDRS$ Tracking and Data Relay System

TLE Two Line Element

1 INTRODUCTION

This section serves as an introduction to the project in which the abstract of the project will be presented, as well as the scope and requirements. Finally the project's objectives will also be enumerated.

In this introduction, the *Plathon* project will also be explained in broad terms so as to see where this particular project fits in.

1.1 Abstract

This project is part of a bigger project called *Plathon* and consists of the study, application and adaptation of NASA's orbital simulator 42 to the *Plathon* project. For this purpose, the program has been studied with several test simulations in order to better understand its inner workings and to test the possibilities and limitations of the simulator.

After that, a series of modifications and standalone codes, using *Matlab*, have been implemented to better fit the simulator into the project and to allow for better post-processing of the obtained data. On top of that, the integration of the simulator with the *Plathon* project will also be discussed.

1.2 Plathon project

The *Plathon* project aims to develop a platform for the simulation of a constellation of nanosatellites (cubesats) with the objective of simulating optical and radio frequency communication amongst the satellites and between satellite and ground station.

This simulation is meant to start by simulating everything with software such as the orbital simulator 42 (the object of this project), but will eventually turn into a hardware in the loop simulation, where the satellite itself will be included in the simulation. This means that the spacecraft dynamics and attitude control will not be simulated but rather read directly from the sensors and actuators on board the satellite.

Moreover, the satellite will also be managing the communications and energy leaving only the orbital propagation and environment to the simulators. In figure 1, the basic idea behind the *Plathon* project is dispalyed.

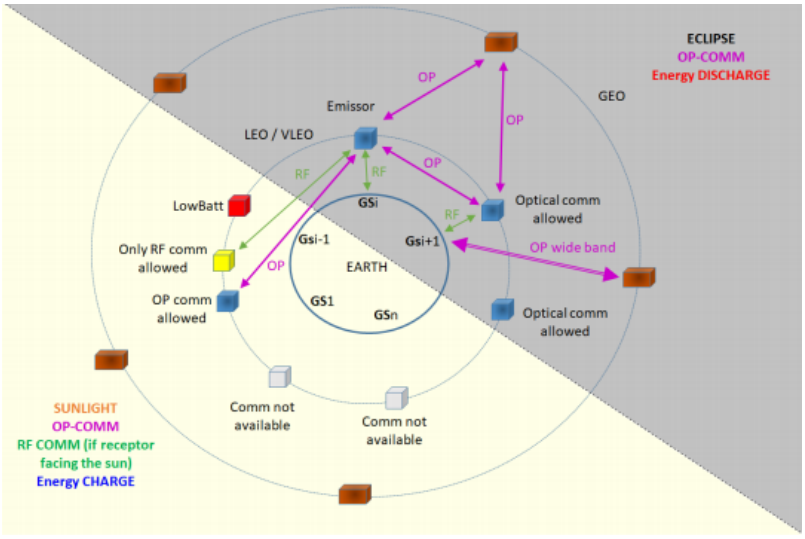


Figure 1: Plathon project schematic

1.3 Scope

The scope of the project covers the following points:

- To understand the inner workings of NASA's orbital simulator 42.
- To implement the necessary functions or models that better fit the *Plathon* project.
- To visualise the results using both the 42 simulator GUI and self-made *Matlab/Simulink* functions.
- To design a simulation of a constellation of cubesats to test communication amongst themselves and the ground station.
- To begin the implementation of 42 into the *Plathon* project.

The project does not intend:

- To design an orbital simulator.
- To manufacture the cubesats or any of its components.

1.4 Requirements

In order to meet the project's objectives, a set of requirements have been set:

- Any work done must be original.
- Any work done must help accomplishing the objectives of the *Plathon* project as a whole.
- Cooperation among students working in the *Plathon* project must be encouraged.

1.5 Objectives

The project objectives can be summarised in the following points:

- To understand the orbital simulator 42
- To provide the necessary postprocessing functions to allow for a better visualization of the simulation results
- To implement all necessary modifications to 42 that may be needed to integrate the simulator into the *Plathon* project
- To simulate a constellation of satellites to test the capabilities of the program
- To demonstrate through a simulation the concept of inter-process communication of several 42 running in parallel and some standalone *AcApp's* controlling the satellites independently

2 PROGRAM OPERATION

In this section, the inner workings of NASA's orbital simulator 42 will be explained in order to expose how the program works, how to input the necessary data to start a simulation and what kind of output it produces.

First, the simulation procedure will be explained so that the way the simulator works can be understood; then, the different inputs will be detailed so that the necessary information to start a simulation is clear; after that, the different outputs produced after a simulation will be explained so as to know what is obtainable; last, the sensor and actuator modelling included in the simulator will be explained.

2.1 Simulation procedure

The Orbital simulator 42 is a very powerful tool made by NASA to simulate satellites in orbit around any of the astral bodies of the solar system which includes orbital dynamics and attitude determination and control [1]. It is part of the OpenSat Kit, a series of programs which are meant to plan space missions. This implementation inside OpenSat Kit allows the use of 42 with other software which would simulate other aspect that are not taken into account inside 42 or that are modelled in a very simple way.

It is capable of simulating several spacecrafts as well as having models implemented for the environment (gravitational and magnetic fields), ephemeris, sensors and actuators. It also includes a GUI to visualise the simulation.

In order to explain how the simulator works, first, the structure of the different files must be explained. Orbital simulator 42 is split into two different kinds of files, includes and source files. The former include all necessary global variables, functions and files that are needed for the program to work. The latter includes the code of the program. Inside the code, three different kinds of functions can be found, top-level functions, mid-level functions and low-level functions. Top-level functions basically call other functions in the determined order; mid-level functions perform most of the operations needed for the simulations; and last, low-level functions are toolkit functions such as matrix multiplication and so forth.

The main simulation algorithm starts by initialising all aspects of the simulation such as loading the celestial bodies, spacecrafts and other variables stated by the inputs files. After that it reads the command inputs for the simulation, and last it performs all necessary simulation calculations in a loop until the simulation is done.

The main simulation loop begins by reporting the simulation progress into the console, followed by managing program flags and reading simulation commands. After that, based on the torques and forces of the previous time step, it updates the spacecraft's position and attitude using the model specified in the inputs. Then, it calculates the orbit's motion, such as drifts from J2. At this point it checks for the simulation completion, in case this is the end of the simulation, it sets a flag as true which will end the simulation at this time step (not at this point specifically, as it still makes

some more calculations).

Next, it updates the spacecrafts' bounding boxes and sends and receives all necessary information from external apps, in case they are being used. NASA's orbital simulator 42 is capable of working together with other programs of the *Open Sat Kit* tools.

Following that, it updates the planets', moons' and other bodies' ephemeris based on the model set at the input. Then, after resetting all forces and torques to 0, for every spacecraft present in the simulation, it checks the environment, perturbations, sensor readings, flight software actuation and actuators response; before continuing, it splits the forces into external and internal.

Finally, the last part of the main simulation loop performed by the program is reporting some data into output files.

The environment function performed checks the magnetic field, atmospheric density and presence of radiation fluxes in radiation belts based on the models defined in the inputs. The perturbations functions computes the perturbation forces and torques that have been enabled in the input files. The sensor function, in case sensors have been defined for the spacecraft in question, takes the truth, i.e. position, quaternions, etc; and through some models adds inaccuracies and noise; if no sensors are defined, the truth is taken without modification. The flight software function applies the attitude control algorithm defined in the inputs. Every one of these algorithms performs different actions, the most basic of them is to comply with the simulation commands relative to attitude. This is done in different ways depending on the particular algorithm but in most cases a simple PD algorithm is used. The actuators function, takes the desired forces and torques from the flight software function and obtains the actuators' output forces and torques based on some simple models; if no actuators are defined, the program uses ideal actuators which output the desired forces and torques no matter how absurd.

To summarise, the main simulation loop can be described in a few steps (see figure 2 for clarification):

1. Initialisation
2. Ephemeris calculations
3. Environmental model
4. Sensor readings
5. Flight software functions
6. Actuator model
7. Spacecraft dynamics
8. If not finished go to 2

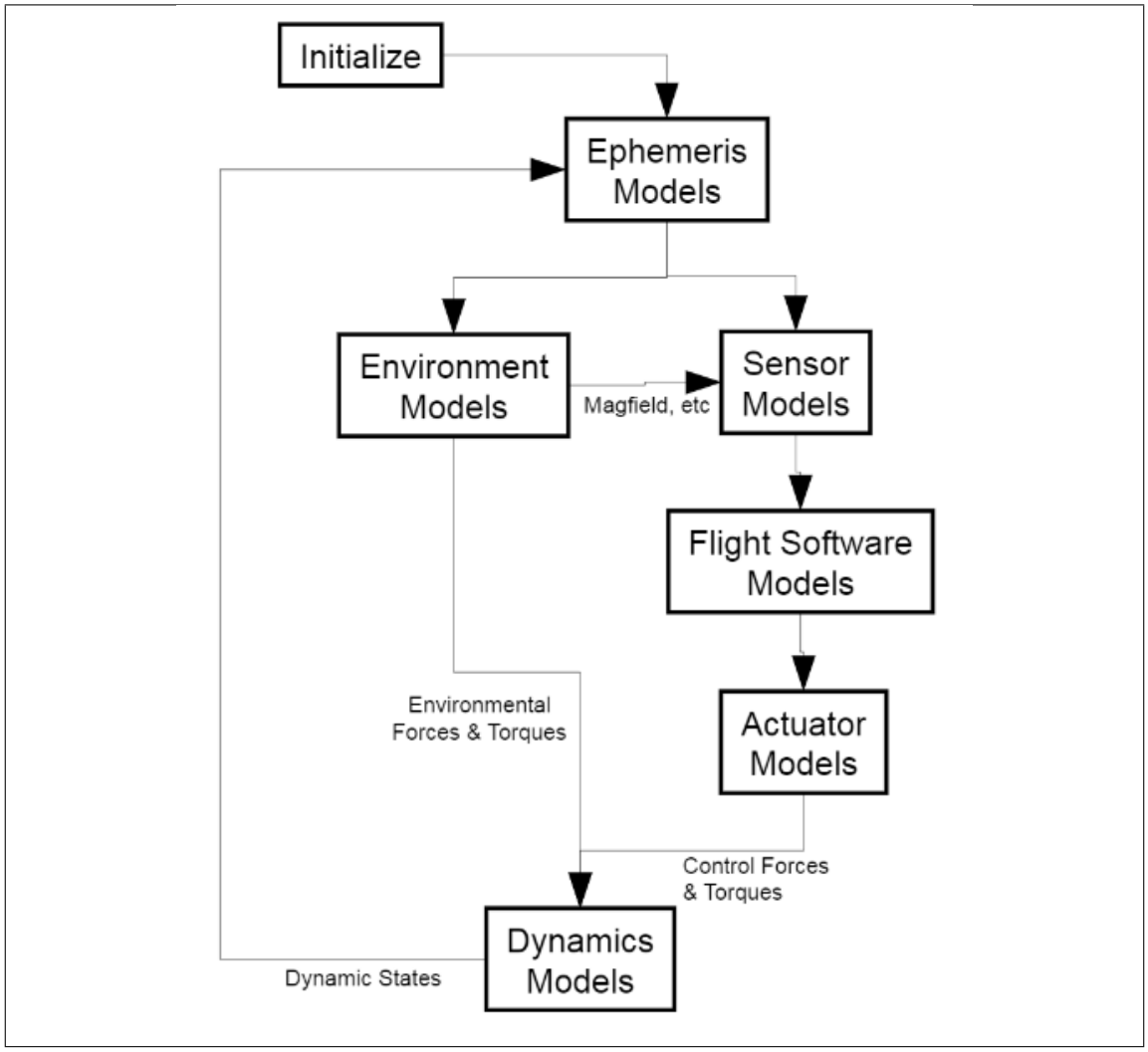


Figure 2: 42 simulation procedure [1]

2.2 Input definition

NASA's orbital simulator 42 uses a series of input text files to load all the desired simulation parameters. These files follow a very specific format which must be followed so that the program interprets every file without problems.

There are 5 main input files, although some other inputs must be defined as well. These main inputs are:

- Inp_Sim.txt
- Inp_Cmd.txt

- Inp_Graphics.txt
- Orb_*.txt
- SC_*.txt

Where * can be any name. Keep in mind that there can be as many *Orb* and *SC* files as is desired for the simulation. In addition to the main inputs, it is also necessary to define 4 extra inputs:

- Inp_FOV.txt
- Inp_IPC.txt
- Inp_Region.txt
- Inp_TDRS.txt

The most basic input file, from which all others derive, is the *Sim* file (See example in appendix A.1), it defines the basic simulation parameters such as the duration, time step, whether the graphical interface is desired, the environment characteristics such as the gravity and magnetic field models, the disturbance forces and torques present, the celestial bodies that will be present as well as the Lagrange point systems present, the ground stations and finally, the orbit and spacecraft files to be used.

The next input file is the *Cmd* file (See appendix A.2 for an example), in this file, all simulations commands are stated. Commands are used to alter the behaviour of the simulation at a specific time. Using commands it is possible to control the attitude of any spacecraft present in the simulation, alter the graphics interface, or even impart Δv to any spacecraft.

The *Graphics* input file controls the graphical interface. It is where the different camera angles are defined as well as indicating whether the different graphical screens are desired. For an example see appendix A.3.
Inside the *Orb* file (See example in appendix A.4), the orbital parameters of the reference orbit in question are defined. There are 4 possible types of orbits according to the program documentation [4]: zero, flight, central and three-body.

The central orbit type is a two-body orbit, those are appropriate for situations where one celestial body dominates the orbital motion of the spacecraft. The three-body orbit type, as its name implies, takes into account three-body orbital mechanics, this applies whenever a spacecraft is near a libration point. The flight orbit type determines the motion relative to a region (regions are defined as a point fixed in a local reference frame). This type of orbits are the preferred orbit type for situations where a reference orbit does not make sense, such as take-off and landing, ground operations. Last the zero orbit type is similar to the flight type but with the reference at the center of the attracting body. This is used when the spacecraft is near a minor body.

The last of the main input files is the *SC* file, in this file the spacecraft definition is stated. This input file can be split into 4 parts, the general definition, the body parameters, the joint parameters and the actuators and sensors definition.

In the general definition, the initial attitude, orbit propagation method, dynamics flags and flight software type are defined. In the body parameters, the mass, inertia and geometry file are set; a spacecraft may be composed of multiple bodies. In the joint parameters, the degrees of freedom, initial angles and displacements, as well as spring and damping coefficients are defined for the joints joining two bodies. And in the actuators and sensors definition the amount and parameters of the different sensors and actuators are included. It is also possible to set a flex file in the body definitions to include a flexible solid model.

With regards to the other inputs, the *FOV* file (See appendix A.6 for an example) adds field of view cones to the defined spacecraft which can be seen using the graphical interface; the *IPC* file (Example in appendix A.7) allows the simulator to run inter-process communication, which allows it to work as a standalone program, or to work together with other programs; the *Region* file (Example in appendix A.8) defines the regions for the flight orbit type; and finally, the *TDRS* file (Example in appendix A.9) includes the TDRS (Tracking and Data Relay Satellite) satellites in the simulation.
As a summary:

- **Inp_Sim**: determines the basic simulation information and points to the other input files
- **Inp_Cmd**: inputs the simulation commands
- **Inp_Graphics**: configures the GUI
- **Orb_***: defines a reference orbit
- **SC_***: defines a spacecraft and its components
- **Inp_FOV**: adds FOV cones to the visualization
- **Inp_IPC**: configures the sockets for inter-process communication between programs
- **Inp_Region**: defines the regions present in the simulation
- **Inp_TDRS**: defines the TDRS satellites

Regard how thanks to the *Inp_IPC* file, the simulator can work in conjunction with other programs as it controls the flow of information coming/going through the sockets defined inside it.

2.3 Output files

After the simulation has finished, the simulator outputs certain information as text files for further analysis by the user. These files contain information such as the position and velocity in different reference frames, the quaternion expressing the attitude of the spacecraft, the angular velocity, and other complementary information such as the sun vector.

The program only outputs this information for the first spacecraft defined in the inputs, not all of the spacecrafts present in the simulation. The only information output for all spacecraft are the tree structure (which indicates the relations between the different bodies and joints present in the spacecraft definition), the dynamic characteristics, such as the mass and inertia, and the dynamics states

(position, velocity, quaternion and rotation rate of the spacecraft with respect to its reference point).

The output files obtained are listed below:

- **Acc**: accelerometer readings
- **Dyn***: dynamic properties of the spacecraft (mass, center of mass, etc)
- **DynTime**: time since J200 of the simulation
- **Hvn**: spacecraft's angular momentum vector in N frame
- **Hwhl**: reaction wheels momentum
- **KE**: spacecraft's kinetic energy
- **MTB**: magnetorquer torque
- **PosN**: spacecraft position in N frame
- **PosR**: spacecraft position in R frame
- **PosW**: spacecraft position in W frame
- **qbn**: spacecraft quaternion in N frame
- **RPY**: spacecraft Euler angles in L frame
- **svb**: Sun pointing vector in B frame
- **svn**: Sun pointing vector in N frame
- **time**: simulation time
- **Tree***: spacecraft body/joint tree representation
- **u***: body dynamic information (angular velocity in N frame, velocity relative to spacecraft center of mass N frame)
- **VelN**: spacecraft velocity in N frame
- **VelR**: spacecraft velocity in R frame
- **VelW**: spacecraft velocity in W frame
- **wbn**: spacecraft angular velocity in N frame
- **x***: body dynamic information (quaternion in N frame, position relative to spacecraft center of mass in N frame)

As seen in the list above, the simulator uses a series of reference frames, these are:

- **N frame**: world-centered inertial frame, in case of Earth it corresponds to the ECI frame

- **W frame**: world-centered world-fixed frame, in case of Earth it corresponds to the ECEF frame
- **L frame**: local-horizon, local-vertical frame
- **R frame**: orbit reference frame, like L frame but follows the reference orbit instead of the spacecraft. If the spacecraft is directly on the reference orbit and no perturbations are present, R frame and L frame are the same

2.4 Sensor modelling

Inside the simulator, in order for the spacecraft to obtain its position, attitude and important vectors such as the sun pointing vector, some simple sensor models are implemented. These include the basic sensor used in satellites for attitude determination and are:

- Accelerometers
- Gyroscopes
- Magnetometers
- CSS
- FSS
- Startrackers
- GPS

The basic idea behind these models is to take the truth values and add some random noise or bias. After that, each different sensor computes different values relevant to attitude determination. In case a particular type of sensor is missing, the simulator will fill in the measured value with the truth directly. This, however, does not occur when using the simulator with the standalone *AcApp*, in this case it will not have access to the value in question.

In the case of the accelerometer, it calculates the acceleration of the point in the body of the spacecraft where the accelerometer is located in the direction of the axis specified in the input file. After calculating the true acceleration, it takes into account certain error sources such as the accelerometer's bias, noise factor and random walk; as well as quantization errors, maximum acceleration measurable and sample time.

The gyroscope modelling is akin to the accelerometer's but the position is not relevant as every point in a rigid solid has the same angular velocity, so only the axis is necessary.

The magnetometer determines the value of the magnetic field in the determined axis, it also noise, scale error and quantization error. After that, it also takes into account the magnetometer's saturation.

The CSS does not compute directly the Sun pointing vector, but instead determines if the CSS is illuminated and at what intensity. After checking whether the spacecraft is in eclipse, it compares

the angle of the true Sun pointing vectorwith respect to the CSS' axis with the half-cone angle determined. After that, it adds some quantization error to the illumination value.

Unlike the CSS, the FSS directly determines the angles of the Sun pointing vector with respect to the FSS' axis. It does so in a similar way to the CSS but instead of calculating the illumination value it directly computes the angles.

The startrackers directly compute the spacecraft's attitude, expressed in quaternions, by taking the truth and adding some errors in a similar way to the other sensors. However, before even starting the computations, using the values set in the inputs for the mounting axis, field of view and exclusion angles, it checks whether it is being blinded by the Sun, Earth (or any other body), and the Moon (only if orbiting the Earth). In case the startracker is being blinded it returns no value.

Finally, the GPS modelling takes the truth values for the position and velocity of the spacecraft, as well as the time, and after adding some noise, computes the measured values in both inertial and Earth-fixed reference frames. It also computes the latitude and longitude and the GPS week and second of the spacecraft.

2.5 Actuator modelling

Just like with the sensors, the simulator includes some actuator modelling so that the forces and torques applied do not directly correspond to the ideal forces/torques indicated by the attitude control PD function. However, the models are much simpler.

There are currently three types of actuators:

- Thrusters
- Reaction wheels
- Magnetorquers

The thruster model, takes the required thrust and pulse width from the flight software function (which is only implemented in one of the example functions) and applies said thrust, previously being compared with the maximum thrust of the thruster, in the axis and the position specified. If the resulting thrust force does not go through the center of mass of the spacecraft, it also computes the generated torque. After all of this, it also checks for the forces that may be caused by the exhaust plume impacting on other spacecraft surfaces.

The reaction wheel model is very simple. It takes the required torque from the flight software function and checks that said torque is not greater than the maximum torque that the wheel is capable of producing. After that, it also checks that the momentum of the wheel is not greater than the maximum. In the input file, some other parameters are also specified such as the imbalance and inertia of the wheel. This is accounted for as a perturbation force rather than part of the wheel model.

The magnetorquer model is also very simple. It limits the torque to the maximum that the magnetorquer can produce and then applies this torque depending on the axis of the magnetorquer and

the direction of the magnetic field vector.

3 CONSTELLATION GENERATION

In this section, the general concept of a constellation, as well as some examples will be provided. Followed by how the generation of the input files for a constellation are done so as to be used in 42.

First, a justification for constellations of nanosatellites will be made while also remarking some of the current concerns regarding the increase on space debris; following that, the explanation of a Matlab program made to generate 42 inputs for a Walker constellation will be presented; last, the input generator for a simulation of the Iridium constellation will be explained as well as the results obtained from said simulation.

3.1 Constellations and nanosatellites

A constellation is a group of satellites of similar design and purpose, distributed over space. This is not to be confused with a cluster formation of satellites, which are satellites in very close proximity to each other [5].

Constellations may have very different characteristics depending on their objective, however, some aspects are common such as the intention of providing total coverage in the desired latitudes or minimizing the number of satellites needed for that purpose due to cost.

Some of the most famous constellations are GNSS constellations like GPS and GLONASS. These constellations have the objective of providing positioning to the users via broadcast messages, although both of them have the same purpose, due to the different objective latitudes in which to give service to, the orbital characteristics of both of them vary. This variation in orbital characteristics is very typical of constellations as each one is tailored to the needs of the mission. A very important capability of a satellite constellation is the ability to precisely track moving objects and provide positioning and information services to several users at a time, regardless of their position in the globe, this is very helpful for both maritime and aerial transportation as while far away from land, traditional localization services are unavailable.

However, there have been many ideas towards achieving a more generalised constellation pattern. There are many constellation patterns providing global coverage, one of the most interesting is the Walker constellation, this constellation achieves global coverage with just 5 satellites at altitudes of over 11482 km, still requiring the least satellites at lower altitudes [6]. This constellation also has the benefit of being uniformly distributed, meaning that these constellations are easy to make and study. Moreover, these constellations can provide several fold coverage without exponentially increasing the numebr of satellites.

Using this kind of constellation, recent studies have found possible to construct a constellation with only 264 satellites at 900 km of altitude with a GDOP of only 2.36 [7]. This number could be lowered by increasing the altitude, which would also decrease the number of satellites needed, but will put the constellation out of LEO.

Keeping the constellation in LEO is important to comply with the current trends in satellite constel-

lations, which is the use of nanosatellites (See figure 3), often cubesats, due to their lower cost and complexity. These satellites are commonly placed in LEO orbits for economic purposes. However, placing satellites in LEO has other advantages besides cost, such as lower latency communications due to the close proximity to the Earth.

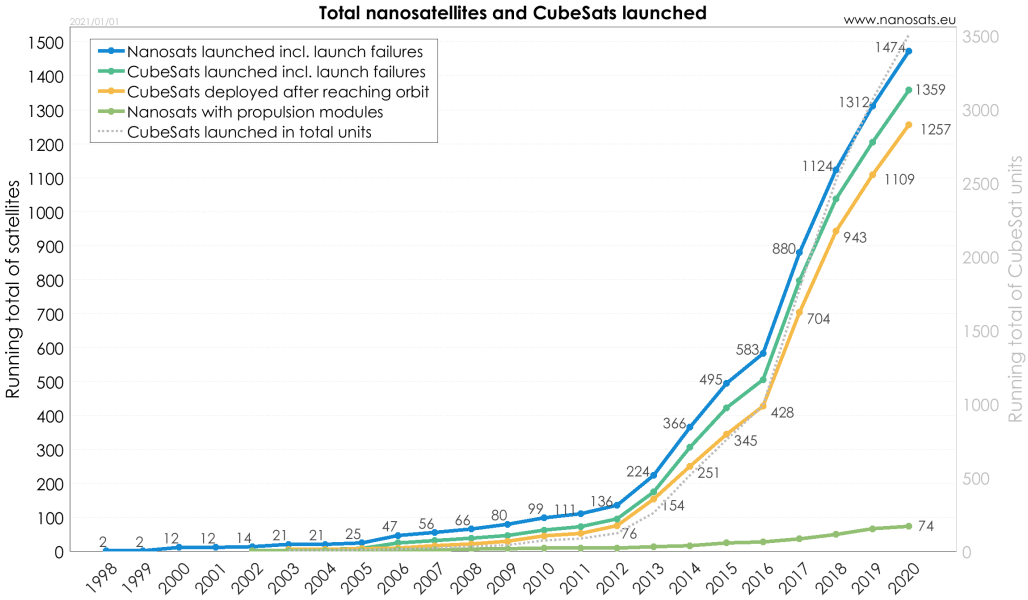


Figure 3: Total nanosatellites launched [2]

The amount of nanosatellites in orbit as of January 1st, 2021, is about 1474 and there are over 2500 planned nanosatellite launches in 6 years [8]. Comparing it with the 10680 satellites in total that have ever been launched since 1957 [9], already places the nanosatellite launches at a 10% of total satellites launched. However, if the trend continues, the number of nanosatellites will continue to increase exponentially, which will eventually make nanosatellites the most prevalent satellite type in orbit.

This numbers are worrying with regards to space debris, as with an increasing number of satellites in orbit, comes and increase in collisions which lead to a chain reaction of debris being catapulted in erratic orbits causing more collisions. Current estimates place the total number of objects greater than 1mm at over 128 million, with more than 28000 regularly tracked due to their size [9]. The reason why nanosatellite constellations are worrying is due to the great amount of new satellites which are being put in orbit, increasing collision chance. Currently, of the 6250 satellites currently in orbit, only 3900 are still functioning, meaning that the rest are inert objects without the capability of dodging any incoming debris that may cross their path, leading to more debris being put into orbit.

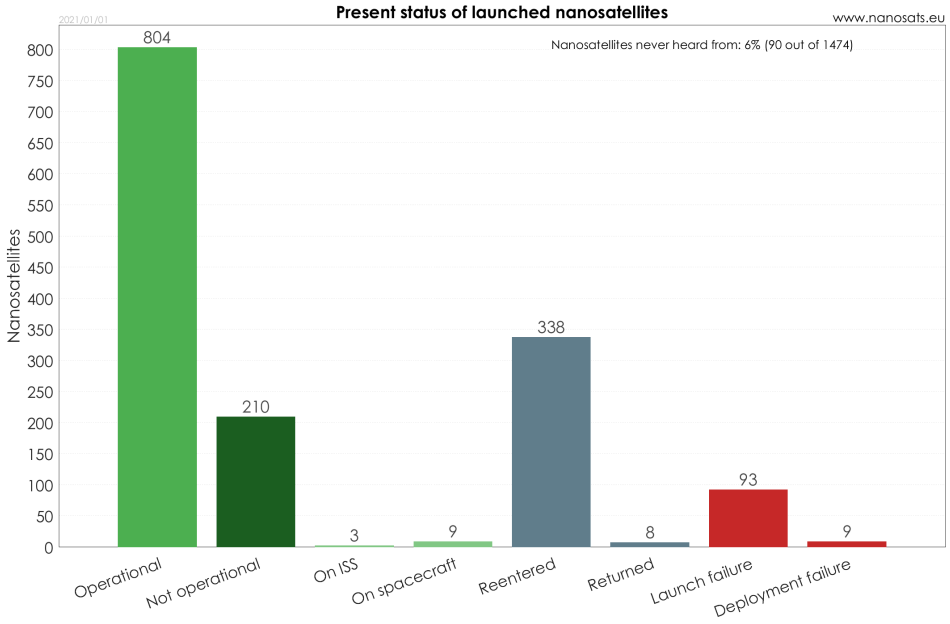


Figure 4: Nanosatellites current status [3]

In the case of nanosatellites, 1014 are still in orbit, among which, only 210 are not operational (Figure 4. This indicates a smaller proportion of abandonment of the satellite in orbit. This is due to the fact that nanosatellites are placed in LEO, meaning that any satellite left there will eventually decay into the atmosphere naturally due to drag, thus reducing the number of satellites in orbit after their decommissioning phase. Nevertheless, as LEO becomes more and more congested, the space debris problem will continue to endanger the current satellite infrastructure.

That being said, there are different initiatives regarding space debris such as NASA's ODPO (Orbital Debris Program Office) [10], or ESA's Clean Space initiative [11].

3.2 Input generator

In order to perform a simulation of a constellation, the input files for the orbits of each individual satellite must be written one by one. This, while possible, would be long, arduous and prone to human error. For that purpose, a matlab file which given the parameters of a Walker constellation, outputs the necessary input files has been written (See appendix B).

For the constellation definition, 6 parameters are required:

- Inclination in degrees, i
- Number of satellites, t

- Number of planes, p
- Relative spacing between satellites in adjacent planes, f
- Periapsis altitude in km, r_{min}
- Eccentricity, e

The former four parameters are what defines the Walker pattern, usually expressed **i:t/p/f**, whereas the latter two define the size and shape of the orbit. Keep in mind that f must be an integer ranging from 0 to p-1.

After that, the number of satellites per plane is simply defined as $s = t/p$. Another important parameter derived from the basic parameters is what is known as the pattern unit, which is simply $PU = 360/t$ (in degrees).

The pattern unit is used to calculate the in-plane spacing between satellites, the node spacing and the phase difference between adjacent planes using equations 1, 2 and 3 respectively, all in degrees.

$$PU_p = PU * p \tag{1}$$

$$PU_s = PU * s \tag{2}$$

$$PU_f = PU * f \tag{3}$$

Following that, it generates an orbit input file per satellite with the same inclination, periapsis altitude and eccentricity, but with the RAAN and true anomaly assigned to each satellite.

As an example, a Walker constellation **60:12/3/2** will result in $s = 4$ and $PU = 30$, resulting in $PU_p = 90$, $PU_s = 120$ and $PU_f = 60$. To see the resulting constellation check figures 5 and 6.

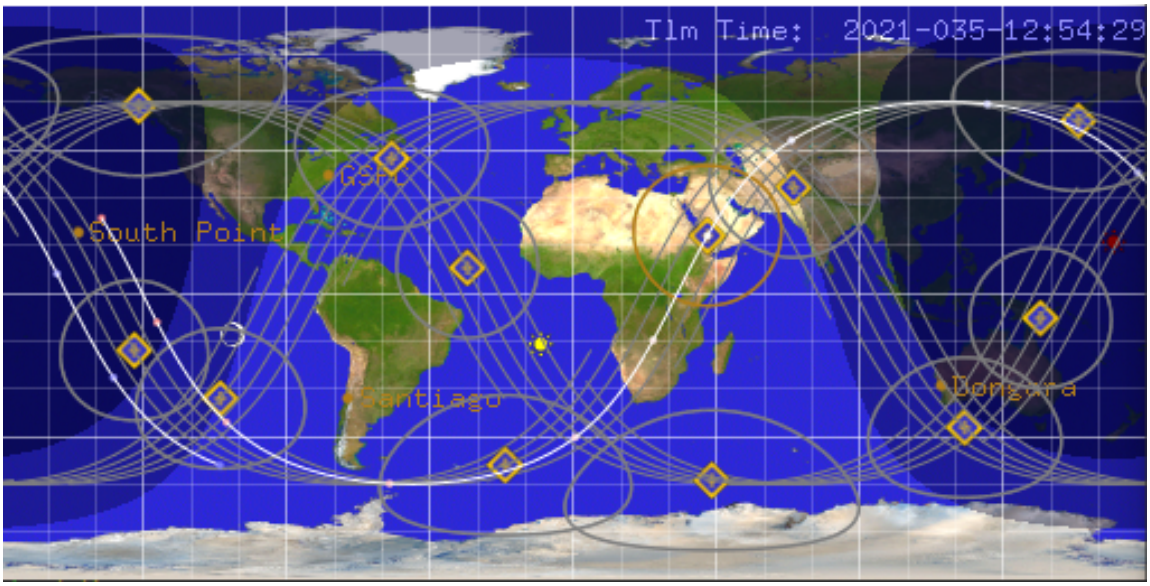


Figure 5: Example constellation ground track

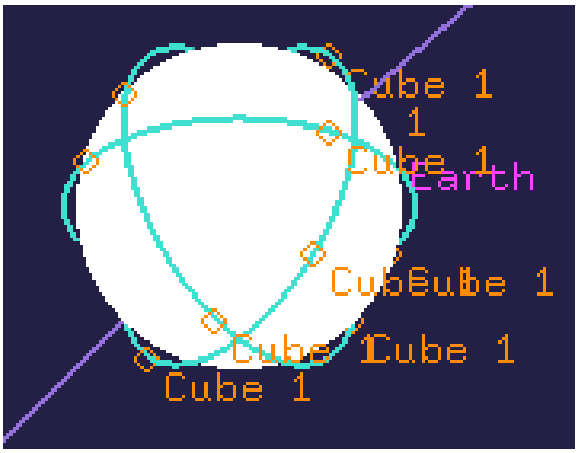


Figure 6: Example constellation orrery view

Finally, it generates the simulation input file with all the orbits and spacecrafts defined.

Keep in mind that in order to change any parameter of both the orbits and the simulation that has not been explicitly defined in the constellation generator, these values must be change directly into the functions which generate the input files.

3.3 Iridium constellation

An interesting exercise performed to demonstrate the capabilities of the orbital simulator 42 was to simulate a real constellation which is currently operating. This constellation has been chosen to be the Iridium constellation.

The Iridium constellation is a constellation of 66 cross-linked communication satellites in LEO, which was replaced completely in 2019 with a new set of satellites [12].

Being a real constellation, the orbital data of all of its satellites is available in different web pages in the form of TLE. For this particular case, the page used was *https://celestrak.com/*.

The contents of a TLE are detailed below [13]:

```
IRIDIUM 106
1 41917U 17003A    21120.32238812  .00000093  00000-0  26065-4 0  9993
2 41917  86.3971 151.6191 0002052  85.2850 274.8580 14.34215972224627
```

Line 0 is the 24-char long name of the satellite, while lines 1 and 2 correspond the the NORAD TLE format described in tables 1 and 2.

Line 1	
Column	**Description**
01	Line number of element data
03-07	Satellite number
08	Classification
10-11	International designator
12-14	International designator
15-17	International designator
19-20	Epoch year
21-32	Epoch
34-43	First time derivative of the mean motion
45-52	Second time derivative of the mean motion
54-61	BSTAR drag term
63	Ephemeris type
65-68	Element number
69	Checksum

Table 1: TLE format description (line 1)

Line 2	
Column	**Description**
01	Line number of element data
03-07	Satellite number
09-16	Inclination (degrees)
18-25	Right ascencion of the ascending node (degrees)
27-33	Eccentricity
35-42	Argument of perigee (degrees)
44-51	Mean anomaly (degrees)
53-63	Mean motion (revolutions per day)
64-68	revolution number at epoch (revolutions)
69	Checksum

Table 2: TLE format description (line 2)

In order to create the simulation, a similar program to the one already described in appendix B was made. However, as in this case the data was available at a web page in the form of a TLE, the program had to be changed.

All matlab files used to generate the inputs of the iridium constellation are found in appendix G. In this program, the TLE are downloaded from the webpage at the moment of execution as the TLE for the Iridium constellation are updated constantly. Then, this file is split into an array of characters to allow some processing. The TLE is then loaded into a class which obtains all the info from it and stores it as variables to be used later. In this class, some functions to obtain other parameters derived from the TLE's information are included in case it is desired to create the inputs directly as keplerian elements, although this is not necessary as 42 can accept TLE data directly. That was the preferred strategy as 42 propagates the information found on the TLE to the required epoch, which lessens the necessary pre-processing due to the fact that the TLE's of all the satellites in the Iridium constellation are from different epochs.

Figure 7: Iridium simulation results

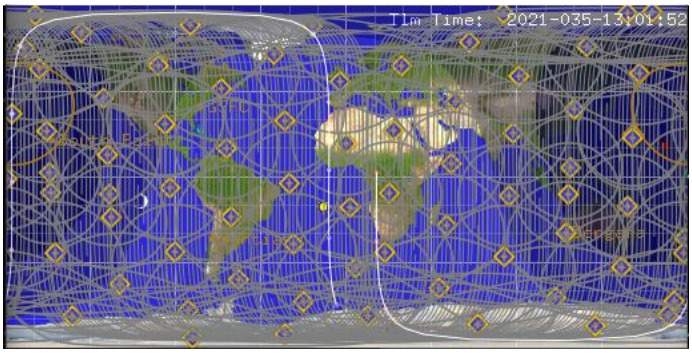


Figure 8: Iridium simulation ground track

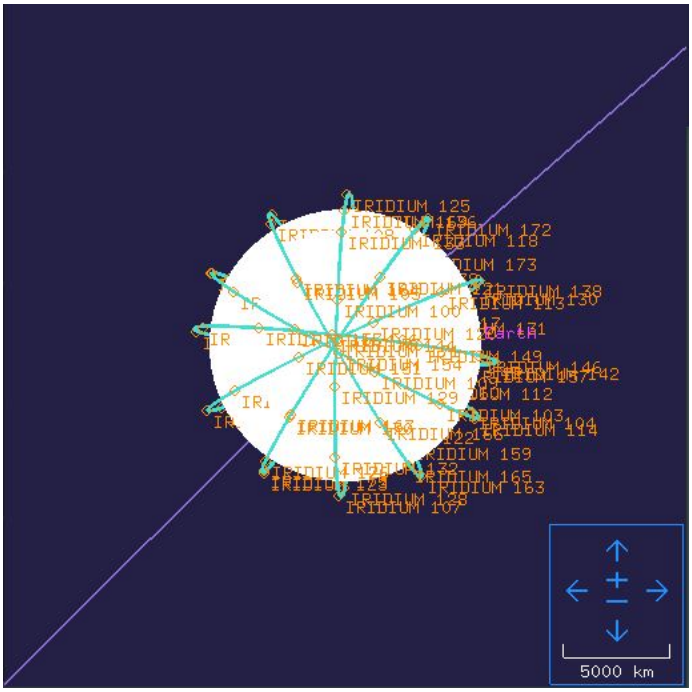


Figure 9: Iridium simulation orbits

In figures 7, 8 and 9, the results of the simulation can be seen represented in the 42 GUI. Keep in mind that the spacecraft models correspond to those of a 1U cubesat and not to the Iridium satellites.

4 ATTITUDE CONTROL SUBROUTINE

In this section, a summary of the simulator's built in attitude control as well as the process followed to design a fully customised function for a cubesat will be explained.

To begin with, the default fsw routines will be explained and enumerated; then, the definition of a new attitude control function tailored for a cubesat will be explained; finally, the testing of said function will be commented with a discussion on its results.

4.1 Default routines

The simulator contains several attitude control functions already programmed. Those vary from a very simple passive control to more complex functions [14]. However, none of these work to the needs of this project, meaning that a new function must be designed.

To begin with, let's analyse the already existing attitude control functions as they provide an insight on the simulator's inner working which is crucial to understand in order to design a new attitude control subroutine.

The most basic control function included is the *Passive* control mode, as its name implies, the satellite will not perform any correction with regards to its attitude, it will essentially be a rock. Obviously, this mode is not interesting for the purpose of this work.

The next function included is the *Prototype* control mode, with this function, the satellite adjusts its attitude using a PD controller. In this function, the satellite uses both ideal sensors and ideal actuators, meaning that the satellite will point where it is commanded without asking how. With regards to the PD controller, the program calculates the k_P and k_D constants for each axis based on the satellite inertia tensor and two constants that can be manually adjusted. Although this control mode would make attitude control possible, it would be interesting to add more realistic models for both the actuators and the sensors.

The rest of the functions included, while more realistic, do not comply with the requirements of this project for different reasons. All of them where tailored specifically to a particular example, meaning that they require a specific spacecraft body structure as they include amongst other things a controller to point the solar panels to the Sun. As a cubesat usually consists on a single body, and in case the solar panels are deployable, they do not have hinges to orientate themselves, these functions do not work. However, it is interesting to see that the sensors and actuators models can be easily integrated in control functions.

The last function included, the *Ad Hoc* function, is nothing more than a template in which to build other functions. Starting from this, the necessary attitude control functions will be built.

A summary of the default fsw function is included in table 3.

Name	Description
Passive	No control
Prototype	Simple PD with ideal sensors and actuators
Spinner	Provides spin stabilization
MomBias	Provides momentum based nadir pointing stabilization and points solar panels towards the Sun. Requires two bodies
ThreeAxis	Provides three axis stabilisation on command and points solar panels towards the Sun. Requires two bodies. Must use both wheels and magnetorquers
Iss	Made for a model of the ISS. Holds position, points solar panels towards the Sun, points radiators away from the Sun and points antenna towards TDRS nearest zenith
Thr	Provides three axis stabilization on command using thusters
Ad Hoc	Template for programming new functions

Table 3: Summary of fsw functions

4.2 Cubesat attitude control definition

As determined, it is necessary to program a customised attitude control function for the cubesats. This has to be done in a certain way in order to integrate the function properly with the rest of the simulator. In appendix C.1, the process needed to include a new function is detailed.

In order to define the function, first it is necessary to state the needs of the attitude control function:

1. The attitude control function shall be capable of determining the attitude of the satellite.
2. The attitude control function shall be capable of adjusting the attitude of the satellite to the desired attitude in all three axes.
3. The attitude acquisition shall be performed using star trackers and gyroscopes.
4. The attitude actuation shall be performed by either reaction wheels or magnetorquers.

In the program, some basic sensor and actuator modelling is included, in the sensor models, the truth is taken and some noise and bias is added to distort the real attitude vectors, in the actuators models, the commanded torques are limited by the maximum torque that the actuator is capable of producing at that moment.

These models are simple yet adequate but they will be modified as needed to improve the accuracy of the simulation with respect to the actuators and sensors designed by the rest of the people working in the *Plathon* project.

4.3 Subroutine testing

Once the attitude control functions have been programmed, they must be tested to ensure that they comply with the requirements. This tests have been performed with a basic simulation tailored to

the specific needs of each function.

For the *CubesatFSW* function (See appendix C.2), the test consisted on the following commands:

- Hold $qrn = [0, 0, 0, 1]$ until 5000 s
- Point Primary vector $[0, 1, 0]$ at the Sun and Secondary vector $[0, 0, 1]$ at the Earth after exiting eclipse
- Hold $qrn = [1, 0, 0, 0]$ from 10000 s

When defining the attitude of a satellite, it is important to define at least two vectors, the Primary vector and the Secondary vector. If only on vector is defined (Primary vector), the spacecraft is free to rotate around the axis defined by it. By fixing another vector (Secondary vector), the spacecraft's attitude is completely defined.

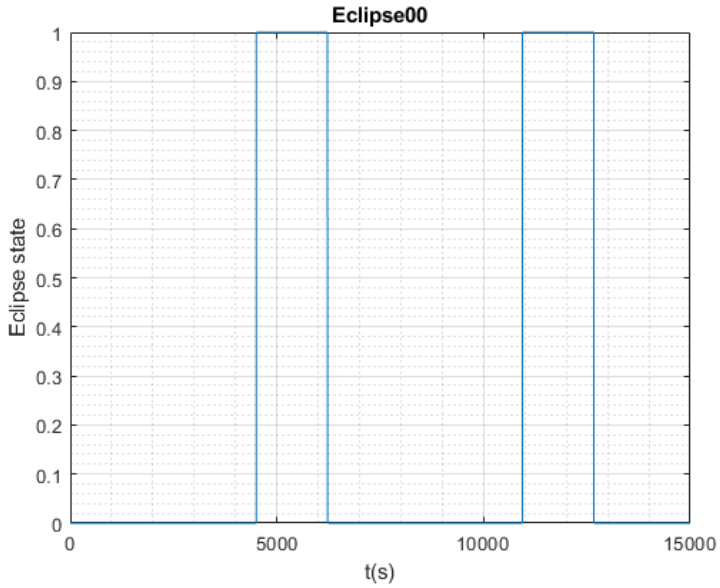


Figure 10: Eclipse status during simulation

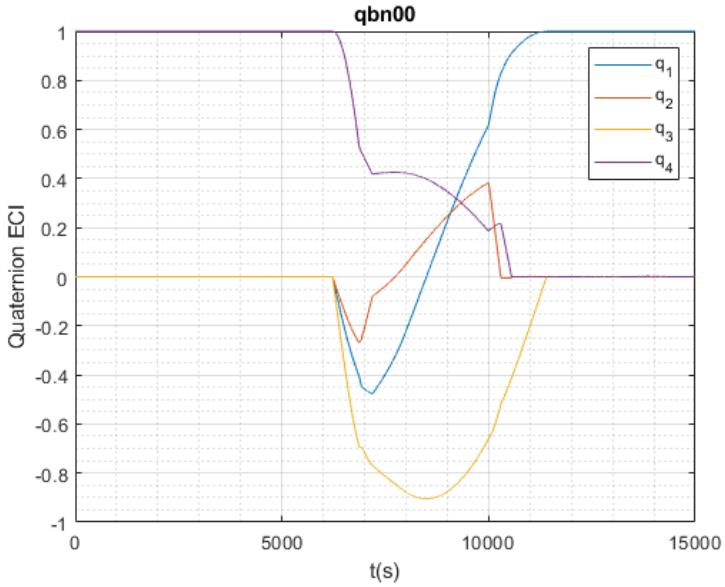


Figure 11: Quaternion body with respect to ECI during simulation

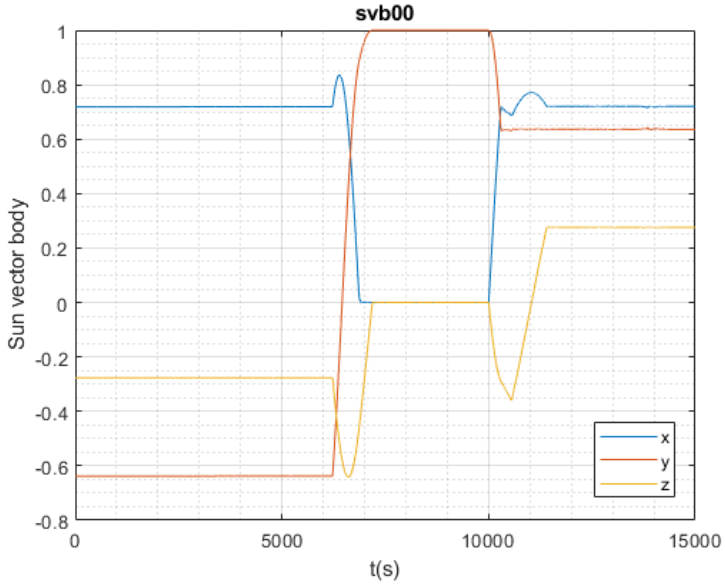


Figure 12: Sun vector in body during simulation

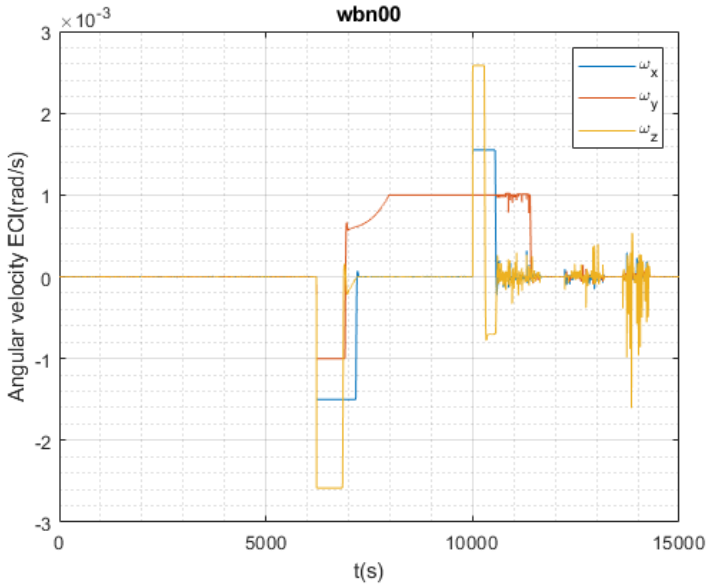


Figure 13: Angular velocity of body in ECI during simulation

In figures 10 11, 12 and 13, the eclipse status, quaternion, sun vector and angular velocity are represented in time respectively.

Note how the sun vector in body turns into $[0, 1, 0]$ as soon as the spacecraft exits the eclipse, thus pointing at the Sun, the secondary vector pointing a the Earth is not directly observable but it was defined just so that the attitude was totally determined as indicating the direction of the primary vector alone does not properly define the orientation of the spacecraft.

With regards to the quaternion, during the parts of the simulation which directly indicated which quaternion the spacecraft should have, it shows that the spacecraft has no issues achieving the desired quaternion. Note that even though the simulation only stated that the spacecraft should hold the quaternion $[0, 0, 0, 1]$ until 5000 s, it held that attitude for longer because the next command required the condition that the spacecraft were no longer in eclipse.

So as to the angular velocity, the graphs clearly show that all of the reaction wheels were saturated during most of the simulation as the graphs are mostly square shaped. Keep in mind that having a constant angular velocity during the time when the spacecraft is pointing to the Sun is a desired condition and does not imply saturation.

Another important regard, is that in the angular velocity graph, during the last 5000 s of simulation, some very rapid and short oscillations occur. This is a consequence of the star trackers being inside the exclusion angle of the Sun/Earth/Moon, which sends erratic information regarding the attitude, as this is modelled in the sensor functions of the simulator, causing the flight software function to try to correct the attitude. However, these movements are very small and do not get reflected in

the quaternion as such. The reason why this was not problematic is that for this simulation, for testing purposes, the exclusion angles were set very small so as to not interfere with the results that much.

One important modification that had to be done to 42 for this test was the addition of a new command as there was no command already programmed to point the satellite at the Sun whenever it exits eclipse (See appendix D). There were, however, a command which set a quaternion whenever the spacecraft exited eclipse and one that pointed the spacecraft at the Sun at a given time. Hence, the command added was a mix of these two.

5 POST-PROCESSING

During this section, the postprocessing done to visualize the results of the simulation will be explained. Starting with the built-in GUI included in 42, followed by the description of a post-processing function made in Matlab to visualise the data obtained through the outputs of the simulator, continuing with the explanation of a Simulink model to recreate the positions and attitudes of two satellites based on the data of the 42's output files from a simulation, and ending by stating the final solution found regarding visualization of the simulation.

5.1 42's GUI

The first visualization method is the 42 simulator's own GUI. This allows for real-time visualization of the simulation with four different viewers. The first one is the cam (See figure 14), in the cam viewer, the spacecraft can be seen as it orbits the celestial body chosen. It is possible to change the point of view, the targeted spacecraft or celestial body and other functions such as showing the reference axes.

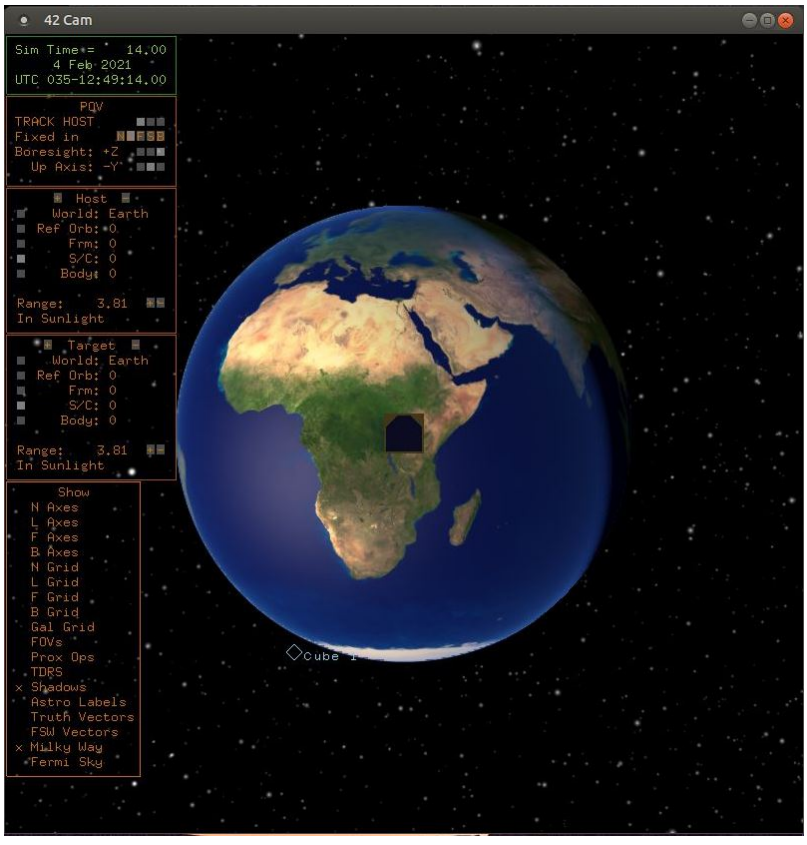


Figure 14: 42 GUI cam viewer

The next viewer is the map viewer (Figure 15). In the map viewer, all the spacecrafts' ground tacks and coverage is represented.

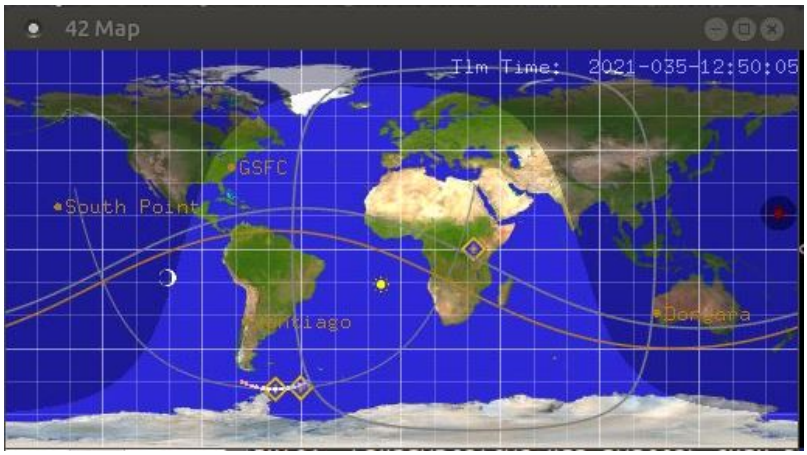


Figure 15: 42 GUI map viewer

The following viewer is the orrery viewer (Figure 16). In this window, the orbits of the celestial bodies and spacecraft currently in the simulation are displayed.

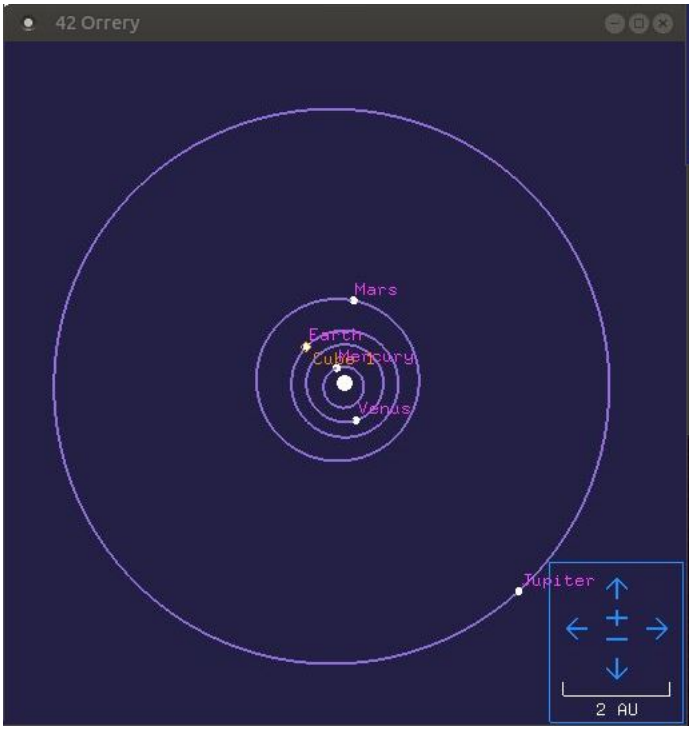


Figure 16: 42 GUI orrery viewer

Lastly, the unit sphere viewer (Figure 17) displays the position of different vector and celestial bodies with respect to the spacecraft.

Figure 17: 42 GUI unit sphere viewer

View	Description
Cam	Represents the spacecraft as it orbits. It is possible to change the angle of the visualization as well as the distance. It is also possible to represent axis systems and several vectors
Map	Represents the ground tracks over the orbited body as well as the coverage
Orrery	Represents the orbits and the spacecrafts over a map of the solar system
Unit-sphere viewer	Represents the different vectors and directions with regards to the spacecraft

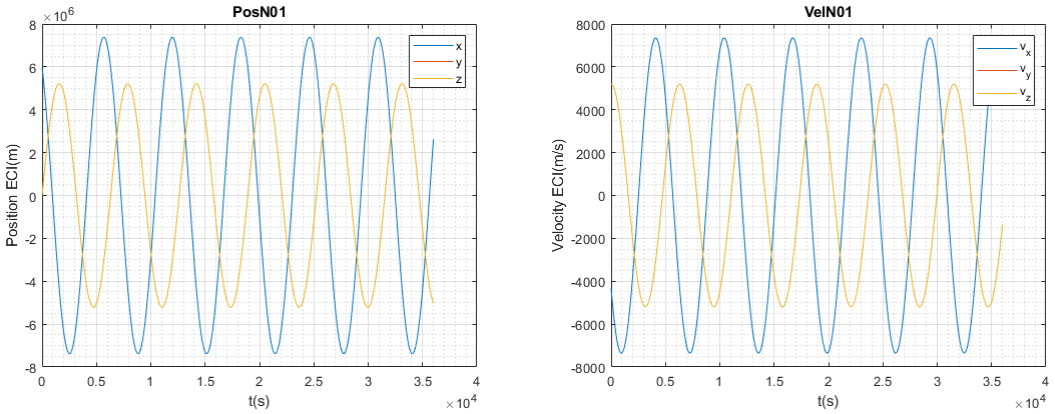
Table 4: 42 GUI view description

In table 4, a summary of the different views of the 42's GUI is presented.

While being able to see what happens in the simulation in real time is an advantage, activating the GUI slow the simulation down to a crawl. This disadvantage heavily outweighs the benefits of observing the simulation at real time, which means that some sort of postprocessing needed to be done with the data obtained through the output files.

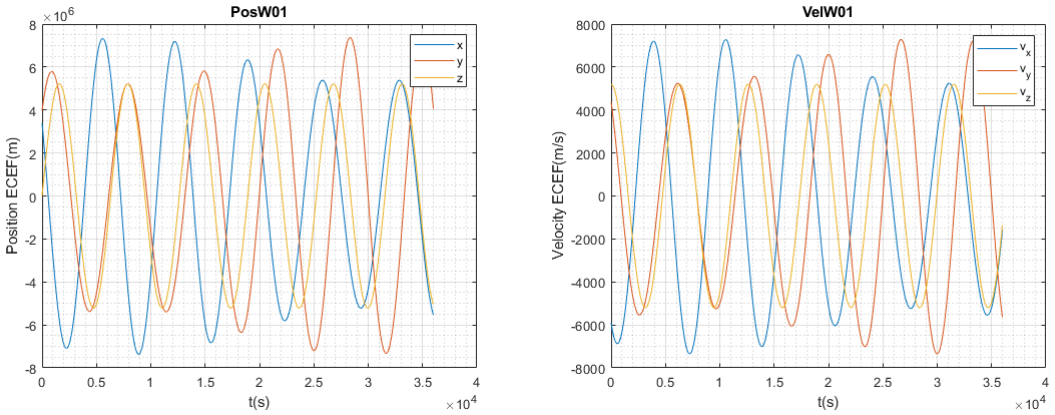
5.2 Post-processing function

For a basic postprocessing, a matlab function has been programmed which reads the data from the output files and plots the different values along the orbit. The plots that can be obtained include those seen in section 4.3, and the position and velocity both in ECI and ECEF (Figures 18 and 19 respectively).



(a) Position (b) Velocity

Figure 18: Position and velocity in ECI



(a) Position (b) Velocity

Figure 19: Position and velocity in ECEF

The code can be found in appendix E.1. The inputs needed for the postprocessing are the directory path where the simulations are stored, the name of the simulation folder for the particular simulation in study, the values of two boolean flags, the former to plot the results and the latter

to initiate a simulink model for a 3D simulation (See section 5.3), the number of the spacecrafts to obtain the data and to simulate in simulink, and the initial time of the simulation in a date vector format (Y/M/D/H/M/S).

Keep in mind that while the plots can be performed for as many spacecrafts as your computer allows, the simulink model is only prepared for two spacecraft.

After defining the inputs, the program loads the output files from 42 into arrays. After that, if the plots are desired, the variables are plotted using matlab's plot functions. Finally, if the simulink simulation is requested, it opens the simulink model after having calculated the quaternions in ECEF, as 42 only outputs the quaternions in ECI and the simulations requires both.

To summarise, the program works in the following way:

1. Define directory path to the output files
2. Define whether plots are desired
3. Define whether 3D simulation is desired
4. Define the spacecrafts desired
5. Define the simulation's initial time
6. Read the date from the output files
7. If 2, plot the results
8. If 3, format variables and open Simulink model

The calculation of the quaternions in ECEF is done by multiplying the quaternion that changes from ECI to ECEF and the quaternion in ECI. The quaternion in ECI is directly obtained from the output files but has to be shifted as matlab and 42 use different notations for quaternions (matlab places the scalar component in the first index of the array whereas 42 places it in the last). The quaternion that changes from ECI to ECEF is obtained using a function from the *aerospace toolkit* with the initial time of the simulation as an input.

5.3 3D simulation

In order to observe the results of the simulation, a 3D simulation using simulink has been made. For that, the cubesat simulation included in the *aerospace toolkit* has been used as a template. This is a simulink block which includes all necessary preparations for a 3D simulation which includes not only the satellites, but also the Earth and the Sun. Using a premade simulation as a tmplate also comes with predefined 3D models for the satellites, which may not be 100% accurate with respect to the satellite being desigend in the Plathon project but serves the purpose of visualizing the attitude changes. The way in which the template has been modified is by adding a second satellite, this required the modification of both the simulink block and the 3D simulation file. The details of the 3D animation block and the modified 3D simulation file can be seen in appendix F

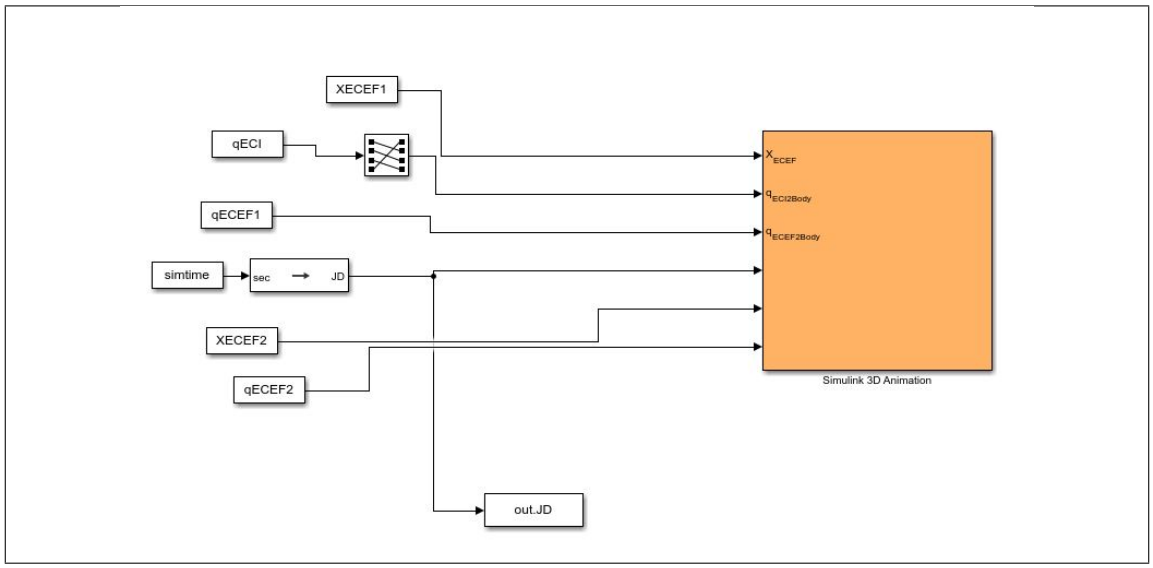


Figure 20: Simulink 3D simulation

In figure 20, the simulink simulation is displayed. The original 3D animation block had only 4 input slots, but in order to allow a two satellite simulation, 2 extra slots have been added. The required inputs are:

- Position of satellite 1 in ECEF
- Quaternion of satellite 1 in ECI
- Quaternion of satellite 1 in ECEF
- Time in JD
- Position of satellite 2 in ECEF
- Quaternion of satellite 2 in ECEF

Note that the quaternion in ECI has to be shifted due to differences in notation between 42 and matlab/simulink.

The model outputs the JD to a matlab array in case it is needed for further postprocessing and produces a 3D animation.

Figure 21: Simulink 3D animation initial moment

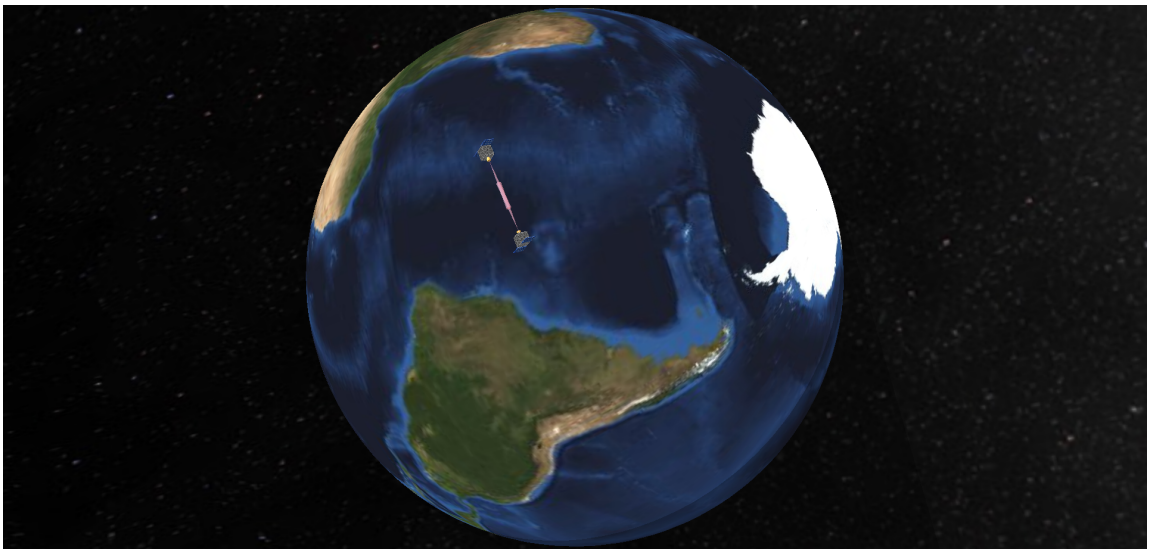


Figure 22: Simulink 3D animation satellites communicating

In figures 21 and 22, two screenshots of a simulation of 2 satellites communicating are presented. In this simulation, the two satellites were instructed to point at each other to simulate a data link via optical communication.

5.4 Final solution

Although initially using 42's GUI was deemed not viable due to the way in which it slowed down the simulation, the proposed Matlab/Simulink alternative presented other problems. First, both the matlab functions and the simulink model were only capable of post-processing and it was necessary for the visualization to be done in real-time. Second, the simulink 3D animation was not easily expandable to multiple satellites due to how 3D animation works in simulink. This presented a problem regarding the visualization of the simulations as the only real-time visualization slowed down the simulation. However, the problem with 42' GUI was found to be hardware-based. As the simulators are inside virtual machines, these do not use the graphical capabilities of any graphics card installed in the host computer but rather the integrated graphics of the CPU. To fix this, the simulator had to be installed directly into the host computer so as to use its graphics card. If that was the case, 42's GUI no longer slowed down the simulation and then it could be used for the real-time visualization of the simulation. The matlab/simulink functions are then only used as post-processing tools.

6 INTER PROCESS COMMUNICATION

In this section, the IPC capabilities of 42 will be explained as well as the proposed solution for the *Plathon* project. First, the different IPC configurations possible with 42 will be explained; then, the standalone *AcApp* will be presented as it plays an important role in the *Plathon* project; and finally, the integration of 42 with *Plathon* will be presented as well as a demonstration simulation made to showcase the capabilities of the simulator.

6.1 IPC configurations

As mentioned previously, one of the input files of the simulator controls the IPC configuration. Using this file, it is possible to allow 42 to send and/or receive information through sockets. This is vital to the *Plathon* project as the long-term objective is to produce a hardware-in-the-loop simulation, meaning that some of the tasks that are originally performed by 42 will be externalised to other software and even to real hardware tests.

There are 7 possible IPC modes for 42, these are:

1. **OFF:** Self-explanatory, it disables the socket.
2. **TX:** Writes data through a socket.
3. **RX:** Reads data through a socket.
4. **TXRX:** Writes and reads data through a socket.
5. **ACS:** Writes and reads ACS data through a socket.
6. **WRITEFILE:** Writes data thorugh a socket into a file.
7. **READFILE:** Reads data through a socket from a file.

It is important to note that the data is sent in different moments of the simulation in ACS mode compared to all others. In ACS mode, 42 sends sensor data through the socket during the ACS step and waits for the actuator data to be received from an external app, whereas the other modes write or read simulation related data before the computations.

For the ACS mode, an ID has to be specified as a unique integer which will be used as an argument for the external app to now which spacecraft has to control.

For the WRITEFILE and READFILE modes, the file name has to be specified in the configuration as well.

Another setup option is defining the socket as either a client, a server or a GMSEC client. The latter allows the simulator to work together with GMSEC, a NASA program which aims to integrate different programs for mission control.

Apart from that, the name and port of the socket must also be specified in the input file as well as two flags, one which allows blocking of the socket and another which writes the traffic in the terminal.

The last configuration needed is the prefix specification. The prefix corresponds to the structures that 42 searches when reading or writing info from/to a socket. Specifying the prefix wanted makes 42 only send/receive the data required. For instance, specifying "SC[0]" as a prefix will make the simulator only send/receive the information corresponding to spacecraft 0.

6.2 Standalone AcApp

When selecting the ACS mode for the IPC configuration, 42 delegates the ACS calculations to an outside app. With 42, comes a small application called *AcApp* which is a means of testing the Standalone ACS capabilities of the simulator. Inside this app, some of the 42's ACS related code is copied and executed in a loop.

In order to execute a simulation with a Standalone AcApp, the IPC input file must be configured in ACS mode. Then, on executing the simulation, 42 will send the sensor data and read the actuator data from an outside app. In this case, to use the Standalone AcApp, it is called from a console with the ID set on the IPC input file as an argument. After that, the AcApp will do any necessary ACS calculation.

The ACS function included in AcApp uses different sensors to calculate the attitude of the spacecraft as well as the sun vector, magnetic field vector, etc. After that, it uses a PD to compute the necessary torque required and sets this as the input for the reaction wheels. Last, it also compensates the momentum gained by the reaction wheels using magnetorquers and orientates the solar panels towards the sun.

In this case, as the models used are 1U cubesats, solar panels are not movable. However this is not an issue for the program.

The main problem of the AcApp is that it does not read commands from 42. That means that the app holds the attitude but cannot make the spacecraft rotate as no command arrives through the socket. Initially, the functions that control the socket traffic were modified to include this data in the communication message, however, as the idea is to control the attitude in real time, instead of sending the data through the socket, it was decided to include a function which reads that data from another different socket to control the cubesat's attitude from another external app.

6.3 Integration with Plathon

As the *Plathon* project aims to produce a hardware-in-the-loop simulation, the function of the 42 simulator is to provide all the data relative to the orbits and environment while delegating all other functions such as ACS, mission design, sensors and actuators to other software or hardware.

With that in mind, the proposed system architecture with regards to 42 is the following:

- A main 42 simulator acting as a conductor
- A 42 simulator in an external host computer acting as a GUI
- An external Standalone AcApp controlling ACS and communicating with the rest of the *Plathon* system

The basic scheme of this system is represented in figure 23. Note how the ACS communication is bidirectional while the socket connecting both 42's only flows in one direction.

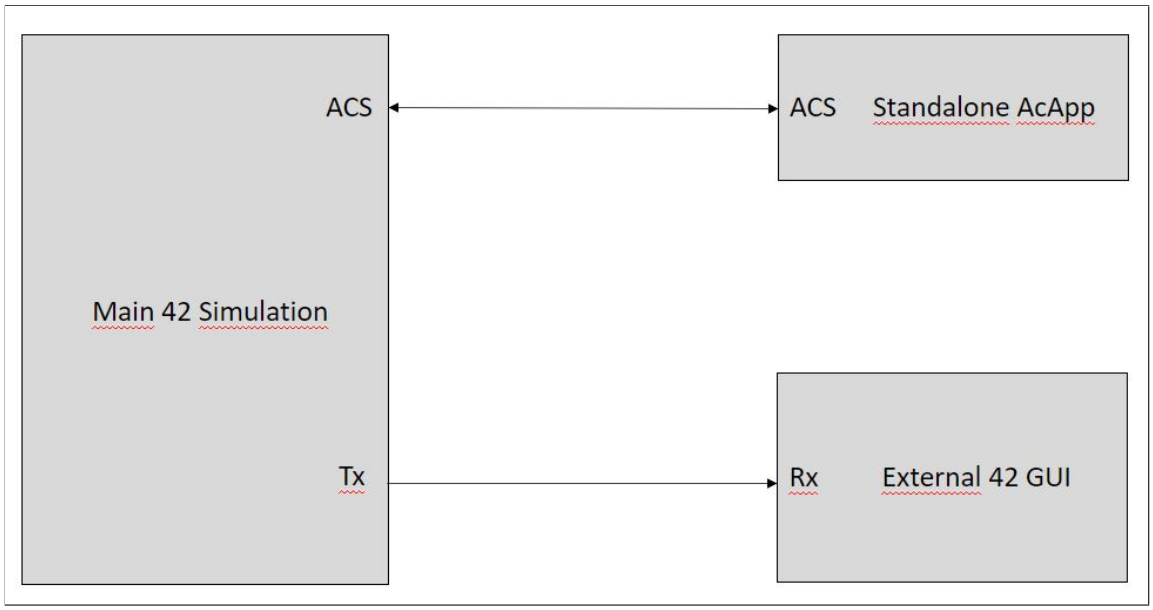


Figure 23: 42 system architecture

When 42 is working in Tx mode, it sends information about the different worlds, such as their position and velocity with respect to the Sun. It also sends information about the reference orbits. Furthermore, it sends information about the spacecrafts present in the simulation such as their position, velocity, momentum, etc. On top of that, for every spacecraft, it sends information regarding their attitude control functions such as the commanded angles, the sensor readings and so on.

When it is working in ACS mode, it only sends information regarding the attitude control of the specified spacecraft, like the sensor readings, and receives the information regarding the actuators of the specified spacecraft such as the commanded angle and actuator torques and forces.

Using this configuration, 42 will simulate with the ACS as an external app through one socket, while sending all the data necessary for another 42 which will be in an external machine with the

42 installed in host whose function will be just to represent the results of the simulation in real-time.

When thinking about the future hardware in the loop simulation, it is important to remark that it is possible to introduce readings from a real sensor or other hardware directly into the main 42 simulation by running it in TxRx mode, as any data received through the socket will overwrite the simulated data before the main simulation sends the sensor readings in ACS mode. It is also possible to modify the *AcApp's* to read that information directly from the hardware, in that way, no intervention of the main simulation will occur in the cases were it is not needed.

6.4 IPC simulation

As mentioned in the previous section, the integration of the simulator with the *Plathon* project was to be done according to figure 23. For that purpose, the proposed simulation was designed as a test for the system's architecture. On top of what has been previously discussed, the simulation's attitude commands were programmed directly into the *AcApp* as a set of instructions. The plan for *Plathon* is to send the commands in real time through another application which would connect directly to the *AcApp* instead of being an input of the main simulation. However, as this application was still not developed and is out of the scope of this project, a simple, temporary solution was found to be sufficient. This solution was to simply add a function on the *AcApp* which would periodically check for the simulation time, which is calculated based on the difference between the current time and the initial time, as *AcApp* does not have direct access to the simulation time, just like many other things, and whenever the time was greater than the value assigned, it would change the command. The modified *AcApp* can be seen in appendix H.

Besides that, the *AcApp* program was changed to meet the characteristics of the spacecrafts present on the simulation as well as the IPC configuration corresponding to the virtual machines used.

Another change that was made to the original plan was the addition of a second spacecraft to the simulation which was controlled by a second *AcApp*. This second spacecraft would receive different commands and would be in very close proximity to the first one so as to be able to see it on the GUI. The system architecture for this simulation can be seen in figure 24.

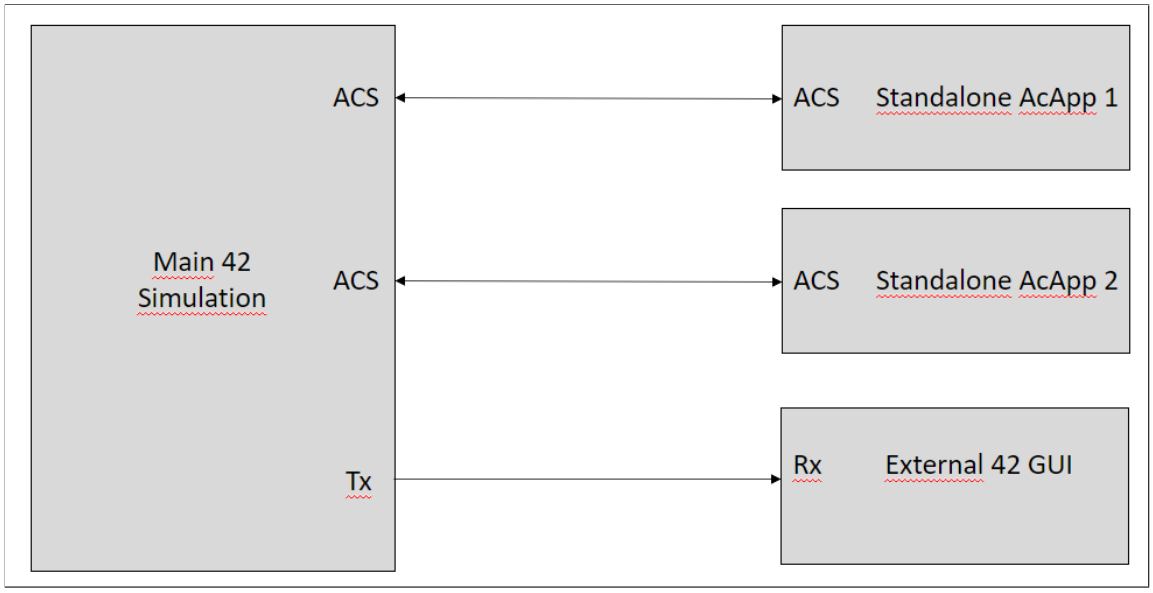


Figure 24: IPC simulation architecture

Due to having 2 different spacecraft's with different commands, as the commands and socket definition are directly programmed into the *AcApp*, two different programs had to be compiled independently, thus the projects makefile also had to be modified to include this second *AcApp* in its set of instructions. This file can be seen in appendix I.

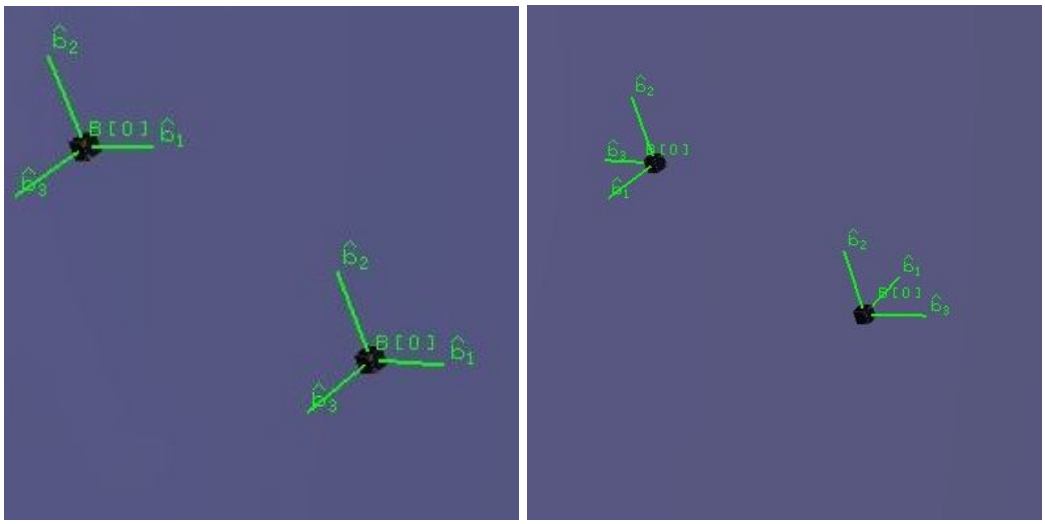
Although this simulation is simple in terms of manoeuvres and is not a full-on constellation, it paves the way to more complex simulations in the future which may be performed following the defined architecture. The simulation serves as a demonstration of the IPC capabilities of the simulator and as a proof of concept for the *Plathon* project architecture. The most important input files for the main simulation are found in appendix J, some input files have been omitted as they are not important or handled externally (such as the graphics or commands).

Figure 25: IPC simulation cam view

Figure 26: IPC simulation map view

In figures 25 and 26, a view of both the cam and the map respectively can be seen. Both satellites share the same reference orbit but are in close proximity as they have been defined as a formation. This has been done for demonstration purposes as having the satellites separated at distances similar to those found in constellation would have made the visualization of both of them impossible.

It is also important to highlight that for this simulation, most planets and other celestial bodies have been deactivated as by having two spacecrafts sending data through the same socket plus having all the celestial bodies saturated the socket causing overflow issues. This could have been avoided by sending the data through different sockets but the ports on the host machine were closed, making that impossible. Nevertheless, this should be an easy fix for the future as changing the socket configuration is relatively simple.



(a) Initial attitude (b) Final attitude

Figure 27: IPC simulation spacecraft attitude change

In figure 27, both satellites are displayed with their body axes to see the change in attitude. As seen on the left, both satellites begin the simulation with the same attitude; at the end, as seen on the right, they have performed a 90 degree rotation in opposite directions. These attitude changes have been controlled externally to the main simulation through both of the *AcApp* independently.

Keep in mind that the visualization is also being provided externally in another machine which was running another 42 simulating in Rx mode, thus not really simulating but representing the data coming in through the socket. The data flow between the application can be seen in figure 28.

Figure 28: IPC simulation data traffic between applications

The two consoles on the top correspond to the 42's while the ones on the bottom correspond to the *AcApp's*. The console on the top right corner is the main simulation, which sends all the information that is being simulated through the socket. The console on the top left corner is the GUI, which receives the information and runs the graphical interface. The one on the bottom right is the first *AcApp* which controls the first spacecraft, hence why it only sends data regarding spacecraft 0. Finally, the console on the bottom left is analogous to the previous one but for the second spacecraft.

With this simulation, the basic *Plathon* architecture was established for its use in future projects with more realistic simulations and for its implementation with other software such as CFS for the completion of the *Plathon* project as a whole.

7 SCHEDULE, BUDGET AND ENVIRONMENTAL IMPACT

In this section, the project's schedule, budget and environmental impact will be discussed. For the schedule, the different activities done during the realization of the project are explained and the number of hours assigned are represented together with a Gantt chart; in the case of the budget, the costs of this project are detailed; and last, the environmental impact of the project is also explained.

7.1 Schedule

As stated in the course guide for the Master's thesis, this project will have a schedule of 300 hours. In table 5, the hours are split into different activities which have been performed during the term. The tasks enumerated have the following description:

- Documentation: Gathering of information about the simulator, programming or any other required information needed to fulfill the objective of the project
- Program comprehension: Time spent studying how the simulator works and what has to be changed in order to obtain the desired performance/behaviour
- Program tests: simple simulations done to learn how to implement simulations and what the different parameters are used for
- Postprocessing subroutine: Implementation of a Matlab code to obtain different graphs/information about the simulation's result
- FSW subroutine: programming of a new FSW function customised to a cubesat and testing of said function
- Satellite visibility simulation: design of a simulation in which two satellites point at each other together with the necessary modifications to 42 and the creation of a Simulink model to visualise the results
- Constellation simulation: creation of a Matlab code to generate the input files for a constellation both from scratch and from TLE data and simulation of the Iridium constellation in 42
- Integration with Plathon: Programming of the necessary modifications of the both the simulator and the AcApp for its use in the Plathon project and demonstration simulation
- Report: writing of the report
- Presentation: preparation of the presentation of the project

Activity	Duration (h)
Documentation	15
Program comprehension	30
Program tests	20
Postprocessing subroutine	20
FSW subroutine	40
Satellite visibility simulation	40
Constellation simulation	20
Integration with Plathon	60
Report	35
Presentation	20
TOTAL	**300**

Table 5: Hours spent per activity

To have a better idea of the amount of time spent with regards to the total, a pie chart is presented in figure 29.

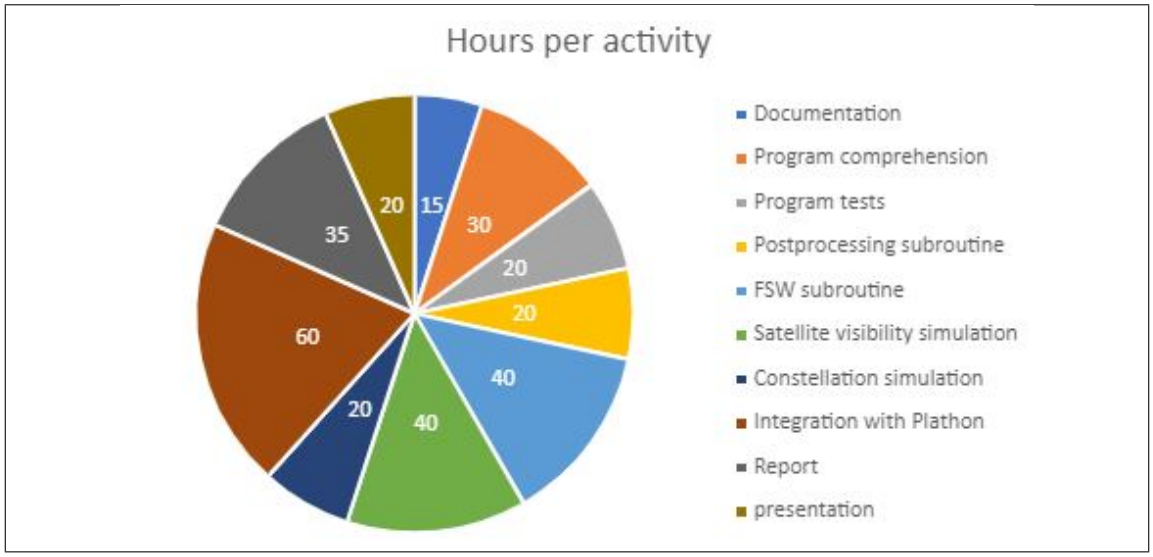


Figure 29: Pie chart: hours spent per activity

These activities have been distributed during the term in the manner shown in figure 30. In the chart, some of the activities are shown to have been done in parallel as in the beginning, the documentation, program comprehension, tests and other activities could be performed at once. After that initial phase, the rest of the activities are done sequentially with the exception of the report which has been done as the project advanced. The presentation is the last task to be performed after the report's deadline.

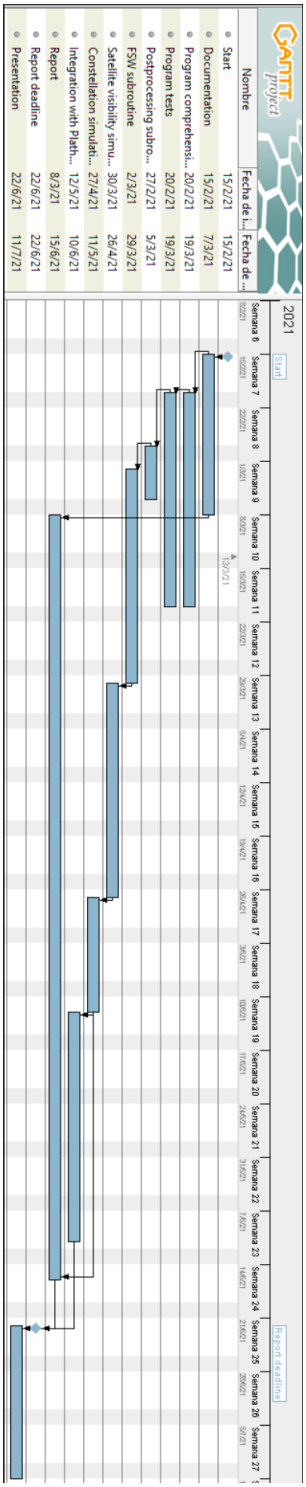


Figure 30: Gantt chart of the project

7.2 Budget

For this project, the only variables for computing the cost were the electricity used and the salary. With regards to the salary, the cost per hour has been calculated using the average salary for a scientific professional from [15], which is 25.42 €/h for a male and 21.54 €/h for a female. In this case, the value taken will be for a male but it is important to note as a society that a 15% difference between the average salary of a male and a female for the same occupation is unacceptable and it should be addressed with urgency.

Besides that, the cost of electricity corresponds to the use of different computers. The computer used has a power supply of 750W but most of the computations were performed by a virtual machine in a server owned by *Plathon*. In some cases, up to 2 virtual machines were used at once (IPC simulation). So, accounting for these variables, for the activities which only required the host computer, as they are not computationally demanding, the power consumption considered is about 50W. Then for the main virtual machine used, an extra 500W will be considered as it was performing intensive computations. The second virtual machine was used as a GUI display, which only uses graphical power, thus an extra 200W will be taken into account.

As for the electricity cost, using [16], a rough average cost has been estimated to be 0.12 €/kWh.

The host computer was on for the entirety of the project but both virtual machines were only on for specific activities. In the case of the main virtual machine, it was on during the program tests, FSW subroutine, satellite visibility simulation, constellation simulation and integration with Plathon; whereas the second virtual machine was only on during the integration with Plathon. The total cost for the project can be seen in table 6.

Concept	Power consumption (W)	Electricity cost (€/kWh)	Unit price (€/h)	Units (h)	Total price (€)
Salary	-	-	25.42	300	7626
Host PC	50	0.12	0.006	300	1.8
Main virtual machine	500	0.12	0.06	180	10.8
Second virtual machine	200	0.12	0.024	60	1.44
TOTAL					**7640.04**

Table 6: Project's budget

7.3 Environmental impact

The environmental impact of a project based on programming is relatively low. The carbon emissions for the project were caused by the electricity used. According to [17], the average CO2 emissions per MWh were about 0.11 t/MWh. In table 7, the environmental impact of this project can be consulted.

Concept	Emissions (t CO2/MWh)	Consumption (kWh)	Total carbon emissions (kg CO2)
Host PC	0.11	15	1.65
Main virtual machine	0.11	90	9.9
Second virtual machine	0.11	12	1.32
TOTAL			**12.87**

Table 7: Project's environmental impact

8 CONCLUSIONS

In this section, the project's conclusions as well as some recommendations for future studies will be presented. In the conclusions, the different results obtained will be commented while in the recommendation and future studies section, some idea for the continuation of the project with regards to *Plathon* will be given.

8.1 Conclusions

The orbital simulator 42 is a powerful tool for orbit propagation and spacecraft dynamics simulation. However, being an open source program means it is designed as a basis for more complex analyses. In the case of the *Plathon* project, the idea was to make a hardware in the loop simulation using 42 just to obtain orbital data such as GPS position.

Due to the demands of the project as a whole, the idea behind this particular project was to adapt this simulator to the needs of the bigger *Plathon* project. For that, a series of modifications have been made to the base code and an architecture consisting on two different simulators, one for the base simulation and another for the GUI visualization were prepared, as well as two standalone *AcApp's* which would be controlling the attitude of two separate spacecrafts.

Besides that, the capabilities of the simulator were tested for constellations of satellites, in particular, the Iridium simulation, as the goal of the *Plathon* project is to test constellations of nanosatellites.

Also, some postprocessing tools have been developed to further asses the results of the simulations performed as well as some Matlab codes to automate the generation of some inputs for the simulator as writing them by hand may become too much of a burden as the number of satellites present in the simulation increases.

Overall the results of the project have been satisfactory as the implementation of the simulator into the architecture of the *Plathon* project has been successful. But there is still much to do in order to fully achieve the requirements of the *Plathon* project as whole. These extra step needed for the fulfillment of the *Plathon* project's requirements will be discussed in section 8.2.

However, although the *Plathon* project still needs some work, this project has paved the way for future studies using the simulator 42 as it establishes the basis for future work, having prepared the simulator for its standalone use and by having determined the way the simulator operates.

To conclude, this project has opened the door to new possibilities regarding 42 and its use in the *Plathon* project by preparing the simulator for its integration with different software through communication via socket and provides some other tools such as preprocessing and postprocessing. It also complements the lacking documentation of the 42 simulator with some useful insight into the program's inner workings.

As a summary of everything accomplished with this project:

- **General knowledge about 42**: The program has been studied extensively via reading the code and performing tests. It has been concluded that it has a lot of potential but still needs some development behind as some parts such as the sensor and actuator modelling have some shortcomings that should be addressed.
- **Constellation simulations**: After studying the implications of constellations of nanosatellites and their benefits and problems, it has been concluded that despite issues with space debris, the benefits they provide outweigh their downsides. Then, a Matlab program to generate Walker constellations has been implemented and the resulting input files, tested. Following that, another Matlab program which obtains TLE data from a webpage and obtains the orbital parameters to generate the inputs was made. However, due to the fact that TLE data was from different epochs, the TLE were finally run through 42 directly as it propagated the orbits to the specified epoch. This method was used to simulate the Iridium constellation to showcase the capabilities of 42 to simulate large constellations.
- **Attitude control**: The default 42 fsw functions were studied and deemed inappropriate for the spacecraft models *Plathon* is aiming for. Thus, an attitude control function was implemented to be used for a cubesat. Although the idea behind *Plathon* is to make a hardware in the loop simulation with real cubesats, only a limited number of cubesats will be constructed, which leaves the remaining cubesats in the constellation to be controlled by the main 42 simulation. While programming this fsw function, some extra attitude commands were programmed into 42, showing that it is possible to alter the behaviour of the simulation as necessary.
- **Postprocessing**: First, the different functionalities of 42's GUI were checked. As in the beginning, the simulations using the GUI were several times slower than real time, some postprocessing tools were developed. However, this issue was found to be hardware limited due to the use of virtual machines without graphical power. Hence, in the end the preferred method for visualisation of the simulation was using the GUI. Nevertheless, the postprocessing tools developed may still be useful for data analysis. The first tool developed was a Matlab script which takes the output files from 42 and graphs all the data requested. After that, using Simulink, a 3D visualization was created which allowed to see the movement of the satellites (which were scaled up so that they can be seen) around the Earth with which a simulation about two satellites pointing at each other was made. Furthermore, in order to obtain the data from more than one spacecraft, the 42's report function had to be modified too as it originally reported only the data from the first spacecraft. This was found to be due to memory use as getting the data from multiple spacecrafts at once may require a lot of RAM, especially as the number of spacecrafts increases such as with constellations.

- **IPC and *Plathon* integration**: The different IPC configurations for 42 were studied in order to see how to properly configure sockets to allow 42 to exchange information with several external simulations or standalone *AcApp's*. When the architecture for the *Plathon* simulation was defined, the need to modify the basic *AcApp* arose. This application was modified in order to adapt it to the desired spacecraft and socket configuration as well as adding a way to command attitude changes to the spacecraft being controlled. In the mean time, the way to command attitude changes is temporary as it will be controlled by another program connecting directly to the *AcApp*, thus the solution is rather simple, but is working

as intended nonetheless. Besides that, as 2 *AcApp's* will be running at the same time with different spacecrafts which will be getting different commands, the compiler Makefile had to be modified too to allow the compilation of a second, different *AcApp*. Finally, a demonstration simulation was prepared which had a 42 simulating everything in Tx mode and sending the data to another 42 in Rx mode to perform the visualization in another machine and to 2 external *AcApp's* controlling the attitude of two separate spacecraft.

8.2 Recommendations and future studies

As mentioned, this project is part of the bigger *Plathon* project. Said project still needs to be developed as the efforts done with several bachelor's and master's thesis during this term only mark the beginning of the journey.

During the realization of this project, some aspects have been identified which will benefit from further study. Moreover, with the results of this project, some paths have been opened for development regarding *Plathon*.

With regards to the simulator itself, the models for both the sensors and actuators are really simple and would benefit from a more detailed model, especially those which will be needed for the *Plathon* project such as the GPS receivers (other sensors or actuators may be included as hardware).

Besides that, as simulating a whole constellation of satellites using standalone applications would not be viable, some satellites will have to be controlled from the main 42 simulation, for that, a study about a more complex attitude control function which not only controls the attitude of the spacecraft on demand but also point the solar arrays to the Sun when needed, or point the antennas to the ground stations or other satellites may be interesting to develop.

With regards to the integration with *Plathon*, the next logical step is to integrate the standalone *AcApp's* with other software such as CFS in order to control the mission planning in real time and allow the integration of the hardware in the loop simulation with the rest of the software.

Another possible improvement is to create a better model of the spacecraft akin to the real cubesat being developed so as to make the any simulation of the spacecraft's dynamics more realistic.

It is also heavily recommended to study the communications between machines as it plays a huge role in the simulations as every aspect of it will be controlled by a different device, especially when the *Plathon* project has been fully realized.

References

[1] E. Stoneking, "What is 42 ? • A simulation of spacecraft attitude and orbital dynamics and control • Intended for use from concept studies through ops," 2017.

[2] "Nanosats_total_2021-01-01_large.png (2560×1440)." [Online]. Available: https://www.nanosats.eu/img/fig/Nanosats_total_2021-01-01_large.png

[3] "Nanosats_status_2021-01-01_large.png (2560×1440)." [Online]. Available: https://www.nanosats.eu/img/fig/Nanosats_status_2021-01-01_large.png

[4] E. Stoneking, "Flight Regimes," pp. 1–5, 2015.

[5] J. R. Wertz, *Orbit and Constellation Design and Management (Space Technology Library)*. Microcosm, Inc, 2001. [Online]. Available: http://gen.lib.rus.ec/book/index.php?md5=03441da51aecb122ae00919fe29fc6f8

[6] J. G. Walker, "CONTINUOUS WHOLE-EARTH COVERAGE BY CIRCULAR ORBIT SATELLITE PATTERNS," 1977.

[7] M. Guan, T. Xu, F. Gao, W. Nie, and H. Yang, "Optimal Walker Constellation Design of LEO-Based Global Navigation and Augmentation System," *Remote Sensing*, vol. 12, no. 11, p. 1845, jun 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/11/1845

[8] "Nanosats Database — Constellations, companies, technologies and more." [Online]. Available: https://www.nanosats.eu/

[9] "ESA - Space debris by the numbers." [Online]. Available: https://www.esa.int/Safety_Security/Space_Debris/Space_debris_by_the_numbers

[10] "ARES — Orbital Debris Program Office." [Online]. Available: https://orbitaldebris.jsc.nasa.gov/

[11] "ESA - Clean Space." [Online]. Available: https://www.esa.int/Safety_Security/Clean_Space

[12] "Global Network — Iridium Satellite Communications." [Online]. Available: https://www.iridium.com/network/globalnetwork/#overview-section

[13] "CelesTrak: NORAD Two-Line Element Set Format." [Online]. Available: https://celestrak.com/NORAD/documentation/tle-fmt.php

[14] E. Stoneking, "Global Variables and the FSW Structure Flight Software Models," pp. 3–6, 2017.

[15] "Idescat. Indicadors anuals. Salari brut anual i guany per hora. Per sexe i tipus d'ocupació." [Online]. Available: https://www.idescat.cat/indicadors/?id=anuals&n=10403

[16] "PVPC — ESIOS electricidad · datos · transparencia." [Online]. Available: https://www.esios.ree.es/es/pvpc

[17] "REData - No renovables detalle emisiones CO2 — Red Eléctrica de España." [Online]. Available: https://www.ree.es/es/datos/generacion/no-renovables-detalle-emisiones-CO2

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

Study of orbital propagator 42 for missions based on constelations of nanosatellites

Contents:
APPENDIX

Master's thesis
MUEA

Lluís Montilla Rodríguez

Supervisor: David González Díez — Delivery date: 22/06/2021

MUEA - ESEIAAT
Universitat Politècnica de Catalunya (UPC)

Contents

Appendix A INPUT FILE EXAMPLES **5**
A.1 Inp_Sim . . . 5
A.2 Inp_Cmd . . . 7
A.3 Inp_Graphics . . . 9
A.4 Orb_* . . . 11
A.5 SC_* . . . 13
A.6 Inp_FOV . . . 18
A.7 Inp_IPC . . . 20
A.8 Inp_Region . . . 22
A.9 Inp_TDRS . . . 23

Appendix B CONSTELLATION INPUT GENERATOR **24**
B.1 Constellation generator . . . 24
B.2 WriteOrbFile . . . 25
B.3 WriteSim . . . 27

Appendix C PROGRAMMING A FSW FUNCTION IN 42 **30**
C.1 Programming process . . . 30
C.2 Cubesat FSW routine . . . 33

Appendix D ADDED SIMULATION COMMANDS **34**
D.1 Point at the Sun when exiting eclipse . . . 34

Appendix E MATLAB POSTPROCESSING FUNCTIONS **36**
E.1 Basic postprocessing . . . 36

Appendix F SIMULINK 3D MODEL **40**
F.1 Simulink 3D simulation block detail . . . 40
F.2 3D animation file . . . 44

Appendix G IRIDIUM SIMULATION **51**
G.1 Iridium generator . . . 51
G.2 txt2array . . . 52
G.3 TLE . . . 53
G.4 WriteOrbFileTLE . . . 56
G.5 WriteSCFile . . . 58
G.6 WriteSimMultSC . . . 64

Appendix H MODIFIED ACAPP **67**

Appendix I MODIFIED MAKEFILE **81**

Appendix J IPC SIMULATION INPUTS **91**
J.1 Inp_IPC 91
J.2 Inp_Sim 93
J.3 Orb_* 95
J.4 SC_*1 97
J.5 SC_*2 104

List of Figures

31 3D simulation block 41
32 3D simulation block top detail 42
33 3D simulation block bottom detail 43

Appendix A INPUT FILE EXAMPLES

In this section, some example input files are presented to show the format of said files.

A.1 Inp_Sim

```
<<<<<<<<<<<<<<<<<<  42: The Mostly Harmless Simulator  >>>>>>>>>>>>>>>>>>
************************* Simulation Control
    *************************
FAST                                ! Time Mode (FAST, REAL, or EXTERNAL)
3600.0       0.1                   ! Sim Duration, Step Size [sec]
10.0                                ! File Output Interval [sec]
TRUE                                ! Graphics Front End?
Inp_Cmd.txt                         ! Command Script File Name
************************* Reference Orbits
    *************************
1                                   ! Number of Reference Orbits
TRUE   Orb_test.txt                 ! Input file name for Orb 0
***************************** Spacecraft
    *****************************
1                                   ! Number of Spacecraft
TRUE  0 SC_test.txt                 ! Existence, RefOrb, Input file for SC 0
***************************** Environment
    *****************************
02 04 2021                          ! Date (UTC) (Month, Day, Year)
12 49 00.00                         ! Time (UTC) (Hr,Min,Sec)
37.0                                ! Leap Seconds (sec)
NOMINAL                             ! F10.7, Ap (USER, NOMINAL or
    TWOSIGMA)
230.0                               ! If USER_DEFINED, enter desired F10.7
     value
100.0                               ! If USER_DEFINED, enter desired AP
    value
IGRF                                ! Magfield (NONE,DIPOLE,IGRF)
8    8                              ! IGRF Degree and Order (<=10)
2    0                              ! Earth Gravity Model N and M (<=18)
2    0                              ! Mars Gravity Model N and M (<=18)
2    0                              ! Luna Gravity Model N and M (<=18)
FALSE   FALSE                       ! Aerodynamic Forces & Torques (
    Shadows)
FALSE                               ! Gravity Gradient Torques
FALSE   FALSE                       ! Solar Pressure Forces & Torques (
    Shadows)
FALSE                               ! Gravity Perturbation Forces
FALSE                               ! Passive Joint Torques
FALSE                               ! Thruster Plume Forces & Torques
```

```
FALSE                                   !  RWA Imbalance Forces and Torques
FALSE                                   !  Contact Forces and Torques
FALSE                                   !  CFD Slosh Forces and Torques
FALSE                                   !  Output Environmental Torques to
    Files
******************** Celestial Bodies of Interest
    ********************
MEAN                                    !  Ephem Option (MEAN or DE430)
TRUE                                    !  Mercury
TRUE                                    !  Venus
TRUE                                    !  Earth and Luna
TRUE                                    !  Mars and its moons
TRUE                                    !  Jupiter and its moons
TRUE                                    !  Saturn and its moons
TRUE                                    !  Uranus and its moons
TRUE                                    !  Neptune and its moons
TRUE                                    !  Pluto and its moons
FALSE                                    !  Asteroids and Comets
***************** Lagrange Point Systems of Interest *****************
FALSE                                   !  Earth-Moon
FALSE                                   !  Sun-Earth
FALSE                                   !  Sun-Jupiter
************************ Ground Stations ************************
5                                                   ! Number of Ground
    Stations
TRUE  EARTH  -77.0   37.0   "GSFC"                  ! Exists, World, Lng, Lat,
     Label
TRUE  EARTH  -155.6  19.0   "South Point"           ! Exists, World, Lng, Lat,
     Label
TRUE  EARTH  115.4   -29.0  "Dongara"               ! Exists, World, Lng, Lat,
     Label
TRUE  EARTH  -71.0   -33.0  "Santiago"              ! Exists, World, Lng, Lat,
     Label
TRUE  LUNA   45.0    45.0   "Moon Base Alpha"       ! Exists, World, Lng, Lat,
     Label
```

A.2 Inp_Cmd

```
<<<<<<<<<<<<<<<<<<  42: Command Script File  >>>>>>>>>>>>>>>>>>
0.0 SC[0] qrl = [0.0  0.0  0.0  1.0]
EOF

###############################################################################
# All lines after EOF are ignored
# Comment lines begin with #, %, or //
# Blank lines are permitted

# Here are recognized command formats.
#  %lf means that a floating-point number is expected
#  %ld means that an integer is expected
#  %s means that a string is expected
#  %c means that a character is expected
#  Look in functions SimCmdInterpreter, GuiCmdInterpreter,
#     and FswCmdInterpreter for strings and characters that
#     are meaningful in a particular context
#  The first %lf is always the SimTime of command execution.

# Sim-related commands
%lf DTSIM = %lf
%lf SC[%ld].RotDOF %s
%lf SC[%ld].G[%ld].RotLocked[%ld] %s
%lf SC[%ld].G[%ld].TrnLocked[%ld] %s
%lf Impart Impulsive Delta-V of [%lf %lf %lf] m/s in Frame %c to Orb[%
    ld]
   %c can be N or L
%lf SC[%ld].LoopGain = %lf
%lf SC[%ld].LoopDelay = %lf
%lf SC[%ld].GainAndDelayActive = %s

# GUI-related commands
%lf POV.Host.SC %ld
%lf CaptureCam %s
%lf CamSnap %s
%lf MapSnap %s
%lf Banner = "Banner in Quotes"
%lf GL Output Step = %lf
%lf POV CmdRange = %lf
%lf POV CmdSeq = %ld
%lf POV CmdAngle = [%lf %lf %lf] deg
%lf POV CmdPermute = [%lf %lf %lf; %lf %lf %lf; %lf %lf %lf]
%lf POV TimeToGo = %lf
%lf POV Frame = %c
```

```
%lf ShowHUD %s
%lf ShowWatermark %s
%lf ShowShadows %s
%lf ShowProxOps %s
%lf ShowFOV %s
%lf FOV[%ld].NearExists = %s
%lf FOV[%ld].FarExists = %s

# FSW-related commands
%lf SC[%ld] FswTag = %s
   # %s is PASSIVE_FSW, PROTOTYPE_FSW, etc.
%lf SC[%ld] qrn = [%lf %lf %lf %lf]
%lf SC[%ld] qrl = [%lf %lf %lf %lf]
%lf SC[%ld] Cmd Angles = [%lf %lf %lf] deg, Seq = %ld wrt %c Frame
   # %c is either N or L
%lf SC[%ld].G[%ld] Cmd Angles = [%lf %lf %lf] deg
#  In the following, the (first) %s is either "Primary" or "Secondary"
%lf Point SC[%ld].B[%ld] %s Vector [%lf %lf %lf] at RA = %lf deg, Dec =
    %lf deg
%lf Point SC[%ld].B[%ld] %s Vector [%lf %lf %lf] at World[%ld] Lng = %
   lf deg, Lat = %lf deg, Alt = %lf km
%lf Point SC[%ld].B[%ld] %s Vector [%lf %lf %lf] at World[%ld]
%lf Point SC[%ld].B[%ld] %s Vector [%lf %lf %lf] at GroundStation[%ld]
%lf Point SC[%ld].B[%ld] %s Vector [%lf %lf %lf] at %s
   # Last %s is SUN, MOON, any planet, VELOCITY, or MAGFIELD
%lf Point SC[%ld].B[%ld] %s Vector [%lf %lf %lf] at SC[%ld]
%lf Point SC[%ld].B[%ld] %s Vector [%lf %lf %lf] at SC[%ld].B[%ld]
    point [%lf %lf %lf]
%lf Align SC[%ld].B[%ld] %s Vector [%lf %lf %lf] with %c-frame Vector
    [%lf %lf %lf]
   # %c-frame can be H, N, or L
%lf Align SC[%ld].B[%ld] %s Vector [%lf %lf %lf] with SC[%ld].B[%ld]
    vector [%lf %lf %lf]
%lf SC[%ld].Thr[%ld] %s
   %s is OFF or ON
Event Eclipse Entry SC[%ld] qrl = [%lf %lf %lf %lf]
Event Eclipse Exit SC[%ld] qrl = [%lf %lf %lf %lf]
Event Eclipse Entry SC[%ld] Cmd Angles = [%lf %lf %lf] deg, Seq = %ld
    wrt %c Frame
   # %c is either N or L
Event Eclipse Exit SC[%ld] Cmd Angles = [%lf %lf %lf] deg, Seq = %ld
    wrt %c Frame
   # %c is either N or L
%lf Set SC[%ld] RampCoastGlide wc = %lf Hz, amax = %lf, vmax = %lf
%lf Spin SC[%ld] about Primary Vector at %lf deg/sec
```

A.3 Inp_Graphics

```
<<<<<<<<<<<<<<<<  42 Graphics Configuration File  >>>>>>>>>>>>>>>>>>>
1.0                                  !  GL Output Interval [sec]
Skymap09.txt                         !  Star Catalog File Name
TRUE                                 !  Map Window Exists
TRUE                                 !  Orrery Window Exists
TRUE                                 !  Unit Sphere Window Exists
****************************** POV *******************************
FALSE                                !  Pause at Startup
TRACK_HOST                           !  POV Mode (TRACK_HOST,
    TRACK_TARGET, FIXED_IN_HOST)
SC                                   !  Host Type (WORLD, REFORB, FRM, SC
    , BODY)
0   0   L                            !  Initial Host SC, Body, POV Frame
SC                                   !  Target Type (WORLD, REFORB, FRM,
    SC, BODY)
0   0   L                            !  Initial Target SC, Body, POV
    Frame
POS_Z                                !  Boresight Axis
NEG_Y                                !  Up Axis
40.0                                 !  Initial POV Range from Target [m]
30.0                                 !  POV Angle (Vertical) [deg]
0.0   0.0   0.0                      !  POV Position in Host [m]
FRONT                                !  Initial POV View (FRONT,
    FRONT_RIGHT, etc)
****************************** CAM *******************************
"42 Cam"                             ! Cam Title [delimited by "]
800   800                            !  Width, Height [pixels]
5.0E-5                               !  Mouse Scale Factor
4.0                                  !  Display's Gamma Exponent
    (1.8-4.0)
************************ CAM Show Menu ***************************
FALSE "N Axes"                       !  Show N Axes
FALSE "L Axes"                       !  Show L Axes
FALSE "F Axes"                       !  Show F Axes
FALSE "B Axes"                       !  Show B Axes
FALSE "N Grid"                       !  Show N Grid
FALSE "L Grid"                       !  Show L Grid
FALSE "F Grid"                       !  Show F Grid
FALSE "B Grid"                       !  Show B Grid
FALSE "Gal Grid"                     !  Show B Grid
FALSE  "FOVs"                        !  Show Fields of View
FALSE "Prox Ops"                     !  Show Prox Ops
FALSE "TDRS"                         !  Show TDRS Satellites
TRUE  "Shadows"                      !  Show Shadows
```

```
FALSE "Astro Labels"                    !  Show Astro Labels
FALSE "Truth Vectors"                   !  Show Truth Vectors
FALSE "FSW Vectors"                     !  Show FSW Vectors
TRUE  "Milky Way"                       !  Show Milky Way
FALSE "Fermi Sky"                       !  Show Fermi Sky
****************************** MAP ******************************
"42 Map"                                ! Map Title [delimited by "]
512  256                                !  Width, Height [pixels]
************************ MAP Show Menu ************************
TRUE  "Clock"                           !  Show Clock
TRUE  "Tlm Clock"                       !  Show Clock
FALSE "Credits"                         !  Show Credits
TRUE  "Night"                           !  Show Night
******************** Unit Sphere Show Menu ********************
TRUE                                    !  Show Major Constellations
TRUE                                    !  Show Zodiac Constellations
FALSE                                   !  Show Minor Constellations
```

A.4 Orb_*

```
<<<<<<<<<<<<<<<<<<  42: Orbit Description File   >>>>>>>>>>>>>>>>>>
Test orbit              !  Description
CENTRAL                          !  Orbit Type (ZERO, FLIGHT, CENTRAL,
    THREE_BODY)
::::::::::::::  Use these lines if ZERO          ::::::::::::::::
MINORBODY_2                      !  World
FALSE                            ! Use Polyhedron Gravity
::::::::::::::  Use these lines if FLIGHT   ::::::::::::::::
0                                !  Region Number
FALSE                            ! Use Polyhedron Gravity
::::::::::::::  Use these lines if Body-Centered Orbit
    ::::::::::::::::
EARTH                      !  Orbit Center
FALSE                            !  Secular Orbit Drift Due to J2
KEP                              !  Use Keplerian elements (KEP) or (RV)
    or FILE
PA                               !  Use Peri/Apoapsis (PA) or min alt/ecc
    (AE)
35786.0     35786.0                    !  Periapsis & Apoapsis Altitude,
     km
200.0   2.0                      !  Min Altitude (km), Eccentricity
0.0                             !  Inclination (deg)
0.0                            !  Right Ascension of Ascending Node (deg)
0.0                             !  Argument of Periapsis (deg)
0.0                            !  True Anomaly (deg)
0.0   0.0   0.0                  !  RV Initial Position (km)
0.0   0.0   0.0                  !  RV Initial Velocity (km/sec)
TRV   "ORB_ID"                   !  TLE or TRV format, Label to find in
    file
"TRV.txt"                        !  File name
::::::::::::  Use these lines if Three-Body Orbit  ::::::::::::::
SUNEARTH                         !  Lagrange system
LAGDOF_MODES                     !  Propagate using LAGDOF_MODES or
    LAGDOF_COWELL or LAGDOF_SPLINE
MODES                            !  Initialize with MODES or XYZ or FILE
L2                               !  Libration point (L1, L2, L3, L4, L5)
800000.0                         !  XY Semi-major axis, km
45.0                             !  Initial XY Phase, deg
CW                               !  Sense (CW, CCW), viewed from +Z
0.0                              !  Second XY Mode Semi-major Axis, km (L4
    , L5 only)
0.0                              !  Second XY Mode Initial Phase, deg (L4,
     L5 only)
CW                               !  Sense (CW, CCW), viewed from +Z (L4,
```

```
   L5 only)
400000.0                        !  Z Semi-axis, km
60.0                            !  Initial Z Phase, deg
1.05   0.5   0.0                !  Initial X, Y, Z (Non-dimensional)
0.0    0.0   0.0                !  Initial Xdot, Ydot, Zdot (Non-
   dimensional)
TRV  "ORB_ID"                   !  TLE, TRV or SPLINE format, Label to
   find in file
"TRV.txt"                       !  File name
****************** Formation Frame Parameters ***********************
N                               !  Formation Frame Fixed in [NL]
0.0   0.0   0.0   123           !  Euler Angles (deg) and Sequence
N                               !  Formation Origin expressed in [NL]
0.0   0.0   0.0                 !  Formation Origin wrt Ref Orbit (m)
```

A.5 SC_*

```
<<<<<<<<<<<<<<<<<<  42: Spacecraft Description File  >>>>>>>>>>>>>>>>>>
1-U Cubesat                         !  Description
"Cube 1"                            !  Label
GenScSpriteAlpha.ppm                !  Sprite File Name
PASSIVE_FSW                         !  Flight Software Identifier
0.2                                 !  FSW Sample Time, sec
************************ Orbit Parameters ****************************
ENCKE                               !  Orbit Prop FIXED, EULER_HILL, or ENCKE
CM                                  !  Pos of CM or ORIGIN, wrt F
    0.00000    0.00000    0.00000 ! Pos wrt F
    0.00000    0.00000    0.00000 ! Vel wrt F
************************** Initial Attitude
    ***************************
NAN                                 ! Ang Vel wrt [NL], Att [QA] wrt [NLF]
0.0     0.0     0.0                 ! Ang Vel (deg/sec)
0.0     0.0     0.0     1.0         ! Quaternion
0.0     0.0     0.0     123         ! Angles (deg) & Euler Sequence
**************************  Dynamics Flags
    ***************************
STEADY                             ! Rotation STEADY, KIN_JOINT, or DYN_JOINT
FALSE                               ! Passive Joint Forces and Torques
    Enabled
FALSE                               ! Compute Constraint Forces and Torques
REFPT_CM                            ! Mass Props referenced to REFPT_CM or
    REFPT_JOINT
FALSE                               ! Flex Active
FALSE                               ! Include 2nd Order Flex Terms
2.0                                 ! Drag Coefficient
*********************************************************************

************************ Body Parameters
    ******************************
*********************************************************************

1                                   ! Number of Bodies
================================== Body 0
    ==================================
1.0                                 ! Mass
2.0   3.0   1.2                     ! Moments of Inertia (kg-m^2)
0.0   0.0   0.0                     ! Products of Inertia (xy,xz,yz)
0.0   0.0   0.0                     ! Location of mass center, m
0.0   0.0   0.0                     ! Constant Embedded Momentum (Nms)
Cubesat_1U.obj                      ! Geometry Input File Name
NONE                                ! Flex File Name
```

```
******************************************************************************

************************** Joint Parameters
    **************************
******************************************************************************

          (Number of Joints is Number of Bodies minus one)
================================== Joint 0 ===================================
0 1                                ! Inner, outer body indices
1    213    GIMBAL                 ! RotDOF, Seq, GIMBAL or SPHERICAL
0    123                           ! TrnDOF, Seq
FALSE   FALSE   FALSE              ! RotDOF Locked
FALSE   FALSE   FALSE              ! TrnDOF Locked
0.0     0.0     0.0                ! Initial Angles [deg]
0.0     0.0     0.0                ! Initial Rates, deg/sec
0.0     0.0     0.0                ! Initial Displacements [m]
0.0     0.0     0.0                ! Initial Displacement Rates, m/sec
0.0    0.0   0.0   312             ! Bi to Gi Static Angles [deg] & Seq
0.0    0.0   0.0   312             ! Go to Bo Static Angles [deg] & Seq
0.0    0.0   0.0                   ! Position wrt inner body origin, m
0.0    0.0   0.0                   ! Position wrt outer body origin, m
0.0    0.0   0.0                   ! Rot Passive Spring Coefficients (Nm/rad
    )
0.0    0.0   0.0                   ! Rot Passive Damping Coefficients (Nms/
    rad)
0.0    0.0   0.0                   ! Trn Passive Spring Coefficients (N/m)
0.0    0.0   0.0                   ! Trn Passive Damping Coefficients (Ns/m)
************************** Wheel Parameters
    **************************
0                                  ! Number of wheels
================================== Wheel 0
    ===================================
0.0                                ! Initial Momentum, N-m-sec
1.0    0.0    0.0                  ! Wheel Axis Components, [X, Y, Z]
0.14    50.0                       ! Max Torque (N-m), Momentum (N-m-sec)
0.012                              ! Wheel Rotor Inertia, kg-m^2
0.48                               ! Static Imbalance, g-cm
13.7                               ! Dynamic Imbalance, g-cm^2
0                                  ! Flex Node Index
************************** MTB Parameters
    **************************
0                                      ! Number of MTBs
================================== MTB 0
    ===================================
180.0                              ! Saturation (A-m^2)
1.0    0.0    0.0                  ! MTB Axis Components, [X, Y, Z]
```

```
0                                   ! Flex Node Index
************************ Thruster Parameters
    ************************
0                                   ! Number of Thrusters
================================  Thr 0
    ================================
 1.0                                ! Thrust Force (N)
0  -1.0   0.0   0.0                 ! Body, Thrust Axis
 1.0   1.0   1.0                    ! Location in B0, m
0                                   ! Flex Node Index
****************************** Gyro
    **********************************
0                                    ! Number of Gyro Axes
================================== Axis 0
    ====================================
0.1                                 ! Sample Time,sec
1.0   0.0   0.0                     ! Axis expressed in Body Frame
1000.0                              ! Max Rate, deg/sec
100.0                               ! Scale Factor Error, ppm
1.0                                 ! Quantization, arcsec
0.07                                ! Angle Random Walk (deg/rt-hr)
0.1   1.0                           ! Bias Stability (deg/hr) over timespan (
    hr)
0.1                                 ! Angle Noise, arcsec RMS
0.1                                 ! Initial Bias (deg/hr)
0                                   ! Flex Node Index
************************** Magnetometer
    ******************************
0                                   ! Number of Magnetometer Axes
================================== Axis 0
    ====================================
0.1                                 ! Sample Time,sec
1.0   0.0   0.0                     ! Axis expressed in Body Frame
60.0E-6                             ! Saturation, Tesla
0.0                                 ! Scale Factor Error, ppm
1.0E-6                              ! Quantization, Tesla
1.0E-6                              ! Noise, Tesla RMS
0                                   ! Flex Node Index
********************** Coarse Sun Sensor
    ******************************
0                                   ! Number of Coarse Sun Sensors
================================== CSS 0
    ====================================
0.1                                 ! Sample Time,sec
0    1.0   1.0   1.0                ! Axis expressed in Body Frame
90.0                                ! Half-cone Angle, deg
```

```
1.0                                 ! Scale Factor
0.001                               ! Quantization
0                                   ! Flex Node Index
************************ Fine Sun Sensor
    ******************************
0                                   ! Number of Fine Sun Sensors
================================== FSS 0
    ==================================
0.2                                 ! Sample Time,sec
30.0   20.0   10.0   213            ! Mounting Angles (deg), Seq in Body
32.0    32.0                        ! X, Y FOV Size, deg
0.1                                 ! Noise Equivalent Angle, deg RMS
0.5                                 ! Quantization, deg
0                                   ! Flex Node Index
************************ Star Tracker
    ********************************
0                                   ! Number of Star Trackers
================================== ST 0
    ==================================
0.25                                ! Sample Time,sec
30.0   20.0   10.0   213            ! Mounting Angles (deg), Seq in Body
8.0    8.0                          ! X, Y FOV Size, deg
30.0   10.0   10.0                  ! Sun, Earth, Moon Exclusion Angles, deg
2.0   2.0   20.0                    ! Noise Equivalent Angle, arcsec RMS
0                                   ! Flex Node Index
****************************** GPS
    ************************************
0                                   ! Number of GPS Receivers
================================ GPSR 0
    ==================================
0.25                                ! Sample Time,sec
4.0                                 ! Position Noise, m RMS
0.02                                ! Velocity Noise, m/sec RMS
20.0E-9                             ! Time Noise, sec RMS
0                                   ! Flex Node Index
************************ Accelerometer
    ******************************
0                                   ! Number of Accel Axes
================================ Axis 0
    ==================================
0.1                                 ! Sample Time,sec
0.5   1.0   1.5                     ! Position in B[0] (m)
1.0   0.0   0.0                     ! Axis expressed in Body Frame
1.0                                 ! Max Acceleration (m/s^2)
0.0                                 ! Scale Factor Error, ppm
0.05                                ! Quantization, m/s^2
```

```
0.0                                    ! DV Random Walk (m/s/rt-hr)
0.0 1.0                                ! Bias Stability (m/s^2) over timespan (
    hr)
0.0                                    ! DV Noise, m/s
0.5                                    ! Initial Bias (m/s^2)
0                                      ! Flex Node Index
```

A.6 Inp_FOV

```
/*                    Note: FOV Boresight is +Z Axis                    */
************************ Fields of View ***************************
4                                       !  Number of FOVs
-------------------------------------------------------------------------
"SOLID"                                 !  Label
4    4.0                                !  Number of Sides, Length [m]
8.0   4.0                               !  X Width, Y Height [deg]
0.0  1.0  0.0  0.5                      !  Color RGB+Alpha
SOLID                                   !  WIREFRAME, SOLID, VECTOR, or
    PLANE
TRUE  TRUE                              !  Draw Near Field, Draw Far Field
0   0                                   !  SC, Body
0.0   0.0   1.0                         !  Position in Body [m]
0.0   0.0   0.0   321                   !  Euler Angles [deg], Sequence
-------------------------------------------------------------------------
"WIRE"                                  !  Label
24    4.0                               !  Number of Sides, Length [m]
10.0   5.0                              !  X Width, Y Height [deg]
0.7  0.7  0.0  1.0                      !  Color RGB+Alpha
WIREFRAME                               !  WIREFRAME, SOLID, VECTOR, or
    PLANE
TRUE  TRUE                              !  Draw Near Field, Draw Far Field
0   0                                   !  SC, Body
1.0   0.0   0.0                         !  Position in Body [m]
90.0   0.0   0.0   213                  !  Euler Angles [deg], Sequence
-------------------------------------------------------------------------
"VECTOR"                                !  Label
0    4.0                                !  Number of Sides, Length [m]
0.0   0.0                               !  X Width, Y Height [deg]
0.0  1.0  1.0  1.0                      !  Color RGB+Alpha
VECTOR                                  !  WIREFRAME, SOLID, VECTOR, or
    PLANE
TRUE  TRUE                              !  Draw Near Field, Draw Far Field
0   0                                   !  SC, Body
1.0   0.0   0.0                         !  Position in Body [m]
135.0   0.0   0.0   213                 !  Euler Angles [deg], Sequence
-------------------------------------------------------------------------
"PLANE"                                 !  Label
24     8.0                               !  Number of Sides, Length [m]
0.0   0.0                               !  X Width, Y Height [deg]
1.0  1.0  1.0  0.3                      !  Color RGB+Alpha
PLANE                                   !  WIREFRAME, SOLID, VECTOR, or
    PLANE
TRUE  TRUE                              !  Draw Near Field, Draw Far Field
```

```
0   0                                   !  SC, Body
0.0   0.0   0.0                         !  Position in Body [m]
-45.0   0.0   0.0   213                 !  Euler Angles [deg], Sequence
```

A.7 Inp_IPC

```
<<<<<<<<<<<<<<<< 42: InterProcess Comm Configuration File
    >>>>>>>>>>>>>>>>
0                                       ! Number of Sockets
********************************  IPC 0
    ****************************
OFF                                     ! IPC Mode (OFF,TX,RX,TXRX,ACS,
    WRITEFILE,READFILE)
0                                       ! AC.ID for ACS mode
"State00.42"                            ! File name for WRITE or READ
CLIENT                                  ! Socket Role (SERVER,CLIENT,
    GMSEC_CLIENT)
localhost       10001                   ! Server Host Name, Port
TRUE                                    ! Allow Blocking (i.e. wait on
    RX)
TRUE                                    ! Echo to stdout
3                                       ! Number of TX prefixes
"SC"                                    ! Prefix 0
"Orb"                                   ! Prefix 1
"World"                                 ! Prefix 2
********************************  IPC 1
    ****************************
OFF                                     ! IPC Mode (OFF,TX,RX,TXRX,ACS,
    WRITEFILE,READFILE)
0                                       ! AC.ID for ACS mode
"State01.42"                            ! File name for WRITE or READ
CLIENT                                  ! Socket Role (SERVER,CLIENT,
    GMSEC_CLIENT)
localhost       10002                   ! Server Host Name, Port
TRUE                                    ! Allow Blocking (i.e. wait on
    RX)
FALSE                                   ! Echo to stdout
1                                       ! Number of TX prefixes
"SC[0].AC"                              ! Prefix 0
********************************  IPC 2
    ****************************
OFF                                     ! IPC Mode (OFF,TX,RX,TXRX,ACS,
    WRITEFILE,READFILE)
1                                       ! AC.ID for ACS mode
"State02.42"                            ! File name for WRITE or READ
CLIENT                                  ! Socket Role (SERVER,CLIENT,
    GMSEC_CLIENT)
localhost       10003                   ! Server Host Name, Port
TRUE                                    ! Allow Blocking (i.e. wait on
    RX)
```

```
FALSE                                   ! Echo to stdout
1                                       ! Number of TX prefixes
"SC[1].AC"                              ! Prefix 0
********************************  IPC 3
    ****************************
OFF                                     ! IPC Mode (OFF,TX,RX,TXRX,ACS,
    WRITEFILE,READFILE)
0                                       ! AC.ID for ACS mode
"State03.42"                            ! File name for WRITE or READ
CLIENT                                  ! Socket Role (SERVER,CLIENT,
    GMSEC_CLIENT)
localhost       10004                   ! Server Host Name, Port
TRUE                                    ! Allow Blocking (i.e. wait on
    RX)
FALSE                                   ! Echo to stdout
1                                       ! Number of TX prefixes
"SC[0].Tach[0]"                         ! Prefix 0
```

A.8 Inp_Region

```
********************  Regions for 42  ********************
1                                   !  Number of Regions
-------------------------------------------------------------

TRUE                                ! Exists
"LZ"                                ! Name
EARTH                               ! World
LLA                                 ! POSW or LLA
0.0     0.0   0.0                   ! Position in W, m
-80.53   28.46  1000.0                    ! Lng, Lat (deg), Alt (m)
1.0E6   1.0E4   0.1                 ! Elasticity, Damping, Friction Coef
Rgn_Terrain.obj                     ! Geometry File Name
```

A.9 Inp_TDRS

```
<<<<<<<<<<<<<<<<<<<<  42 TDRS Configuration File  >>>>>>>>>>>>>>>>>>>>
FALSE   "TDRS-1"              ! TDRS-1 Exists, Designation
FALSE   "In Memorium"         ! TDRS-2 was lost along with Challenger
TRUE    "TDZ"                 ! TDRS-3 Exists, Designation
FALSE   "TDS"                 ! TDRS-4 Exists, Designation
FALSE   "TD171"               ! TDRS-5 Exists, Designation
TRUE    "TDW"                 ! TDRS-6 Exists, Designation
FALSE   "TDRS-7"              ! TDRS-7 Exists, Designation
FALSE   "TD271"               ! TDRS-8 Exists, Designation
FALSE   "TDRS-9"              ! TDRS-9 Exists, Designation
TRUE    "TDE"                 ! TDRS-10 Exists, Designation
```

Appendix B CONSTELLATION INPUT GENERATOR

In this appendix, the code fro the constellation input generator are displayed.

B.1 Constellation generator

```
clc
clear
close all

i=45; %inclination in deg
t=50; %num of sats
p=5; %num of planes
f=4; %rel spacing between sats in adjacent planes (from 0 to p-1)
rmin=500; %min altitude in km
e=0; %eccentricity

s=t/p; %num of sats per plane
PU=360/t; %in deg

PUp=PU*p; %in-plane spacing between sats in deg
PUs=PU*s; %node spacing in deg
PUf=PU*f; %phase diff between adjacent planes in deg

sat=1:t;
planes=1:p;

RAAN(planes)=0:PUs:360-PUs;
dphi(planes)=PUf*(planes-1);
spacing(1:s)=0:PUp:360-PUp;

filename(1:t)="inputs/Orb_"+sat+".txt";
orbname(1:t)="Orb_"+sat+".txt";
SCname="SC_cubesat.txt";

k=1;
for j=1:p
    for jj=1:s
        w=spacing(jj)+dphi(j);
        WriteOrbFile(filename(k),i,RAAN(j),w,rmin,e)
        k=k+1;
    end
end
WriteSim(orbname,SCname,t);
```

B.2 WriteOrbFile

```
function WriteOrbFile(name,i,raan,w,r,e)
    fid=fopen(name,'w');
    fprintf(fid,'<<<<<<<<<<<<<<<<<<< 42: Orbit Description File
       >>>>>>>>>>>>>>>>>>>\n');
    fprintf(fid,'Test orbit ! Description\n');
    fprintf(fid,'CENTRAL ! Orbit Type (ZERO,
       FLIGHT, CENTRAL, THREE_BODY)\n');
    fprintf(fid,':::::::::::::: Use these lines if ZERO
       ::::::::::::::::\n');
    fprintf(fid,'MINORBODY_2 ! World\n');
    fprintf(fid,'FALSE ! Use Polyhedron Gravity
       \n');
    fprintf(fid,':::::::::::::: Use these lines if FLIGHT
       ::::::::::::::::\n');
    fprintf(fid,'0 ! Region Number\n');
    fprintf(fid,'FALSE ! Use Polyhedron Gravity
       \n');
    fprintf(fid,':::::::::::::: Use these lines if Body-Centered Orbit
       :::::::::::::::::\n');
    fprintf(fid,'EARTH ! Orbit Center\n');
    fprintf(fid,'FALSE ! Secular Orbit Drift
       Due to J2\n');
    fprintf(fid,'KEP ! Use Keplerian
       elements (KEP) or (RV) or FILE\n');
    fprintf(fid,'AE ! Use Peri/Apoapsis (PA
       ) or min alt/ecc (AE)\n');
    fprintf(fid,'35786.0 35786.0 ! Periapsis &
       Apoapsis Altitude, km\n');
    fprintf(fid,'%s %s ! Min Altitude (km),
       Eccentricity\n',num2str(r),num2str(e));
    fprintf(fid,'%s ! Inclination (deg)\n',
       num2str(i));
    fprintf(fid,'%s ! Right Ascension of
       Ascending Node (deg)\n',num2str(raan));
    fprintf(fid,'0.0 ! Argument of Periapsis
       (deg)\n');
    fprintf(fid,'%s ! True Anomaly (deg)\n',
       num2str(w));
    fprintf(fid,'0.0 0.0 0.0 ! RV Initial Position (
       km)\n');
    fprintf(fid,'0.0 0.0 0.0 ! RV Initial Velocity (
       km/sec)\n');
    fprintf(fid,'TRV "ORB_ID" ! TLE or TRV format,
       Label to find in file\n');
```

```
    fprintf(fid,'"TRV.txt"                    !  File name\n');
    fprintf(fid,':::::::::::::  Use these lines if Three-Body Orbit  
        ::::::::::::::::\n');
    fprintf(fid,'SUNEARTH                     !  Lagrange system\n');
    fprintf(fid,'LAGDOF_MODES                 !  Propagate using 
        LAGDOF_MODES or LAGDOF_COWELL or LAGDOF_SPLINE\n');
    fprintf(fid,'MODES                        !  Initialize with MODES
        or XYZ or FILE\n');
    fprintf(fid,'L2                           !  Libration point (L1, 
        L2, L3, L4, L5)\n');
    fprintf(fid,'800000.0                     !  XY Semi-major axis, 
        km\n');
    fprintf(fid,'45.0                         !  Initial XY Phase, deg
        \n');
    fprintf(fid,'CW                           !  Sense (CW, CCW), 
        viewed from +Z\n');
    fprintf(fid,'0.0                          !  Second XY Mode Semi-
        major Axis, km (L4, L5 only)\n');
    fprintf(fid,'0.0                          !  Second XY Mode 
        Initial Phase, deg (L4, L5 only)\n');
    fprintf(fid,'CW                           !  Sense (CW, CCW), 
        viewed from +Z (L4, L5 only)\n');
    fprintf(fid,'400000.0                     !  Z Semi-axis, km\n');
    fprintf(fid,'60.0                         !  Initial Z Phase, deg\
        n');
    fprintf(fid,'1.05   0.5   0.0             !  Initial X, Y, Z (Non-
        dimensional)\n');
    fprintf(fid,'0.0    0.0   0.0             !  Initial Xdot, Ydot, 
        Zdot (Non-dimensional)\n');
    fprintf(fid,'TRV  "ORB_ID"                !  TLE, TRV or SPLINE 
        format, Label to find in file\n');
    fprintf(fid,'"TRV.txt"                    !  File name\n');
    fprintf(fid,'******************  Formation Frame Parameters 
        **********************\n');
    fprintf(fid,'N                            !  Formation Frame Fixed
        in [NL]\n');
    fprintf(fid,'0.0   0.0   0.0   123        !  Euler Angles (deg) 
        and Sequence\n');
    fprintf(fid,'N                            !  Formation Origin 
        expressed in [NL]\n');
    fprintf(fid,'0.0   0.0   0.0              !  Formation Origin wrt 
        Ref Orbit (m)\n');
    fclose(fid);
end
```

B.3 WriteSim

```
function WriteSim(orb,SC,t)
    fid=fopen('inputs/Inp_Sim.txt','w');
    fprintf(fid,'<<<<<<<<<<<<<<<<<<  42: The Mostly Harmless Simulator  
        >>>>>>>>>>>>>>>>>>\n');
    fprintf(fid,'************************ Simulation Control 
        ************************\n');
    fprintf(fid,'FAST                              ! Time Mode (FAST, 
        REAL, or EXTERNAL)\n');
    fprintf(fid,'36000.0       0.1                 ! Sim Duration, Step 
        Size [sec]\n');
    fprintf(fid,'10.0                              ! File Output 
        Interval [sec]\n');
    fprintf(fid,'FALSE                             ! Graphics Front End 
        ?\n');
    fprintf(fid,'Inp_Cmd.txt                       ! Command Script File 
        Name\n');
    fprintf(fid,'************************  Reference Orbits  
        ************************\n');
    fprintf(fid,'%i                                ! Number of 
        Reference Orbits\n',t);
    for i=1:t
        fprintf(fid,'TRUE   %s                     ! Input file name for Orb 
            0\n',orb(i));
    end
    fprintf(fid,'*****************************  Spacecraft  
        *****************************\n');
    fprintf(fid,'%i                                ! Number of 
        Spacecraft\n',t);
    for i=1:t
        fprintf(fid,'TRUE  %i %s                   ! Existence, RefOrb, Input 
            file for SC  0\n',i-1,SC);
    end
    fprintf(fid,'***************************  Environment  
        ***************************\n');
    fprintf(fid,'02 04 2021                        ! Date (UTC) (Month, 
        Day, Year)\n');
    fprintf(fid,'12 49 00.00                       ! Time (UTC) (Hr, Min, 
        Sec)\n');
    fprintf(fid,'37.0                              ! Leap Seconds (sec)\
        n');
    fprintf(fid,'NOMINAL                           ! F10.7, Ap (USER, 
        NOMINAL or TWOSIGMA)\n');
    fprintf(fid,'230.0                             ! If USER_DEFINED, 
        enter desired F10.7 value\n');
```

```
fprintf(fid,'100.0                    !  If USER_DEFINED, enter desired AP value\n');
fprintf(fid,'IGRF                     !  Magfield (NONE, DIPOLE,IGRF)\n');
fprintf(fid,'8   8                    !  IGRF Degree and Order (<=10)\n');
fprintf(fid,'2   0                    !  Earth Gravity Model N and M (<=18)\n');
fprintf(fid,'2   0                    !  Mars Gravity Model N and M (<=18)\n');
fprintf(fid,'2   0                    !  Luna Gravity Model N and M (<=18)>\n');
fprintf(fid,'FALSE   FALSE            !  Aerodynamic Forces & Torques (Shadows)\n');
fprintf(fid,'FALSE                    !  Gravity Gradient Torques\n');
fprintf(fid,'FALSE   FALSE            !  Solar Pressure Forces & Torques (Shadows)\n');
fprintf(fid,'FALSE                    !  Gravity Perturbation Forces\n');
fprintf(fid,'FALSE                    !  Passive Joint Torques\n');
fprintf(fid,'FALSE                    !  Thruster Plume Forces & Torques\n');
fprintf(fid,'FALSE                    !  RWA Imbalance Forces and Torques\n');
fprintf(fid,'FALSE                    !  Contact Forces and Torques\n');
fprintf(fid,'FALSE                    !  CFD Slosh Forces and Torques\n');
fprintf(fid,'FALSE                    !  Output Environmental Torques to Files\n');
fprintf(fid,'********************  Celestial Bodies of Interest  ********************\n');
fprintf(fid,'MEAN                     !  Ephem Option (MEAN or DE430)\n');
fprintf(fid,'TRUE                     !  Mercury\n');
fprintf(fid,'TRUE                     !  Venus\n');
fprintf(fid,'TRUE                     !  Earth and Luna\n');
fprintf(fid,'TRUE                     !  Mars and its moons\n');
fprintf(fid,'TRUE                     !  Jupiter and its moons\n');
fprintf(fid,'TRUE                     !  Saturn and its moons\n');
fprintf(fid,'TRUE                     !  Uranus and its
```

```
        moons\n');
    fprintf(fid, 'TRUE                          !  Neptune and its
        moons\n');
    fprintf(fid, 'TRUE                          !  Pluto and its moons
        \n');
    fprintf(fid, 'FALSE                         !  Asteroids and
        Comets\n');
    fprintf(fid, '****************  Lagrange Point Systems of Interest
        *****************\n');
    fprintf(fid, 'FALSE                         !  Earth-Moon\n');
    fprintf(fid, 'FALSE                         !  Sun-Earth\n');
    fprintf(fid, 'FALSE                         !  Sun-Jupiter\n');
    fprintf(fid, '************************  Ground Stations
        *************************\n');
    fprintf(fid, '5                                                 !  Number
        of Ground Stations\n');
    fprintf(fid, 'TRUE  EARTH   -77.0   37.0   "GSFC"               !  Exists,
        World, Lng, Lat, Label\n');
    fprintf(fid, 'TRUE  EARTH   -155.6  19.0   "South Point"       !  Exists,
        World, Lng, Lat, Label\n');
    fprintf(fid, 'TRUE  EARTH   115.4   -29.0  "Dongara"           !  Exists,
        World, Lng, Lat, Label\n');
    fprintf(fid, 'TRUE  EARTH   -71.0   -33.0  "Santiago"          !  Exists,
        World, Lng, Lat, Label\n');
    fprintf(fid, 'TRUE  LUNA    45.0    45.0   "Moon Base Alpha"    !  Exists,
        World, Lng, Lat, Label\n');
    fclose(fid);
end
```

Appendix C PROGRAMMING A FSW FUNCTION IN 42

In this appendix, the process for writing a new attitude control function inside 42 simulator will be detailed and at the end, the cubesat's attitude control function will be displayed.

C.1 Programming process

Due to how the simulator is programmed, adding a new attitude control function is not as simple as adding a new subroutine, there are several files that need to be modified in order to include said function.

The first step is to add a new FSW tag in *42defines.h*. In here, a tag for the attitude controller is added next to a uniquely defined integer. In this case, the line *#define CUBESAT_FSW 11* has been added. The name of the definition (CUBESAT_FSW) will be the name of the controller called from the inputs.

```
/* FSW Tags */
#define PASSIVE_FSW 0
#define PROTOTYPE_FSW 1
#define AD_HOC_FSW 2
#define SPINNER_FSW 3
#define MOMBIAS_FSW 4
#define THREE_AXIS_FSW 5
#define ISS_FSW 6
#define CMG_FSW 7
#define THR_FSW 8
#define CFS_FSW 9
#define NOS3_FSW 10
#define CUBESAT_FSW 11
```

The next step is to add a new structure for the controller in question, this is done in *fswtypes.h* and in *Actypes.h*. This file contains all the structures used in the attitude control subroutine of the simulator. First, add a structure inside the general *AcType* structure. In this case, the structure is of the type *AcCubesatCtrlType* and is called *CubesatCtrl*.

```
    /* Control Modes */
    struct AcPrototypeCtrlType PrototypeCtrl;
    struct AcAdHocCtrlType AdHocCtrl;
    struct AcSpinnerCtrlType SpinnerCtrl;
    struct AcMomBiasCtrlType MomBiasCtrl;
    struct AcThreeAxisCtrlType ThreeAxisCtrl;
    struct AcIssCtrlType IssCtrl;
    struct AcCmgCtrlType CmgCtrl;
    struct AcThrCtrlType ThrCtrl;
    struct AcCfsCtrlType CfsCtrl;
    struct AcCubesatCtrlType CubesatCtrl;
```

After adding the structure, define it above the main structure. The components of the structure will depend on the attitude control function being programmed but there is one compulsory variable, *long Init*, which will be used by the simulator to initialise the controller.

```
struct AcCubesatCtrlType {
   /*- Parameters -*/
   double Kr[3];
   double Kp[3];

   /*- Internal Variables -*/
   long Init;
   double therr[3];
   double werr[3];
   double Tcmd[3];
}
```

After that, in the file *42init.c*, a new entry for the *DecodeString* function must be added so that the program can interpret the new controller in the inputs. This entry is an else if statement which returns the integer defined in the #define added in the first step of this process if the string read at the input coincides with the controller definition.

```
      else if (!strcmp(s,"PASSIVE_FSW")) return PASSIVE_FSW;
      else if (!strcmp(s,"PROTOTYPE_FSW")) return PROTOTYPE_FSW;
      else if (!strcmp(s,"AD_HOC_FSW")) return AD_HOC_FSW;
      else if (!strcmp(s,"SPINNER_FSW")) return SPINNER_FSW;
      else if (!strcmp(s,"MOMBIAS_FSW")) return MOMBIAS_FSW;
      else if (!strcmp(s,"THREE_AXIS_FSW")) return THREE_AXIS_FSW;
      else if (!strcmp(s,"ISS_FSW")) return ISS_FSW;
      else if (!strcmp(s,"CMG_FSW")) return CMG_FSW;
      else if (!strcmp(s,"THR_FSW")) return THR_FSW;
      else if (!strcmp(s,"CFS_FSW")) return CFS_FSW;
      else if (!strcmp(s,"CUBESAT_FSW")) return CUBESAT_FSW;
```

Finally, in the file *42fsw.c*, three modification must be done. First, in the *FlightSoftWare* function, a new entry inside the controller tag switch must be added to call the new attitude control function. In this case, the function will be called *CubesatFSW* and has as an input the spacecraft structure *S*.

```
            case CUBESAT_FSW:
               CubesatFSW(S);
               break;
```

Then, in the function *InitAC*, the new controller must be initialised by assigning 1 to the *Init* variable inside the controller structure.

```
/* Controllers */
AC->PrototypeCtrl.Init = 1;
AC->AdHocCtrl.Init = 1;
AC->SpinnerCtrl.Init = 1;
AC->MomBiasCtrl.Init = 1;
AC->ThreeAxisCtrl.Init = 1;
AC->IssCtrl.Init = 1;
AC->CmgCtrl.Init = 1;
AC->ThrCtrl.Init = 1;
AC->CfsCtrl.Init = 1;
AC->ThrSteerCtrl.Init = 1;
AC->CubesatCtrl.Init = 1;
```

Lastly, add the new attitude control function anywhere above *FlightSoftWare* (See appendix C.2).

C.2 Cubesat FSW routine

```
void CubesatFSW(struct SCType *S)
{
    struct AcType *AC;
    struct AcCubesatCtrlType *C;
    struct CmdType *Cmd;
    double qbr[4];
    long i;

    AC = &S->AC;
    C = &AC->CubesatCtrl;
    Cmd = &AC->Cmd;

    if (C->Init) {
        C->Init = 0;
        for (i=0;i<3;i++) {
            FindPDGains(AC->MOI[i][i],0.1,0.7,&C->Kr[i],&C->Kp[i]);
        }
    }
    ThreeAxisAttitudeCommand(S);
    SpinnerCommand(S);
    /*Star tracker*/
    StarTrackerProcessing(AC);
    GyroProcessing(AC);

    /* Form attitude error signals */
    QxQT(AC->qbn,Cmd->qrn,qbr);
    RECTIFYQ(qbr);
    for(i=0;i<3;i++){
        C->therr[i] = 2.0*qbr[i];
        C->werr[i] = AC->wbn[i] - Cmd->wrn[i];
    }

    /*Closed-loop attitude control */
    for(i=0;i<3;i++) {
        C->Tcmd[i] = -C->Kr[i]*C->werr[i]-C->Kp[i]*C->therr[i];
    }

    for(i=0;i<3;i++) AC->Tcmd[i] = C->Tcmd[i];

    /*Reaction wheels*/
    WheelProcessing(AC);
}
```

Appendix D ADDED SIMULATION COMMANDS

In this section, all added simulation commands to 42 will be displayed. To do so, an else if clause must be added to the *FswCmdInterpreter* function in *42fsw.c*.

D.1 Point at the Sun when exiting eclipse

```
else if (sscanf(CmdLine,"Event Eclipse Exit Point SC[%ld].B[%ld]
    %s Vector [%lf %lf %lf] at %s",
    &Isc,&Ib,VecString,&VecR[0],&VecR[1],&VecR[2],TargetString) ==
        7) {
   *CmdTime = SimTime+DTSIM;
   if (!SC[Isc].Eclipse) {
     NewCmdProcessed = TRUE;
     if (Ib == 0) {
        Cmd = &SC[Isc].AC.Cmd;
     }
     else {
        Ig = SC[Isc].B[Ib].Gin;
        Cmd = &SC[Isc].AC.G[Ig].Cmd;
     }
     Cmd->Parm = PARM_VECTORS;
     Cmd->Frame = FRAME_N;
     if (!strcmp(VecString,"Primary")) CV = &Cmd->PriVec;
     else CV = &Cmd->SecVec;
     CV->Mode = CMD_TARGET;
     CV->Frame = FRAME_N;
     if (!strcmp(TargetString,"EARTH")) {
        CV->TrgType = TARGET_WORLD;
        CV->TrgWorld = EARTH;
     }
     else if (!strcmp(TargetString,"MOON")) {
        CV->TrgType = TARGET_WORLD;
        CV->TrgWorld = LUNA;
     }
     else if (!strcmp(TargetString,"LUNA")) {
        CV->TrgType = TARGET_WORLD;
        CV->TrgWorld = LUNA;
     }
     else if (!strcmp(TargetString,"MERCURY")) {
        CV->TrgType = TARGET_WORLD;
        CV->TrgWorld = MERCURY;
     }
     else if (!strcmp(TargetString,"VENUS")) {
        CV->TrgType = TARGET_WORLD;
```

```
            CV->TrgWorld = VENUS;
        }
        else if (!strcmp(TargetString,"MARS")) {
            CV->TrgType = TARGET_WORLD;
            CV->TrgWorld = MARS;
        }
        else if (!strcmp(TargetString,"JUPITER")) {
            CV->TrgType = TARGET_WORLD;
            CV->TrgWorld = JUPITER;
        }
        else if (!strcmp(TargetString,"SATURN")) {
            CV->TrgType = TARGET_WORLD;
            CV->TrgWorld = SATURN;
        }
        else if (!strcmp(TargetString,"URANUS")) {
            CV->TrgType = TARGET_WORLD;
            CV->TrgWorld = URANUS;
        }
        else if (!strcmp(TargetString,"NEPTUNE")) {
            CV->TrgType = TARGET_WORLD;
            CV->TrgWorld = NEPTUNE;
        }
        else if (!strcmp(TargetString,"PLUTO")) {
            CV->TrgType = TARGET_WORLD;
            CV->TrgWorld = PLUTO;
        }
        else if (!strcmp(TargetString,"VELOCITY")) {
            CV->TrgType = TARGET_VELOCITY;
        }
        else if (!strcmp(TargetString,"MAGFIELD")) {
            CV->TrgType = TARGET_MAGFIELD;
        }
        else if (!strcmp(TargetString,"TDRS")) {
            CV->TrgType = TARGET_TDRS;
        }
        else {
            CV->TrgType = TARGET_WORLD;
            CV->TrgWorld = SOL;
        }
        UNITV(VecR);
        for(i=0;i<3;i++) CV->R[i] = VecR[i];
    }
}
```

Appendix E MATLAB POSTPROCESSING FUNCTIONS

In this section, the postprocessing functions programmed with matlab will be displayed.

E.1 Basic postprocessing

```
clc
clear
close all

addpath(genpath('C:\Users\Lluis\Desktop\UPC\2B\TFM\simulations')) %
    simulations folder
a='Constellationtest'; %particular simulation folder
Simflag=true;
Plotflag=true;
SC=[0,1];
simSC=[0,1];
t0=[2021 2 28 10 0 0];


time=load(strcat(a,'/time.42'),'-ascii');
for i=1:length(SC)
    PosN(i,:,:)=load(strcat(a,'/PosN',num2str(SC(i),'%02u'),'.42'),'-
        ascii');
    VelN(i,:,:)=load(strcat(a,'/VelN',num2str(SC(i),'%02u'),'.42'),'-
        ascii');
    PosW(i,:,:)=load(strcat(a,'/PosW',num2str(SC(i),'%02u'),'.42'),'-
        ascii');
    VelW(i,:,:)=load(strcat(a,'/VelW',num2str(SC(i),'%02u'),'.42'),'-
        ascii');
    qbn(i,:,:)=load(strcat(a,'/qbn',num2str(SC(i),'%02u'),'.42'),'-
        ascii');
    wbn(i,:,:)=load(strcat(a,'/wbn',num2str(SC(i),'%02u'),'.42'),'-
        ascii');
    svn(i,:,:)=load(strcat(a,'/svn',num2str(SC(i),'%02u'),'.42'),'-
        ascii');
    svb(i,:,:)=load(strcat(a,'/svb',num2str(SC(i),'%02u'),'.42'),'-
        ascii');
    Eclipse(i,:,:)=load(strcat(a,'/Eclipse',num2str(SC(i),'%02u'),'.42'
        ),'-ascii');
end
lim=length(time);

if Plotflag
    for i=1:length(SC)
```

```
figure
plot(time(1:lim),PosN(i,1:lim,1),time(1:lim),PosN(1,1:lim,2),
    time(1:lim),PosN(1,1:lim,3))
title(['PosN',num2str(SC(i),'%02u')])
xlabel('t(s)')
ylabel('Position ECI(m)')
legend('x','y','z')
grid on
grid minor
figure
plot(time(1:lim),VelN(i,1:lim,1),time(1:lim),VelN(1,1:lim,2),
    time(1:lim),VelN(1,1:lim,3))
title(['VelN',num2str(SC(i),'%02u')])
xlabel('t(s)')
ylabel('Velocity ECI(m/s)')
legend('v_x','v_y','v_z')
grid on
grid minor
figure
plot(time(1:lim),PosW(i,1:lim,1),time(1:lim),PosW(1,1:lim,2),
    time(1:lim),PosW(1,1:lim,3))
title(['PosW',num2str(SC(i),'%02u')])
xlabel('t(s)')
ylabel('Position ECEF(m)')
legend('x','y','z')
grid on
grid minor
figure
plot(time(1:lim),VelW(i,1:lim,1),time(1:lim),VelW(1,1:lim,2),
    time(1:lim),VelW(1,1:lim,3))
title(['VelW',num2str(SC(i),'%02u')])
xlabel('t(s)')
ylabel('Velocity ECEF(m/s)')
legend('v_x','v_y','v_z')
grid on
grid minor
figure
plot(time(1:lim),qbn(i,1:lim,1),time(1:lim),qbn(1,1:lim,2),time
    (1:lim),qbn(1,1:lim,3),time(1:lim),qbn(1,1:lim,4))
title(['qbn',num2str(SC(i),'%02u')])
xlabel('t(s)')
ylabel('Quaternion ECI')
legend('q_1','q_2','q_3','q_4')
grid on
grid minor
figure
```

```
        plot(time(1:lim),wbn(i,1:lim,1),time(1:lim),wbn(1,1:lim,2),time
            (1:lim),wbn(1,1:lim,3))
        title(['wbn',num2str(SC(i),'%02u')])
        xlabel('t(s)')
        ylabel('Angular velocity ECI(rad/s)')
        legend('\omega_x','\omega_y','\omega_z')
        grid on
        grid minor
        figure
        plot(time(1:lim),svn(i,1:lim,1),time(1:lim),svn(1,1:lim,2),time
            (1:lim),svn(1,1:lim,3))
        title(['svn',num2str(SC(i),'%02u')])
        xlabel('t(s)')
        ylabel('Sun vector ECI')
        legend('x','y','z')
        grid on
        grid minor
        figure
        plot(time(1:lim),svb(i,1:lim,1),time(1:lim),svb(1,1:lim,2),time
            (1:lim),svb(1,1:lim,3))
        title(['svb',num2str(SC(i),'%02u')])
        xlabel('t(s)')
        ylabel('Sun vector body')
        legend('x','y','z')
        grid on
        grid minor
        figure
        plot(time(1:lim),Eclipse(i,1:lim))
        title(['Eclipse',num2str(SC(i),'%02u')])
        xlabel('t(s)')
        ylabel('Eclipse state')
        grid on
        grid minor
    end
end

%Simulink variable formating%

if Simflag
    for i=1:lim
        dcm=dcmeci2ecef('IAU-2000/2006',datevec(datetime(t0)+seconds(
            time(i))));
        q(i,:)=dcm2quat(dcm);
        qbn03=[qbn(simSC(1)+1,i,4) qbn(simSC(1)+1,i,1) qbn(simSC(1)+1,i
            ,2) qbn(simSC(1)+1,i,3)];
        qbw1(i,:)=quatmultiply(quatinv(q(i,:)),qbn03);
```

```
        qbn03=[qbn(simSC(2)+1,i,4) qbn(simSC(2)+1,i,1) qbn(simSC(2)+1,i
            ,2) qbn(simSC(2)+1,i,3)];
        qbw2(i,:)=quatmultiply(quatinv(q(i,:)),qbn03);
    end


    XECEF1=[time,reshape(PosW(simSC(1)+1,:,:),[],3)];
    XECEF2=[time,reshape(PosW(simSC(2)+1,:,:),[],3)];
    qECI=[time,reshape(qbn(1,:,:),[],4)];
    qECEF1=[time,qbw1];
    qECEF2=[time,qbw2];
    simtime=[time,time];


    %%

    open_system('Viewer')
    set_param('Viewer','StopTime',int2str((lim-1)*(time(2)-time(1))))
end
```

Appendix F SIMULINK 3D MODEL

In this section, the Simulink 3D model will be displayed as well as the 3D animation file for the simulink simulation.

F.1 Simulink 3D simulation block detail

Figure 31, shows the inside of the simulink 3D animation block modified to display two satellites. Figures 32 and 33 show the detail of the top and bottom respectively of the 3D animation block.

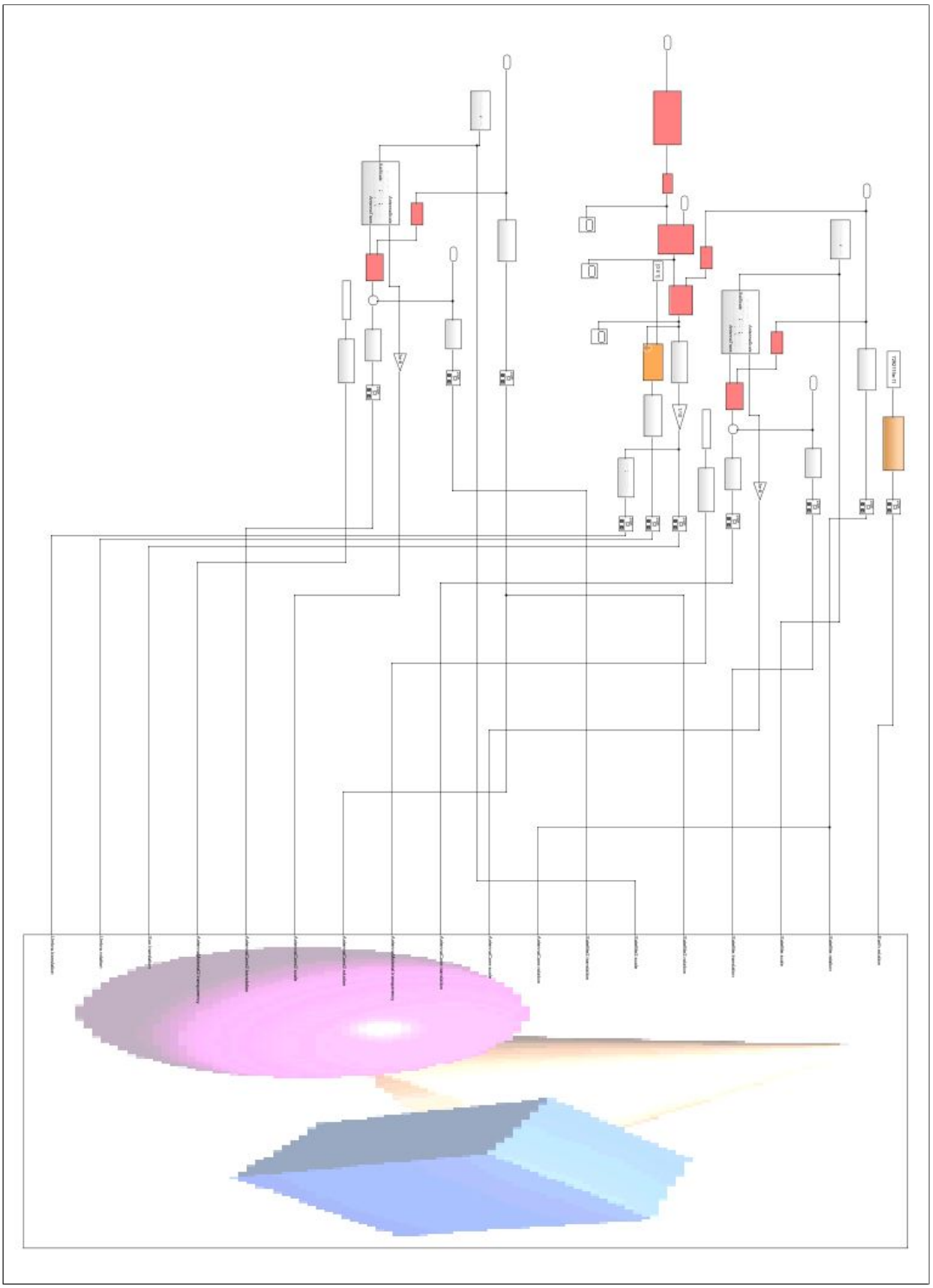


Figure 31: 3D simulation block

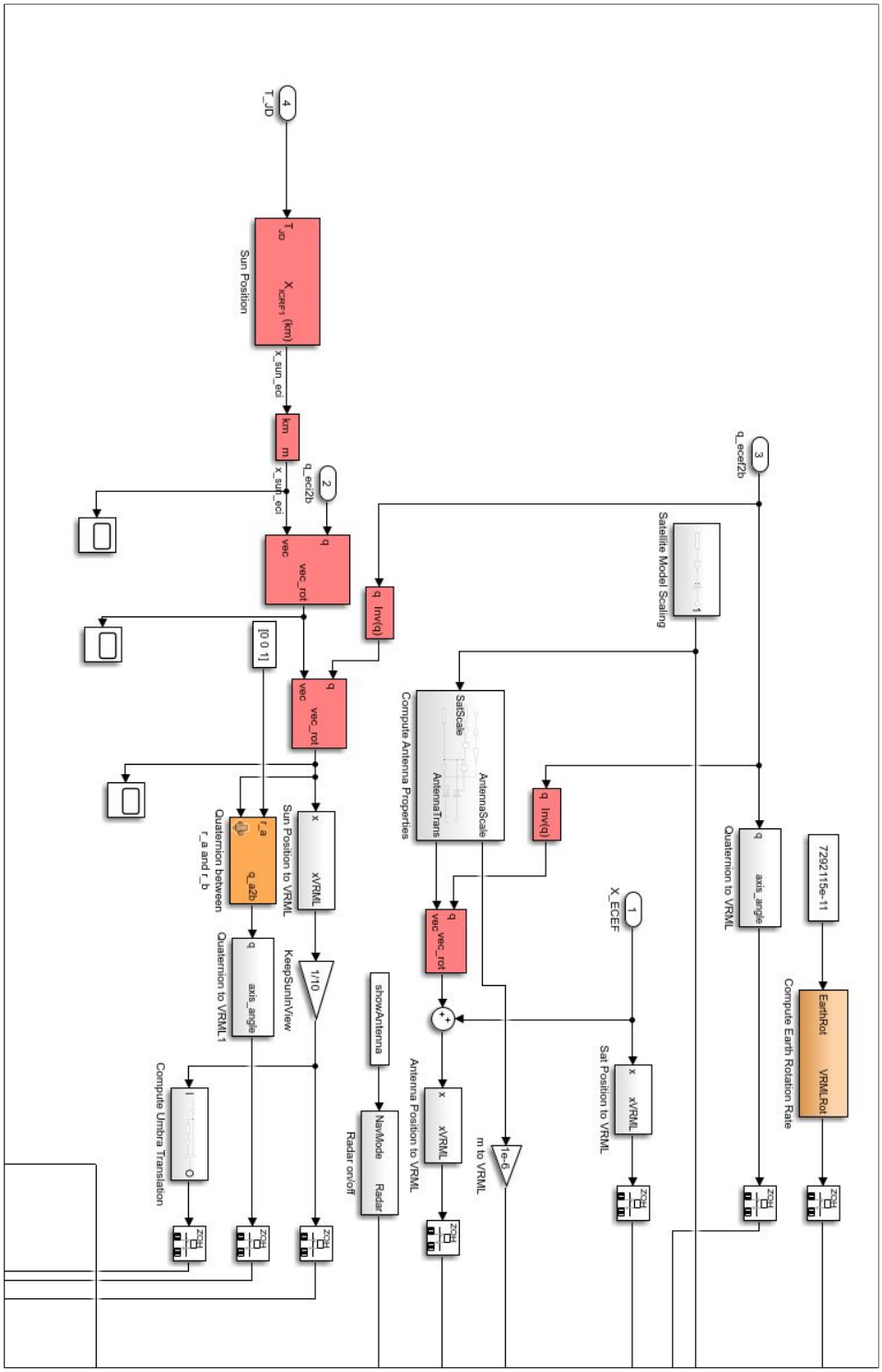


Figure 32: 3D simulation block top detail

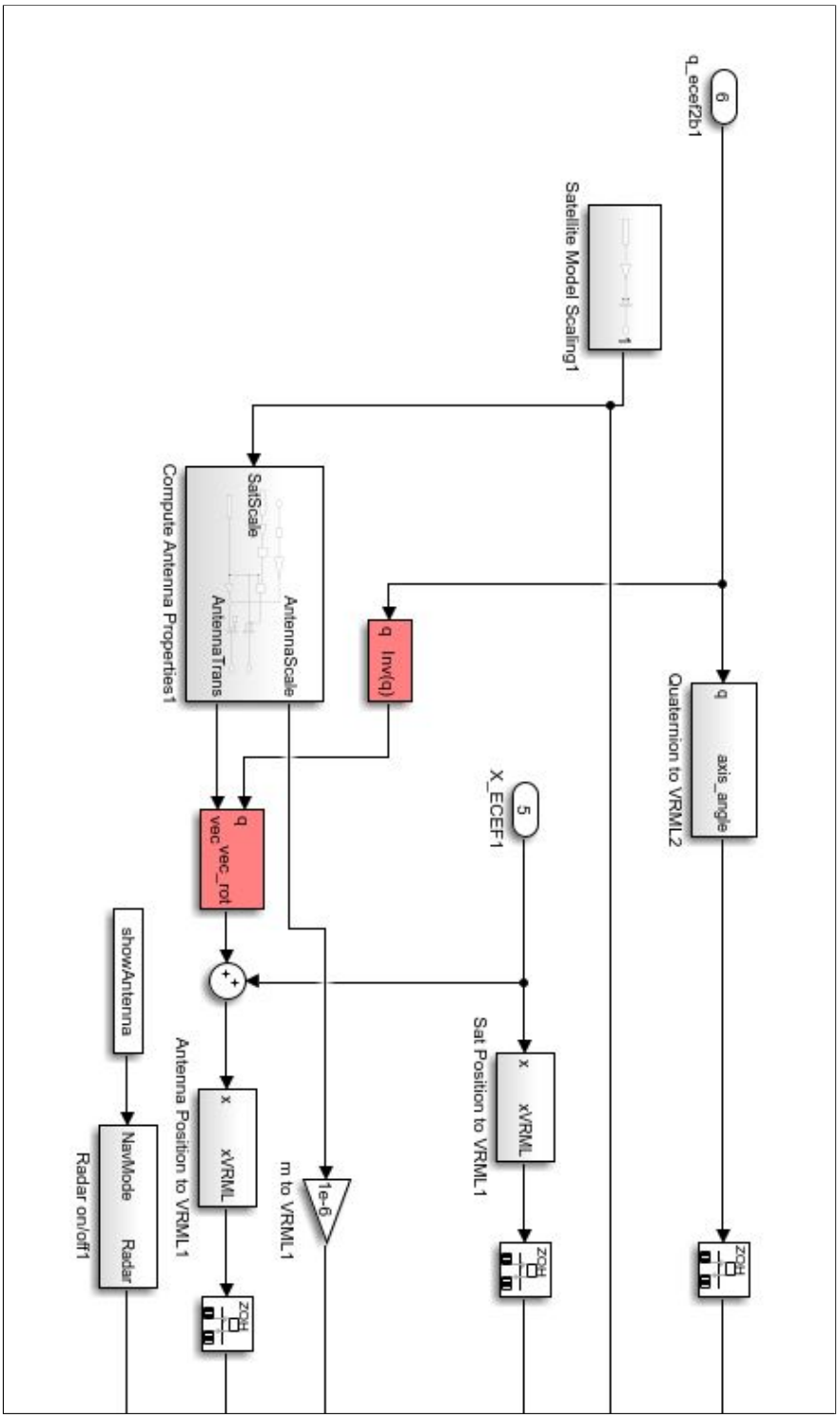


Figure 33: 3D simulation block bottom detail

F.2 3D animation file

```
#VRML V2.0 utf8
# VRML model of the Earth
# Texture images used are from the NASA Visible Earth catalog:
# http://visibleearth.nasa.gov

WorldInfo {
  title "CubeSat Orbit Visualization"
  info "Copyright 2016-2019 HUMUSOFT s.r.o. and MathWorks, Inc."
}
NavigationInfo {
  visibilityLimit 200000
  type "EXAMINE"
  speed 10
  # avatarSize [ 10 10 10 ]
  headlight TRUE
}
DirectionalLight {
  direction 1 -2 1
  ambientIntensity 0.2
}
DirectionalLight {
  direction -1 -2 -1
  ambientIntensity 0.2
}
DEF SkyDome Transform {
  # SkyDome diameter in the Inlined file is 10000
  translation 0 0 0
  scale 8 8 8
  children Inline {
    url "skyAnim.wrl"
  }
}
# Sun
PointLight {
  radius 10000
  location 10000 0 0
  # attenuation 0 0 0
  color 1 1 1
  attenuation 0 0 0
  ambientIntensity 0.5
}
DEF NorthPole Transform {
  translation 0 18 0
  rotation -1 0 0 1.5708
```

```
  children Viewpoint {
    description "NorthPole"
    position 0 0 18
    fieldOfView 0.5
  }
}
DEF EarthSide Transform {
  translation 0 0 12
  children Viewpoint {
    description "EarthSide"
    fieldOfView 0.7854
  }
}
DEF Earth Transform {
  scale 1 0.999999 0.999999
  rotation 0 0 0 1
  children [
    DEF EarthTopo Transform {
      scale 6.37814 6.35675 6.37814
      children Inline {
        url "earthAnim.wrl"
      }
    }
    DEF Satellite Transform {
      scaleOrientation 0 1 0 0
      scale 0.6667 0.6667 0.6667
      rotation 0 0 0 1
      children [
        DEF BusBody Shape {
          appearance Appearance {
            texture ImageTexture {
              url "texture/cubesat.jpg"
            }
            material Material {
              specularColor 1 1 1
              shininess 1
              emissiveColor 0.2 0.2 0.2
              diffuseColor 0.2 0.2 0.2
              ambientIntensity 0.6
            }
          }
          geometry Box {
            size 0.3 0.3 0.3
          }
        }
        DEF SolarPanel2 Transform {
```

```
  translation 0 0.16 -0.3
  children Shape {
    appearance Appearance {
      texture ImageTexture {
        url "texture/solarpanel.jpg"
      }
      material Material {
        shininess 1
        diffuseColor 0.9 0.767329 0.619635
        ambientIntensity 0
      }
    }
    geometry Box {
      size 0.3 0.02 0.3
    }
  }
}
DEF SolarPanel1 Transform {
  translation 0 0.16 0.3
  children Shape {
    appearance Appearance {
      texture ImageTexture {
        url "texture/solarpanel.jpg"
      }
      material Material {
        shininess 1
        diffuseColor 0.9 0.767329 0.619635
        ambientIntensity 0
      }
    }
    geometry Box {
      size 0.3 0.02 0.3
    }
  }
}
DEF Antenna Transform {
  children DEF Radar Transform {
    translation 0 -0.15 0
    children Shape {
      appearance Appearance {
        material Material {
          specularColor 1 1 1
          shininess 1
          diffuseColor 0.9 0.655038 0.319173
        }
      }
```

```
            geometry Cone {
              height 0.2
              bottomRadius 0.06
            }
          }
        }
      }
      DEF FollowSat Transform {
        translation -5.5 2 0
        rotation -0.0923 -0.7011 -0.0923 1.5878
        children DEF Eeek Viewpoint {
          description "Satellite1-fixed"
          position 0 0 0
        }
      }
    ]
  }
  DEF Satellite2 Transform {
    scaleOrientation 0 1 0 0
    scale 0.6667 0.6667 0.6667
    rotation 0 0 0 1
    children [
      DEF BusBody2 Shape {
        appearance Appearance {
          texture ImageTexture {
            url "texture/cubesat.jpg"
          }
          material Material {
            specularColor 1 1 1
            shininess 1
            emissiveColor 0.2 0.2 0.2
            diffuseColor 0.2 0.2 0.2
            ambientIntensity 0.6
          }
        }
        geometry Box {
          size 0.3 0.3 0.3
        }
      }
      DEF SolarPanel22 Transform {
        translation 0 0.16 -0.3
        children Shape {
          appearance Appearance {
            texture ImageTexture {
              url "texture/solarpanel.jpg"
            }
```

```
        material Material {
          shininess 1
          diffuseColor 0.9 0.767329 0.619635
          ambientIntensity 0
        }
      }
      geometry Box {
        size 0.3 0.02 0.3
      }
    }
  }
  DEF SolarPanel12 Transform {
    translation 0 0.16 0.3
    children Shape {
      appearance Appearance {
        texture ImageTexture {
          url "texture/solarpanel.jpg"
        }
        material Material {
          shininess 1
          diffuseColor 0.9 0.767329 0.619635
          ambientIntensity 0
        }
      }
      geometry Box {
        size 0.3 0.02 0.3
      }
    }
  }
  DEF Antenna2 Transform {
    children Transform {
      translation 0 -0.15 0
      children Shape {
        appearance Appearance {
          material Material {
            specularColor 1 1 1
            shininess 1
            diffuseColor 0.9 0.655038 0.319173
          }
        }
        geometry Cone {
          height 0.2
          bottomRadius 0.06
        }
      }
    }
```

```
      }
      DEF FollowSat2 Transform {
        translation -5.5 2 0
        rotation -0.0923 -0.7011 -0.0923 1.5878
        children DEF Eeek2 Viewpoint {
          description "Satellite2-fixed"
          position 0 0 0
        }
      }
    ]
  }
  DEF AntennaCone Transform {
    translation 0 -0.3333 0
    rotation 0 0 0 1
    children DEF ConeShape Shape {
      appearance DEF ConeOn Appearance {
        material DEF AntennaMaterial Material {
          transparency 0.4
          emissiveColor 0.99 0.424286 0.424286
          diffuseColor 0.8 0.255075 0.333323
        }
      }
      geometry Cone {
        height 1
        bottomRadius 0.1
      }
    }
  }
  DEF AntennaCone2 Transform {
    translation 0 -0.3333 0
    rotation 0 0 0 1
    children DEF ConeShape2 Shape {
      appearance DEF ConeOn2 Appearance {
        material DEF AntennaMaterial2 Material {
          transparency 0.4
          emissiveColor 0.99 0.424286 0.424286
          diffuseColor 0.8 0.255075 0.333323
        }
      }
      geometry Cone {
        height 1
        bottomRadius 0.1
      }
    }
  }
  DEF Sun Transform {
```

```
      translation -13528.3 -5687.39 1011.79
      children [
        DEF SunOrb Shape {
          appearance Appearance {
            texture ImageTexture {
              url "texture/sun.jpg"
            }
            material Material {
              specularColor 1 0.8 0
              shininess 1
              emissiveColor 1 1 0
              diffuseColor 1 0.8 0
              ambientIntensity 0.5
            }
          }
          geometry Sphere {
            radius 695.508
          }
        }
        DEF Umbra Transform {
          translation 14164.7 5954.94 -1059.38
          rotation 0 -0.062101 0.55379 5.10102
          children Shape {
            appearance Appearance {
              material Material {
                transparency 0.75
                diffuseColor 0 0 0
                ambientIntensity 1
              }
            }
            geometry Cone {
              height 1384
              bottomRadius 6.5
            }
          }
        }
      ]
    }
  ]
}
```

Appendix G IRIDIUM SIMULATION

In this section, the matlab files which create the inputs of the Iridium constellation are displayed.

G.1 Iridium generator

```
clc
clear
close all

url='https://celestrak.com/NORAD/elements/iridium-NEXT.txt';
TLEdata=webread(url);
name='IridiumTLE';
name=strcat(name,'.txt');
dir='C:\Users\Lluis\Desktop\UPC\2B\TFM\TLE';
loc=strcat(dir,filesep,name);
fid=fopen(loc,'w');
fprintf(fid,TLEdata);
fclose(fid);

TLEarray=txt2array(TLEdata);
for i=1:size(TLEarray,1)/3
    Iridium(i)=TLE(TLEarray([3*i-2,3*i-1,3*i],:));
    Iridium(i)=getMinAlt(Iridium(i));
    Iridium(i)=getTrueAnomaly(Iridium(i));
end

t=length(Iridium);
sat=1:t;
filename(1:t)="inputs/Orb_"+sat+".txt";
orbname(1:t)="Orb_"+sat+".txt";
SCname(1:t)="SC_cubesat"+sat+".txt";
SCfilename(1:t)="inputs/SC_cubesat"+sat+".txt";

for i=sat
    WriteOrbFileTLE(filename(i),name,Iridium(i).name);
    WriteSCFile(SCfilename(i),Iridium(i).name)
end
WriteSimMultSC(orbname,SCname,t);
```

G.2 txt2array

```
function TLEarray=txt2array(TLEfile)
    j=1;
    k=1;
    for i=1:length(TLEfile)
        if TLEfile(i)==newline
            k=k+1;
            j=1;
        else
            TLEarray(k,j)=TLEfile(i);
            j=j+1;
        end
    end
end
```

G.3 TLE

```
classdef TLE
    properties (SetAccess=private)
        name='', number='', designation='', epoch='', Der1='', Der2='',
            Drag='', ephType='', elNum='',
        inc='', RAAN='', ecc='', omega='', M='', n='', revNum=''
    end
    properties (Constant=true)
        RE=6371, muE=3.986004418e14
    end
    properties
        minAlt, trueAnomaly, E
    end
    methods
        function obj=TLE(array)
            obj.name=array(1,[1:24]);
            obj.number=array(2,[3:8]);
            obj.designation=array(2,[10:17]);
            obj.epoch=array(2,[19:32]);
            obj.Der1=array(2,[34:43]);
            obj.Der2=['.',array(2,[45:52])];
            obj.Drag=array(2,[54:61]);
            obj.ephType=array(2,63);
            obj.elNum=array(2,[65:68]);
            obj.inc=array(3,[9:16]);
            obj.RAAN=array(3,[18:25]);
            obj.ecc=['.',array(3,[27:33])];
            obj.omega=array(3,[35:42]);
            obj.M=array(3,[44:51]);
            obj.n=array(3,[53:63]);
            obj.revNum=array(3,[64:68]);
        end
        function obj=getMinAlt(obj)
            a=(obj.muE/(str2double(obj.n)*(2*pi)/(24*3600))^2)^(1/3);
            rp=a*(1-str2double(obj.ecc));
            obj.minAlt=rp-obj.RE*1e3;
        end
        function obj=getTrueAnomaly(obj)
            obj.E=obj.KeplerSolver(str2double(obj.ecc),str2double(obj.M
                )/180*pi,1e-5,'Newton');
            obj.trueAnomaly=2*atan(sqrt((1+str2double(obj.ecc)/(1-
                str2double(obj.ecc))))*tan(obj.E/2))/pi*180;
        end
    end
    methods (Static)
```

```
function E=KeplerSolver(e,M,delta,solver)
    err=1000;
    E0=pi;
    maxit=1000;
    it=0;
    Eprev=E0;
    while err>delta&&it<maxit
        it=it+1;
        f=Eprev-e*sin(Eprev)-M;
        df=1-e*cos(Eprev);
        ddf=e*sin(Eprev);
        switch solver
            case "Newton"
                E=NewtonMethod(Eprev,f,df);
            case "Halley"
                E=HalleyMethod(Eprev,f,df,ddf);
            otherwise
                error("Invalid solver.")
        end
        err=abs(E-Eprev);
        Eprev=E;
    end
    if it>=maxit
        warning("Kepler equation did not converge")
    end
end
function X=HalleyMethod(Xprev,f,df,ddf)
    if -1e-8<(2*df^2-f*ddf)&&(2*df^2-f*ddf)<=0
        df=0;
        f=1;
        ddf=1e-8;
    elseif 0<(2*df^2-f*ddf)&&(2*df^2-f*ddf)<1e-8
        df=0;
        f=-1;
        ddf=1e-8;
    end
    X=Xprev-(2*f*df)/(2*df^2-f*ddf);
end
function X=NewtonMethod(Xprev,f,df)
    if -1e-8<df&&df<=0
        df=-1e-8;
    elseif 0<df&&df<1e-8
        df=1e-8;
    end
    X=Xprev-f/df;
end
```

```
    end
end
```

G.4 WriteOrbFileTLE

```
function WriteOrbFileTLE(name,TLE,id)
    fid=fopen(name,'w');
    fprintf(fid,'<<<<<<<<<<<<<<<<<<  42: Orbit Description File  >>>>>>>>>>>>>>>>>>\n');
    fprintf(fid,'Test orbit           ! Description\n');
    fprintf(fid,'CENTRAL              ! Orbit Type (ZERO, FLIGHT, CENTRAL, THREE_BODY)\n');
    fprintf(fid,':::::::::::::: Use these lines if ZERO ::::::::::::::::::\n');
    fprintf(fid,'MINORBODY_2          ! World\n');
    fprintf(fid,'FALSE                ! Use Polyhedron Gravity\n');
    fprintf(fid,':::::::::::::: Use these lines if FLIGHT ::::::::::::::::::\n');
    fprintf(fid,'0                    ! Region Number\n');
    fprintf(fid,'FALSE                ! Use Polyhedron Gravity\n');
    fprintf(fid,':::::::::::::: Use these lines if Body-Centered Orbit ::::::::::::::::::\n');
    fprintf(fid,'EARTH                ! Orbit Center\n');
    fprintf(fid,'FALSE                ! Secular Orbit Drift Due to J2\n');
    fprintf(fid,'FILE                 ! Use Keplerian elements (KEP) or (RV) or FILE\n');
    fprintf(fid,'AE                   ! Use Peri/Apoapsis (PA) or min alt/ecc (AE)\n');
    fprintf(fid,'35786.0    35786.0   ! Periapsis & Apoapsis Altitude, km\n');
    fprintf(fid,'0  0                 ! Min Altitude (km), Eccentricity\n');
    fprintf(fid,'0                    ! Inclination (deg)\n');
    fprintf(fid,'0                    ! Right Ascension of Ascending Node (deg)\n');
    fprintf(fid,'0                    ! Argument of Periapsis (deg)\n');
    fprintf(fid,'0                    ! True Anomaly (deg)\n');
    fprintf(fid,'0.0  0.0  0.0        ! RV Initial Position (km)\n');
    fprintf(fid,'0.0  0.0  0.0        ! RV Initial Velocity (km/sec)\n');
    fprintf(fid,'TLE  "%s"            ! TLE or TRV format, Label to find in file\n',id);
    fprintf(fid,'"%s"                 ! File name\n',TLE);
    fprintf(fid,':::::::::::: Use these lines if Three-Body Orbit  
```

```
        ::::::::::::::::\n');
    fprintf(fid,'SUNEARTH                    ! Lagrange system\n');
    fprintf(fid,'LAGDOF_MODES                ! Propagate using
        LAGDOF_MODES or LAGDOF_COWELL or LAGDOF_SPLINE\n');
    fprintf(fid,'MODES                       ! Initialize with MODES
        or XYZ or FILE\n');
    fprintf(fid,'L2                          ! Libration point (L1,
        L2, L3, L4, L5)\n');
    fprintf(fid,'800000.0                    ! XY Semi-major axis,
        km\n');
    fprintf(fid,'45.0                        ! Initial XY Phase, deg
        \n');
    fprintf(fid,'CW                          ! Sense (CW, CCW),
        viewed from +Z\n');
    fprintf(fid,'0.0                         ! Second XY Mode Semi-
        major Axis, km (L4, L5 only)\n');
    fprintf(fid,'0.0                         ! Second XY Mode
        Initial Phase, deg (L4, L5 only)\n');
    fprintf(fid,'CW                          ! Sense (CW, CCW),
        viewed from +Z (L4, L5 only)\n');
    fprintf(fid,'400000.0                    ! Z Semi-axis, km\n');
    fprintf(fid,'60.0                        ! Initial Z Phase, deg\
        n');
    fprintf(fid,'1.05  0.5  0.0              ! Initial X, Y, Z (Non-
        dimensional)\n');
    fprintf(fid,'0.0   0.0  0.0              ! Initial Xdot, Ydot,
        Zdot (Non-dimensional)\n');
    fprintf(fid,'TRV  "ORB_ID"               ! TLE, TRV or SPLINE
        format, Label to find in file\n');
    fprintf(fid,'"TRV.txt"                   ! File name\n');
    fprintf(fid,'******************** Formation Frame Parameters
        **********************\n');
    fprintf(fid,'N                           ! Formation Frame Fixed
        in [NL]\n');
    fprintf(fid,'0.0  0.0  0.0  123          ! Euler Angles (deg)
        and Sequence\n');
    fprintf(fid,'N                           ! Formation Origin
        expressed in [NL]\n');
    fprintf(fid,'0.0  0.0  0.0               ! Formation Origin wrt
        Ref Orbit (m)\n');
    fclose(fid);
end
```

G.5 WriteSCFile

```
function WriteSCFile(name,id)
    fid=fopen(name,'w');
    fprintf(fid,'<<<<<<<<<<<<<<<<<<< 42: Spacecraft Description File
       >>>>>>>>>>>>>>>>>>>\n');
    fprintf(fid,'1-U Cubesat                    ! Description\n');
    fprintf(fid,'"%s"                           ! Label\n',id);
    fprintf(fid,'GenScSpriteAlpha.ppm           ! Sprite File Name\n');
    fprintf(fid,'PASSIVE_FSW                    ! Flight Software
       Identifier\n');
    fprintf(fid,'0.2                            ! FSW Sample Time, sec\
       n');
    fprintf(fid,'************************ Orbit Parameters
       ***************************\n');
    fprintf(fid,'ENCKE                          ! Orbit Prop FIXED,
       EULER_HILL, or ENCKE\n');
    fprintf(fid,'CM                             ! Pos of CM or ORIGIN,
       wrt F\n');
    fprintf(fid,'    0.00000    0.00000    0.00000 ! Pos wrt F\n');
    fprintf(fid,'    0.00000    0.00000    0.00000 ! Vel wrt F\n');
    fprintf(fid,'************************** Initial Attitude
       ***************************\n');
    fprintf(fid,'NAN                            ! Ang Vel wrt [NL], Att
       [QA] wrt [NLF]\n');
    fprintf(fid,'0.0    0.0    0.0              ! Ang Vel (deg/sec)\n');
    fprintf(fid,'0.0    0.0    0.0    1.0       ! Quaternion\n');
    fprintf(fid,'0.0    0.0    0.0    123       ! Angles (deg) & Euler
       Sequence\n');
    fprintf(fid,'************************** Dynamics Flags
       ***************************\n');
    fprintf(fid,'DYN_JOINT                      ! Rotation STEADY,
       KIN_JOINT, or DYN_JOINT\n');
    fprintf(fid,'FALSE                          ! Passive Joint Forces
       and Torques Enabled\n');
    fprintf(fid,'FALSE                          ! Compute Constraint
       Forces and Torques\n');
    fprintf(fid,'REFPT_CM                       ! Mass Props referenced
       to REFPT_CM or REFPT_JOINT\n');
    fprintf(fid,'FALSE                          ! Flex Active\n');
    fprintf(fid,'FALSE                          ! Include 2nd Order Flex
       Terms\n');
    fprintf(fid,'2.0                            ! Drag Coefficient\n');
    fprintf(fid,'
       ****************************************************************\
       n');
```

```
fprintf(fid,'************************ Body Parameters
    ****************************\n');
fprintf(fid,'
    ************************************************************************\
    n');
fprintf(fid,'1                              ! Number of Bodies\n');
fprintf(fid,'==================================== Body 0
    ====================================\n');
fprintf(fid,'1.0                            ! Mass\n');
fprintf(fid,'2.0   3.0   1.2                ! Moments of Inertia (kg
    -m^2)\n');
fprintf(fid,'0.0   0.0   0.0                ! Products of Inertia (
    xy,xz,yz)\n');
fprintf(fid,'0.0   0.0   0.0                ! Location of mass
    center, m\n');
fprintf(fid,'0.0   0.0   0.0                ! Constant Embedded
    Momentum (Nms)\n');
fprintf(fid,'Cubesat_1U.obj                 ! Geometry Input File
    Name\n');
fprintf(fid,'NONE                           ! Flex File Name\n');
fprintf(fid,'
    ************************************************************************\
    n');
fprintf(fid,'************************** Joint Parameters
    **************************\n');
fprintf(fid,'
    ************************************************************************\
    n');
fprintf(fid,'          (Number of Joints is Number of Bodies minus
    one)\n');
fprintf(fid,'==================================== Joint 0
    ====================================\n');
fprintf(fid,'0 1                            ! Inner, outer body
    indices\n');
fprintf(fid,'1   213   GIMBAL               ! RotDOF, Seq, GIMBAL or
    SPHERICAL\n');
fprintf(fid,'0   123                        ! TrnDOF, Seq\n');
fprintf(fid,'FALSE  FALSE  FALSE            ! RotDOF Locked\n');
fprintf(fid,'FALSE  FALSE  FALSE            ! TrnDOF Locked\n');
fprintf(fid,'0.0    0.0    0.0              ! Initial Angles [deg]\n
    ');
fprintf(fid,'0.0    0.0    0.0              ! Initial Rates, deg/sec
    \n');
fprintf(fid,'0.0    0.0    0.0              ! Initial Displacements
    [m]\n');
fprintf(fid,'0.0    0.0    0.0              ! Initial Displacement
```

```
    Rates, m/sec\n');
fprintf(fid, '0.0   0.0   0.0   312            ! Bi to Gi Static Angles
    [deg] & Seq\n');
fprintf(fid, '0.0   0.0   0.0   312            ! Go to Bo Static Angles
    [deg] & Seq\n');
fprintf(fid, '0.0   0.0   0.0                  ! Position wrt inner
    body origin, m\n');
fprintf(fid, '0.0   0.0   0.0                  ! Position wrt outer
    body origin, m\n');
fprintf(fid, '0.0   0.0   0.0                  ! Rot Passive Spring
    Coefficients (Nm/rad)\n');
fprintf(fid, '0.0   0.0   0.0                  ! Rot Passive Damping
    Coefficients (Nms/rad)\n');
fprintf(fid, '0.0   0.0   0.0                  ! Trn Passive Spring
    Coefficients (N/m)\n');
fprintf(fid, '0.0   0.0   0.0                  ! Trn Passive Damping
    Coefficients (Ns/m)\n');
fprintf(fid, '************************ Wheel Parameters
    ************************\n');
fprintf(fid, '0                                ! Number of wheels\n');
fprintf(fid, '================================  Wheel 0
    ================================\n');
fprintf(fid, '0.0                              ! Initial Momentum, N-m-
    sec\n');
fprintf(fid, '1.0   0.0   0.0                  ! Wheel Axis Components,
    [X, Y, Z]\n');
fprintf(fid, '0.14   50.0                      ! Max Torque (N-m),
    Momentum (N-m-sec)\n');
fprintf(fid, '0.012                            ! Wheel Rotor Inertia,
    kg-m^2\n');
fprintf(fid, '0.48                             ! Static Imbalance, g-cm
    \n');
fprintf(fid, '13.7                             ! Dynamic Imbalance, g-
    cm^2\n');
fprintf(fid, '0                                ! Flex Node Index\n');
fprintf(fid, '************************ MTB Parameters
    ************************\n');
fprintf(fid, '0                                ! Number of MTBs\n
    ');
fprintf(fid, '================================  MTB 0
    ================================\n');
fprintf(fid, '180.0                            ! Saturation (A-m^2)\n')
    ;
fprintf(fid, '1.0   0.0   0.0                  ! MTB Axis Components, [
    X, Y, Z]\n');
fprintf(fid, '0                                ! Flex Node Index\n');
```

```
fprintf(fid,'************************ Thruster Parameters
    ************************\n');
fprintf(fid,'0                              ! Number of Thrusters\n'
    );
fprintf(fid,'================================  Thr 0
    ================================\n');
fprintf(fid,' 1.0                           ! Thrust Force (N)\n');
fprintf(fid,'0   -1.0   0.0   0.0           ! Body, Thrust Axis \n')
    ;
fprintf(fid,' 1.0   1.0   1.0               ! Location in B0, m\n');
fprintf(fid,'0                              ! Flex Node Index\n');
fprintf(fid,'****************************** Gyro
    ******************************\n');
fprintf(fid,'0                              ! Number of Gyro Axes
    \n');
fprintf(fid,'================================ Axis 0
    ================================\n');
fprintf(fid,'0.1                            ! Sample Time, sec\n');
fprintf(fid,'1.0   0.0   0.0                ! Axis expressed in Body
    Frame\n');
fprintf(fid,'1000.0                         ! Max Rate, deg/sec\n');
fprintf(fid,'100.0                          ! Scale Factor Error,
    ppm\n');
fprintf(fid,'1.0                            ! Quantization, arcsec \
    n');
fprintf(fid,'0.07                           ! Angle Random Walk (deg
    /rt-hr)\n');
fprintf(fid,'0.1   1.0                      ! Bias Stability (deg/hr
    ) over timespan (hr)\n');
fprintf(fid,'0.1                            ! Angle Noise, arcsec
    RMS\n');
fprintf(fid,'0.1                            ! Initial Bias (deg/hr)\
    n');
fprintf(fid,'0                              ! Flex Node Index\n');
fprintf(fid,'************************ Magnetometer
    ******************************\n');
fprintf(fid,'0                              ! Number of Magnetometer
    Axes\n');
fprintf(fid,'================================ Axis 0
    ================================\n');
fprintf(fid,'0.1                            ! Sample Time, sec\n');
fprintf(fid,'1.0   0.0   0.0                ! Axis expressed in Body
    Frame\n');
fprintf(fid,'60.0E-6                        ! Saturation, Tesla\n');
fprintf(fid,'0.0                            ! Scale Factor Error,
    ppm\n');
```

```
fprintf(fid,'1.0E-6                    ! Quantization, Tesla \n
    ');
fprintf(fid,'1.0E-6                    ! Noise, Tesla RMS\n');
fprintf(fid,'0                         ! Flex Node Index\n');
fprintf(fid,'********************** Coarse Sun Sensor
    ****************************\n');
fprintf(fid,'0                         ! Number of Coarse Sun
    Sensors\n');
fprintf(fid,'================================ CSS 0
    ========================================\n');
fprintf(fid,'0.1                       ! Sample Time, sec\n');
fprintf(fid,'0   1.0  1.0  1.0         ! Axis expressed in Body
    Frame\n');
fprintf(fid,'90.0                      ! Half-cone Angle, deg\n
    ');
fprintf(fid,'1.0                       ! Scale Factor\n');
fprintf(fid,'0.001                     ! Quantization\n');
fprintf(fid,'0                         ! Flex Node Index\n');
fprintf(fid,'********************** Fine Sun Sensor
    ****************************\n');
fprintf(fid,'0                         ! Number of Fine Sun
    Sensors\n');
fprintf(fid,'================================ FSS 0
    ========================================\n');
fprintf(fid,'0.2                       ! Sample Time, sec\n');
fprintf(fid,'30.0  20.0  10.0  213     ! Mounting Angles (deg),
    Seq in Body\n');
fprintf(fid,'32.0   32.0               ! X, Y FOV Size, deg\n')
    ;
fprintf(fid,'0.1                       ! Noise Equivalent Angle
    , deg RMS\n');
fprintf(fid,'0.5                       ! Quantization, deg\n');
fprintf(fid,'0                         ! Flex Node Index\n');
fprintf(fid,'********************** Star Tracker
    ****************************\n');
fprintf(fid,'0                         ! Number of Star
    Trackers\n');
fprintf(fid,'================================ ST 0
    ========================================\n');
fprintf(fid,'0.25                      ! Sample Time, sec\n');
fprintf(fid,'30.0  20.0  10.0  213     ! Mounting Angles (deg),
    Seq in Body\n');
fprintf(fid,'8.0   8.0                 ! X, Y FOV Size, deg\n')
    ;
fprintf(fid,'30.0  10.0  10.0          ! Sun, Earth, Moon
    Exclusion Angles, deg\n');
```

```
    fprintf(fid,'2.0  2.0  20.0                ! Noise Equivalent Angle
        , arcsec RMS\n');
    fprintf(fid,'0                             ! Flex Node Index\n');
    fprintf(fid,'***************************** GPS
        ***********************************\n');
    fprintf(fid,'0                             ! Number of GPS
        Receivers\n');
    fprintf(fid,'================================ GPSR 0
        ==========================================\n');
    fprintf(fid,'0.25                          ! Sample Time, sec\n');
    fprintf(fid,'4.0                           ! Position Noise, m RMS\
        n');
    fprintf(fid,'0.02                          ! Velocity Noise, m/sec
        RMS\n');
    fprintf(fid,'20.0E-9                       ! Time Noise, sec RMS\n'
        );
    fprintf(fid,'0                             ! Flex Node Index\n');
    fprintf(fid,'************************** Accelerometer
        ****************************\n');
    fprintf(fid,'0                             ! Number of Accel Axes\n
        ');
    fprintf(fid,'================================ Axis 0
        ==========================================\n');
    fprintf(fid,'0.1                           ! Sample Time, sec\n');
    fprintf(fid,'0.5  1.0  1.5                 ! Position in B[0] (m)\n
        ');
    fprintf(fid,'1.0  0.0  0.0                 ! Axis expressed in Body
        Frame\n');
    fprintf(fid,'1.0                           ! Max Acceleration (m/s
        ^2)\n');
    fprintf(fid,'0.0                           ! Scale Factor Error,
        ppm\n');
    fprintf(fid,'0.05                          ! Quantization, m/s^2\n'
        );
    fprintf(fid,'0.0                           ! DV Random Walk (m/s/rt
        -hr)\n');
    fprintf(fid,'0.0 1.0                       ! Bias Stability (m/s^2)
        over timespan (hr)\n');
    fprintf(fid,'0.0                           ! DV Noise, m/s\n');
    fprintf(fid,'0.5                           ! Initial Bias (m/s^2)\n
        ');
    fprintf(fid,'0                             ! Flex Node Index\n');
    fclose(fid);
end
```

G.6 WriteSimMultSC

```
function WriteSimMultSC(orb,SC,t)
    fid=fopen('inputs/Inp_Sim.txt','w');
    fprintf(fid,'<<<<<<<<<<<<<<<<<<  42: The Mostly Harmless Simulator  >>>>>>>>>>>>>>>>>>\n');
    fprintf(fid,'************************ Simulation Control ************************\n');
    fprintf(fid,'FAST                                    ! Time Mode (FAST, REAL, or EXTERNAL)\n');
    fprintf(fid,'36000.0       0.1                       ! Sim Duration, Step Size [sec]\n');
    fprintf(fid,'10.0                                    ! File Output Interval [sec]\n');
    fprintf(fid,'TRUE                                    ! Graphics Front End?\n');
    fprintf(fid,'Inp_Cmd.txt                             ! Command Script File Name\n');
    fprintf(fid,'************************  Reference Orbits  ************************\n');
    fprintf(fid,'%i                                      ! Number of Reference Orbits\n',t);
    for i=1:t
        fprintf(fid,'TRUE   %s                           ! Input file name for Orb 0\n',orb(i));
    end
    fprintf(fid,'****************************  Spacecraft  ****************************\n');
    fprintf(fid,'%i                                      ! Number of Spacecraft\n',t);
    for i=1:t
        fprintf(fid,'TRUE  %i %s                         ! Existence, RefOrb, Input file for SC  0\n',i-1,SC(i));
    end
    fprintf(fid,'**************************** Environment ****************************\n');
    fprintf(fid,'02 04 2021                              ! Date (UTC) (Month, Day, Year)\n');
    fprintf(fid,'12 49 00.00                             ! Time (UTC) (Hr, Min, Sec)\n');
    fprintf(fid,'37.0                                    ! Leap Seconds (sec)\n');
    fprintf(fid,'NOMINAL                                 ! F10.7, Ap (USER, NOMINAL or TWOSIGMA)\n');
    fprintf(fid,'230.0                                   ! If USER_DEFINED, enter desired F10.7 value\n');
```

```
fprintf(fid,'100.0                    !  If USER_DEFINED,
    enter desired AP value\n');
fprintf(fid,'IGRF                     !  Magfield (NONE,
    DIPOLE,IGRF)\n');
fprintf(fid,'8   8                    !  IGRF Degree and
    Order (<=10)\n');
fprintf(fid,'2   0                    !  Earth Gravity Model
    N and M (<=18)\n');
fprintf(fid,'2   0                    !  Mars Gravity Model
    N and M (<=18)\n');
fprintf(fid,'2   0                    !  Luna Gravity Model
    N and M (<=18)>\n');
fprintf(fid,'FALSE   FALSE            !  Aerodynamic Forces
    & Torques (Shadows)\n');
fprintf(fid,'FALSE                    !  Gravity Gradient
    Torques\n');
fprintf(fid,'FALSE   FALSE            !  Solar Pressure
    Forces & Torques (Shadows)\n');
fprintf(fid,'FALSE                    !  Gravity
    Perturbation Forces\n');
fprintf(fid,'FALSE                    !  Passive Joint
    Torques\n');
fprintf(fid,'FALSE                    !  Thruster Plume
    Forces & Torques\n');
fprintf(fid,'FALSE                    !  RWA Imbalance
    Forces and Torques\n');
fprintf(fid,'FALSE                    !  Contact Forces and
    Torques\n');
fprintf(fid,'FALSE                    !  CFD Slosh Forces
    and Torques\n');
fprintf(fid,'FALSE                    !  Output
    Environmental Torques to Files\n');
fprintf(fid,'********************  Celestial Bodies of Interest
    ********************\n');
fprintf(fid,'MEAN                     !  Ephem Option (MEAN
    or DE430)\n');
fprintf(fid,'TRUE                     !  Mercury\n');
fprintf(fid,'TRUE                     !  Venus\n');
fprintf(fid,'TRUE                     !  Earth and Luna\n');
fprintf(fid,'TRUE                     !  Mars and its moons\
    n');
fprintf(fid,'TRUE                     !  Jupiter and its
    moons\n');
fprintf(fid,'TRUE                     !  Saturn and its
    moons\n');
fprintf(fid,'TRUE                     !  Uranus and its
```

```
        moons\n');
    fprintf(fid, 'TRUE                              !  Neptune and its
        moons\n');
    fprintf(fid, 'TRUE                              !  Pluto and its moons
        \n');
    fprintf(fid, 'FALSE                             !  Asteroids and
        Comets\n');
    fprintf(fid, '****************  Lagrange Point Systems of Interest
        ****************\n');
    fprintf(fid, 'FALSE                             !  Earth-Moon\n');
    fprintf(fid, 'FALSE                             !  Sun-Earth\n');
    fprintf(fid, 'FALSE                             !  Sun-Jupiter\n');
    fprintf(fid, '************************  Ground Stations
        ************************\n');
    fprintf(fid, '5                                                 !  Number
        of Ground Stations\n');
    fprintf(fid, 'TRUE  EARTH   -77.0   37.0   "GSFC"               !  Exists,
        World, Lng, Lat, Label\n');
    fprintf(fid, 'TRUE  EARTH   -155.6  19.0   "South Point"       !  Exists,
        World, Lng, Lat, Label\n');
    fprintf(fid, 'TRUE  EARTH   115.4   -29.0  "Dongara"           !  Exists,
        World, Lng, Lat, Label\n');
    fprintf(fid, 'TRUE  EARTH   -71.0   -33.0  "Santiago"          !  Exists,
        World, Lng, Lat, Label\n');
    fprintf(fid, 'TRUE  LUNA    45.0    45.0   "Moon Base Alpha"   !  Exists,
        World, Lng, Lat, Label\n');
    fclose(fid);
end
```

Appendix H MODIFIED ACAPP

In this section, the modified *AcApp* program will be displayed. The second *AcApp* created for the second spacecraft is identical with the exception of the commands and socket configuration. As such, it has not been included.

```
#include "Ac.h"

/* #ifdef __cplusplus
** namespace _42 {
** using namespace Kit;
** #endif
*/

extern void WriteToFile(FILE *StateFile, struct AcType *AC);
extern void WriteToGmsec(struct AcType *AC);
extern void WriteToSocket(SOCKET Socket, struct AcType *AC);
extern void ReadFromFile(FILE *StateFile, struct AcType *AC);
extern void ReadFromGmsec(struct AcType *AC);
extern void ReadFromSocket(SOCKET Socket, struct AcType *AC);



#ifdef _AC_STANDALONE_
/**********************************************************************
   */
/* This function copies needed parameters from the SC structure to
   */
/* the AC structure.  This is a crude first pass.  It only allocates
   */
/* memory for the structures, and counts on the data to be filled in
   */
/* via messages.
   */
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 */


}
/*****************************************************************
    */
void InitAC(struct AcType *AC)
{
    AC->Init = 1;

    AC->EchoEnabled = 1;

    /* Controllers */
    AC->CfsCtrl.Init = 1;
}
#endif
/*****************************************************************
    */
/*  Some Simple Sensor Processing Functions
    */
```

```
/*  corresponding to the Sensor Models in 42sensors.c
    */
/*  Note!  These are simple, sometimes naive.  Use with care.
    */
/*****************************************************************
    */
void GyroProcessing(struct AcType *AC)
{
      struct AcGyroType *G;
      double A0xA1[3];
      double A[3][3],b[3],Ai[3][3];
      double AtA[3][3] = {{0.0,0.0,0.0},{0.0,0.0,0.0},{0.0,0.0,0.0}};
      double Atb[3] = {0.0,0.0,0.0};
      double AtAi[3][3];
      long Ig,i,j;

      if (AC->Ngyro == 0) {
         /* AC->wbn populated by true S->B[0].wn in 42sensors.c */
      }
      else if (AC->Ngyro == 1) {
         G = &AC->Gyro[0];
         for(i=0;i<3;i++) AC->wbn[i] = G->Rate*G->Axis[i];
      }
      else if (AC->Ngyro == 2) {
         VxV(AC->Gyro[0].Axis,AC->Gyro[1].Axis,A0xA1);
         for(i=0;i<3;i++) {
            A[0][i] = AC->Gyro[0].Axis[i];
            A[1][i] = AC->Gyro[1].Axis[i];
            A[2][i] = A0xA1[i];
         }
         b[0] = AC->Gyro[0].Rate;
         b[1] = AC->Gyro[1].Rate;
         b[2] = 0.0;
         MINV3(A,Ai);
         MxV(Ai,b,AC->wbn);
      }
      else if (AC->Ngyro > 2) {
         /* Normal Equations */
         for(Ig=0;Ig<AC->Ngyro;Ig++) {
            G = &AC->Gyro[Ig];
            for(i=0;i<3;i++) {
               Atb[i] += G->Rate*G->Axis[i];
               for(j=0;j<3;j++) {
                  AtA[i][j] += G->Axis[i]*G->Axis[j];
               }
            }
```

```
            }
            MINV3(AtA,AtAi);
            MxV(AtAi,Atb,AC->wbn);
        }
}
/**********************************************************************
    */
void MagnetometerProcessing(struct AcType *AC)
{
        struct AcMagnetometerType *M;
        double A0xA1[3];
        double A[3][3],b[3],Ai[3][3];
        double AtA[3][3] = {{0.0,0.0,0.0},{0.0,0.0,0.0},{0.0,0.0,0.0}};
        double Atb[3] = {0.0,0.0,0.0};
        double AtAi[3][3];
        long Im,i,j;

        if (AC->Nmag == 0) {
            /* AC->bvb populated by true S->bvb in 42sensors.c */
        }
        else if (AC->Nmag == 1) {
            M = &AC->MAG[0];
            for(i=0;i<3;i++) AC->bvb[i] = M->Field*M->Axis[i];
        }
        else if (AC->Nmag == 2) {
            VxV(AC->MAG[0].Axis,AC->MAG[1].Axis,A0xA1);
            for(i=0;i<3;i++) {
                A[0][i] = AC->MAG[0].Axis[i];
                A[1][i] = AC->MAG[1].Axis[i];
                A[2][i] = A0xA1[i];
            }
            b[0] = AC->MAG[0].Field;
            b[1] = AC->MAG[1].Field;
            b[2] = 0.0;
            MINV3(A,Ai);
            MxV(Ai,b,AC->bvb);
        }
        else if (AC->Nmag > 2) {
            /* Normal Equations */
            for(Im=0;Im<AC->Nmag;Im++) {
                M = &AC->MAG[Im];
                for(i=0;i<3;i++) {
                    Atb[i] += M->Field*M->Axis[i];
                    for(j=0;j<3;j++) {
                        AtA[i][j] += M->Axis[i]*M->Axis[j];
                    }
```

```
            }
         }
         MINV3(AtA,AtAi);
         MxV(AtAi,Atb,AC->bvb);
      }
}
/*****************************************************************
   */
void CssProcessing(struct AcType *AC)
{
      struct AcCssType *Css;
      double AtA[3][3] = {{0.0,0.0,0.0},{0.0,0.0,0.0},{0.0,0.0,0.0}};
      double Atb[3] = {0.0,0.0,0.0};
      double AtAi[3][3];
      double A[2][3],b[2];
      long Ic,i,j;
      long Nvalid = 0;
      double InvalidSVB[3] = {1.0,0.0,0.0}; /* Safe vector if SunValid
         == FALSE */

      if (AC->Ncss == 0) {
         /* AC->svb populated by true S->svb in 42sensors.c */
      }
      else {
         for(Ic=0;Ic<AC->Ncss;Ic++) {
            Css = &AC->CSS[Ic];
            if (Css->Valid) {
               Nvalid++;
               /* Normal equations, assuming Nvalid will end up > 2 */
               for(i=0;i<3;i++) {
                  Atb[i] += Css->Axis[i]*Css->Illum/Css->Scale;

                  for(j=0;j<3;j++) {
                     AtA[i][j] += Css->Axis[i]*Css->Axis[j];
                  }
               }
               /* In case Nvalid ends up == 2 */
               for(i=0;i<3;i++) {
                  A[0][i] = A[1][i];
                  A[1][i] = Css->Axis[i];
               }
               b[0] = b[1];
               b[1] = Css->Illum/Css->Scale;
            }
         }
         if (Nvalid > 2) {
```

```
            AC->SunValid = TRUE;
            MINV3(AtA,AtAi);
            MxV(AtAi,Atb,AC->svb);
            UNITV(AC->svb);
         }
         else if (Nvalid == 2) {
            AC->SunValid = TRUE;
            for(i=0;i<3;i++) AC->svb[i] = b[0]*A[0][i] + b[1]*A[1][i];
            UNITV(AC->svb);
         }
         else if (Nvalid == 1) {
            AC->SunValid = TRUE;
            for(i=0;i<3;i++) AC->svb[i] = Atb[i];
            UNITV(AC->svb);
         }
         else {
            AC->SunValid = FALSE;
            for(i=0;i<3;i++) AC->svb[i] = InvalidSVB[i];
         }
      }
}
/*
    ***************************************************************************
    */
/* This function assumes FSS FOVs don't overlap, and FSS overwrites CSS
    */
void FssProcessing(struct AcType *AC)
{
      struct AcFssType *FSS;
      double tanx,tany,z;
      long Ifss,i;

      for(Ifss=0;Ifss<AC->Nfss;Ifss++) {
         FSS = &AC->FSS[Ifss];
         if (FSS->Valid) {
            AC->SunValid = 1;
            tanx = tan(FSS->SunAng[0]);
            tany = tan(FSS->SunAng[1]);
            z = 1.0/sqrt(1.0+tanx*tanx+tany*tany);
            FSS->SunVecS[0] = z*tanx;
            FSS->SunVecS[1] = z*tany;
            FSS->SunVecS[2] = z;
            MTxV(FSS->CB,FSS->SunVecS,FSS->SunVecB);
            for(i=0;i<3;i++) AC->svb[i] = FSS->SunVecB[i];
         }
      }
```

```
}
/*********************************************************************
   */
/* TODO: Weight measurements to reduce impact of "weak" axis */
void StarTrackerProcessing(struct AcType *AC)
{
      long Ist,i;
      struct AcStarTrackerType *ST;
      long Nvalid = 0;
      double qbn[4];

      if (AC->Nst == 0) {
         /* AC->qbn populated by true S->B[0].qn in 42sensors.c */
         AC->StValid = TRUE;
      }
      else {
         /* Naive averaging */
         for(i=0;i<4;i++) AC->qbn[i] = 0.0;
         for(Ist=0;Ist<AC->Nst;Ist++) {
            ST = &AC->ST[Ist];
            if (ST->Valid) {
               Nvalid++;
               QTxQ(ST->qb,ST->qn,qbn);
               RECTIFYQ(qbn);
               for(i=0;i<4;i++) AC->qbn[i] += qbn[i];
            }
         }
         if (Nvalid > 0) {
            AC->StValid = TRUE;
            UNITQ(AC->qbn);
         }
         else {
            AC->StValid = FALSE;
            AC->qbn[3] = 1.0;
         }
      }
}
/*********************************************************************
   */
void GpsProcessing(struct AcType *AC)
{
      struct AcGpsType *G;
      double DaysSinceWeek,DaysSinceRollover,DaysSinceEpoch,JD;
      long i;

      if (AC->Ngps == 0) {
```

```
            /* AC->Time, AC->PosN, AC->VelN */
            /* populated in 42sensors.c */
        }
        else {
            G = &AC->GPS[0];
            /* GPS Time is seconds since 6 Jan 1980 00:00:00.0, which is
                JD = 2444244.5 */
            DaysSinceWeek = G->Sec/86400.0;
            DaysSinceRollover = DaysSinceWeek + 7.0*G->Week;
            DaysSinceEpoch = DaysSinceRollover + 7168.0*G->Rollover;
            JD = DaysSinceEpoch + 2444244.5;
            /* AC->Time is seconds since J2000, which is JD = 2451545.0 */
            AC->Time = (JD-2451545.0)*86400.0;

            /* Position, Velocity */
            for(i=0;i<3;i++) {
                AC->PosN[i] = AC->GPS[0].PosN[i];
                AC->VelN[i] = AC->GPS[0].VelN[i];
            }
        }
}
/**********************************************************************
    */
void AccelProcessing(struct AcType *AC)
{
}
/**********************************************************************
    */
/*  End Sensor Processing Functions
    */
/**********************************************************************
    */
/*  Some Actuator Processing Functions
    */
/**********************************************************************
    */
void WheelProcessing(struct AcType *AC)
{
        struct AcWhlType *W;
        long Iw;

        for(Iw=0;Iw<AC->Nwhl;Iw++) {
            W = &AC->Whl[Iw];
            W->Tcmd = Limit(-VoV(AC->Tcmd,W->DistVec),-W->Tmax,W->Tmax);
        }
}
```

```
/*****************************************************************
    */
void MtbProcessing(struct AcType *AC)
{
       struct AcMtbType *M;
       long Im;

       for(Im=0;Im<AC->Nmtb;Im++) {
          M = &AC->MTB[Im];
          M->Mcmd = Limit(VoV(AC->Mcmd,M->DistVec),-M->Mmax,M->Mmax);
       }
}
/*****************************************************************
    */
/*  End Actuator Processing Functions
    */
/*****************************************************************
    */
void AcFsw(struct AcType *AC)
{
       struct AcCfsCtrlType *C;
       struct AcJointType *G;
       double L1[3],L2[3],L3[3];
       double HxB[3];
       long i,j;

       C = &AC->CfsCtrl;
       G = &AC->G[0];

       if (C->Init) {
          C->Init = 0;
          for(i=0;i<3;i++) FindPDGains(AC->MOI[i][i],0.1,0.7,&C->Kr[i],&
              C->Kp[i]);
          C->Kunl = 1.0E6;
          FindPDGains(100.0,0.2,1.0,&G->AngRateGain[0],&G->AngGain[0]);
          G->MaxAngRate[0] = 1.0*D2R;
          G->MaxTrq[0] = 10.0;
       }

/* .. Sensor Processing */
       GyroProcessing(AC);
       MagnetometerProcessing(AC);
       CssProcessing(AC);
       FssProcessing(AC);
       StarTrackerProcessing(AC);
       GpsProcessing(AC);
```

```
/* .. Commanded Attitude */
      if (AC->GPS[0].Valid) {
         CopyUnitV(AC->PosN,L3);
         VxV(AC->PosN,AC->VelN,L2);
         UNITV(L2);
         UNITV(L3);
         for(i=0;i<3;i++) {
            L2[i] = -L2[i];
            L3[i] = -L3[i];
         }
         VxV(L2,L3,L1);
         UNITV(L1);
         for(i=0;i<3;i++) {
            AC->CLN[0][i] = L1[i];
            AC->CLN[1][i] = L2[i];
            AC->CLN[2][i] = L3[i];
         }
         C2Q(AC->CLN,AC->qln);
         AC->wln[1] = -MAGV(AC->VelN)/MAGV(AC->PosN);
      }
      else {
         for(i=0;i<3;i++) {
            for(j=0;j<3;j++) {
               AC->CLN[i][j] = 0.0;
            }
            AC->CLN[i][i] = 1.0;
            AC->qln[i] = 0.0;
            AC->wln[i] = 0.0;
         }
         AC->qln[3] = 1.0;
      }

/* .. Attitude Control */
      if (AC->StValid) {
         QxQT(AC->qbn,AC->Cmd.qrn,AC->qbr);
         RECTIFYQ(AC->qbr);
      }
      else {
         for(i=0;i<3;i++) AC->qbr[i] = 0.0;
         AC->qbr[3] = 1.0;
      }
      for(i=0;i<3;i++) {
         C->therr[i] = 2.0*AC->qbr[i];
         C->werr[i] = AC->wbn[i] - AC->Cmd.wrn[i];
         AC->Tcmd[i] = -C->Kr[i]*C->werr[i] - C->Kp[i]*C->therr[i];
```

```
        }
/* .. Momentum Management */
        for(i=0;i<3;i++) {
            AC->Hvb[i] = AC->MOI[i][i]*AC->wbn[i];
            for(j=0;j<AC->Nwhl;j++) AC->Hvb[i] += AC->Whl[j].Axis[i]*AC->
                Whl[j].H;
        }
        VxV(AC->Hvb,AC->bvb,HxB);
        for(i=0;i<3;i++) AC->Mcmd[i] = C->Kunl*HxB[i];

/* .. Solar Array Steering */
        G->Cmd.Ang[0] = atan2(AC->svb[0],AC->svb[2]);

/* .. Actuator Processing */
        WheelProcessing(AC);
        MtbProcessing(AC);
}

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 = 213;
        }

        else {
           AC->Cmd.Ang[0] = 90;
           AC->Cmd.Ang[1] = 0;
           AC->Cmd.Ang[2] = 0;
           AC->Cmd.RotSeq = 213;
        }


        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;

}
#ifdef _AC_STANDALONE_
/*********************************************************************
    */
int main(int argc, char **argv)
{
        FILE *ParmDumpFile;
        char FileName[120];
        struct AcType AC;
        SOCKET Socket;
        char hostname[20] = "localhost";
        int Port = 10301;

        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%02ld.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);
}
#endif
/* #ifdef __cplusplus
** }
** #endif
*/
```

Appendix I MODIFIED MAKEFILE

In this section, the modified makefile will be displayed, this modification was necessary to compile a second *AcApp*.

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

# Let's try to auto-detect what platform we're on.
# If this fails, set 42PLATFORM manually in the else block.
AUTOPLATFORM = Failed
ifeq ($(MSYSTEM),MINGW32)
    AUTOPLATFORM = Succeeded
    42PLATFORM = __MSYS__
endif
UNAME_S := $(shell uname -s)
ifeq ($(UNAME_S),Linux)
    AUTOPLATFORM = Succeeded
    42PLATFORM = __linux__
endif
ifeq ($(UNAME_S),Darwin)
    AUTOPLATFORM = Succeeded
    42PLATFORM = __APPLE__
endif
ifeq ($(AUTOPLATFORM),Failed)
    # Autodetect failed.  Set platform manually.
    #42PLATFORM = __APPLE__
    #42PLATFORM = __linux__
    42PLATFORM = __MSYS__
endif

GUIFLAG = -D _USE_GUI_
#GUIFLAG =

SHADERFLAG = -D _USE_SHADERS_
#SHADERFLAG =

CFDFLAG =
#CFDFLAG = -D _ENABLE_CFD_SLOSH_

FFTBFLAG =
#FFTBFLAG = -D _ENABLE_FFTB_CODE_

#GSFCFLAG =
GSFCFLAG = -D _USE_GSFC_WATERMARK_
```

```
#STANDALONEFLAG =
STANDALONEFLAG = -D _AC_STANDALONE_

NOS3FSWFLAG =
#NOS3FSWFLAG = -D _ENABLE_NOS3_FSW_

#GLFWFLAG =
GLFWFLAG = -D _USE_GLFW_

GMSECFLAG =
#GMSECFLAG = -D _ENABLE_GMSEC_
ifeq ($(strip $(GMSECFLAG)),)
    GMSECDIR =
    GMSECINC =
    GMSECBIN =
    GMSECLIB =
else
    GMSECDIR = ~/GMSEC/
    GMSECINC = -I $(GMSECDIR)include/
    GMSECBIN = -L $(GMSECDIR)bin/
    GMSECLIB = -lGMSECAPI
endif

# Basic directories
HOMEDIR = ./
PROJDIR = ./
KITDIR = $(PROJDIR)Kit/
OBJ = $(PROJDIR)Object/
INC = $(PROJDIR)Include/
SRC = $(PROJDIR)Source/
KITINC = $(KITDIR)Include/
KITSRC = $(KITDIR)Source/
INOUT = $(PROJDIR)InOut/
GSFCSRC = $(PROJDIR)/GSFC/Source/
IPCSRC = $(SRC)IPC/

#EMBEDDED = -D EMBEDDED_MATLAB
EMBEDDED =

ifeq ($(42PLATFORM),__APPLE__)
    # Mac Macros
    CINC = -I /usr/include -I /usr/local/include
    EXTERNDIR =
    GLINC = -I /System/Library/Frameworks/OpenGL.framework/Headers/ -I /
        System/Library/Frameworks/GLUT.framework/Headers/
    # ARCHFLAG = -arch i386
```

```
    ARCHFLAG = -arch x86_64

    LFLAGS = -bind_at_load
    ifeq ($(strip $(GLFWFLAG)),)
        LIBS = -framework System -framework Carbon -framework OpenGL -
            framework GLUT
        GUIOBJ = $(OBJ)42GlutGui.o $(OBJ)glkit.o
    else
        LIBS = -lglfw -framework System -framework Carbon -framework
            OpenGL -framework GLUT
        GUIOBJ = $(OBJ)42glfwgui.o $(OBJ)glkit.o
    endif
    EXENAME = 42
    CC = gcc
endif

ifeq ($(42PLATFORM),__linux__)
    # Linux Macros
    CINC =
    EXTERNDIR =

    ifneq ($(strip $(GUIFLAG)),)
        GUIOBJ = $(OBJ)42GlutGui.o $(OBJ)glkit.o
        #GLINC = -I /usr/include/
        GLINC = -I $(KITDIR)/include/GL/
        LIBS = -lglut -lGLU -lGL -ldl -lm
        LFLAGS = -L $(KITDIR)/GL/lib/
        ARCHFLAG =
    else
        GUIOBJ =
        GLINC =
        LIBS = -ldl -lm
        LFLAGS =
        ARCHFLAG =
    endif
    ifneq ($(strip $(NOS3FSWFLAG)),)
        LIBS += -lpthread
    endif
    EXENAME = 42
    CC = gcc
endif

ifeq ($(42PLATFORM),__MSYS__)
    CINC =
    EXTERNDIR = /c/42ExternalSupport/
```

```
    ifneq ($(strip $(GUIFLAG)),)
       GLEW = $(EXTERNDIR)GLEW/
       GLUT = $(EXTERNDIR)freeglut/
       LIBS = -lopengl32 -lglu32 -lfreeglut -lws2_32 -lglew32
       LFLAGS = -L $(GLUT)lib/ -L $(GLEW)lib/
       GUIOBJ = $(OBJ)42GlutGui.o $(OBJ)glkit.o
       GLINC = -I $(GLEW)include/GL/ -I $(GLUT)include/GL/
       ARCHFLAG = -D GLUT_NO_LIB_PRAGMA -D GLUT_NO_WARNING_DISABLE -D
           GLUT_DISABLE_ATEXIT_HACK
    else
       GUIOBJ =
       GLINC =
       LIBS = -lws2_32
       LFLAGS =
       ARCHFLAG =
    endif
    EXENAME = 42.exe
    CC = gcc
endif

# If not using GUI, don't compile GUI-related files
ifeq ($(strip $(GUIFLAG)),)
    GUIOBJ =
endif

# If not in FFTB, don't compile FFTB-related files
ifneq ($(strip $(FFTBFLAG)),)
    FFTBOBJ = $(OBJ)42fftb.o
else
    FFTBOBJ =
endif

ifneq ($(strip $(CFDFLAG)),)
    SLOSHOBJ = $(OBJ)42CfdSlosh.o
else
    SLOSHOBJ =
endif

# 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
```

```
ifneq ($(strip $(GMSECFLAG)),)
  GMSECOBJ = $(OBJ)gmseckit.o
  ACIPCOBJ = $(OBJ)AppReadFromFile.o $(OBJ)AppWriteToGmsec.o $(OBJ)
      AppReadFromGmsec.o \
      $(OBJ)AppWriteToSocket.o $(OBJ)AppReadFromSocket.o $(OBJ)
          AppWriteToFile.o
  SIMIPCOBJ = $(OBJ)SimWriteToFile.o $(OBJ)SimWriteToGmsec.o $(OBJ)
      SimWriteToSocket.o \
      $(OBJ)SimReadFromFile.o $(OBJ)SimReadFromGmsec.o $(OBJ)
          SimReadFromSocket.o
else
  GMSECOBJ =
  ACIPCOBJ = $(OBJ)AppReadFromFile.o \
      $(OBJ)AppWriteToSocket.o $(OBJ)AppReadFromSocket.o $(OBJ)
          AppWriteToFile.o
  SIMIPCOBJ = $(OBJ)SimWriteToFile.o $(OBJ)SimWriteToSocket.o \
      $(OBJ)SimReadFromFile.o $(OBJ)SimReadFromSocket.o
endif

42OBJ = $(OBJ)42main.o $(OBJ)42exec.o $(OBJ)42actuators.o $(OBJ)42cmd.o
    \
$(OBJ)42dynamics.o $(OBJ)42environs.o $(OBJ)42ephem.o $(OBJ)42fsw.o \
$(OBJ)42init.o $(OBJ)42ipc.o $(OBJ)42perturb.o $(OBJ)42report.o \
$(OBJ)42sensors.o \
$(OBJ)42nos3.o

KITOBJ = $(OBJ)dcmkit.o $(OBJ)envkit.o $(OBJ)fswkit.o $(OBJ)geomkit.o \
$(OBJ)iokit.o $(OBJ)mathkit.o $(OBJ)nrlmsise00kit.o $(OBJ)msis86kit.o \
$(OBJ)orbkit.o $(OBJ)radbeltkit.o $(OBJ)sigkit.o $(OBJ)sphkit.o $(OBJ)
    timekit.o

ACKITOBJ = $(OBJ)dcmkit.o $(OBJ)mathkit.o $(OBJ)fswkit.o $(OBJ)iokit.o
    $(OBJ)timekit.o

ACIPCOBJ = $(OBJ)AppReadFromFile.o \
$(OBJ)AppWriteToSocket.o $(OBJ)AppReadFromSocket.o $(OBJ)AppWriteToFile
    .o

#ANSIFLAGS = -Wstrict-prototypes -pedantic -ansi -Werror
ANSIFLAGS =

CFLAGS = -Wall -Wshadow -Wno-deprecated -g $(ANSIFLAGS) $(GLINC) $(
    CINC) -I $(INC) -I $(KITINC) -I $(KITSRC) $(GMSECINC) -O0 $(ARCHFLAG
    ) $(GUIFLAG) $(SHADERFLAG) $(CFDFLAG) $(FFTBFLAG) $(GSFCFLAG) $(
    GMSECFLAG) $(STANDALONEFLAG) $(NOS3FSWFLAG) $(GLFWFLAG)
```

```
############################  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)42main.o          : $(SRC)42main.c
        $(CC) $(CFLAGS) -c $(SRC)42main.c -o $(OBJ)42main.o

$(OBJ)42exec.o          : $(SRC)42exec.c $(INC)42.h
        $(CC) $(CFLAGS) -c $(SRC)42exec.c -o $(OBJ)42exec.o

$(OBJ)42actuators.o : $(SRC)42actuators.c $(INC)42.h $(INC)Ac.h $(INC)
    AcTypes.h
        $(CC) $(CFLAGS) -c $(SRC)42actuators.c -o $(OBJ)42actuators.o

$(OBJ)42cmd.o : $(SRC)42cmd.c $(INC)42.h $(INC)Ac.h $(INC)AcTypes.h
        $(CC) $(CFLAGS) -c $(SRC)42cmd.c -o $(OBJ)42cmd.o

$(OBJ)42dynamics.o      : $(SRC)42dynamics.c $(INC)42.h
        $(CC) $(CFLAGS) -c $(SRC)42dynamics.c -o $(OBJ)42dynamics.o

$(OBJ)42environs.o  : $(SRC)42environs.c $(INC)42.h
        $(CC) $(CFLAGS) -c $(SRC)42environs.c -o $(OBJ)42environs.o

$(OBJ)42ephem.o     : $(SRC)42ephem.c $(INC)42.h
        $(CC) $(CFLAGS) -c $(SRC)42ephem.c -o $(OBJ)42ephem.o

$(OBJ)42fsw.o       : $(SRC)42fsw.c $(INC)Ac.h $(INC)AcTypes.h
        $(CC) $(CFLAGS) -c $(SRC)42fsw.c -o $(OBJ)42fsw.o
```

```
$(OBJ)42glfwgui.o          : $(SRC)42glfwgui.c $(INC)42.h $(INC)42glfwgui.
    h
        $(CC) $(CFLAGS) -c $(SRC)42glfwgui.c -o $(OBJ)42glfwgui.o

$(OBJ)42GlutGui.o          : $(SRC)42GlutGui.c $(INC)42.h $(INC)42GlutGui
    .h
        $(CC) $(CFLAGS) -c $(SRC)42GlutGui.c -o $(OBJ)42GlutGui.o

$(OBJ)42init.o         : $(SRC)42init.c $(INC)42.h
        $(CC) $(CFLAGS) -c $(SRC)42init.c -o $(OBJ)42init.o

$(OBJ)42ipc.o          : $(SRC)42ipc.c $(INC)42.h
        $(CC) $(CFLAGS) -c $(SRC)42ipc.c -o $(OBJ)42ipc.o

$(OBJ)42perturb.o      : $(SRC)42perturb.c $(INC)42.h
        $(CC) $(CFLAGS) -c $(SRC)42perturb.c -o $(OBJ)42perturb.o

$(OBJ)42report.o       : $(SRC)42report.c $(INC)42.h
        $(CC) $(CFLAGS) -c $(SRC)42report.c -o $(OBJ)42report.o

$(OBJ)42sensors.o      : $(SRC)42sensors.c $(INC)42.h $(INC)Ac.h $(INC)
    AcTypes.h
        $(CC) $(CFLAGS) -c $(SRC)42sensors.c -o $(OBJ)42sensors.o

$(OBJ)dcmkit.o         : $(KITSRC)dcmkit.c
        $(CC) $(CFLAGS) -c $(KITSRC)dcmkit.c -o $(OBJ)dcmkit.o

$(OBJ)envkit.o         : $(KITSRC)envkit.c
        $(CC) $(CFLAGS) -c $(KITSRC)envkit.c -o $(OBJ)envkit.o

$(OBJ)fswkit.o         : $(KITSRC)fswkit.c
        $(CC) $(CFLAGS) -c $(KITSRC)fswkit.c -o $(OBJ)fswkit.o

$(OBJ)glkit.o          : $(KITSRC)glkit.c $(KITINC)glkit.h
        $(CC) $(CFLAGS) -c $(KITSRC)glkit.c -o $(OBJ)glkit.o

$(OBJ)geomkit.o        : $(KITSRC)geomkit.c $(KITINC)geomkit.h
        $(CC) $(CFLAGS) -c $(KITSRC)geomkit.c -o $(OBJ)geomkit.o

$(OBJ)gmseckit.o       : $(KITSRC)gmseckit.c $(KITINC)gmseckit.h
        $(CC) $(CFLAGS) -c $(KITSRC)gmseckit.c -o $(OBJ)gmseckit.o

$(OBJ)iokit.o          : $(KITSRC)iokit.c
        $(CC) $(CFLAGS) -c $(KITSRC)iokit.c -o $(OBJ)iokit.o

$(OBJ)mathkit.o        : $(KITSRC)mathkit.c
```

```
	$(CC) $(CFLAGS) -c $(KITSRC)mathkit.c -o $(OBJ)mathkit.o

$(OBJ)nrlmsise00kit.o : $(KITSRC)nrlmsise00kit.c
	$(CC) $(CFLAGS) -c $(KITSRC)nrlmsise00kit.c -o $(OBJ)
	    nrlmsise00kit.o

$(OBJ)msis86kit.o : $(KITSRC)msis86kit.c $(KITINC)msis86kit.h
	$(CC) $(CFLAGS) -c $(KITSRC)msis86kit.c -o $(OBJ)msis86kit.o

$(OBJ)orbkit.o : $(KITSRC)orbkit.c
	$(CC) $(CFLAGS) -c $(KITSRC)orbkit.c -o $(OBJ)orbkit.o

$(OBJ)radbeltkit.o : $(KITSRC)radbeltkit.c
	$(CC) $(CFLAGS) -c $(KITSRC)radbeltkit.c -o $(OBJ)radbeltkit.o

$(OBJ)sigkit.o : $(KITSRC)sigkit.c
	$(CC) $(CFLAGS) -c $(KITSRC)sigkit.c -o $(OBJ)sigkit.o

$(OBJ)sphkit.o : $(KITSRC)sphkit.c
	$(CC) $(CFLAGS) -c $(KITSRC)sphkit.c -o $(OBJ)sphkit.o

$(OBJ)timekit.o : $(KITSRC)timekit.c
	$(CC) $(CFLAGS) -c $(KITSRC)timekit.c -o $(OBJ)timekit.o

$(OBJ)42CfdSlosh.o : $(GSFCSRC)42CfdSlosh.c $(INC)42.h
	$(CC) $(CFLAGS) -c $(GSFCSRC)42CfdSlosh.c -o $(OBJ)42CfdSlosh.o

$(OBJ)42fftb.o : $(GSFCSRC)42fftb.c $(INC)42.h
	$(CC) $(CFLAGS) -c $(GSFCSRC)42fftb.c -o $(OBJ)42fftb.o

$(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

$(OBJ)SimWriteToFile.o : $(IPCSRC)SimWriteToFile.c $(INC)42.h $(INC)
    AcTypes.h
	$(CC) $(CFLAGS) -c $(IPCSRC)SimWriteToFile.c -o $(OBJ)
	    SimWriteToFile.o

$(OBJ)SimWriteToGmsec.o : $(IPCSRC)SimWriteToGmsec.c $(INC)42.h $(INC)
    AcTypes.h
	$(CC) $(CFLAGS) -c $(IPCSRC)SimWriteToGmsec.c -o $(OBJ)
	    SimWriteToGmsec.o
```

```
$(OBJ)SimWriteToSocket.o : $(IPCSRC)SimWriteToSocket.c $(INC)42.h $(
    INC)AcTypes.h
        $(CC) $(CFLAGS) -c $(IPCSRC)SimWriteToSocket.c -o $(OBJ)
            SimWriteToSocket.o

$(OBJ)SimReadFromFile.o : $(IPCSRC)SimReadFromFile.c $(INC)42.h $(INC)
    AcTypes.h
        $(CC) $(CFLAGS) -c $(IPCSRC)SimReadFromFile.c -o $(OBJ)
            SimReadFromFile.o

$(OBJ)SimReadFromGmsec.o : $(IPCSRC)SimReadFromGmsec.c $(INC)42.h $(
    INC)AcTypes.h
        $(CC) $(CFLAGS) -c $(IPCSRC)SimReadFromGmsec.c -o $(OBJ)
            SimReadFromGmsec.o

$(OBJ)SimReadFromSocket.o : $(IPCSRC)SimReadFromSocket.c $(INC)42.h $(
    INC)AcTypes.h
        $(CC) $(CFLAGS) -c $(IPCSRC)SimReadFromSocket.c -o $(OBJ)
            SimReadFromSocket.o

$(OBJ)AppWriteToFile.o : $(IPCSRC)AppWriteToFile.c $(INC)42.h $(INC)
    AcTypes.h
        $(CC) $(CFLAGS) -c $(IPCSRC)AppWriteToFile.c -o $(OBJ)
            AppWriteToFile.o

$(OBJ)AppWriteToGmsec.o : $(IPCSRC)AppWriteToGmsec.c $(INC)42.h $(INC)
    AcTypes.h
        $(CC) $(CFLAGS) -c $(IPCSRC)AppWriteToGmsec.c -o $(OBJ)
            AppWriteToGmsec.o

$(OBJ)AppWriteToSocket.o : $(IPCSRC)AppWriteToSocket.c $(INC)42.h $(
    INC)AcTypes.h
        $(CC) $(CFLAGS) -c $(IPCSRC)AppWriteToSocket.c -o $(OBJ)
            AppWriteToSocket.o

$(OBJ)AppReadFromFile.o : $(IPCSRC)AppReadFromFile.c $(INC)42.h $(INC)
    AcTypes.h
        $(CC) $(CFLAGS) -c $(IPCSRC)AppReadFromFile.c -o $(OBJ)
            AppReadFromFile.o

$(OBJ)AppReadFromGmsec.o : $(IPCSRC)AppReadFromGmsec.c $(INC)42.h $(
    INC)AcTypes.h
        $(CC) $(CFLAGS) -c $(IPCSRC)AppReadFromGmsec.c -o $(OBJ)
            AppReadFromGmsec.o
```

```
$(OBJ)AppReadFromSocket.o   : $(IPCSRC)AppReadFromSocket.c $(INC)42.h $(
    INC)AcTypes.h
        $(CC) $(CFLAGS) -c $(IPCSRC)AppReadFromSocket.c -o $(OBJ)
            AppReadFromSocket.o

$(OBJ)42nos3.o          : $(SRC)42nos3.c
        $(CC) $(CFLAGS) -c $(SRC)42nos3.c -o $(OBJ)42nos3.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 $(INOUT)*.42 ./Standalone
            /*.42 ./Demo/*.42 ./Rx/*.42 ./Tx/*.42
endif
```

Appendix J IPC SIMULATION INPUTS

In this section the most important input files for the IPC simulation will be displayed. These inputs include:

- Inp_IPC
- Inp_Sim
- Orb_*
- SC_*1
- SC_*2

J.1 Inp_IPC

```
<<<<<<<<<<<<<<< 42: InterProcess Comm Configuration File
    >>>>>>>>>>>>>>>
3                                   ! Number of Sockets
********************************  IPC 0
    ****************************
TX                                  ! IPC Mode (OFF,TX,RX,TXRX,ACS,
    WRITEFILE,READFILE)
0                                   ! AC.ID for ACS mode
"State01.42"                        ! File name for WRITE or READ
SERVER                              ! Socket Role (SERVER,CLIENT,
    GMSEC_CLIENT)
localhost       10301               ! Server Host Name, Port
TRUE                                ! Allow Blocking (i.e. wait on
    RX)
TRUE                                ! Echo to stdout
3                                   ! Number of TX prefixes
"SC"                                ! Prefix 1
"Orb"                               ! Prefix 2
"World"                             ! Prefix 3
********************************  IPC 1
    ****************************
ACS                                 ! IPC Mode (OFF,TX,RX,TXRX,ACS,
    WRITEFILE,READFILE)
0                                   ! AC.ID for ACS mode
"State00.42"                        ! File name for WRITE or READ
SERVER                              ! Socket Role (SERVER,CLIENT,
    GMSEC_CLIENT)
localhost       10301               !Server Host Name, Port
TRUE                                ! Allow Blocking (i.e. wait on
    RX)
TRUE                                ! Echo to stdout
```

```
1                                                    ! Number of TX prefixes
"SC[0].AC"                                           ! Prefix 0
********************************  IPC 1
    ****************************
ACS                                                  ! IPC Mode (OFF,TX,RX,TXRX,ACS,
    WRITEFILE,READFILE)
1                                                    ! AC.ID for ACS mode
"State01.42"                                         ! File name for WRITE or READ
SERVER                                               ! Socket Role (SERVER,CLIENT,
    GMSEC_CLIENT)
localhost       10301                                !Server Host Name, Port
TRUE                                                 ! Allow Blocking (i.e. wait on
    RX)
TRUE                                                 ! Echo to stdout
1                                                    ! Number of TX prefixes
"SC[1].AC"                                           ! Prefix 0
```

J.2 Inp_Sim

```
<<<<<<<<<<<<<<<<<<  42: The Mostly Harmless Simulator  >>>>>>>>>>>>>>>>>>
************************* Simulation Control
    *************************
FAST                                ! Time Mode (FAST, REAL, or EXTERNAL)
5000.0    0.1                      ! Sim Duration, Step Size [sec]
1.0                                 ! File Output Interval [sec]
FALSE                               ! Graphics Front End?
Inp_Cmd.txt                         ! Command Script File Name
************************* Reference Orbits
    *************************
1                                   ! Number of Reference Orbits
TRUE   Orb_LEO.txt                  ! Input file name for Orb 0
***************************** Spacecraft
    *****************************
2                                   ! Number of Spacecraft
TRUE  0 SC_test.txt                 ! Existence, RefOrb, Input file for SC 0
TRUE  0 SC_test1.txt                ! Existence, RefOrb, Input file for SC 0
***************************** Environment
    *****************************
03 21 2016                          ! Date (UTC) (Month, Day, Year)
12 00 00.00                         ! Time (UTC) (Hr,Min,Sec)
0.0                                 ! Leap Seconds (sec)
USER                                ! F10.7, Ap (USER, NOMINAL or TWOSIGMA
    )
230.0                               ! If USER_DEFINED, enter desired F10.7
     value
100.0                               ! If USER_DEFINED, enter desired AP
    value
IGRF                                ! Magfield (NONE,DIPOLE,IGRF)
8    8                              ! IGRF Degree and Order (<=10)
8    8                              ! Earth Gravity Model N and M (<=18)
2    0                              ! Mars Gravity Model N and M (<=18)
2    0                              ! Luna Gravity Model N and M (<=18)
FALSE   FALSE                       ! Aerodynamic Forces & Torques (
    Shadows)
FALSE                               ! Gravity Gradient Torques
FALSE   FALSE                       ! Solar Pressure Forces & Torques (
    Shadows)
FALSE                               ! Gravity Perturbation Forces
FALSE                               ! Passive Joint Forces & Torques
FALSE                               ! Thruster Plume Forces & Torques
FALSE                               ! RWA Imbalance Forces and Torques
FALSE                               ! Contact Forces and Torques
FALSE                               ! CFD Slosh Forces and Torques
```

```
FALSE                              !  Output Environmental Torques to
    Files
********************* Celestial Bodies of Interest
    *********************
MEAN                               !  Ephem Option (MEAN or DE430)
FALSE                              !  Mercury
FALSE                              !  Venus
TRUE                               !  Earth and Luna
FALSE                              !  Mars and its moons
FALSE                              !  Jupiter and its moons
FALSE                              !  Saturn and its moons
FALSE                              !  Uranus and its moons
FALSE                              !  Neptune and its moons
FALSE                              !  Pluto and its moons
FALSE                              !  Asteroids and Comets
***************** Lagrange Point Systems of Interest ******************
FALSE                              !  Earth-Moon
FALSE                              !  Sun-Earth
FALSE                              !  Sun-Jupiter
************************ Ground Stations ***************************
5                                                  ! Number of Ground
    Stations
TRUE  EARTH  -77.0   37.0   "GSFC"                 ! Exists, World, Lng, Lat,
    Label
TRUE  EARTH  -155.6  19.0   "South Point"          ! Exists, World, Lng, Lat,
    Label
TRUE  EARTH  115.4   -29.0  "Dongara"              ! Exists, World, Lng, Lat,
    Label
TRUE  EARTH  -71.0   -33.0  "Santiago"             ! Exists, World, Lng, Lat,
    Label
TRUE  LUNA   45.0    45.0   "Moon Base Alpha"      ! Exists, World, Lng, Lat,
    Label
```

J.3 Orb_*

```
<<<<<<<<<<<<<<<<<  42: Orbit Description File  >>>>>>>>>>>>>>>>>
Low Earth Orbit                 !  Description
CENTRAL                         !  Orbit Type (ZERO, FLIGHT, CENTRAL,
    THREE_BODY)
:::::::::::::  Use these lines if ZERO          ::::::::::::::::
MINORBODY_2                     !  World
FALSE                           ! Use Polyhedron Gravity
:::::::::::::  Use these lines if FLIGHT        ::::::::::::::::
0                               !  Region Number
FALSE                           ! Use Polyhedron Gravity
:::::::::::::  Use these lines if CENTRAL       ::::::::::::::::
EARTH                           !  Orbit Center
FALSE                           !  Secular Orbit Drift Due to J2
KEP                             !  Use Keplerian elements (KEP) or (RV)
    or FILE
PA                              !  Use Peri/Apoapsis (PA) or min alt/ecc
    (AE)
500.0       500.0               !  Periapsis & Apoapsis Altitude, km
400.0   2.0                     !  Min Altitude (km), Eccentricity
60.0                            !  Inclination (deg)
0.0                             !  Right Ascension of Ascending Node (deg
    )
0.0                             !  Argument of Periapsis (deg)
0.0                             !  True Anomaly (deg)
0.0   0.0   0.0                 !  RV Initial Position (km)
0.0   0.0   0.0                 !  RV Initial Velocity (km/sec)
TRV   "ORB_ID"                  !  TLE or TRV format, Label to find in
    file
"TRV.txt"                       !  File name
:::::::::::::  Use these lines if THREE_BODY      ::::::::::::::::
SUNEARTH                        !  Lagrange system
LAGDOF_MODES                    !  Propagate using LAGDOF_MODES or
    LAGDOF_COWELL or LAGDOF_SPLINE
MODES                           !  Initialize with MODES or XYZ or FILE
L2                              !  Libration point (L1, L2, L3, L4, L5)
800000.0                        !  XY Semi-major axis, km
45.0                            !  Initial XY Phase, deg  (CCW from -Y)
CW                              !  Sense (CW, CCW), viewed from +Z
0.0                             !  Second XY Mode Semi-major Axis, km (L4
    , L5 only)
0.0                             !  Second XY Mode Initial Phase, deg (L4,
     L5 only)
CW                              !  Sense (CW, CCW), viewed from +Z (L4,
    L5 only)
```

```
400000.0                          !  Z Semi-axis, km
60.0                              !  Initial Z Phase, deg
1.05   0.5   0.0                  !  Initial X, Y, Z (Non-dimensional)
0.0    0.0   0.0                  !  Initial Xdot, Ydot, Zdot (Non-
    dimensional)
TRV  "ORB_ID"                     !  TLE, TRV or SPLINE format, Label to
    find in file
"TRV.txt"                         !  File name
****************** Formation Frame Parameters **********************
L                                 !  Formation Frame Fixed in [NL]
0.0   0.0   0.0   123             !  Euler Angles (deg) and Sequence
L                                 !  Formation Origin expressed in [NL]
0.0   0.0   0.0                   !  Formation Origin wrt Ref Orbit (m)
```

J.4 SC_*1

```
<<<<<<<<<<<<<<<<<<  42: Spacecraft Description File   >>>>>>>>>>>>>>>>>>
1-U Cubesat                         !  Description
"Cube 1"                            !  Label
GenScSpriteAlpha.ppm                !  Sprite File Name
CFS_FSW                         !  Flight Software Identifier
0.1                                 !  FSW Sample Time, sec
************************ Orbit Parameters ****************************
ENCKE                               !  Orbit Prop FIXED, EULER_HILL, or ENCKE
CM                                  !  Pos of CM or ORIGIN, wrt F
   0.00000    0.00000    0.00000 ! Pos wrt F
   0.00000    0.00000    0.00000 ! Vel wrt F
************************** Initial Attitude
   ***************************
NAN                                 ! Ang Vel wrt [NL], Att [QA] wrt [NLF]
0.0     0.0     0.0                 ! Ang Vel (deg/sec)
0.0     0.0     0.0     1.0         ! Quaternion
0.0     0.0     0.0     123         ! Angles (deg) & Euler Sequence
**************************  Dynamics Flags
   ***************************
DYN_JOINT                           ! Rotation STEADY, KIN_JOINT, or
   DYN_JOINT
FALSE                               ! Passive Joint Forces and Torques
   Enabled
FALSE                               ! Compute Constraint Forces and Torques
REFPT_CM                            ! Mass Props referenced to REFPT_CM or
   REFPT_JOINT
FALSE                               ! Flex Active
FALSE                               ! Include 2nd Order Flex Terms
2.0                                 ! Drag Coefficient
*********************************************************************

************************ Body Parameters
   *****************************
*********************************************************************

1                                   ! Number of Bodies
==================================== Body 0
   ====================================
1.0                                 ! Mass
2.0   3.0   1.2                   ! Moments of Inertia (kg-m^2)
0.0   0.0   0.0                     ! Products of Inertia (xy,xz,yz)
0.0   0.0   0.0                     ! Location of mass center, m
0.0   0.0   0.0                     ! Constant Embedded Momentum (Nms)
Cubesat_1U.obj                      ! Geometry Input File Name
```

```
NONE                                ! Flex File Name
************************************************************************

************************** Joint Parameters
    **************************
************************************************************************

          (Number of Joints is Number of Bodies minus one)
================================ Joint 0 ================================
0 1                                 ! Inner, outer body indices
1   213   GIMBAL                    ! RotDOF, Seq, GIMBAL or SPHERICAL
0   123                             ! TrnDOF, Seq
FALSE  FALSE  FALSE                 ! RotDOF Locked
FALSE  FALSE  FALSE                 ! TrnDOF Locked
0.0    0.0    0.0                   ! Initial Angles [deg]
0.0    0.0    0.0                   ! Initial Rates, deg/sec
0.0    0.0    0.0                   ! Initial Displacements [m]
0.0    0.0    0.0                   ! Initial Displacement Rates, m/sec
0.0   0.0  0.0  312                 ! Bi to Gi Static Angles [deg] & Seq
0.0   0.0  0.0  312                 ! Go to Bo Static Angles [deg] & Seq
0.0   0.0  0.0                      ! Position wrt inner body origin, m
0.0   0.0  0.0                      ! Position wrt outer body origin, m
0.0   0.0  0.0                      ! Rot Passive Spring Coefficients (Nm/rad
    )
0.0   0.0  0.0                      ! Rot Passive Damping Coefficients (Nms/
    rad)
0.0   0.0  0.0                      ! Trn Passive Spring Coefficients (N/m)
0.0   0.0  0.0                      ! Trn Passive Damping Coefficients (Ns/m)
************************** Wheel Parameters
    **************************
3                                   ! Number of wheels
================================= Wheel 0
    ==================================
0.0                                 ! Initial Momentum, N-m-sec
1.0   0.0   0.0                     ! Wheel Axis Components, [X, Y, Z]
1E3   3E3                         ! Max Torque (N-m), Momentum (N-m-sec)
0.012                               ! Wheel Rotor Inertia, kg-m^2
0.48                                ! Static Imbalance, g-cm
13.7                                ! Dynamic Imbalance, g-cm^2
0                                   ! Flex Node Index
================================= Wheel 1
    ==================================
0.0                                 ! Initial Momentum, N-m-sec
0.0   1.0   0.0                     ! Wheel Axis Components, [X, Y, Z]
1E3   3E3                         ! Max Torque (N-m), Momentum (N-m-sec)
0.012                               ! Wheel Rotor Inertia, kg-m^2
```

```
0.48                                ! Static Imbalance, g-cm
13.7                                ! Dynamic Imbalance, g-cm^2
0                                   ! Flex Node Index
================================  Wheel 2
   ================================
0.0                                 ! Initial Momentum, N-m-sec
0.0    0.0    1.0                   ! Wheel Axis Components, [X, Y, Z]
1E3    3E3                         ! Max Torque (N-m), Momentum (N-m-sec)
0.012                               ! Wheel Rotor Inertia, kg-m^2
0.48                                ! Static Imbalance, g-cm
13.7                                ! Dynamic Imbalance, g-cm^2
0                                   ! Flex Node Index
************************** MTB Parameters
   **************************
3                                       ! Number of MTBs
================================  MTB 0
   ================================
180.0                               ! Saturation (A-m^2)
1.0    0.0    0.0                   ! MTB Axis Components, [X, Y, Z]
0                                   ! Flex Node Index
================================  MTB 1
   ================================
180.0                               ! Saturation (A-m^2)
0.0    1.0    0.0                   ! MTB Axis Components, [X, Y, Z]
0                                   ! Flex Node Index
================================  MTB 2
   ================================
180.0                               ! Saturation (A-m^2)
0.0    0.0    1.0                   ! MTB Axis Components, [X, Y, Z]
0                                   ! Flex Node Index
************************ Thruster Parameters
   ************************
0                                   ! Number of Thrusters
================================  Thr 0
   ================================
 1.0                                ! Thrust Force (N)
0  -1.0   0.0   0.0                 ! Body, Thrust Axis
 1.0   1.0   1.0                   ! Location in B0, m
0                                   ! Flex Node Index
****************************** Gyro
   ********************************
3                                    ! Number of Gyro Axes
================================ Axis 0
   ================================
0.1                                 ! Sample Time,sec
1.0   0.0   0.0                     ! Axis expressed in Body Frame
```

```
1000.0                          ! Max Rate, deg/sec
10.0                            ! Scale Factor Error, ppm
0.01                             ! Quantization, arcsec
0.007                            ! Angle Random Walk (deg/rt-hr)
0.01  1.0                        ! Bias Stability (deg/hr) over timespan
   (hr)
0.01                             ! Angle Noise, arcsec RMS
0.01                             ! Initial Bias (deg/hr)
0                               ! Flex Node Index
================================ Axis 1
   ====================================
0.1                             ! Sample Time,sec
0.0  1.0  0.0                   ! Axis expressed in Body Frame
1000.0                          ! Max Rate, deg/sec
10.0                           ! Scale Factor Error, ppm
0.01                             ! Quantization, arcsec
0.007                            ! Angle Random Walk (deg/rt-hr)
0.01  1.0                        ! Bias Stability (deg/hr) over timespan
   (hr)
0.01                             ! Angle Noise, arcsec RMS
0.01                             ! Initial Bias (deg/hr)
0                               ! Flex Node Index
================================ Axis 2
   ====================================
0.1                             ! Sample Time,sec
0.0  0.0  1.0                   ! Axis expressed in Body Frame
1000.0                          ! Max Rate, deg/sec
10.0                           ! Scale Factor Error, ppm
0.01                             ! Quantization, arcsec
0.007                            ! Angle Random Walk (deg/rt-hr)
0.01  1.0                        ! Bias Stability (deg/hr) over timespan
   (hr)
0.01                             ! Angle Noise, arcsec RMS
0.01                             ! Initial Bias (deg/hr)
0                               ! Flex Node Index
************************** Magnetometer
   ********************************
3                               ! Number of Magnetometer Axes
================================ Axis 0
   ====================================
0.1                             ! Sample Time,sec
1.0  0.0  0.0                   ! Axis expressed in Body Frame
60.0E-6                         ! Saturation, Tesla
0.0                             ! Scale Factor Error, ppm
1.0E-6                          ! Quantization, Tesla
1.0E-6                          ! Noise, Tesla RMS
```

```
0                                   ! Flex Node Index
================================== Axis 1
    ======================================
0.1                                 ! Sample Time,sec
0.0   1.0   0.0                     ! Axis expressed in Body Frame
60.0E-6                             ! Saturation, Tesla
0.0                                 ! Scale Factor Error, ppm
1.0E-6                              ! Quantization, Tesla
1.0E-6                              ! Noise, Tesla RMS
0                                   ! Flex Node Index
================================== Axis 2
    ======================================
0.1                                 ! Sample Time,sec
0.0   0.0   1.0                     ! Axis expressed in Body Frame
60.0E-6                             ! Saturation, Tesla
0.0                                 ! Scale Factor Error, ppm
1.0E-6                              ! Quantization, Tesla
1.0E-6                              ! Noise, Tesla RMS
0                                   ! Flex Node Index
********************** Coarse Sun Sensor
    ******************************
3                                   ! Number of Coarse Sun Sensors
================================== CSS 0
    ======================================
0.1                                 ! Sample Time,sec
0    1.0   0.0   0.0                ! Axis expressed in Body Frame
180.0                               ! Half-cone Angle, deg
1.0                                 ! Scale Factor
0.001                               ! Quantization
0                                   ! Flex Node Index
================================== CSS 1
    ======================================
0.1                                 ! Sample Time,sec
0    0.0   1.0   0.0                ! Axis expressed in Body Frame
180.0                               ! Half-cone Angle, deg
1.0                                 ! Scale Factor
0.001                               ! Quantization
0                                   ! Flex Node Index
================================== CSS 2
    ======================================
0.1                                 ! Sample Time,sec
0    0.0   0.0   1.0                ! Axis expressed in Body Frame
180.0                               ! Half-cone Angle, deg
1.0                                 ! Scale Factor
0.001                               ! Quantization
0                                   ! Flex Node Index
```

```
************************ Fine Sun Sensor
    ******************************
0                                   ! Number of Fine Sun Sensors
================================== FSS 0
    ==================================
0.2                                 ! Sample Time,sec
30.0   20.0   10.0   213            ! Mounting Angles (deg), Seq in Body
32.0    32.0                        ! X, Y FOV Size, deg
0.1                                 ! Noise Equivalent Angle, deg RMS
0.5                                 ! Quantization, deg
0                                   ! Flex Node Index
************************* Star Tracker
    ********************************
3                                   ! Number of Star Trackers
================================== ST 0
    ==================================
0.1                                  ! Sample Time,sec
0.0   0.0   0.0   213               ! Mounting Angles (deg), Seq in Body
8.0    8.0                           ! X, Y FOV Size, deg
0.0   0.0   0.0                     ! Sun, Earth, Moon Exclusion Angles, deg
2.0   2.0   20.0                    ! Noise Equivalent Angle, arcsec RMS
0                                   ! Flex Node Index
================================== ST 0
    ==================================
0.1                                  ! Sample Time,sec
90.0   0.0   0.0   213               ! Mounting Angles (deg), Seq in Body
8.0    8.0                           ! X, Y FOV Size, deg
0.0   0.0   0.0                     ! Sun, Earth, Moon Exclusion Angles, deg
2.0   2.0   20.0                    ! Noise Equivalent Angle, arcsec RMS
0                                   ! Flex Node Index
================================== ST 0
    ==================================
0.1                                  ! Sample Time,sec
0.0   90.0   0.0   213               ! Mounting Angles (deg), Seq in Body
8.0    8.0                           ! X, Y FOV Size, deg
0.0   0.0   0.0                     ! Sun, Earth, Moon Exclusion Angles, deg
2.0   2.0   20.0                    ! Noise Equivalent Angle, arcsec RMS
0                                   ! Flex Node Index
***************************** GPS
    ***********************************
1                                   ! Number of GPS Receivers
================================= GPSR 0
    ==================================
0.25                                ! Sample Time,sec
4.0                                 ! Position Noise, m RMS
0.02                                ! Velocity Noise, m/sec RMS
```

```
20.0E-9                          ! Time Noise, sec RMS
0                                ! Flex Node Index
************************* Accelerometer
    ******************************
0                                ! Number of Accel Axes
================================ Axis 0
    ==================================
0.1                              ! Sample Time,sec
0.5   1.0   1.5                  ! Position in B[0] (m)
1.0   0.0   0.0                  ! Axis expressed in Body Frame
1.0                              ! Max Acceleration (m/s^2)
0.0                              ! Scale Factor Error, ppm
0.05                             ! Quantization, m/s^2
0.0                              ! DV Random Walk (m/s/rt-hr)
0.0 1.0                          ! Bias Stability (m/s^2) over timespan (
    hr)
0.0                              ! DV Noise, m/s
0.5                              ! Initial Bias (m/s^2)
0                                ! Flex Node Index
```

J.5 SC_*2

```
<<<<<<<<<<<<<<<<<<  42: Spacecraft Description File   >>>>>>>>>>>>>>>>>>
1-U Cubesat                         !  Description
"Cube 2"                            !  Label
GenScSpriteAlpha.ppm                !  Sprite File Name
CFS_FSW                         !  Flight Software Identifier
0.1                                 !  FSW Sample Time, sec
************************ Orbit Parameters ****************************
ENCKE                               !  Orbit Prop FIXED, EULER_HILL, or ENCKE
CM                                  !  Pos of CM or ORIGIN, wrt F
   1.00000    1.00000    1.00000 ! Pos wrt F
   0.00000    0.00000    0.00000 ! Vel wrt F
************************** Initial Attitude
   ***************************
NAN                                 ! Ang Vel wrt [NL], Att [QA] wrt [NLF]
0.0     0.0     0.0                 ! Ang Vel (deg/sec)
0.0     0.0     0.0     1.0         ! Quaternion
0.0     0.0     0.0     123         ! Angles (deg) & Euler Sequence
**************************  Dynamics Flags
   ***************************
DYN_JOINT                           ! Rotation STEADY, KIN_JOINT, or
   DYN_JOINT
FALSE                               ! Passive Joint Forces and Torques
   Enabled
FALSE                               ! Compute Constraint Forces and Torques
REFPT_CM                            ! Mass Props referenced to REFPT_CM or
   REFPT_JOINT
FALSE                               ! Flex Active
FALSE                               ! Include 2nd Order Flex Terms
2.0                                 ! Drag Coefficient
*********************************************************************

************************ Body Parameters
   *****************************
*********************************************************************

1                                   ! Number of Bodies
=================================== Body 0
   ===================================
1.0                                 ! Mass
2.0   3.0   1.2                   ! Moments of Inertia (kg-m^2)
0.0   0.0   0.0                     ! Products of Inertia (xy,xz,yz)
0.0   0.0   0.0                     ! Location of mass center, m
0.0   0.0   0.0                     ! Constant Embedded Momentum (Nms)
Cubesat_1U.obj                      ! Geometry Input File Name
```

```
NONE                                ! Flex File Name
!------------------------------------------------------------------------

!------------------------- Joint Parameters
   -------------------------
!------------------------------------------------------------------------

          (Number of Joints is Number of Bodies minus one)
================================ Joint 0 ================================
0 1                                 ! Inner, outer body indices
1   213   GIMBAL                    ! RotDOF, Seq, GIMBAL or SPHERICAL
0   123                             ! TrnDOF, Seq
FALSE  FALSE  FALSE                 ! RotDOF Locked
FALSE  FALSE  FALSE                 ! TrnDOF Locked
0.0    0.0    0.0                   ! Initial Angles [deg]
0.0    0.0    0.0                   ! Initial Rates, deg/sec
0.0    0.0    0.0                   ! Initial Displacements [m]
0.0    0.0    0.0                   ! Initial Displacement Rates, m/sec
0.0   0.0  0.0  312                 ! Bi to Gi Static Angles [deg] & Seq
0.0   0.0  0.0  312                 ! Go to Bo Static Angles [deg] & Seq
0.0   0.0  0.0                      ! Position wrt inner body origin, m
0.0   0.0  0.0                      ! Position wrt outer body origin, m
0.0   0.0  0.0                      ! Rot Passive Spring Coefficients (Nm/rad
   )
0.0   0.0  0.0                      ! Rot Passive Damping Coefficients (Nms/
   rad)
0.0   0.0  0.0                      ! Trn Passive Spring Coefficients (N/m)
0.0   0.0  0.0                      ! Trn Passive Damping Coefficients (Ns/m)
!------------------------- Wheel Parameters
   -------------------------
3                                   ! Number of wheels
================================= Wheel 0
   ==================================
0.0                                 ! Initial Momentum, N-m-sec
1.0   0.0   0.0                     ! Wheel Axis Components, [X, Y, Z]
1E3   3E3                         ! Max Torque (N-m), Momentum (N-m-sec)
0.012                               ! Wheel Rotor Inertia, kg-m^2
0.48                                ! Static Imbalance, g-cm
13.7                                ! Dynamic Imbalance, g-cm^2
0                                   ! Flex Node Index
================================= Wheel 1
   ==================================
0.0                                 ! Initial Momentum, N-m-sec
0.0   1.0   0.0                     ! Wheel Axis Components, [X, Y, Z]
1E3   3E3                         ! Max Torque (N-m), Momentum (N-m-sec)
0.012                               ! Wheel Rotor Inertia, kg-m^2
```

```
0.48                                ! Static Imbalance, g-cm
13.7                                ! Dynamic Imbalance, g-cm^2
0                                   ! Flex Node Index
================================  Wheel 2
   ================================
0.0                                 ! Initial Momentum, N-m-sec
0.0    0.0    1.0                   ! Wheel Axis Components, [X, Y, Z]
1E3    3E3                         ! Max Torque (N-m), Momentum (N-m-sec)
0.012                               ! Wheel Rotor Inertia, kg-m^2
0.48                                ! Static Imbalance, g-cm
13.7                                ! Dynamic Imbalance, g-cm^2
0                                   ! Flex Node Index
************************** MTB Parameters
   **************************
3                                       ! Number of MTBs
================================  MTB 0
   ================================
180.0                               ! Saturation (A-m^2)
1.0    0.0    0.0                   ! MTB Axis Components, [X, Y, Z]
0                                   ! Flex Node Index
================================  MTB 1
   ================================
180.0                               ! Saturation (A-m^2)
0.0    1.0    0.0                   ! MTB Axis Components, [X, Y, Z]
0                                   ! Flex Node Index
================================  MTB 2
   ================================
180.0                               ! Saturation (A-m^2)
0.0    0.0    1.0                   ! MTB Axis Components, [X, Y, Z]
0                                   ! Flex Node Index
************************ Thruster Parameters
   ************************
0                                   ! Number of Thrusters
================================  Thr 0
   ================================
 1.0                                ! Thrust Force (N)
0  -1.0   0.0   0.0                 ! Body, Thrust Axis
 1.0   1.0   1.0                   ! Location in B0, m
0                                   ! Flex Node Index
****************************** Gyro
   ********************************
3                                    ! Number of Gyro Axes
================================ Axis 0
   ================================
0.1                                 ! Sample Time,sec
1.0   0.0   0.0                     ! Axis expressed in Body Frame
```

```
1000.0                          ! Max Rate, deg/sec
10.0                            ! Scale Factor Error, ppm
0.01                             ! Quantization, arcsec
0.007                            ! Angle Random Walk (deg/rt-hr)
0.01  1.0                        ! Bias Stability (deg/hr) over timespan
   (hr)
0.01                             ! Angle Noise, arcsec RMS
0.01                             ! Initial Bias (deg/hr)
0                               ! Flex Node Index
================================ Axis 1
   ====================================
0.1                             ! Sample Time,sec
0.0  1.0  0.0                   ! Axis expressed in Body Frame
1000.0                          ! Max Rate, deg/sec
10.0                           ! Scale Factor Error, ppm
0.01                             ! Quantization, arcsec
0.007                            ! Angle Random Walk (deg/rt-hr)
0.01  1.0                        ! Bias Stability (deg/hr) over timespan
   (hr)
0.01                             ! Angle Noise, arcsec RMS
0.01                             ! Initial Bias (deg/hr)
0                               ! Flex Node Index
================================ Axis 2
   ====================================
0.1                             ! Sample Time,sec
0.0  0.0  1.0                   ! Axis expressed in Body Frame
1000.0                          ! Max Rate, deg/sec
10.0                           ! Scale Factor Error, ppm
0.01                             ! Quantization, arcsec
0.007                            ! Angle Random Walk (deg/rt-hr)
0.01  1.0                        ! Bias Stability (deg/hr) over timespan
   (hr)
0.01                             ! Angle Noise, arcsec RMS
0.01                             ! Initial Bias (deg/hr)
0                               ! Flex Node Index
************************** Magnetometer
   ********************************
3                               ! Number of Magnetometer Axes
================================ Axis 0
   ====================================
0.1                             ! Sample Time,sec
1.0  0.0  0.0                   ! Axis expressed in Body Frame
60.0E-6                         ! Saturation, Tesla
0.0                             ! Scale Factor Error, ppm
1.0E-6                          ! Quantization, Tesla
1.0E-6                          ! Noise, Tesla RMS
```

```
0                                   ! Flex Node Index
================================== Axis 1
    ======================================
0.1                                 ! Sample Time,sec
0.0   1.0   0.0                     ! Axis expressed in Body Frame
60.0E-6                             ! Saturation, Tesla
0.0                                 ! Scale Factor Error, ppm
1.0E-6                              ! Quantization, Tesla
1.0E-6                              ! Noise, Tesla RMS
0                                   ! Flex Node Index
================================== Axis 2
    ======================================
0.1                                 ! Sample Time,sec
0.0   0.0   1.0                     ! Axis expressed in Body Frame
60.0E-6                             ! Saturation, Tesla
0.0                                 ! Scale Factor Error, ppm
1.0E-6                              ! Quantization, Tesla
1.0E-6                              ! Noise, Tesla RMS
0                                   ! Flex Node Index
********************** Coarse Sun Sensor
    ******************************
3                                   ! Number of Coarse Sun Sensors
================================== CSS 0
    ======================================
0.1                                 ! Sample Time,sec
0    1.0   0.0   0.0                ! Axis expressed in Body Frame
180.0                               ! Half-cone Angle, deg
1.0                                 ! Scale Factor
0.001                               ! Quantization
0                                   ! Flex Node Index
================================== CSS 1
    ======================================
0.1                                 ! Sample Time,sec
0    0.0   1.0   0.0                ! Axis expressed in Body Frame
180.0                               ! Half-cone Angle, deg
1.0                                 ! Scale Factor
0.001                               ! Quantization
0                                   ! Flex Node Index
================================== CSS 2
    ======================================
0.1                                 ! Sample Time,sec
0    0.0   0.0   1.0                ! Axis expressed in Body Frame
180.0                               ! Half-cone Angle, deg
1.0                                 ! Scale Factor
0.001                               ! Quantization
0                                   ! Flex Node Index
```

```
************************ Fine Sun Sensor
    ******************************
0                                   ! Number of Fine Sun Sensors
================================== FSS 0
    ==================================
0.2                                 ! Sample Time,sec
30.0   20.0   10.0   213            ! Mounting Angles (deg), Seq in Body
32.0    32.0                        ! X, Y FOV Size, deg
0.1                                 ! Noise Equivalent Angle, deg RMS
0.5                                 ! Quantization, deg
0                                   ! Flex Node Index
************************* Star Tracker
    ********************************
3                                   ! Number of Star Trackers
================================== ST 0
    ==================================
0.1                                 ! Sample Time,sec
0.0   0.0   0.0   213             ! Mounting Angles (deg), Seq in Body
8.0    8.0                          ! X, Y FOV Size, deg
0.0   0.0   0.0                   ! Sun, Earth, Moon Exclusion Angles, deg
2.0   2.0   20.0                    ! Noise Equivalent Angle, arcsec RMS
0                                   ! Flex Node Index
================================== ST 0
    ==================================
0.1                                 ! Sample Time,sec
90.0   0.0   0.0   213            ! Mounting Angles (deg), Seq in Body
8.0    8.0                          ! X, Y FOV Size, deg
0.0   0.0   0.0                   ! Sun, Earth, Moon Exclusion Angles, deg
2.0   2.0   20.0                    ! Noise Equivalent Angle, arcsec RMS
0                                   ! Flex Node Index
================================== ST 0
    ==================================
0.1                                 ! Sample Time,sec
0.0   90.0   0.0   213            ! Mounting Angles (deg), Seq in Body
8.0    8.0                          ! X, Y FOV Size, deg
0.0   0.0   0.0                   ! Sun, Earth, Moon Exclusion Angles, deg
2.0   2.0   20.0                    ! Noise Equivalent Angle, arcsec RMS
0                                   ! Flex Node Index
***************************** GPS
    ************************************
1                                   ! Number of GPS Receivers
================================= GPSR 0
    ==================================
0.25                                ! Sample Time,sec
4.0                                 ! Position Noise, m RMS
0.02                                ! Velocity Noise, m/sec RMS
```

```
20.0E-9                          ! Time Noise, sec RMS
0                                ! Flex Node Index
************************* Accelerometer
    ******************************
0                                ! Number of Accel Axes
================================ Axis 0
    ==================================
0.1                              ! Sample Time,sec
0.5   1.0   1.5                  ! Position in B[0] (m)
1.0   0.0   0.0                  ! Axis expressed in Body Frame
1.0                              ! Max Acceleration (m/s^2)
0.0                              ! Scale Factor Error, ppm
0.05                             ! Quantization, m/s^2
0.0                              ! DV Random Walk (m/s/rt-hr)
0.0 1.0                          ! Bias Stability (m/s^2) over timespan (
    hr)
0.0                              ! DV Noise, m/s
0.5                              ! Initial Bias (m/s^2)
0                                ! Flex Node Index
```