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 changes in the format of INPUT.txt file. Because of the fact that we duplicating 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 whilst providing the data file path in the INPUT.txt file.

The data input file format is abided with the Ntrend[1] system. Dedicated trending profile on Ntrend has been created under the name of *SAAS_"MODE"*, request the trending profile and download to the */data* folder. Slight modification to the downloaded trending data might be required to remove the following data information and satistical data (shown as below in Table VI), and leave only the data sections.

```
S/C                    : FS7T      Request ID   :
 20241570003     Job ID    : 001
SOH Type               : VC1
Title                  : F7T_SE_AOCS_STATE.tdt
(2024-156-00:00:00 ~ 2024-157-00:00:00)
Current Date           : 2024-157-02:25:01
2024-06-05-02:25:01
Start Time             : 2024-156-00:00:00
Stop Time              : 2024-157-00:00:00
Profile Name           : F7T_SE_AOCS_STATE.tdt
Workstation            : trdfs7t
Database               : trending.trd_triton
DB Table               : trd_fs7t_vc1_long_202406
Sample Rate            : AllSample

Time of Point          :        AOCS_MGR_STATE
-----------------------  ------------------------
Units                  :
Maximum                :              2.000000
Minimum                :              2.000000
Median                 :              2.000000
Arithmetic Mean        :              2.000000
Standard Deviation     :              0.000000
Geometric Mean         :              2.000000
Root Mean Square       :              2.000000
Number of Samples      :                  2700
Variance               :              0.000000
Mode                   :              2.000000
...
```

Table VI: Delete portion of the Ntrend trending data file

The main difference between the input file of simulation and propagation is the TREND module (shown in Table VII). The TREND module contains the necessary data for constructing the attitude, orbit position, and sun vector during orbit. The parameters are:

- **QUAT**: Quaternion data filename
- **ECI**: ECI position data filename
- **LLA**: Geographic coordinate data filename
- **ECLIPSE**: Eclipse status data filename
- **SUN**: Sun vector data filename

[1]TASA data trending system for on-board data

```
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 VII: Propagation Input Data

After setting up the INPUT_PROP.txt file, the program should prompt the following for input data check on the command window.

```
>> 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 VIII: Terminal Simulation Data Display

II THEORY

The following sections will be dicussing on the centre pilar of this program – the theory and calculations.

II-A Quaternions

A quaternion is a four-tuple of real numbers $\{s,\ x,\ y,\ z\}$, a mathematically convenient alternative to the euler angle representation without experiencing gimbal lock. The four tuple consists of a scalar part (s) and a vector part ($x\mathbf{i} + y\mathbf{j} + z\mathbf{k}$) and has the following form,

$$q = [s, x\mathbf{i} + y\mathbf{j} + z\mathbf{k}] \quad s, x, y, z \in \mathbb{R} \tag{1}$$

One can interpret quaternion physically as a rotation (s rad) about the rotation axis transcribed by the point of the vector part ($[x, y, z]$) connecting to the origin. One of the advantages of a quaternion is the capability of concatenating a series of rotation into one set of quaternion.

II-B Energy calculations (FS9)

One of the design scenario included in this program is to find the optimised attitude by applying a rotation while maintaining nadir pointing, with the objective of maximising solar energy input (change DESIGN module to 1).

We start out by assuming the satellite in the Local Vertical Local Horizontal (LVLH) frame, when both the surfaces of the solar array are pointing in the negative and positive $y-$direction, by assiging the solar array's normal vector as

$$SA_{\perp} = [0, -1, 0]^T \tag{2}$$

To set up the environment, we assume that the orbit has a 10:00 LTDN. When facing towards the descending node, the sun beta angle, β will be 30 deg, i.e., 30 deg to the right. Therefore, we can characterise the sun vector as the following through the value of β,

$$SUN = [\cos(\beta), \sin(\beta), 0]^T \tag{3}$$

To find the optimised yaw angle at difference orbit location, we first introduce a parameter, ϕ, that describes the location of the satellite on the orbit. $\phi = 0$ when the satellite is located above the north pole, and $\phi = -90$ when the satellite is at the equator, etc. Although we are working in the LVLH frame, the process of finding the optimal yaw angle still requires to take into account of the satellite's orientation due to the requirement of always maintaining nadir pointing.

Here we simply construct two rotational matrices (pitch and yaw) to fulfill the gap. Note that becuase of a constrained z-axis, no roll movement was considered here, the pitch rotation matrix is simply presented to characterise different orientation at different orbit position. In some sense similar to transforming rotation in LVLH frame into ECI frame.

$$R_{pitch} = \begin{bmatrix} cos(\phi) & 0 & sin(\phi) \\ 0 & 1 & 0 \\ -sin(\phi) & 0 & cos(\phi) \end{bmatrix} \tag{4}$$

$$R_{yaw} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

To rotate the normal vector of the solar array, simply

$$SA_{\perp,rotated} = R_{pitch}R_{yaw} * SA_{\perp} \tag{6}$$

Note that matrix multiplication are not commutative, so it is necessary to comply with the order.

To find the angle between the SA normal vector and the incoming SUN vector, it is convenient to obtain the result by constrcuting a dot product of the two vectors,

$$\begin{aligned} DP &= \overrightarrow{SA}_{\perp,rotated} \cdot \overrightarrow{SUN} \\ &= \cos(\beta)\cos(\phi)\sin(\theta) - \sin(\beta)\cos(\theta) \end{aligned} \tag{7}$$

Finding the maximum of the product, i.e., when the angle bewteen the SA normal vector and SUN vector are nearly align, we simply take the derivative of the expression wrt to θ and let the expression equal to zero.

$$\frac{d(\mathrm{DP})}{d\theta} = \sin(\beta)\sin(\theta) + \cos(\beta)\cos(\psi)\cos(\theta) \tag{8}$$

$$0 \overset{\text{LET}}{=} \sin(\beta)\sin(\theta) + \cos(\beta)\cos(\psi)\cos(\theta) \tag{9}$$

$$\therefore\ \theta = \tan^{-1}\Big(-\frac{\cos\phi}{\tan\beta}\Big) \tag{10}$$

Once we have the optimal rotation and the angle between the normal vector and the SUN vector, we could then proceed to calculate our energy at each point. We can see that the energy equation is identical to the dot product calculated in Equation 7

$$\mathcal{E} = DP = \cos(\beta)\cos(\phi)\sin(\theta) - \sin(\beta)\cos(\theta) \tag{11}$$

III UPDATES/TODO

This section will include expected updates and add-ons scheduled in the future. Modifications during each release version will be written in detail here or on girhub's README.md file. Note that permission must be applied and granted if one wishes to have access to the gihub repository. Please contact Cooper <cooper@tasa.org.tw>, systems deparment, for more information and access permission.