Satellite Attitude Animation and Simulation (SAAS)

Cooper Chang Chien
Taiwan Space Agency (TASA), Systems Engineer

Abstract

This is a documentation for the Satellite Attitude Animation and Simulation (SAAS) program written in MATLAB. This documentation covers the overall structure of the program, including the input file format, different simulation mode, and the algorithm behind the simulation, etc.

Note that this is still an ongoing project with updates to be expected in the future. The source code is managed by git, permission will be required to gain access to the code, either to download or fork.(TODO)

NOTE: Some of the concepts here require extensive literature reading, serve this either as a starting point or collection of knowledge, as I'll do my best to include materials that are helpful during my studies.

Index Terms — **Satellite Attitude, quaternion, propagation**

I OVERALL STRUCTURE

Satellite Attitude Animation and Simulation (*SAAS*) is a tool for preliminary deisgns of satellite on-orbit attitude, with the capability to transform numerical results into visualisation with the corresponding imported CAD model. The following sections provide detail explanations and procedures to execute SAAS.

I-A Code Structure

```
SAAS
+— /src
    +— SAAS.m
    +— READ_INPUT.m
    +— /MAIN
    |   +— init.m
    |   +— ATT_sim.m
    |   +— ATT_file_prop.m
    +— /component
    |   +— create_comp.m
+— /input
    +— INPUT_SIM.txt
    +— INPUT_PROP.txt
+— /model
    +— model_setup.m
+— /output
```

Table I: Code Structure

Before diving right into the instructions, it is essential to grasp an overview of the code structure as shown below in Table I. The code is mainly executed by a single command file called *SAAS.m* (file path: /SAAS/src), while the inputs to the program is dictated by the user input in the *INPUT.txt* files. Specific format and entrices have to be followed in order for the program to initialise the simulation correctly.

The *init.m* (file path: /SAAS/src/MAIN) function initialises the simulation and distributes variable data and model parameter depending on the mode selected, which currently contains two, ***Simulation*** or ***Propagation*** mode. The Simulation mode allows the user to import either single or profiles of quaternion that characterises the rotation, or in more general terms, attitude on-orbit. Users can also select the *DESIGN* mode, to design a profile of quaternion control law that maximises the solar energy input based on a preselected sun beta angle. The Propagation mode allows input of large trending data from actual satellite data to visualise the attitude during service, some complementary files might also be needed for this mode. Details for each mode will be expanded and further discussed in the later sections.

I-B Inputs

To improve user experience and ease the load for constant variable changes in different functions, I have created an generalised input file that can handle this. The two modes each required separate *INPUT.txt* files, to better distinguish and avoid variable overload problem.

The input file is structured in such a way that a module "MOD" is defined first, followed by the variables and data, and ends with an "END". The MODs required in the INPUT.txt are MODE, MODEL, ENV, COMPONENT, OPTION, and VALUE/TREND.

- MODE: SAAS mode
- MODEL: Define model parameter & CAD file location
- ENV: Environment, e.g. sun beta angle
- COMPONENT: Define Star-tracker and desired vector
- OPTION: STR view cone visualisation options

The VALUE and TREND correspond to the modes of simulation, simulation and propagation, respectively. This module mainly contains data, either single-column vector or multiple row assembling series of action. The following section focuses on these two modules in a more thorough explanation with example inputs and resulting terminal outputs.

I-B1 INPUT_SIM.txt

To operate under simulation mode, the required module is VALUE shown below. It requires 5 inputs in general but differ when DESIGN sub-mode is selected. For DESIGN purposes (**DESIGN 1**), it is compulsory that the following parameter must be set and/with appropriate inputs shown in Table II.

- **DESIGN**: Built-in design option (0: N, 1: Y)
- **FRAME**: Designed quaternion frame (ECI or LVLH)
- **QUAT_DESIGN**: [start position, end position, interval]

Context for the parameter in **QUAT_DESIGN** will be discussed in the later section.

```
TITLE INPUT_SIM.txt

MODE   simulation
MODEL
        NAME      FS9
        CG        [2.77e+00   1.82e+01   8.99e+02]
        FILE      'model/fs9_SADA.stl'
END
ENV
        BETA_ANGLE 15
END
COMPONENT
        STR1      [474.918, 479.593, 931.902]
                  [473.641, 478.316, 934.298]
        STR2      [-486.347, 537.236, 332.014]
                  [-443.741, 494.732, 411.878]
        USER      [-0.388, -0.198,-0.01]
                  [0, 0, 0]
END
OPTION
        STR_VIEW     0
END
VALUE
        DESIGN          1
        FRAME           'ECI'
        QUAT_DESIGN     [0, -240, 50]
END
```

Table II: Simulation Input (DESIGN 1)

Putting the INPUT.txt file into plain context, the input file prompts a simulation on FS9 with two pre-defined STR (STR1 & STR2) and a USER defined vector (USER), DESIGN sub-mode has been selected to design a set of quaternion in the BODY2ECI frame, with the design variables shown in the vector.

If the DESIGN parameter has been set to 0 (**DESIGN 0**, i.e. to use an user-input quaternion), the only parameter that needs to be filled out is either **QUAT_SINGLE** or **QUAT_PROF**, and leave the other one as an empty array. Input example for both simulation and propagation are shown in Table III and Table IV respectively.

- **QUAT_SINGLE**: Single quaternion vector, or
- **QUAT_PROF**: File name for the quaternion profile

```
VALUE
        DESIGN          0
        FRAME           ''
        QUAT_DESIGN     []
        QUAT_SINGLE     [0.8580  0  0  -0.5136]
        QUAT_PROF       ''
END
```

Table III: Simulation Input (Simulation, DESIGN 0)

```
VALUE
        DESIGN          0
        FRAME           ''
        QUAT_DESIGN     []
        QUAT_SINGLE     []
        QUAT_PROF       'QUAT_PROFILE.txt'
END
```

Table IV: Simulation Input (Propagation, DESIGN 0)

After setting up the INPUT.txt file, you are set and ready to go. Run the command *SAAS()* in the command window, and you should see the following prompt appearing for sanity checks of the inputs. The program automatically generates profiles of vertices and faces based on the .stl file specified in the input file.

```
>> SAAS

|-MODE:              simulation
|-MODEL:
|-        NAME:      FS9
|-        CG:        [2.77e+00   1.82e+01   8.99e+02]
|-        FILE:      model/fs9_SADA.stl
|-        CAD:
|-                   vert:    [52958x3 double]
|-                   faces:   [110808x3 double]


|-ENV:
|-        BETA_ANGLE:     0.26

|-COMPONENT:
|-        STR1:   [2x3 double]
|-        STR2:   [2x3 double]
|-        USER:   [2x3 double]

|-OPTION:
|-        STR_VIEW:       0

|-VALUE:
|-        DESIGN_opt:     1
|-        FRAME:          ECI
|-        QUAT_DESIGN:    [0  -240  50]
|-        QUAT_SINGLE:    []
|-        QUAT_PROF:      []

Orbit SA energy percentage:      48.34%

Rotation Simulation Completed!
```

Table V: Terminal Simulation Data Display

I-B2 INPUT_PROP.txt

Now we take a look at the propagation mode. The steps are more or less similar to the simulation mode, except the additional required input and slight change in format of the INPUT.txt file. Because of the fact that we are trying to duplicate the attitude and the position of the satellite on-orbit, the amount and size of the data might be considerably large. Conveniently, there is a custom *import()* function in SAAS that is capable of reading data files (.txt prefably) and allocate them as vectors, and this will be handle automatically through providing the data file path in the INPUT.txt file.

The main difference between the input file of simulation and propagation is the VALUE module, or for propagation, the TREND module. The TREND module contains the necessary data for constructing the attitude, orbit position, and sun vector during orbit. The parameters are:

- QUAT
- ECI
- LLA
- ECLIPSE
- SUN

```
TITLE INPUT_PROP.txt

MODE  propagation
MODEL
        NAME     FS9
        CG      [2.77e+00   1.82e+01   8.99e+02]
        FILE    'model/fs9_SADA.stl'
END
ENV
        BETA_ANGLE 15
END
COMPONENT
        STR1     [474.918, 479.593, 931.902]
                 [473.641, 478.316, 934.298]
        STR2     [-486.347, 537.236, 332.014]
                 [-443.741, 494.732, 411.878]
        USER     [-0.388, -0.198,-0.01]
                 [0, 0, 0]
END
OPTION
        STR_VIEW    0
END
TREND
        QUAT         '${FILE_PATH}$.txt'
        ECI          '${FILE_PATH}$.txt'
        LLA          '${FILE_PATH}$.txt'
        ECLIPSE      '${FILE_PATH}$.txt'
        SUN          '${FILE_PATH}$.txt'
END
```

Table VI: Propagation Input Data

```
>> SAAS

|-MODE:            propagation
|-MODEL:
|-        NAME:    FS9
|-        CG:      [2.77e+00   1.82e+01   8.99e+02]
|-        FILE:    model/fs9_SADA.stl
|-        CAD:
|-                 vert:    [52958x3 double]
|-                 faces:   [110808x3 double]


|-ENV:
|-        BETA_ANGLE:      0.261799387799149

|-COMPONENT:
|-        STR1:    [2x3 double]
|-        STR2:    [2x3 double]
|-        USER:    [2x3 double]

|-OPTION:
|-        STR_VIEW:        0

|-TREND:
|-        QUAT:    [89x4 double]
|-        ECI:     [1350x4 double]
|-        LLA:     [1350x2 double]
|-        ECLI:    [1350x1 double]
|-        SUN:     [1350x3 double]
|-        DATE:    [1350x1 datetime]


Trending File Propagation Completed!
```

Table VII: Terminal Simulation Data Display

II THEORY

II-A Quaternions

II-B Energy calculations

III UPDATES/TODO