

An Open Source Satellite Attitude and Orbit Simulator Toolbox for Matlab

Valdemir Carrara¹

¹ Instituto Nacional de Pesquisas Espaciais – INPE, Av. dos Astronautas, 1758, São José dos Campos, SP, 12227-010

Abstract: In recent years computer simulation has evolved to accomplish to newer and better technologies, taking the benefit of fast computers, multi core processors and new visualization tools, not to mention new paradigm in man-machine interfaces. Although interpreted languages, like Matlab, are still slow when compared with compiled ones (Fortran and C, for instance), the number of real-time applications with Matlab has increased over time, due mainly its intrinsic ability to handle matrices and the large amount of libraries, toolboxes and ready-to-use applications for both Matlab and Simulink. These facts suggest that Matlab has to be considered as a promising host for any simulation module, which reinforces the necessity of reliable simulation packages written in this language. Following these guidelines, the PROPAT package of Matlab functions has been developed, to simulate Earth's satellites orbit and attitude. The package includes an analytical orbit propagator, allied to a numerical attitude propagator. Nutation dampers and reaction wheels can be easily added to the rigid body dynamics. Also available are functions to compute the position of the Sun relative to Earth, the geomagnetic field in a given satellite position, the Greenwich sidereal time and a function to check if the satellite is in the Earth shadow. Several functions to compute coordinate transformation and ephemeris calculation were included into the package, both for orbital elements (state vector, Keplerian elements or geocentric coordinates) and attitude (Euler angles, quaternion, cosine directors matrix, and Euler's axis and angle), among others. The PROPAT package has being employed at INPE (National Institute for Space Research, in Brazil) for attitude control simulation and orbit and attitude analysis. This paper presents the description of the PROPAT conception, as well as some simulations performed with this package, including the design of the attitude control system of the CubeSat NBR-2 and the CONASAT constellation satellites.

Keywords: attitude control, orbit propagation, attitude simulation

INTRODUCTION

In 1979 the Brazilian Government has approved funding for the design, development, integration and operation of 4 satellites and an in-house launcher. The first 2 satellites would relay data collected from automated on-ground platforms and the following two would carry a remote sensing payload. That was the beginning of the MECB, the Complete Brazilian Space Mission. The development of the satellites was in charge of the National Institute for Space Research (INPE), today's part of the Ministry of Science, Technology and Innovation (MCTI), while the launcher development was in charge of the DCTA under the Ministry of Aeronautics. In its original conception, the first satellite should be launched in 1984. Since that time, 35 years has passed and only the two data collecting satellites have been launched, both by a U.S. rocket, since the national launcher (VLS - Satellite Launcher Vehicle) is still not operational. Remote sensing missions had continuity with the China-Brazil agreement for the development of CBERS (China Brazil Earth Remote Sensing) with 2 successful launches.

For the satellite design, INPE has invested, since that time, in postgraduate courses (MSC and PhD) in space sciences for human resources capability, that were gradually incorporated into the engineering group. These courses have evolved and branched, and still today they are contributing to form experts in the space sciences. One of these courses became what is today the Division of Space Mechanics and Control (DMC), which aims to act in the areas of space structures, thermal control, and dynamics and control of satellite orbit and attitude. In the early 80's, the Orbit Dynamics Group under DMC has developed and implemented a set of computational tools to support the launch, early and operational phases of the Brazilian satellites. The software tools were coded in Fortran, initially running on a Burroughs computer and on Digital Vax computers just after 1985. Currently these programs are running on PC, and they retain many of its original features, since they have been fully tested and qualified for space operations. Among the implemented tools there are programs to estimate the satellite orbital parameters from measurements of azimuth, elevation, range and range-rate, carried out by the INPE's Cuiabá ground station. There was also software for attitude determination and estimation based on the magnetometer and solar sensor measurements sent by the satellite telemetry. Based on the estimated attitude, the attitude control software propagates the attitude and verifies if an attitude maneuver is needed. It also computes the onboard magnetic coil polarity necessary to perform the maneuver. Finally, the satellite contact prediction software uses the estimated orbital elements to forecast the location (direction) and time of the upcoming satellite passages over the mission ground stations. In addition to these programs, several others were coded with specific functions to support the launching and operation phases of the satellites. They all still use a set of basic tools to perform coordinate transformations, orbital ephemeris calculation, corrections of coordinate systems, time and

date transformations, in addition to functions to locate the Sun and Moon, besides computer models of the Earth's magnetic field and atmospheric density in orbital altitudes (Kuga et al., 1981; Lopes et al., 1983; Kuga et al., 1983).

In 2006 the codes of the basic tools were converted to C by this author to support the development of an attitude simulator (Carrara, 2011), which shall be used to qualify the onboard attitude control software of the next generation of satellites in development at INPE. This simulator has several features to simulate physically the attitude kinematics and dynamics with high precision and in real-time. However, because it was coded in the C language, and since it is very complex due to the requirements of fidelity of the space environment, it has being little used for scholar works, in spite of its availability to the students. One of the possible reasons for that is the lack of knowledge of the C language by the students. In fact, there is clearly a gradual changing in the engineering schools, which have migrated from teaching Fortran until the end of the last century, to C during the first decade of this century, and from the second decade on there is a strong appeal to Matlab. Even considering the inherent advantages of this application, such as facility to perform operations with matrices, graphical tools, etc., it is important to note that Matlab is essentially an interpreted language, which can present timing issues when required for real-time operations. Even though programs made in Matlab can be converted to and compiled in C, they present performance degradation when compared with programs originally coded in C. However, for all other applications whose processing time is not critical, as, for example, for academic usage, Matlab is a powerful tool for development and prototyping. This fact has led to the development of Matlab tools for orbit and attitude simulation of artificial satellites. Following these guidelines, some of the tools described previously, and who served to support INPE's space missions, were converted manually from the C language to Matlab, thus creating a set of functions to simulate, with relative fidelity, both the orbit as the attitude of artificial satellites.

The following sections will describe this set of tools (Toolbox), called PROPAT (Attitude Propagator), along with some usage examples. Although they were developed for academic purposes, the library can also be used as to support the mission analysis phase, planning and preliminary design of space missions. Certainly this package will suffer improvements and new features in the future, to make it more versatile and far-reaching. The current version is available for free use in Carrara (2014).

PROPAT TOOLBOX

The PROPAT functions can be grouped in functions to perform coordinate transforms, time and ephemeris conversion, space environment properties, orbit propagation, attitude transforms and attitude propagation. The functions are described in the next sections.

Coordinate Systems

The orbital motion of an Earth satellite is described by means of differential equations expressed in an inertial coordinate system. Non-inertial systems are also employed to locate the satellite with respect to Earth, or to check mutual visibility between the satellite and a given ground station, for instance. The major systems used in the PROPAT toolbox are the inertial geocentric system, the geocentric terrestrial system, and the orbital system. The first two systems are shown in Fig. 1(a), and their origin is located at the Earth's center of mass. The geocentric inertial system has its inertial axis z_i coincident with the Earth's rotation axis and the x_i axis points towards the vernal equinox, which is defined by the intersection of the Equator plane with the Ecliptic plane. The geocentric terrestrial frame has its z_t axis coincident with the z_i axis, and its x_t axis lies on the Equator, over the Greenwich meridian. The angle between x_i and x_t is known as Greenwich sidereal time θ_G and is function of the UTC (Coordinated Universal Time), also known as Greenwich Mean Time.

The orbital system has its origin in the Earth's center, like the previous systems, with the x_o axis pointing to the satellite, as seen in Fig. 1(b). The z_o axis is perpendicular to the orbital plane of the satellite, with same direction of orbital angular momentum, and the y_o axis lies in the orbital plane. The orbital systems is related to the inertial frame by the angular orbital elements: the right ascension of the ascending node, Ω , the orbit inclination i , and the node anomaly given by the sum of the perigee argument, ω , and the true anomaly f . This last angle is computed by solving the Kepler equation

$$M = u - e \text{sen} u, \quad (1)$$

for the eccentric anomaly u , given the mean anomaly M and the orbit eccentricity e . Then, the true anomaly can be obtained from (Kuga et al., 2012):

$$\tan(f / 2) = \sqrt{\frac{1+e}{1-e}} \tan(u / 2). \quad (2)$$

Finally, the local horizontal system, or NED (North-East-Down), has origin at the satellite, with its z_{ned} axis pointing towards the Earth center (Down). The x_{ned} axis lies in the local horizontal plane and points to the geographic North pole. The y_{ned} axis is also contained in the horizontal plane and points to the East.

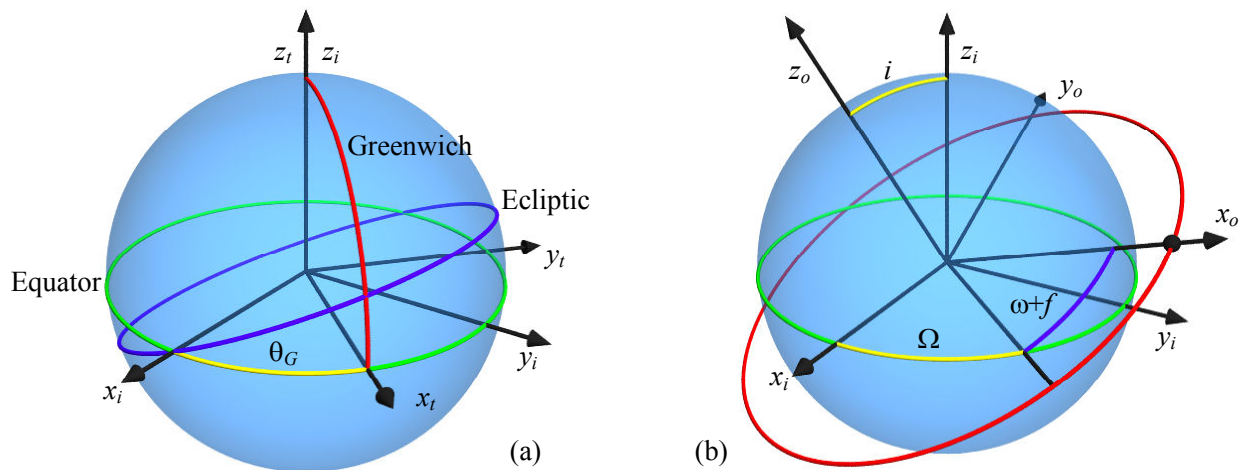


Figure 1 – (a) Geocentric inertial (x_i, y_i, z_i) and terrestrial (x_t, y_t, z_t) frames. (b) Orbital frame (x_o, y_o, z_o).

Time and Date Conversions

Both the Gregorian calendar and the time standard are not based on the decimal system, even considering that they are written in decimal numbers. As a result, 10 days after January 25 is not January 35, but instead February 4th. The same happens with the time counting, since 8 hours after 20 PM is 6 AM instead of 26 hours. Just to make days counting a little bit more friendly, it is customary to employ the Julian Date, which is a sequential count of days from year 4713 BC on. Also, to make computation easy, since the number of days elapsed from that date is around 2.5 million, it is used alternatively the Modified Julian Date (MJD), which starts on January, 1st, 1950. The function `djm` computes the MJD corresponding to the input parameters of the day, month and year. The `djm_inv` function performs the inverse conversion, and so it calculates the day, month and year corresponding to the MJD provided as input parameter. The calculated date is returned in the form of a vector whose components are the day, the month and the corresponding year. Therefore, to propagate the date on PROPAT library this code in Matlab can be used:

```
mjdi = djm(13, 4, 2014);           % day, month and year
for d = 1:365
    mj = mjdi + d;                 % propagated MJD
    data = djm_inv(mj);           % corresponding day, month and year of MJD
end
```

Although the Julian Date can contain fractions of a day, it is usual to adopt the fraction of the day measured in seconds to improve the accuracy of calculations, starting from midnight, 0:00:00 h, until 86400, as shown in this example:

```
t = 0;
i = 0;
mjdi = djm(13, 4, 2014);         % day, month and year
while i < 10
    teta = gst(mjdi, t);         % Greenwich sidereal time
    t = t + 100;
    if (t >= 86400)
        t = 0;                  % new day
        mjdi = mjdi + 1;
        i = i + 1;
    end
    data = djm_inv(mjdi);       % corresponding day, month and year of MJD
    hms = dayf_to_time(t);     % hour, minute and second of the day
end
```

Two functions have been created for converting time units: `dayf_to_time` transforms the fraction of the day measured in seconds to hours, minutes and seconds of the day, and `time_to_dayf` performs the inverse transform. In the latter, the hours, minutes and seconds of the day return as a vector with 3 components. It should be stated that the time should be provided in UTC.

Ephemeris and Space Environment

Satellites usually employ sensors to determine, in orbit, its location in space and its orientation (or attitude). These sensors, obviously, rely on several properties present in the space environment to perform their measurements, such as the direction of the Earth, Sun, Moon, stars and the geomagnetic field, among others. It is important to simulate these sensors, therefore, that computer models must be available for each of these phenomena. The PROPAT library has 3 models in the current version: the position of the Sun in the inertial geocentric system, the Earth orientation at a given

time (Greenwich sidereal time), and the Earth's magnetic field. Additionally, there is also a function to check if the satellite is in the shadow of the Earth.

The `sun` function calculates the position of the Sun in inertial rectangular or spherical coordinates as function of the MJD and time of the day in seconds. The `gst` function gets the Greenwich sidereal time, θ_G , shown in Figure 1 and described in the previous section. The input parameters are also the Modified Julian Date and the time in seconds.

The model adopted for the Earth's magnetic field is the IGRF11, defined by the IAGA (International Association of Geomagnetism and Aeronomy) working group. This model uses a spherical harmonics expansion of the field in terrestrial geocentric coordinates. The coefficients of the expansion depend on the epoch and are set every five years or less, and interpolated linearly in the intervals. The coefficients are adjusted from measures of the geomagnetic field strength collected by several magnetic observatories scattered around the world, by ships, aircraft and satellites. The IGRF11 model was originally coded in Fortran by Susan Macmillan (IGRF, 2013), later converted into C language and Matlab by this author (Carrara, 2011). The series expansion coefficients are stored in the `igrf11.dat` file so it is easy to update them without having to modify the source code. Two functions are available to compute the Earth's magnetic field: `igrffield` and `magfield`. The `igrffield` function takes as arguments the year in the Gregorian calendar plus the fraction of the year (for example, 2014.2385), the geocentric distance in miles, the co-latitude (measured from the Earth's North Pole) and the terrestrial longitude. This function returns the vector of the geomagnetic field in nT (nano Tesla), in the local horizontal coordinate system (NED). The `magfield` function needs the MJD and the terrestrial geocentric position of the satellite in meters as input parameters. On returning the output magnetic field of `magfield` is given in the NED system and in Tesla units.

The `earthshadow` function checks whether the satellite is in the Earth's shadow or sunlit. Its input parameters are the position of the satellite, \mathbf{p}_i (explained in next section), and the position of the Sun, both referred to the inertial system, in meters. If the satellite is sunlit, the `earthshadow` function returns with a unit value, or a null value if it is in the umbra. The function can return values between 0 and 1, if the satellite is in penumbra. The fraction indicates the proportion of light seen by satellite, which means that this function is continuous at all points.

The following sample code illustrates the use of these functions:

```
dayf = time_to_dayf(8, 35, 44);
sun_pos = sun(mjdi, dayf)
tsid = gst(mjdi, dayf)
sat_t = [6000000; 3000000; 1000000];
magf = mag_field(mjdi, sat_t)
sat_i = [4000000; 5000000; 2000000];
esha = earth_shadow(sat_i, sun_pos)
```

In general, as will be seen below, the satellite position is achieved by orbit propagation along with functions to compute coordinate conversions.

Orbit Propagation

When in orbit, a satellite is subject to a set of forces that gradually modify the orbit. Also, acting on attitude as torques, these forces also contribute to change the satellite orientation, or attitude. The main disturbing effect on low Earth orbits is the Earth's gravitational field, which is uniform only at first approximation. The flattening of the Earth poles causes a torque in orbit, which in turn makes a precession in orbit. Other gravitational effects, including the non-uniformity of the gravitational field along with attractions of the Sun and the Moon, tend to modify the orbital parameters or Keplerian elements, such as the semi-major axis a , eccentricity e , inclination i , argument of perigee ω , and also the mean anomaly M , besides the right ascension of the ascending node Ω . Beyond the gravitational effects, in decreasing magnitude order, are the atmospheric drag, solar radiation pressure, and a myriad of small perturbations of electromagnetic origin. For short time intervals, the effects of atmospheric drag and the radiation pressure can be neglected. Considering only the effects of the Earth's gravitational field with the flattening of the poles, an analytical solution to the orbital motion can be achieved, known as Brower model (Brower and Clemence, 1961). By this model changes will occur only on the right ascension of the node, the argument of perigee and in the mean anomaly. Based on the Keplerian elements \mathbf{k}_o at a given time t_o , the `delkep` function computes the variations $\dot{\Omega}$, $\dot{\omega}$, \dot{e} , \dot{M} , and orbit propagation can be done by

$$\dot{\mathbf{k}} = \Delta\mathbf{K}(\mathbf{k}_o) \quad (3)$$

$$\mathbf{k} = \mathbf{k}_o + \dot{\mathbf{k}}(t - t_o) \quad (4)$$

to obtain the keplerian elements at time t . The $\Delta\mathbf{K}$, or `delkep` function takes as input the Keplerian elements vector $\mathbf{k} = (a \ e \ i \ \Omega \ \omega \ M)^T$ at time t_o , with the semi-major axis of the orbit given in meters and all the angles in degrees, including the mean anomaly.

It is quite useful to know the orbital position of the satellite in Cartesian coordinates. The `kepel_statevec` function converts the Keplerian elements \mathbf{k} to the state vector (position and velocity) $\mathbf{p}_i = (x_i \ y_i \ z_i \ \dot{x}_i \ \dot{y}_i \ \dot{z}_i)$ in inertial coordinates, in units of meters and meters per second (Kuga et al., 2012). Since the inertial system is related to the geocentric terrestrial frame by the Greenwich sidereal time θ_G , hence the state vector can also be expressed in the terrestrial coordinates \mathbf{p}_t by the function `inertial_to_terrestrial`, whose input parameters are θ_G obtained by the `gst` function, and the inertial state vector \mathbf{p}_i . The inverse transformation is calculated by `terrestrial_to_inertial` function, with \mathbf{p}_t and θ_G as input arguments.

Sometimes it is necessary to know the satellite altitude, longitude and latitude at a given point in orbit. The `geocentric_to_sph_geodetic` function computes the spherical geodetic coordinates $\mathbf{p}_{sg} = (\lambda \ \varphi \ h)^T$ from the terrestrial geocentric state vector \mathbf{p}_t (only the position is needed) with both the east longitude λ and latitude φ in radians, and the altitude h in meters. The inverse transformation is computed by the function `sph_geodetic_to_geocentric` whose input parameter is \mathbf{p}_{sg} and the output is \mathbf{p}_t . Also useful are the `rectangular_to_spherical` and the `spherical_to_rectangular` functions that convert, respectively, terrestrial coordinates in spherical coordinates (east longitude, geocentric latitude and geocentric distance) and vice versa. The input and output units are meters and radians.

Unlike previous transformations, the transformation to the orbital system is obtained by a matrix rather than directly on the vector coordinates. The reason is due to the fact that usually one wants to perform the conversion in vectors such as the direction of the Sun or of the geomagnetic field, rather than the satellite coordinates. The `orbital_to_inertial_matrix` thus obtains the rotation matrix \mathbf{C}_{i_o} that transforms a vector known in the orbital system in a vector referred to the inertial system:

$$\mathbf{u}_i = \mathbf{C}_{i_o} \mathbf{u}_o. \quad (5)$$

This function takes as input parameter the Keplerian vector \mathbf{k} . The inverse transformation is performed obviously by the transpose of the \mathbf{C}_{i_o} matrix.

Attitude Transformations

The attitude or orientation of a body in space is defined by angles that relate a body fixed coordinate system with the inertial frame. The attitude may be represented by many different ways, but the most common are the direction cosine matrix, the Euler angles, the Euler angle and axis, and the quaternions. The Euler angles consist of a sequence of three successive rotations about the Cartesian axes, which, starting from the inertial system, goes up to the body fixed frame, as is illustrated in Fig. 2, for a 3-1-3 or z - x - z rotation sequence. There are altogether six sequences that can be performed on different axes, and other 6 sequences in which the first rotation axis is repeated in the third rotation. Although the angles obtained for each one of these sequences are different for the same attitude, the direction cosine matrix, which relates both systems, is the same.

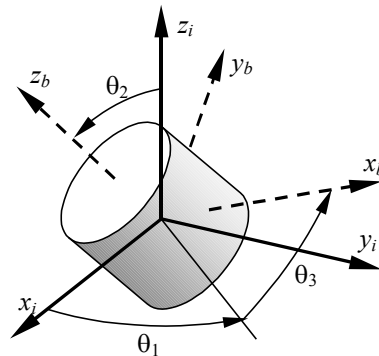


Figure 2 – Euler angles θ_1 , θ_2 and θ_3 of a 3-1-3 rotation sequence.

By the Euler's theorem, whatever is the orientation of a body in space, there will always be a direction and an angle relative to the inertial system such that if the body is rotated around this direction with that angle, starting from the inertial frame, it will reach its final attitude. While it is quite evident the existence of the axis and the rotation angle, they are not easily found by just looking the attitude. On the other hand, quaternion is similar to the Euler axis and angle, and, like that one, it has four components. The advantage of quaternion in relation to other representations of attitude is because they can perform compositions of transformations by simple algebra without any trigonometry. The kinematic equations in quaternion are also more compact and do not require trigonometric calculation.

To uniquely define the attitude only 3 parameters are required, since the attitude has three degrees of freedom. Therefore, the matrix of the director cosines has 6 redundant components, while the quaternion and the Euler axis and

angle present only one redundant parameter. Euler angles are more compact and do not have any redundancy. Whatever representation can be, it can always be converted to any other. However, it is often convenient first to obtain the direction cosine matrix, since there is close relationship between the matrix and the remaining representations. Figure 3 shows the transformations between different attitude representations implemented on PROPAT. In these functions, quaternion is abbreviated by `quat`, direction cosine matrix is `rmx`, Euler axis and angle is `euler`, and finally Euler angles are denoted by `exyz` or `ezxz` for transformations 1-2-3 or 3-1-3, respectively. To transform `r_input` to `r_output` one can just append the tokens to get the function name: `r_inputr_output`, where both `r_input` and `r_output` can be one of the attitude representation tokens: `quat`, `euler`, `rmx`, `exyz`, `ezxz`, `ezyx` and `ezyx`. For instance, to transform the Euler axis and angle to quaternion, the function is `eulerquat`. Not all of the 12 possible combinations of Euler angles have been implemented, but only those most usual. The arrows in Fig. 3 show the transformations included in PROPAT.

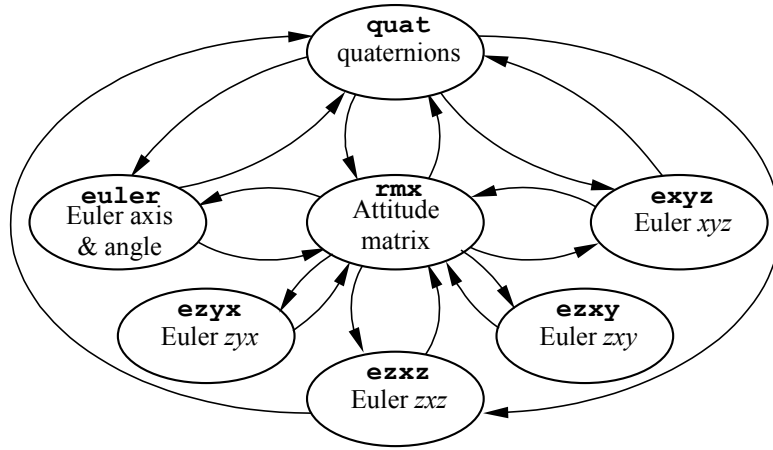


Figure 3 – PROPAT functions to perform attitude transformations.

In addition to these transformations, `rotmax`, `rotmay` and `rotmaz` functions calculate the direction cosine matrix around the Cartesian axes x , y and z , respectively, for a given angle of rotation.

Attitude Kinematics and Dynamics

The kinematic equations allow integration of the attitude motion from a known body initial attitude and angular velocity as a function of time. The kinematics equations express the changes in attitude, and, as there are different representations, there are different expressions for the kinematics. The most widely used of these is the Euler angles, and again there are 12 different sets of differential equations (Wertz, 1978). The great advantage of Euler angles is the lack of redundancy. In contrast, all other representations need to be normalized after the numerical integration, since the integration errors cause them to lose their properties. For quaternion, for example, it should be ensured that its module is unitary whenever necessary. However, even considering the required normalization, the kinematics expressed in quaternion is advantageous because it avoids the trigonometry, as mentioned. The kinematics equation in quaternion is given by (Wertz, 1978; Hughes, 1986):

$$\dot{\mathbf{q}} = \frac{1}{2} \boldsymbol{\Omega} \mathbf{q} \quad (6)$$

where $\mathbf{q} = (\boldsymbol{\varepsilon} \ \eta)^T$ is the quaternion attitude; $\boldsymbol{\varepsilon}$ is the vector part of the quaternion and η is the scalar. The skew-symmetric matrix $\boldsymbol{\Omega}$ is defined by:

$$\boldsymbol{\Omega} \triangleq \begin{pmatrix} -\boldsymbol{\omega}^\times & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{pmatrix} = \begin{pmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{pmatrix} \quad (7)$$

where $\boldsymbol{\omega} = (\omega_1 \ \omega_2 \ \omega_3)^T$ is the angular velocity measured in the body fixed frame.

The dynamic equations come from Newton's law applied to rotating bodies. However, it is highly difficult to obtain the dynamic model in bodies that are non-rigid, flexible, have articulated parts or variable mass distribution. In a first approach the rigid body model may be assumed, and it is really valid for most satellites. The presence of liquids or rotating solar arrays to track the sun, or robotic arms, or even attitude control components like reaction wheels will invalidate the rigid body hypothesis. In few cases the equations of motion can be obtained in particular situations, but

there is no general solution to the equation of motion if the body is composed by several individually rigid parts or the degrees of freedom depend on the construction details. The PROPAT toolbox has distinct dynamic equations, specific for the most common satellite configurations. For a single rigid satellite the Euler dynamics equation result in:

$$\dot{\boldsymbol{\omega}}^b = \mathbf{I}^{-1} (\mathbf{g}_{cm} + \mathbf{g}_{mag} - \boldsymbol{\omega} \times \mathbf{I} \boldsymbol{\omega}). \quad (8)$$

where \mathbf{I} is the inertia matrix with respect to the center of mass and relative to the body fixed coordinates, and \mathbf{g}_{cm} is the total external torque applied to the satellite, except by the magnetic torque. The `rigbody` function obtains the vector changes from the kinematics and dynamic equations, and its parameters are the inertia matrix, its inverse, the torque \mathbf{g}_{cm} , the Earth's magnetic field vector \mathbf{B} in body fixed frame, which, along with the magnetic moment \mathbf{m} from the satellite (arising from the attitude control coils, for instance) is used to calculate the magnetic torque through

$$\mathbf{g}_{mag} = \mathbf{m} \times \mathbf{B}. \quad (9)$$

Any of the Matlab numeric integrator can be used to propagate the attitude. However is strongly recommended to employ a variable step size integrator, such as the ODE45 since they can handle with discontinuities in the torque normally caused by the attitude control algorithm. The attitude state vector is defined by the quaternion and the satellite angular velocity: $\mathbf{x} = (\mathbf{q} \ \boldsymbol{\omega})^T$. The magnetic moment and the Earth's magnetic field strength are optional; if not provided it is assumed that there is no magnetic torque. A simple Matlab code to integrate the attitude motion could be:

```
% Attitude elements in Euler angles of a 3-1-3 (z-x-z) rotation
eulzxx = [30, 50, 20]*pi/180; % converted from degrees to radians
% Attitude in quaternions
quat = ezxzquat(eulzxx); % converted from Euler angles
% Angular velocity vector in body frame:
w_ang = [0.1, 0, 0.5]'; % in radians/sec
% Inertia matrix of a axis-symmetric rigid body:
iner = [8 0 0; 0 8 0; 0 0 12]; % in kg*m*m
% Inverse inertia matrix:
invin = inv(iner);
% ODE solver precision:
options = odeset('abstol', 1e-4, 'reltol', 1e-4);

for t = 0: 0.5: 600
    % External torques (perturbation + control)
    ext_torq = [0 0 0]';
    % Initial attitude vector:
    att_vec = [quat; w_ang]'; % quaternions and angular velocity
    % ODE Solver parameters
    tspan = [t, t+0.25, t+0.5];
    % Numeric integration (ODE45)
    [T, Y] = ode45('rigbody', tspan, att_vec, options, ext_torq, iner, invin);
    att_vec = Y(3, :); % propagated attitude vector
    quat = att_vec(1:4); % propagated quaternion
    w_ang = att_vec(5:7); % propagated angular velocity
    eulzxx = quatezxx(quat); % euler angles
end
```

Satellites that have rotating solar panels or reaction wheels, which are devices that exchange angular momentum with the satellite or when there are moving parts like liquids or nutation dampers (which remove the excess of rotating kinetic energy, causing it to rotate around its greatest moment of inertia) can no longer be regarded as rigid bodies. In all these cases a formulation that takes into account such effects will be needed. As a rule the adopted solutions combine various parts that can be considered individually rigid, in which the conventional formulation can be applied, with forces and torques acting on the links that keep together all these parts. Each rigid part introduces its own degrees of freedom, which increases the order of the motion differential equations. It follows from this that there is no general formulation for the attitude motion, but individual cases can be considered, since much of the satellites adopt similar solutions. In the case of a satellite equipped with reaction wheels for attitude control, each wheel inserts one degree of freedom (the rotation around its axis). Satellites generally have 3 or 4 reaction wheels with rotating axis in any direction but it is always possible to decompose the reaction wheel angular momentum and net torque on the satellite reference frame, so that only the body fixed axes are important from the viewpoint of the motion equations. Therefore the dynamic equations for a satellite the with any number of reaction wheels have six degrees of freedom, expressed by the set of equations (Carrara, 2012; Wertz, 1978; Hughes, 1986):

$$\mathbf{I}_b \dot{\boldsymbol{\omega}}^b = \mathbf{g}_{cm} - \mathbf{g}_w - \boldsymbol{\omega} \times (\mathbf{I}_b \boldsymbol{\omega} + \mathbf{h}_w), \quad (10)$$

$$\dot{\mathbf{h}}_w = \mathbf{g}_w, \quad (11)$$

where \mathbf{h}_w is the total angular momentum stored by reaction wheels, and \mathbf{g}_w is the torque applied to the wheels. \mathbf{I}_b is the inertia of the fixed part of the satellite, that is the total inertia decreased by the rotating inertia of the reaction wheels, given by

$$\mathbf{I}_b \triangleq \left(\mathbf{I} - \sum_{n=1}^N I_{n,s} \mathbf{a}_n \mathbf{a}_n^T \right), \quad (12)$$

in which $I_{n,s}$ is the inertia of the wheel n ($n = 1, \dots, N$) with respect to its rotation axis, and \mathbf{a}_n is the direction (unit vector) of this axis, in the positive direction of the angular velocity. In case there are more than three wheels, or when the direction of their rotating axes does not coincide with the satellite axes, the dynamic equations shall be replaced by:

$$\mathbf{I}_b \dot{\boldsymbol{\omega}}^b = \mathbf{g}_{cm} - \sum_{n=1}^N \mathbf{g}_n \mathbf{a}_n - \boldsymbol{\omega}^\times \left(\mathbf{I}_b \boldsymbol{\omega} + \sum_{n=1}^N h_{w,n} \mathbf{a}_n \right), \quad (13)$$

$$\dot{h}_{w,n} = \mathbf{g}_n, \quad (14)$$

and $h_{w,n}$ and \mathbf{g}_n are respectively the angular momentum (including the moment due to rotation of the satellite itself), and the torque applied to the reaction wheel n . The wheel angular velocity can be calculated by (Carrara, 2012; Hughes, 1986):

$$\boldsymbol{\omega}_n = I_{n,s}^{-1} h_{w,n} - \mathbf{a}_n^T \boldsymbol{\omega}, \quad (15)$$

The equations of motion considering the reaction wheels were implemented in `rb_reaction_wheel` and `rb_reaction_wheel_n` functions, respectively for a set of 3 reaction wheels aligned with the satellite axes, and for a satellite with any number of wheels. The input parameters of `rb_reaction_wheel` are the state vector $\mathbf{x} = (\mathbf{q} \ \boldsymbol{\omega} \ \mathbf{h}_w)^T$, the external torque \mathbf{g}_{cm} , the inertia tensor \mathbf{I}_b , its inverse \mathbf{I}_b^{-1} , and the torque applied to the wheels, \mathbf{g}_w . As for `rb_reaction_wheel_n`, the input are the state vector $\mathbf{x} = (\mathbf{q} \ \boldsymbol{\omega} \ h_{w,1} \ \dots \ h_{w,N})^T$, \mathbf{g}_{cm} , \mathbf{I}_b , \mathbf{I}_b^{-1} , N , \mathbf{g}_n and a matrix $\mathbf{A}_N = (\mathbf{a}_1 \ \dots \ \mathbf{a}_N)$ with the direction of the wheel's rotation axes. The angular velocity of the 3 orthogonal wheels can be obtained by the `rw_speed` function, whose inputs are the satellite angular velocity vector $\boldsymbol{\omega}$, the wheel's stored momentum \mathbf{h}_w and their inertia moment about the rotation axis, $I_{n,s}$, or by the `rw_speed_n` function, which obtains the angular velocity of the n^{th} reaction wheel, given as input $\boldsymbol{\omega}$, $h_{w,n}$, its inertia moment $I_{n,s}$ and the rotation axis unit vector, \mathbf{a}_n .

Spin stabilized satellites can have passive devices to decrease the attitude nutation motion, which occurs whenever the satellite angular momentum does not coincide with the largest principal inertia moment. The excess of the rotating kinetic energy can be removed by a nutation dumper device, which generally consists of a mass-spring-damper set. The dynamic motion equations to integrate the satellite attitude provided with a nutation damper can be found in Carrara (2012) and Hughes (1986), and is based on a rotary inertia, spring and damper. The state vector in this case is given by $\mathbf{x} = (\mathbf{q} \ \boldsymbol{\omega} \ h_{nd} \ \theta_{nd})^T$ where h_{nd} is the angular momentum of the nutation damper and θ_{nd} is the angular position of the rotor. The derivative function for a satellite with a nutation damper in PROPAT toolbox is the `rb_nutation_damper` function, whose input parameters are the state vector, the inertia \mathbf{I}_b , its inverse, the unit vector direction of the rotor rotation axis, \mathbf{a}_{nd} , the rotor inertia, I_{nd} , the spring constant k_{nd} and the viscous damping constant b_{nd} . Except for some of the parameters that change in the differential functions of the toolbox, the sequence to integrate the attitude motion follows the same sequence as the code shown above for the `rigbody` function.

PROPAT TOOLBOX USE EXAMPLES

The PROPAT toolbox is being widely used in the postgraduate course in Space Mechanics and Control of INPE (National Institute for Space Research), with diversified use. 3 usage examples, contained in already published papers will be shown here. The description of the programs that generated these examples will not be discussed here, nor the parameters that were used. The reader is encouraged to see the original works to get additional information.

CubeSat Attitude Control Simulation

This first example of PROPAT shows the attitude control for a CubeSat 1U (a cube of 100 mm), whose attitude sensors are a 3 axis magnetometer and coarse solar sensors allied with a set of 3 orthogonal magnetic torque coils for attitude control. Attitude sensor data are corrupted with noise and bias before being processed by a TRIAD algorithm (Shuster and Oh, 1981) to perform attitude determination, with any kind of filtering. The TRIAD algorithm computes the attitude matrix $\mathbf{C}_{b,orb}$ between the satellite frame and the orbital system, as function of the attitude vectors of the Sun and the Earth's magnetic field as measured by the sensors and compared with these vectors calculated by onboard models of Sun position, geomagnetic field and satellite orbit. When the satellite is in the Earth shadow, there are no valid sun sensor measurements, so there is no attitude determination. In this case attitude control relies on the B-dot

algorithm (Reijneveld and Choukroun, 2013), but only if the satellite angular velocity is greater than 4 revolutions per orbit, otherwise there is no control at all. In the sunshine the attitude is determined and the magnetic moment of the coils is computed to point one of the satellite faces towards the Earth. If the Earth's magnetic strength and the required torque vector are close enough, the control action is nullified because the magnetic torque results too small. The control law is a conventional PD, based on the attitude error and the angular velocity with respect to the orbital system, given by:

$$\mathbf{u} = k_p \theta \mathbf{a} + k_d \boldsymbol{\omega}_{orb}, \quad (16)$$

where θ and \mathbf{a} are the Euler angle and Euler axis computed by the `rmxeuler` function with the matrix $\mathbf{C}_{b,orb}$, and k_p and k_d are the proportional and derivative gains, respectively. The angular velocity $\boldsymbol{\omega}_{orb}$ is calculated by numerical differentiation of the attitude matrix $\mathbf{C}_{b,orb}$, from (Hughes, 1986; Carrara, 2012)

$$\boldsymbol{\omega}_{orb}^\times = -\frac{\Delta \mathbf{C}_{b,orb}}{\Delta t} \mathbf{C}_{b,orb}^T, \quad (17)$$

in which the superscript “ \times ” indicates the cross product matrix (Hughes, 1986). The step time Δt must be carefully chosen. A short time interval makes the angular velocity too noisy, due to the noise in the sensor measurements. A large value, on the other side, can miss the changes in both module and direction of $\boldsymbol{\omega}$, caused by the attitude dynamics. To illustrate this problem, if two consecutive measurements are equal, then $\Delta \mathbf{C}_{b,orb}$ is zero, but that does not mean that $\boldsymbol{\omega}$ is equally zero; instead, it can be $2k\pi/\Delta t$, for k integer. Some empirical results have shown that a good value for Δt is such that the angular motion between two measurements is around 5 to 10°. But, of course, it depends on the signal-to-noise ratio of the sensors, and depends also from a previous estimation of the angular rate. The required magnetic moment for the coils is computed by

$$\mathbf{m} = \frac{\mathbf{B} \times \mathbf{u}}{B}, \quad (18)$$

that gives \mathbf{m} perpendicular to the magnetic field \mathbf{B} . It can be noted that the applied control torque actually differs from the required torque \mathbf{u} , except if \mathbf{u} is orthogonal to \mathbf{B} . This means that the magnetic torque cannot completely control the satellite attitude. However, since the magnetic field changes its direction with respect to the body fixed frame when the satellite travels its orbit, a “mean torque” can be generated in the required direction. The control action is turned off whenever the angle between the magnetic field and the required torque is below 10°, to avoid generating a low control torque.

The attitude acquisition of a 1U CubeSat satellite is shown in Fig. 4, in terms of the angle from an Euler axis and angle attitude. After 6 orbits (10 hours) approximately, the attitude is stabilized and an Earth pointing is achieved, with error below 10°. The simulated orbit was sun-synchronous at 630 km altitude. It was considered an initial angular rate of 0.2 rpm as can be seen in Fig. 5. It can be noted that the inertial angular rate around the z axis (orthogonal to the orbit plane) after stabilization is close to the satellite orbital rate, approximately 0.01 rpm.

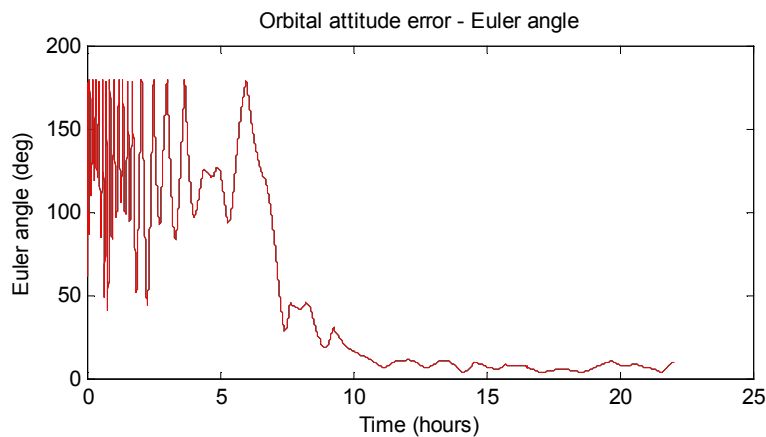


Figure 4 – Geocentric inertial frame (x_i, y_i, z_i) and geocentric terrestrial frame (x_t, y_t, z_t) .

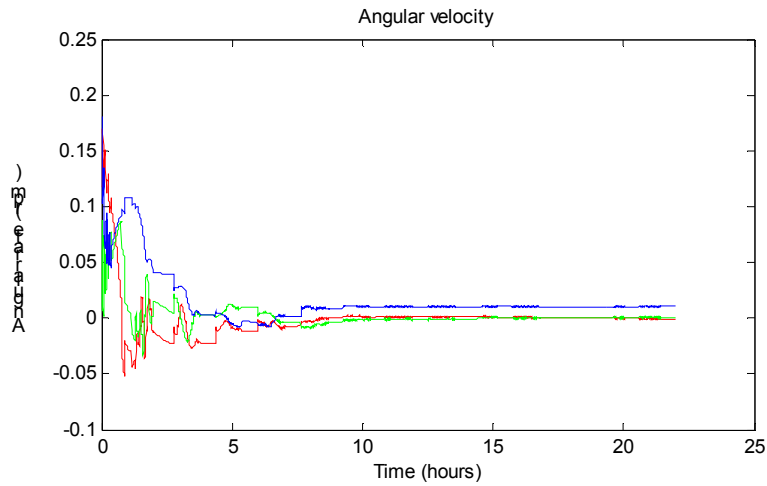


Figure 5 – CubeSat inertial angular rate in Earth pointing attitude.

CONASAT

The Conasat are 8U CubeSat satellites that will replace the Brazilian Data Collecting Satellites SCD1 and SCD2, launched in 1993 and 1998, respectively, and still working at the time of this writing. Each Conasat is composed of two 2U units, main and redundant, both arranged in an 8U cubic structure. The payload shall be a transponder to repeat to the mission center ground station environmental data received from remote autonomous platforms that collect environmental data such as temperature, humidity, pressure, rain volume, river flow, etc. A planar antenna will be used to transmit the data to the ground station. This antenna is directional, which means that a precise attitude pointing will be required. The solution to the pointing requirements was to employ 3-axis attitude stabilization, based on a set of 3 small reaction wheels and 3 magnetic torque coils. Attitude determination will be based on a 3-axis magnetometer and 6 coarse analog sun sensors, fixed on each of the 6 cube faces, and a 3-axis MEMS gyroscope.

The simulation of Conasat attitude control was presented in Carrara et al. (2014), for two operating modes: Earth pointing (nominal) and attitude acquisition together with safe mode. Some results of the nominal mode are presented below. Simulation started with a satellite angular rate of $\omega = (0.6 \ 0.3 \ 0.9)^T$ rpm and initial Euler angles of $(60^\circ \ 30^\circ \ 40^\circ)^T$ of a 1-2-3 rotation sequence. A Kalman filter process the attitude quaternion coming from the attitude determination algorithm, in order to estimate both the quaternion and the gyro biases. A PID controller was implemented based on the attitude error in Euler angles with respect to the orbital system, and the angular rate was given by the simulated gyro measurements, corrected by the estimated gyro biases. Figure 6 shows the Euler angles of the Conasat attitude simulation just after nominal mode entrance. The reaction wheels absorb the residual satellite angular momentum and correct the attitude after just 5 minutes, approximately. The wheel's angular speeds are shown in Fig. 7, for a time span of 5 hours. It was considered in the simulation a disturbance torque caused by some residual magnetic moment of the satellite. Due to the satellite attitude motion in the orbit, the pitch (x) and roll (y) wheels exchange their stored momentum, while the z reaction wheel absorb most of the magnetic disturbance torque.

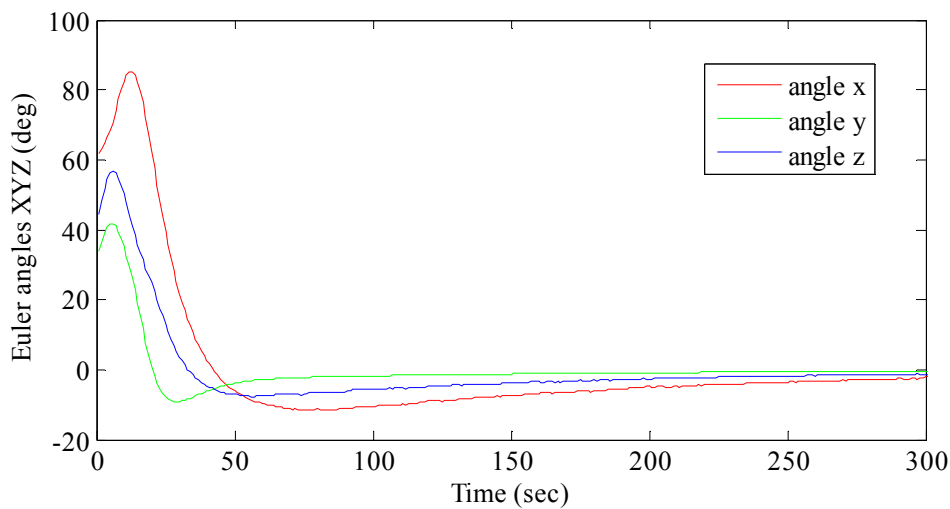


Figure 6 – Conasat attitude in nominal mode coming from acquisition mode. Source: Carrara et al. (2014).

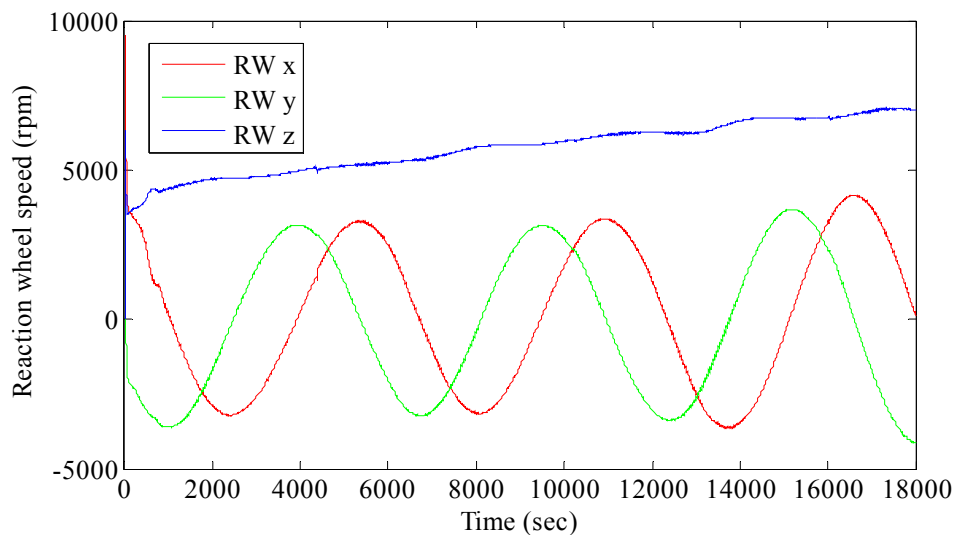


Figure 7 – Reaction wheel speed of CONASAT in nominal mode. Source: Carrara et al. (2014)

Boomerang Motion

This example shows a modification in the PROPAT rotational motion to include also the translational kinematics and dynamic equations of a boomerang. The forces and torques are due to the gravitational weight \mathbf{W} , aerodynamic lift \mathbf{L} and drag. The trajectory of a boomerang is almost circular, because its angular momentum precesses around the vertical caused by the asymmetry of the aerodynamic lift torque. In fact, as the boomerang travels its trajectory, the experienced lift force \mathbf{L} is greater in the wing whose angular velocity has the same sense of the translational motion. The lift force resultant is not coincident with the boomerang center of mass, and so it causes a torque opposite to the velocity vector. The variation of the angular momentum follows the torque, causing the boomerang nose to climb, and to describe a circular path, since it is thrown with its wings slight from the vertical, as shown in Fig. 8(a). The lift force lays almost on the horizontal plane and keeps the boomerang from falling, while, at the same time, gives the centripetal force to the circular trajectory. The initial conditions that influences the trajectory are the translational velocity vector \mathbf{v} , the angular rate ω , the boomerang inclination i , and the shooting angle α . Figure 8(b) shows some trajectories for several initial conditions. It can be noted that the trajectory is not always closed, but exhibits the characteristic circular path.

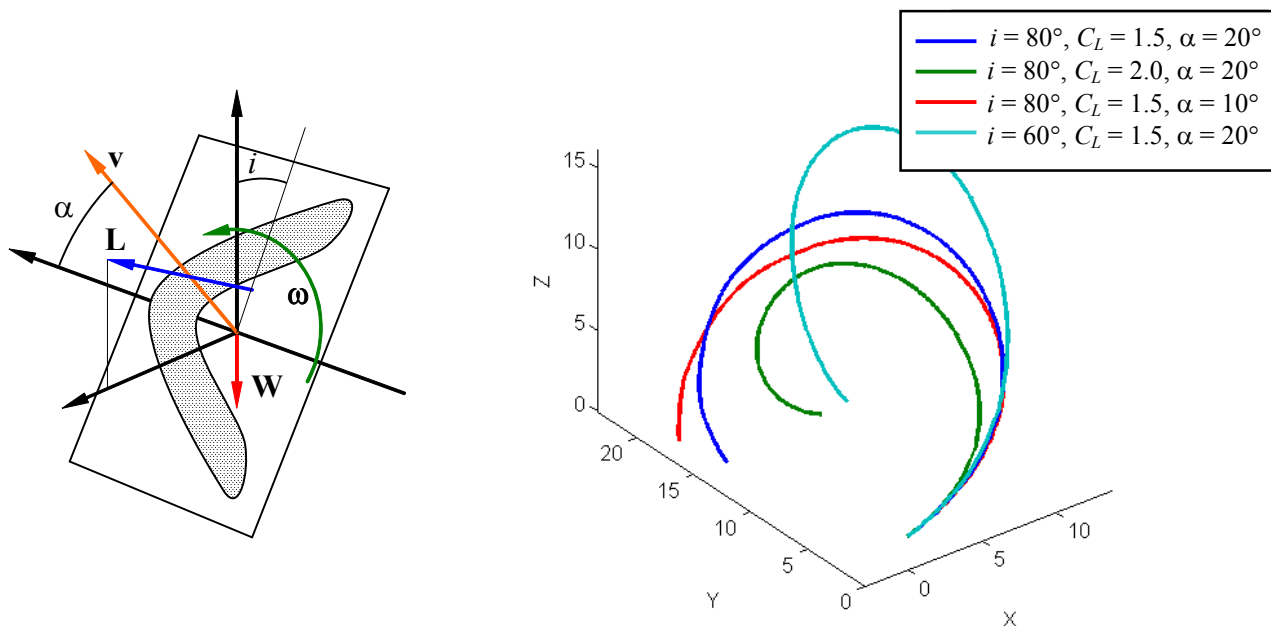


Figure 8 – Boomerang parameters (a) and trajectories for different inclination, lift coefficient and shooting angle. Source: Bigot et al. (2013).

CONCLUSIONS

This paper presented a Matlab toolbox to simulate the attitude and orbit of artificial satellites. This toolbox PROPAT is freely available at INPE's site (Carrara, 2014). It supports rigid body dynamics, as well as satellites equipped with nutation dumpers and reaction wheels. The kinematics is expressed in quaternion, which is more efficient than Euler angles, although quaternion needs some kind of renormalization from time to time during numeric integration. Presently, only gravity gradient and residual magnetic disturbance torques were modeled in PROPAT, but there are plans to include the aerodynamic and solar radiation pressure in the toolbox, besides a high order geopotential model to allow precise numeric orbit propagation. The library also presents coordinate transformation functions of orbit and attitude, which can be used to ease graphical visualization of the interest parameters. Equally available are models of the Earth's orbit around the Sun (sun positioning), the Earth's magnetic field at a given position, and the Greenwich sidereal time to locate the Earth reference axis in inertial coordinates. A fast analytical orbit propagator based on the Brower model (Brower and Clemence, 1961) completes the toolbox. The toolbox contains a demonstration program (`demo_propat`) that shows an example of orbit and attitude simulation.

Three examples of using the toolbox were presented: a complete orbit and attitude simulation of a 1U CubeSat satellite with a magnetic attitude controller, the attitude control of the Conasat satellite (an 8U CubeSat) (Carrara et al., 2014), and a simulation of a boomerang motion (Bigot et al., 2013), that shows the circular trajectory achieved with this peculiar object.

REFERENCES

- Bigot, P.; Carrara, V.; Massad, P. "Numerical calculation algorithm of attitude and movement of a boomerang". XI Conferência Brasileira de Dinâmica, Controle e Aplicações – DINCON 2013. Fortaleza, Out. 2013.
- Brower, D.; Clemence, G. M. "Methods of Celestial Mechanics". Academic, New York, NY, 1961.
- Carrara, V. "Attitude Simulation Software to Support Brazilian Space Missions". Journal of Aerospace Engineering, Sciences and Applications, Vol 3, n. 3, 2011. p. 11-20. ISSN 2236-577X. <<http://www.aeroespacial.org.br/jaes/a/editions/repository/v03/n03/02-Carrara.pdf>>.
- Carrara, V. "Cinemática e dinâmica da atitude de satélites artificiais". Instituto Nacional de Pesquisas Espaciais, INPE. São José dos Campos, 2012. 111 p. (sid.inpe.br/mtc-m19/2012/01.26.19.13-PUD). Disponível em: <<http://urlib.net/8JMKD3MGP7W/3B96GD8>>. Access: July 2012.
- Carrara, V. PROPAT Satellite Attitude and Orbit Toolbox for Matlab. Instituto Nacional de Pesquisas Espaciais – INPE, Available in <<http://www2.dem.inpe.br/val/projetos/propat/>>, Access: Sep., 2014.
- Carrara, V.; Kuga, H. K.; Bringhenti, P. M.; Carvalho, M. J. M. Attitude Determination, Control and Operating Modes for CONASAT Cubesats. 24th International Symposium on Space Flight Dynamics (ISSFD). Laurel, MD, 5-9 May 2014.
- Hughes, P. C. "Spacecraft Attitude Dynamics". Mineola: Dover, 1986.
- IGRF – "International Geomagnetic Reference Field". IAGA Division V-MOD Geomagnetic Field Modeling, 2010 <<http://www.ngdc.noaa.gov/IAGA/vmod/igrf.html>> Access: April 2013.
- Kuga, H. K.; Carrara, V.; Medeiros, V. M. "Rotinas auxiliares de mecânica celeste e geração de órbita". São José dos Campos, INPE, julho 1981. (INPE-2180-RPE/392).
- Kuga, H. K.; Medeiros, V. M.; Carrara, V. "Cálculo recursivo da aceleração do geopotencial". São José dos Campos, INPE, maio 1983. (INPE-2735-RPE/433). Apresentado na 35a. Reunião Anual da SBPC, Belém, PA, julho de 1983.
- Kuga, H. K.; Kondapalli, R. R.; Carrara, V. "Introdução à mecânica orbital - 2a Edição". 2. ed. São José dos Campos: INPE, 2012. 67 p. (sid.inpe.br/mtc-m05/2012/06.28.14.21.24-PUD). Disponível em: <<http://urlib.net/8JMKD3MGP7W/3C76K98>>. Access: July 2012.
- Lopes, R. V. F.; Carrara, V.; Kuga, H. K.; Medeiros, V. M. "Cálculo recursivo do campo geomagnético". São José dos Campos, INPE, setembro 1983. (INPE-2865-PRE/400). Apresentado na 35a. Reunião Anual da SBPC, Belém, PA, julho 1983.
- Reijneveld, J.; Choukroun, D. "Attitude Control System of the DELFI-N3XT Satellite". Progress in Flight Dynamics, GNC, and Avionics, Vol. 6, 2013, pp. 189-208. DOI: 10.1051/eucass/201306189.
- Shuster, M. D.; Oh, S. D. "Three-Axis Attitude Determination from Vector Observations". Journal of Guidance, and Control. Vol. 4, No. 1, 1981, pp. 70-77.
- Wertz, J. R. "Spacecraft attitude determination and control". D. Reidel Publishing, 1978.

RESPONSIBILITY NOTICE

The author is the only responsible for the printed material included in this paper.