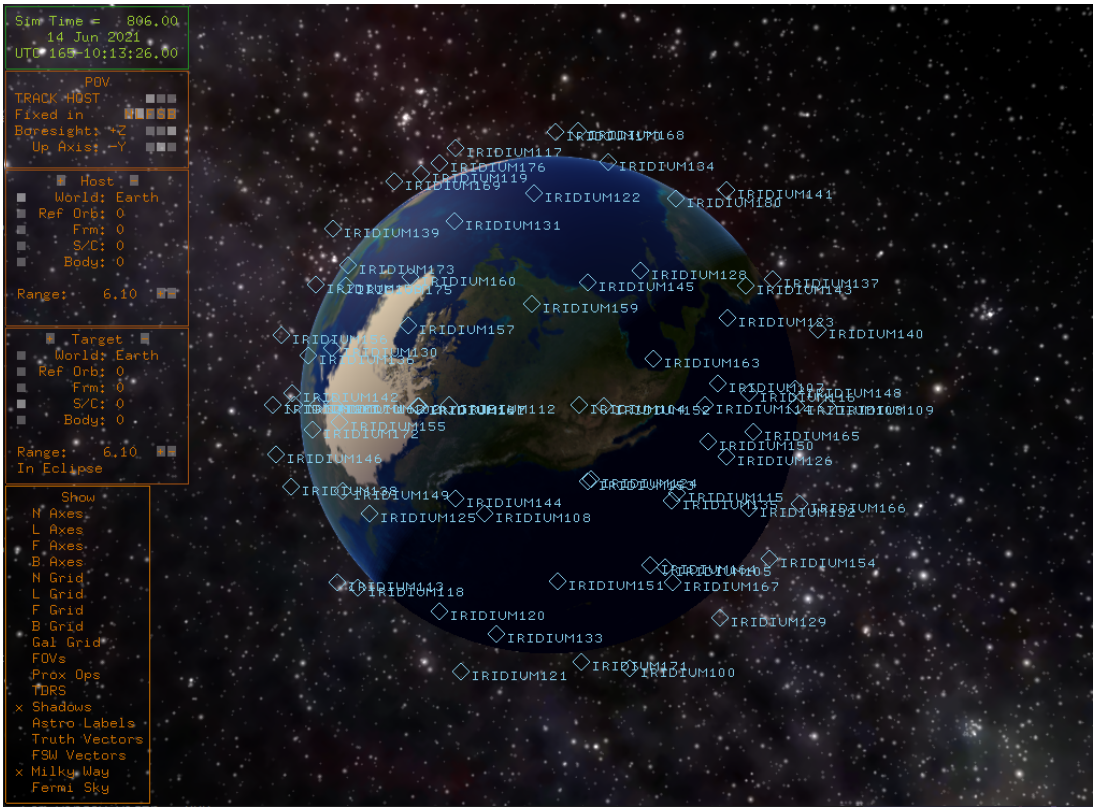
UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Superior d'Enginyeries Industrial,
Aeroespacial i Audiovisual de Terrassa

INTRODUCTION TO CUBESATS

INTRODUCTION TO NASA 42



Author: IVÁN SERMANOUKIAN MOLINA

Professor: DAVID GONZÁLEZ DIEZ
Professor: GORKA UNZUETA GARCÍA

Universitat Politècnica de Catalunya

2021-2022 Q1

Contents

1 Introduction

1.1 NASA 42

NASA's open-source software named 42 is a comprehensive, general-purpose simulation of attitude and trajectory dynamics and control which can be applied to numerous spacecraft composed of multiple rigid or flexible bodies [1].

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

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

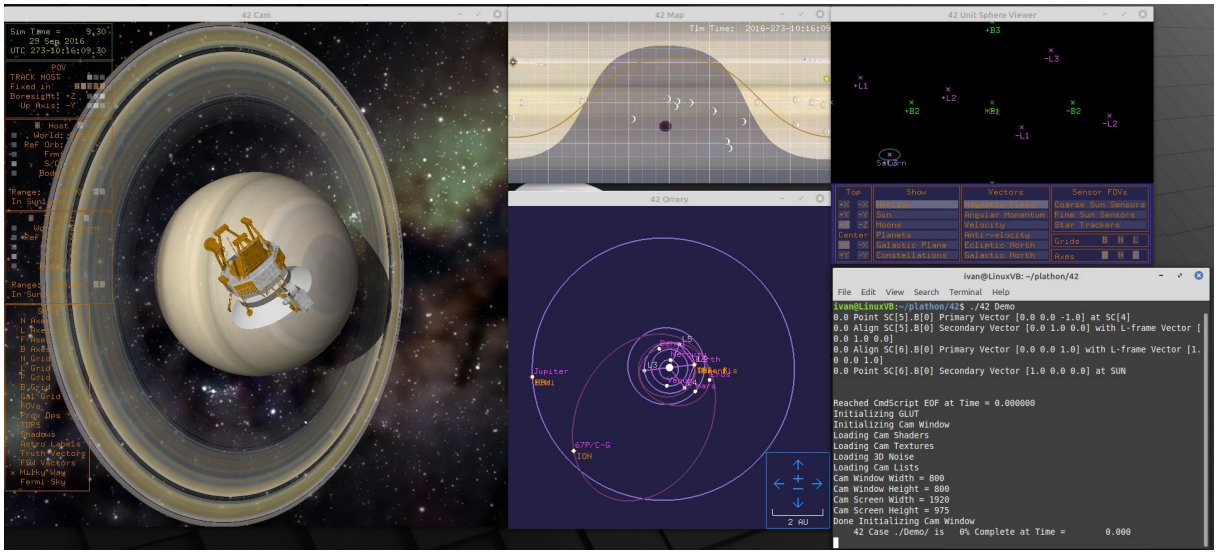


Figure 1 42 complete demonstration with GUI. Source: [2]

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

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

Table 1 42 GUI window description

1.2 Installation and testing

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

In order to install 42 [1], the following steps ought to be taken:

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

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

3. The libraries of OpenGL and other tools have to be installed into the system by executing the following commands as superuser:
   - Enter sudo apt-get update
   - Enter sudo apt-get install freeglut3
   - Enter sudo apt-get install freeglut3-dev
   - Enter sudo apt-get install binutils-gold
   - Enter sudo apt-get install g++ cmake
   - Enter sudo apt-get install libglew-dev
   - Enter sudo apt-get install mesa-common-dev
   - Enter sudo apt-get install build-essential
   - Enter sudo apt-get install libglew1.5-dev libglm-dev
4. Access the directory where 42 is installed and execute the command: make

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

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

If this method has been set up once, future directory creation will only need to clone the designated repository to a new location.

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

1.2.1 Validation

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

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

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

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

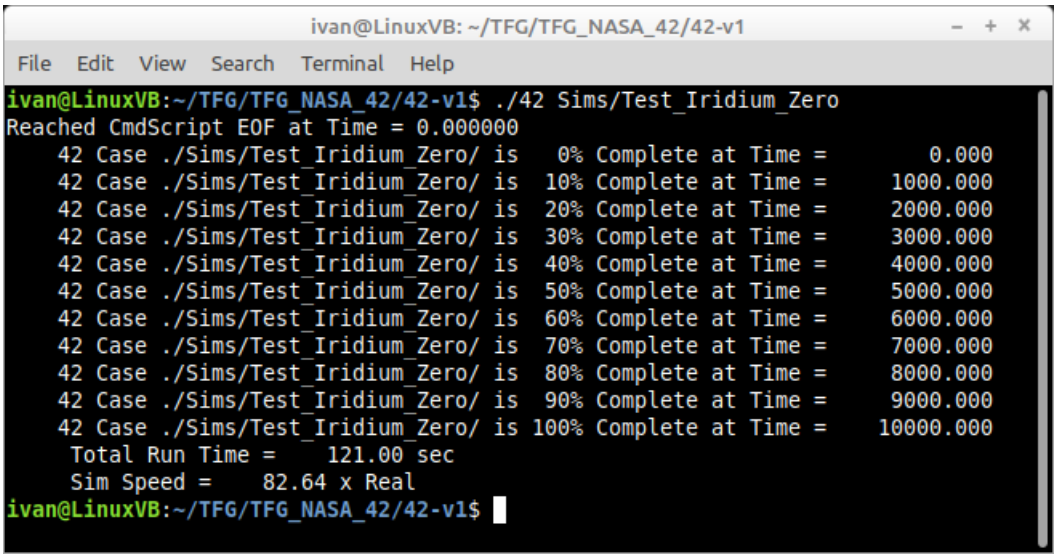


Figure 2 Linux terminal running example programme without GUI. Source: [2]

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

[2] If the software is installed in a VM and the GUI is not displayed correctly, disabling 3D Acceleration might fix it.

1.3 Input files

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

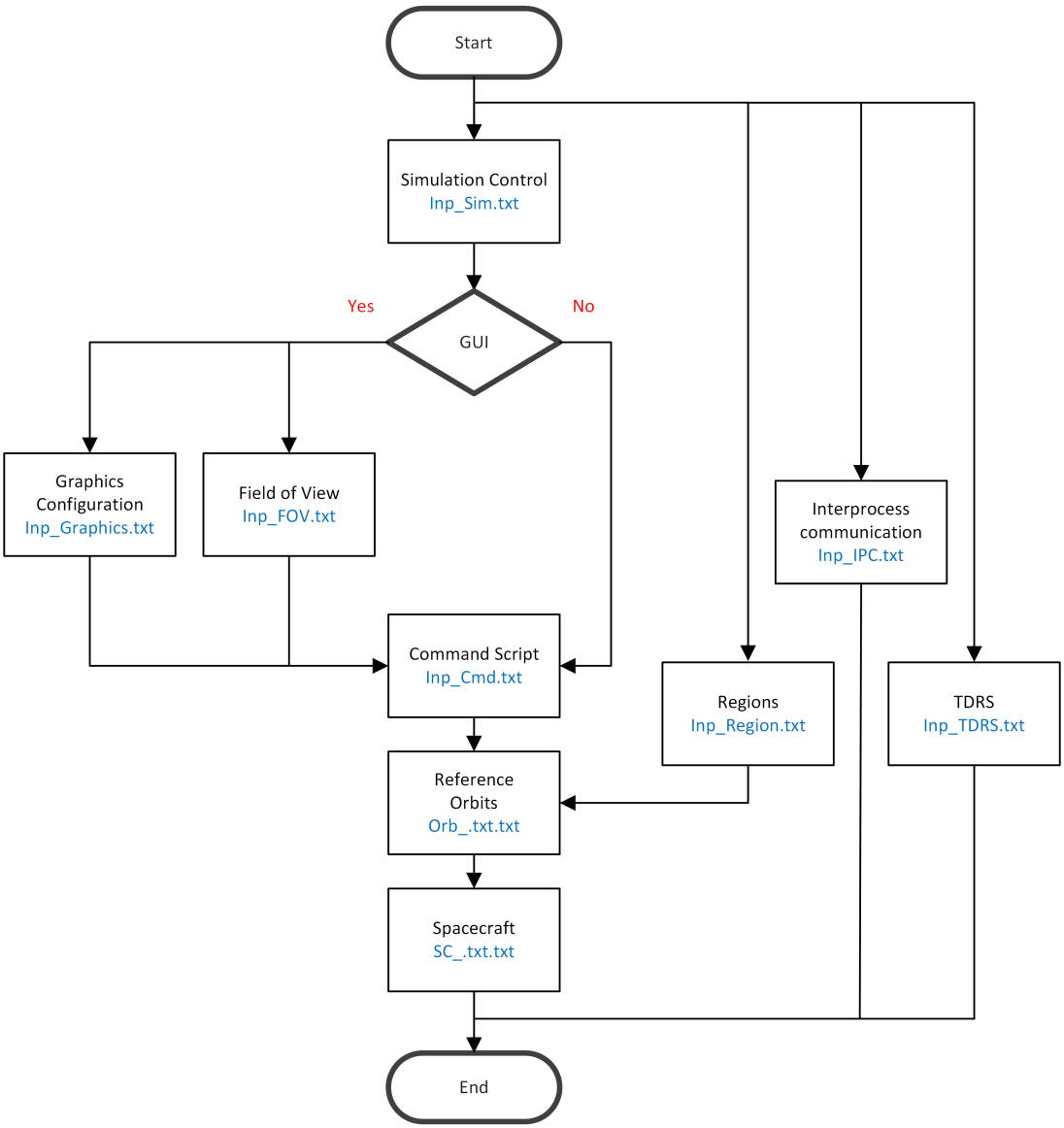


Figure 3 Input files flowchart with dependencies. Source: [2]

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

1.4 Output files

Once the simulation has reached an end, 42 saves certain variables in text files with the .42 extension for further analysis. The raw programme only outputs certain data for all spacecraft and all the available data about the first one. In order to obtain all data for every spacecraft, some changes are introduced in Exercise 3.

For the entire simulation, two time variables are obtained:

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

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

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

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

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

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

2 Guided Example

Following, an example is shown to introduce the reader to the programme capabilities:

The required files to perform a simulation are the documents that can be found on Figure 4. Independently of the GUI usage, the graphics configuration and field of view documents will not be modified.

More information about each file can be found on [2] between pages 34 and 50.

1. Create a folder that will contain both input and output files inside the 42 folder.
2. Copy the necessary files to the folder to run the programme without GUI. Make sure that only one body is set when using the unaltered source code. (Hint: Look at the flowchart from Fig. 3 and the options inside the Simulation Control file)
3. Open the Linux terminal and locate 42's directory.
4. Once inside, run the following command, where Sims/Example is the folder directory created on step 1.

```
./42 Sims/Example
```

5. Once the programme starts, the terminal ought to be similar to Fig. 2. (Warning: with the source version of the code only 1 body will save all the simulation data into data files)
6. Verify that .42 files have been created and open them to see their structure.
7. Subsequently, create another folder with the initial input files used on step 2 and add the ones needed to activate the GUI.
8. Run the programme as explained on step 4 and you should be able to visualise all the windows shown on Fig. 1.

2.1 Post-processing with MATLAB

1. In order analyse the obtained files, a Matlab code is given to read all the data and create the necessary variables. It can be downloaded from the Introduction to CubeSats repository.
2. Once downloaded, change the folder of the operating system section to adapt to your case study.
3. Run the programme and plot the results.

3 Exercise 1

This exercise introduces the reader into the 42 environment choosing a satellite of interest and performing the steps written in the guided example.

For this first exercise you can search information about a current satellite or create your own. You will have to change the parameters of every input file!

3.1 42 visualisation

Start 42 visualising 1 satellite in orbit, for instance OPS-SAT, and move the camera so that both the satellite and Earth are visible.

Save both the orbital parameters and the spacecraft values and take some screenshots during the process.

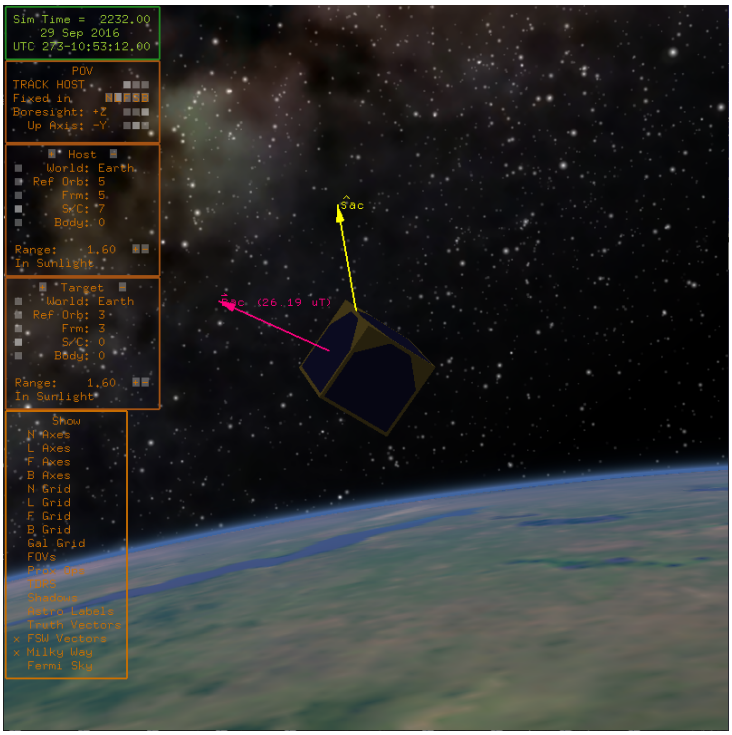


Figure 4 CubeSat orbiting Earth

3.2 Gpredict

- **Compare the results obtained with Gpredict.**
- **Mark the position of TGS (Terrassa Ground Station).**

The Gpredict software is a real-time satellite tracking and orbit prediction application. It can track a large number of satellites and display their position and other data in lists, tables, maps, and polar plots (radar view). Gpredict can also predict the time of future passes for a satellite, and provide you with detailed information about each pass.

4 Exercise 2

This exercise introduces the reader to formation flight and satellite pointing.

The mission is made of two satellites flying in an undefined formation. While one has to point to Earth at all times, the other works as a space telescope. The mission objective is to point at three different Solar System planets with the telescope and then point it to the CubeSat to transmit the gathered data.

Compared to the previous example, two files have to be created for both the spacecrafts and one for the orbit. The templates that come with the source programme can be used for spacecraft definition.

Requirements:

- Communication relay: 1U CubeSat
- Telescope: 3U CubeSat

5 Exercise 3 - Constellation analysis

5.1 File creation automation

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

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

Algorithm 1 Input_Programme.m

```
Choose simulation folder
Select Windows or Linux path
Input initial data
Input orbit function
Input spacecraft function
Input ground stations
Select Solar System variables
Input command script
if GUI activated then
    Input graphics function
    Input FOV function
end if
Input Tracking and Data Relay Satellite System function
Input Region function
Input InterProcess Comunication function
Input Simulation function
if Linux OS then
    Select simulation folder
    Run 42
end if
```

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

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

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

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

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

```
%% Input data

% Input Command Script

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

% Input Simulation Control

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

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

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

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

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

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

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

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

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

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

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

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

The following algorithm represents the structure of the code:

Algorithm 2 Output_Programme.m

```
Choose data folder
Choose simulation folder
Select number of spacecraft
Read time variables
for The first spacecraft to the last do
    Read spacecraft variables
    Save variables on satellite matrix
end for
```

It is important to outline that the reading process is performed in ASCII format.

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

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

```
ulimit -n
```

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

```
ulimit -n 20000
```

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

5.2 Report modification

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

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

Algorithm 3 Report modifications

```
FILE definitions
Memory allocation for each FILE
for Every Spacecraft do
    if Spacecraft exists then
        for Every variable do
            Create a FILE with write permission
        end for
    end if
end for

if Output writing is enabled then
    Save Time data
    for Every Spacecraft do
        if Spacecraft exists then
            for Every variable do
                Save data
            end for
        end if
    end for
end if
```

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

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

File creation for new variables

Once this file is changed, the programme needs to be compiled again.

5.3 Introduction to TLE

This exercise introduces the reader into constellation simulation.

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

The following algorithm explains the TLE download method:

Algorithm 4 TLE_to_42.m

```
Select TLE data link
Read data from TLE
Save data into a matrix by lines
Create TLE document file for 42
Write TLE data into document
Extract satellite ID (delete blank spaces)
```

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

Algorithm 5 Input_SC.m

```
if Option 1 then
    Get satellite IDs
    Select spacecraft template
    Read template data and save into a matrix
    for Every active spacecraft do
        Change satellite label
        Create spacecraft text document
        Add values to spacecraft matrix
    end for
end if
if Option 2 then
    for Every active spacecraft do
        Assign satellite file
        Add values to spacecraft matrix
    end for
end if
```

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

5.4 IRIDIUM NEXT constellation

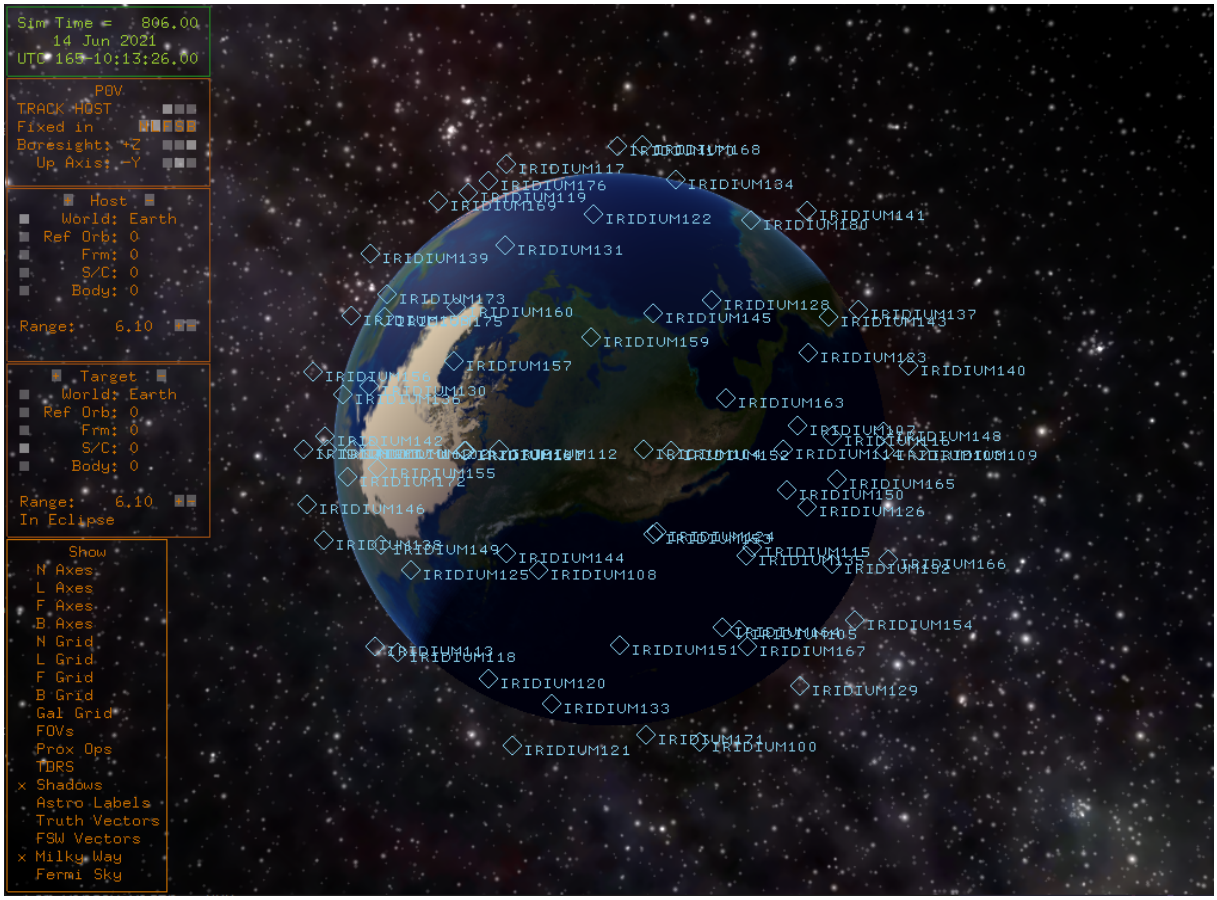


Figure 5 Iridium NEXT simulation (2021/06/14). 3D Camera view.

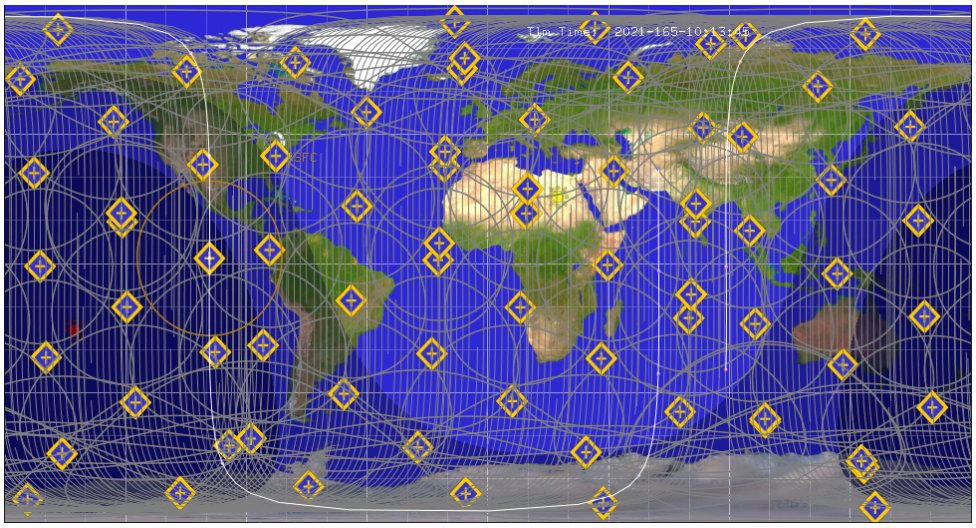


Figure 6 Iridium NEXT simulation (2021/06/14). 2D Map view.

5.5 TLE data format

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

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

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

Table 2 Line 0: Title line of Starlink-1919 TLE

5.5.1 TLE Line 1

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

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

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

Table 3 Line 1: Starlink-1919 TLE

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

Table 4 Line 1: Content description. Data from CelesTrak [3].

5.5.2 TLE Line 2

01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
2		4	6	7	6	9			5	3	.	0	5	6	9		3	4	6	.	0	8	2	4	
1		2					3										4								

27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52
0	0	0	1	6	7	3			8	0	.	0	9	5	7		2	8	0	.	0	2	2	1	
5							6										7								

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

Table 5 Line 2: Starlink-1919 TLE

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

Table 6 Line 2: Content description. Data from CelesTrak [3].

5.6 Exercise guide

- Go to Celestrak and select an active constellation.
- Copy the URL, paste it to the TLE_to_42.m file and modify the name settings.
- Set up the Input_Constellation.m file with the data extracted from the TLEs. If the dates are different, compute the mean value.
- Run 42 with the modified version

References

[1] Goddard Space Flight Center. (n.d.). *42: A comprehensive general-purpose simulation of attitude and trajectory dynamics and control of multiple spacecraft composed of multiple rigid or flexible bodies.* https://software.nasa.gov/software/GSC-16720-1. (Cit. on p. 2)

[2] Sermanoukian Molina, Iván. (2021). Study on orbital propagators. Constellation analysis with NASA 42 and MATLAB/SIMULINK. UPC. (Cit. on pp. 2, 4, 5, 7).

[3] Kelso, T. (2019). *Norad two-line element set format.* https://www.celestrak.com/NORAD/documentation/tle-fmt.php. (Cit. on pp. 16, 17)