NANOSTAR SUITE

Activity Profile Module

Module developed by ienai SPACE

Contents

1. INTRODUCTION

NANOSTAR is a European SUDOE project. One of its goals is to develop a software suite to enable student teams and university staff to perform preliminary analyses and design small satellites.

Nanostar software suite (NSS) needs a visualization module that allows the team to communicate internally, to check and validate mission analysis and scenario and to get pretty communication outputs. . This visualization module shall interact in terms of input/output with other software in the NSS.

The goal of this module is:

- to facilitate satellite mode management (the activity profile of the spacecraft during the mission)
- to ease NSS interaction with VTS, a multi-purpose visualization tool developed by CNES.

2. OVERVIEW.

The activity profile module consists of a user friendly interface (front-end) for manipulating a schematics representation of an activity profile, i.e., a sequence of working modes for a satellite. In particular, the infrastructure allows the user to carry out the following tasks:

- Timeline creation, manipulation
- Drag and drop of satellite modes on the timeline
- Visualize eclipse scheduled on the timeline.
- Navigate through the timeline.

The activity profile package is organized in a standard project folder structure as follows:

- doc/
- README.md
- src/
- test/

The doc/ folder contains the documentation files including additional information regarding the standard protocols referred in this manual . The software is distributed along with an open source license.

The source code can be found in the src/ directory. It contains two main files:

- **MAIN_activity_profile_NOTEBOOK.ipynb**: it is the main file to be run on the jupyterlab application. It can be used for debugging purposes in the local host. **Important!: this file will not run in Jupyter, only in JupiterLab.**

- **MAIN_activity_profile_SERVER.py**: this file can be run via command line. This file is to be running in the server to deploy the application on the web.

The test/ folder contains example files (both inputs and outputs) for interacting with the activity profile module. Further information on the specifications and format for the inputs and outputs will be provided hereafter. Note that all the output files generated by the tool will be automatically saved in the test/output/ folder with a timestamp including the generation epoch.

3. INPUTS.

The following inputs have been defined. Some of them are optional, other are required:

- Satellite position data with the CCSDS standard OEM data in a .txt format [**optional]**. For further information about this protocol, please check the CCSDS document in the doc/requirements folder.

- NSS database project exported as a json file [**required**]. From this file, information regarding:
    - Initial date (days in MJD format)
    - Orbit parameters:
        - Semimajor axis [km]
        - Inclination [deg],
        - Init_cond_on_true_anomaly [deg],
        - Argument of perigee [deg],
        - RAAN [deg],
        - Eccentricity
    - Mission Time: number of orbits where the trajectory is to be propagated and the activity profile is to be generated.
    - Ground stations:
        - Latitude [deg]

- Longitude [deg]
- Altitude [km]
- Elevation [deg]

Important!: all previous parameters has to be defined in the json file. In case they are not defined using the same structure and nomenclature as in example provided, the activity profile application will fail.

- Mode listing data in a csv format file [**optional**]. The first column is the beginning epoch (in MJD format) for the mode expressed in integer days. The second column is the number of seconds passed after the integer day number. The third column corresponds to the name of the mode.

  Important!: The mode name cannot include white spaces. If the name include the word Payload, the mode will be displayed in red color. If it contains the word Downlink, it will be displayed in blue color. If it contains the word Eclipse, it will be displayed in black color. In the default case, the mode will be displayed in yellow color.

  For example: a mode named “Payload”, a mode named “PayloadRadmon”, a mode named “RadmonPayload” or a mode named “Payload1” will be displayed in red. On the other hand, a mode named “Radmon”, will be displayed in yellowe and a mode named “Payload Radmon” will raise an error.

  An example of a user-provided mode-listing csv file is shown in **Figure 1**.

```
55279.0	40799.9999997206	Payload
55315.0	77785.10301222559	Payload
55418.0	31200.000013690442	Downlink
55487.0	69600.00002067536	Eclipse
55557.0	21600.00002766028	Payload
55626.0	60000.0000346452	Payload
55696.0	12000.000003911555	Downlink
55765.0	50400.000010896474	Eclipse
55835.0	2400.0000178813934	Payload
55904.0	40800.00002486631	Payload
```

Figure 1: Example of a user-provided mode-listing csv file

4. OUTPUTS

The following outputs have been defined:

- Activity profile with the CCSDS standard in MEM format.
- Activity profile with the CCSDS standard in CSV format.
- VTS file (.vts) (respecting the VTS input/output format)

5. INSTALLATION

The Activity profile module is programmed using python 3.7 programming language to be run within the JupyterLab computing environment. The interactive visualization is build upon the dash framework. Dash is a productive Python framework for building web applications. Written on top of Flask, Plotly.js, and React.js, Dash is ideal for building data visualization apps with highly custom user interfaces in pure Python. It's particularly suited for anyone who works with data in Python.

The following python packages has to be installed before running the web-based application in Jupyter Lab:

```
$ pip install plotly==4.5.0
$ pip install jupyterlab
$ pip install dash==1.8.0
$ pip install dash-html-components
$ pip install dash-core-components
$ pip install json
$ pip install numpy
$ pip install flask
$ pip install pandas
$ pip install dash_bootstrap_components
$ pip install jupyterlab-dash
$ pip install jupyter_plotly_dash
```

For problems during the the installation or suggestions please send an email to davidmg@ienai.space or saracp@ienai.space for technical support.

6. USAGE

As a first step, open Jupyterlab (NOTE: to open JupyterLab type in terminal *jupyter lab or use anaconda*) . Then navigate to the src/ folder and open the file *MAIN_activity_profile_NOTEBOOK.ipynb* and run it. Thereafter, a new browser tab will be automatically opened as it is illustrated in **Figure 2**. It represents the main interface of the activity profile. There are two main plots:

- In the upper figure the sequence of spacecraft modes is illustrated with respect to the elapsed time from the beginning epoch for a default scenario. The boxes are draggable and resizable, allowing the user to interactively modify the mode sequence. By hovering on the upper side or lower side of each rectangle, the label of each mode will be displayed (e.g., Payload, Eclipse, Downlink)

- In the lower figure, a set of parameters describing the evolution of the semimajor-axis, the inclination, the eccentricity, the cartesian position and velocity, the visibility with respect to the ground station, and the eclipses during the trajectory. The specific parameters to be plotted can be selected with the dropdown menu.

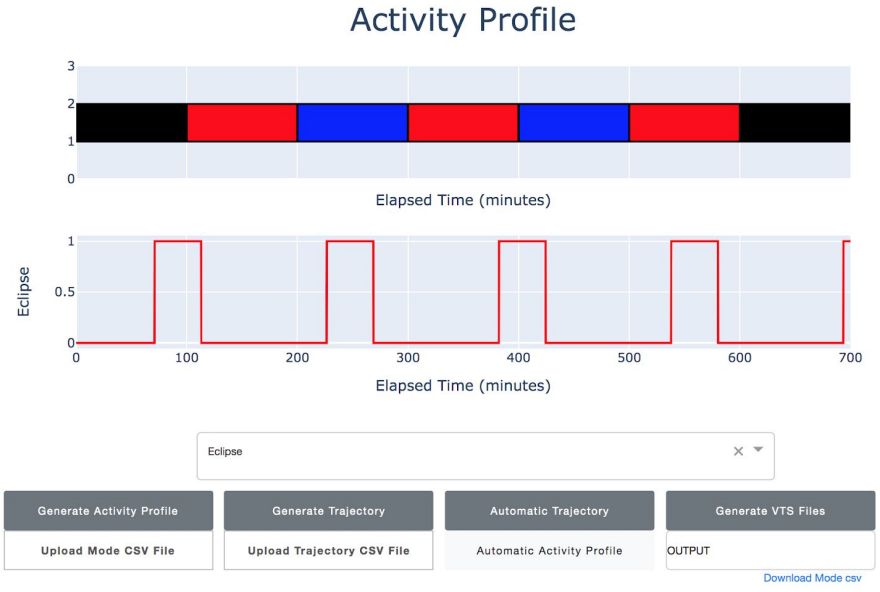

Figure 2: Activity profile default main window.

The are different ways of using the tool:

- In the first case, the user can provide a csv file describing the sequence of modes. The user has to load the csv file by clicking on the button **Upload Mode CSV File.** Then, when clicking on the **Generate Activity Profile**, the new activity will be displayed in the upper figure. In the folder /test/inputs there are two examples files: *mode_sequence_test.csv* that can be used to test this feature. The result for the provided example included in the distribution is depicted in **Figure 3**.

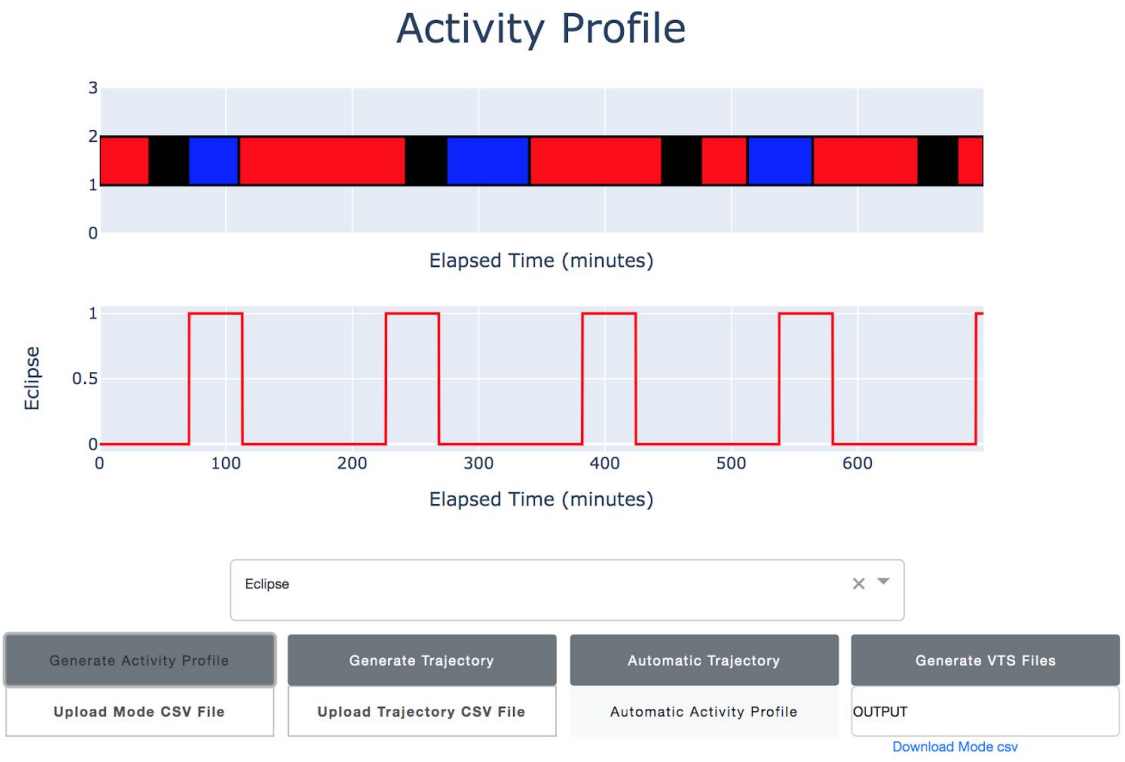

Figure 3: User Provided Activity Profile example

- In the second case, the user can provide a csv file describing the position and velocity of the satellite over time. In order to generate the orbital element plots, the user has to upload the ephemerides file by clicking on **Upload Trajectory CSV File**. then, by pressing the **Generate Trajectory** button, the new eclipse, visibility and orbital elements are plotted in the lower figure. In the folder /test/inputs there are two examples files: *trajectory_test.csv* that can be used to test this feature. The result for the provided example included in the distribution is depicted in **Figure 4**.

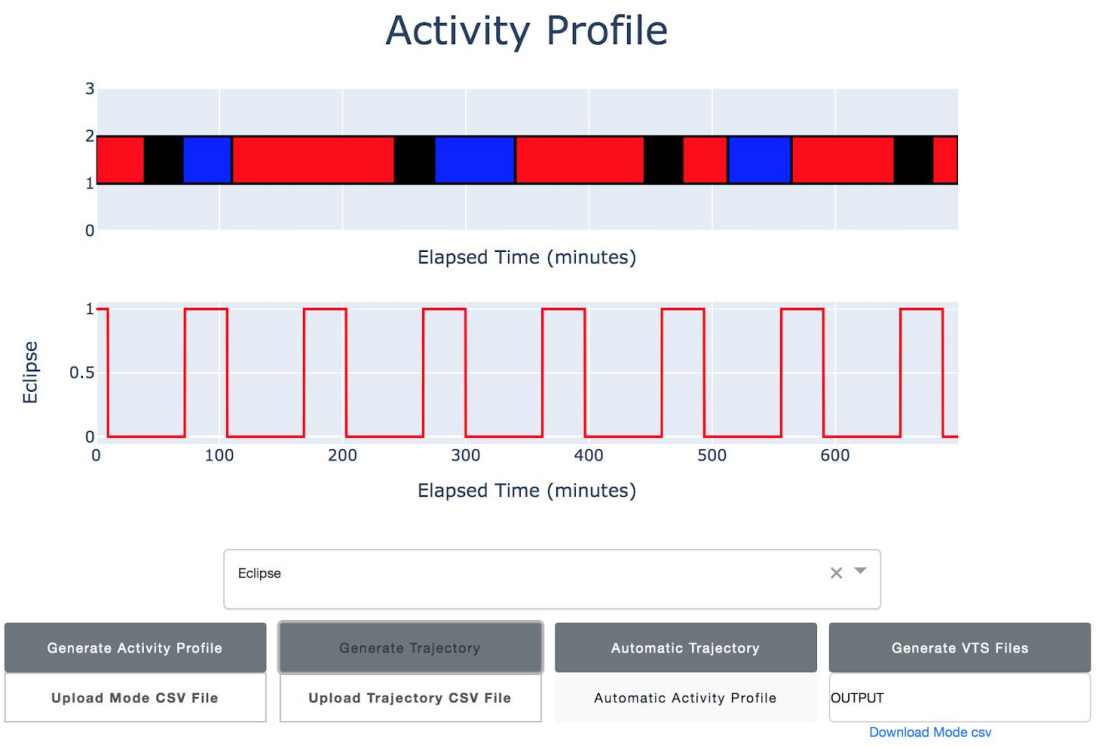

Figure 4: User Provided Trajectory Example

- In the third case, the application is able to automatically generate a spacecraft trajectory. In order to generate this profile, the user has to provide a .json file that includes the orbital parameters, the initial epoch, and the ground station location. By default, the file is located in /test/inputs and is named *Mission_example.json*. This file follows the NSS standard, and can be directly downloaded from the NSS database. Then, the user has to click on the button **Automatic Trajectory** to visualize the result.

- In the fourth case, the application is able to automatically generate a feasible sequence of modes. Eclipse modes happen when in Earth's shadow, downlink modes are adjusted to happen when the satellite is visible from the ground station as long as there is no eclipse. The remaining time is distributed among the available payload modes. In order to generate this profile, the last eclipse and visibility data generated will be used, i.e., the automatic mode egeneration can be used for a user provided trajectory (second case) or for a automatically generated trajectory (third case). The user has to click on the button **Automatic Activity Profile** to visualize the result. The labels for each mode will be displayed while hovering, the different boxed can be dragged and dropped. **Figure 5** exhibits the output obtained by using the default file.

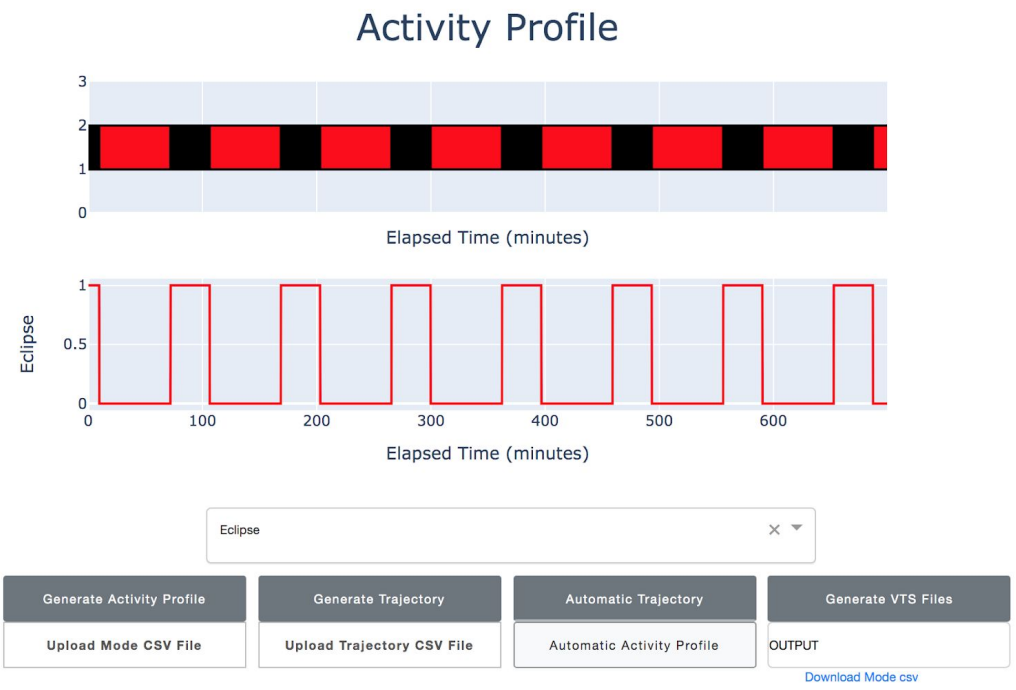

Figure 5: Automatic Activity Profile Example

- In the fifth case the user is able to generate the required VTS Files. They include the new spacecraft activity profile file, and the .vts file that can be loaded in the VTS time loop tool. The user has to type the desired filename in the Input box **Type VTS File names** and click on **Generate VTS Files** to output the files. The generated files are automatically created in the folder test/output.

The application can also be run via the command line prompt by running the file MAIN_activity_profile_SERVER.py. For such purpose, navigate to the src folder, open a terminal and type:

```
$ python MAIN_activity_profile_SERVER.py
```

The web browser will be opened automatically in the port where the dahs application is deployed.

The same operations as in the JupyterLab app can be done. Additionally, in the server application, a download link for retrieving the satellite modes csv files can be found. This can be useful for manually editing the cvs file and later upload it back to the server with the new configuration.

```
CIC_MEM_VERS = 1.0
CREATION_DATE  = 2009-12-08T09:00:00
ORIGINATOR     = SPACEBEL
META_START
COMMENT Cic infos
OBJECT_NAME = CubeSat
OBJECT_ID = CubeSat
USER_DEFINED_PROTOCOL = CIC
USER_DEFINED_CONTENT = SATELLITE_MODES
TIME_SYSTEM = UTC
META_STOP
55279.0 40799.9999997206 Payload
55315.0 77785.10301222559 Payload
55418.0 31200.000013690442 Downlink
55487.0 69600.00002067536 Eclipse
55557.0 21600.00002766028 Payload
55626.0 60000.0000346452 Payload
55696.0 12000.000003911555 Downlink
55765.0 50400.000010896474 Eclipse
55835.0 2400.0000178813934 Payload
55904.0 40800.00002486631 Payload
```