# ATTITUDE SIMULATION SOFTWARE TO SUPPORT BRAZILIAN SPACE MISSIONS

**Valdemir Carrara**
*Instituto Nacional de Pesquisas Espaciais, INPE.*
*Av. dos Astronautas, 1758, São José dos Campos, SP, 12227-010 – val@dem.inpe.br*

***Abstract:*** *Cost reduction on system tests with hardware in the loop during qualification phase of the on-board attitude control software imply in increasingly simulation realism. Attitude simulation software not only has to mimic the dynamical behavior of the spacecraft, but also shall be easily configurable, to adapt to different orbit, attitude pointing requirements, environment disturbances, sensors, actuators, and to run in real-time among other minor requirements. As the requirements for accuracy and stability for attitude pointing has increased in many modern missions, so do the complexity of the simulation algorithms and necessity for dynamical realism. Rigid body dynamics can be considered excessively simple to model a satellite with several reaction wheels, non-rigid structures, unbalanced solar arrays, fuel and liquid sloshing, not to mention crew motion and vehicle docking. This work presents the efforts being carried out to fulfill the requirements of the Brazilian space missions, nominally Multi Mission Platform, with a simulation environment capable to accomplish from early definition and mission analysis phase up to acceptance tests of the attitude control software. Development is being conducted in order to achieve a high degree of realism and range of applicability of the simulation software, compatible with the expected cost reduction of the simulation hardware (ordinary PCs). The kinematics and dynamic models to support AOCS (Attitude and Orbit Control Subsystem) simulation and testing with or without hardware-in-the-loop for Brazilian space missions are presented. Normally the dynamic equations are expressed in angular momentum components, due to the high complexity of these equations when derived in terms of the angular velocity. This, of course, is particularly important in satellites composed by several articulated rigid bodies as solar panels, robotic arms, space booms, deployable antennas, etc. Nevertheless, the inverse of the inertia matrix shall be calculated on the run, since it varies in time. This can be somewhat slow, mainly considering the high degree of freedom in dynamics of satellites with articulated panels and reaction wheels. In this case a more complex but fast set of equations in terms of the angular velocity is the best choice. This work describes the attitude dynamic equations expressed in angular velocities for satellites with appendages (solar panels, telescopes, directional antennas, etc.) and reaction wheels. They can be easily extended, however, to include nutation dampers and robotic arms. As usual, this formulation requires the knowledge of the acting torque in the connection joint, which is not always understood, modeled or known. So the dynamic model presented in this work uses the angular acceleration instead of the torque at the joint, which simplifies and reduces the order of the differential equations. Simulation results are presented, emphasizing the difference between the behavior of rigid body and articulated rigid bodies.*

***Keywords:*** *Attitude simulation, Attitude dynamics, Real time simulation.*

## 1. Introduction

The equations of motion of a rigid body are normally expressed in terms of its inertia matrix. Because the control theory also uses arrays and vectors to express the control of a linear system, it is easer to simulate both the dynamics and control by means of a computer language that avoids the tedious work of algorithm construction to perform a simple array multiplication. Languages such as Matlab or Labview not only operate easily with vectors and matrices, but also have many resources to handle hardware interfaces and to run in real time. Programs written in these languages can be compiled to increase performance, since they usually are interpreted. Add to this a block-programming environment, and apparently the solution for the simulation environment was found. However, this simplification hides a code often oversized, slow and inefficient. Moreover, much of the time spent in attitude simulation is lost in calculation of space environment parameters, such as atmospheric density, geomagnetic and gravitational fields, whose algorithms do not take benefit from operations between matrices or block-programming packages. In fact, some tests performed by

the author showed that the processing time of a rigid body attitude simulation programmed in C is at least 10 times faster than one in compiled Matlab, and at least 10,000 times faster than in interpreted Matlab. It is obvious that the increasing of the computing processing power tips the scales to the side of the graphical programming environment. However, it is still likely the next generation of satellites shall have the on board attitude control programmed in conventional way, due to code size, performance and reliability requirements.

This paper presents the efforts being made in the sense to develop a simulation environment capable to act in the Attitude and Orbit Control Subsystem (AOCS) design since early phases (conception and mission analysis) up to qualification and integration tests. Next sections present the package overview, the dynamic model, some results coming from simulation and the conclusions.

## 2. Simulation package

This section will present a description of the simulation package designed to qualify the attitude control software for Brazilian's space program MMP (Multi Mission Platform) both with and without hardware in the loop. The package was developed in C++ and includes the following features:

- Dynamic equations for a single rigid body or multiple bodies articulated to the satellite, such as solar panels or robotic arms.
- Simplified and functional model for sensors and actuators.
- Modular construction to allow easy reconfiguration of the simulation process.
- Extensive library, which includes coordinate transformation and analytical orbit propagation.
- Real time functions allow to compute attitude at high frequencies and synchronous with the computer clock.

Even considering the package has been developed in order to support the MMP, it is, in fact, generic. This means it can be freely configured to simulate any satellite that has similar or uncommon characteristics, including spin or gravity-gradient stabilized satellites. Package functions are grouped in several modules: attitude propagator, coordinate conversion, attitude control functions, orbital and environmental ephemeris, sensors and actuators simulation, environmental disturbances and real-time processing control. Additionally it was implemented a set of structures to perform operations involving matrices and vectors, in order to simplify program coding and debugging. Among this structures are `matrix3`, `vector3` and `quaternion`. Some math operators were overridden in order to achieve a code similar to that in Matlab. These include operations between matrices and vectors to allow the symbolic representation of these operations, like product, sum, dot (internal) and cross products. Thus, the code for valid operations between matrices and vectors is something like this:

```
matrix3 rot_mat1, rot_mat2, rot_mat3; // matrix declaration
vector3 vec1, vec_pos;              // vector declaration
quaternion q;                       // quaternion declaration
...
rot_mat1 = {0.2, 0, -1, 0.6, 1, -2, 2, 1.5, 3}; // matrix values
vec_pos = {1543.3, 764.0, -2414.8};       // vector values
vec_pos._1 = rot_mat1._1._3;              // or this way
vec_pos._2 = rot_mat1._2._3;
vec_pos._3 = rot_mat1._3._3;
...
mat_rot = (rot_mat1 + rot_mat2)*rot_mat3*vec_pos;    // valid
```

The product of quaternions, in turn, provides direct attitude transformation between three coordinate systems, as explained in [1]. The quaternion structure assumes that the first three values (`q._1`, `q._2`

and `q._3`) are the vector components while the fourth (`q._4`) stores the scalar value. Additionally, some functions were added to provide matrix manipulation, such as to invert or to transpose a matrix, to normalize a vector or a quaternion, to assign values to a matrix or vector, etc.

The package was developed with the sense of "state machine" where the simulation parameters are stored in memory and then used during attitude propagation. Several functions to assign specific values to variables were implemented, as well as functions that restore previously stored values. For instance, the functions `int set_number_wheels (int nwheels)`, and `int get_number_wheels ()` allow to configure the number of momentum or reaction wheels in simulation and retrieve this value, respectively. Also, `int set_wheel_vector (int ind_wheel, vector3 vec_wheel)` and `vector3 get_wheel_vector (int ind_wheel)` allow to set and to retrieve the axis direction of a given wheel. Some values are available only after starting the simulation process, like the wheel speed which can be retrieved by `double get_wheel_speed (int ind_wheel)`.

Sensors and actuators were modeled based in their characteristics instead of specific equipment. All the parameters are configurable, so the package can virtually mimic any sensor whose characteristics are modeled. For example, a star sensor has two parameters: the output standard deviation for each axis and the transformation matrix of the sensor, including misalignments. Functions were implemented to simulate the following attitude control equipment: thrusters (cold or hot gas jets), magnetic coils, reaction or momentum wheels, magnetometers, star sensor, inertial unit, analog sun sensors and a GPS receiver. All these equipment can be selected individually.

Orbit propagation employs an analytical model with includes the effect of Earth oblateness [2], and a SGP8 [3] model, selected by the user. The inertial position of Sun is also achieved by an analytical model from [4] and [5]. It was implemented in the package a C version of the IGFF10 geomagnetic model, adapted by the author from a FORTRAN version produced by Susan Macmillan from British Geological Survey [6]. A complete description of the package can be seen in [7].

### 3. Kinematic and dynamic equations of motion for multi body satellite

The kinematic equations of a rotational body are written in quaternion, as usual, and are given by:

$$\dot{\mathbf{Q}} = \frac{1}{2}\begin{pmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{pmatrix}\mathbf{Q}, \tag{1}$$

where $\mathbf{Q}$ is the attitude quaternion and $\boldsymbol{\omega} = \begin{pmatrix} \omega_1 & \omega_1 & \omega_1 \end{pmatrix}^T$ is the angular velocity vector [8]. Expressing the satellite angular momentum with respect to the center of mass, the dynamic equations of a rigid body result

$$\dot{\boldsymbol{\omega}}^b = \mathbf{I}^{-1}(\mathbf{g}_{cm} - \boldsymbol{\omega}^{\times}\mathbf{I}\boldsymbol{\omega}), \tag{2}$$

where $\mathbf{I}$ is the inertia matrix and $\mathbf{g}_{cm}$ is the total external torque applied to the satellite (no internal reaction wheel torques at this moment). However, considering a satellite with $N$ articulated appendages to the main body through spherical joints, each one considered also rigid, the total angular momentum can be written with respect to a fixed point in the main body, since the center of mass changes in satellite frame due to appendage motions. Hughes [8] states that the additional complexity in the equations of motion does not justify deriving the angular momentum with respect

to the mass center. We shall see that the angular momentum relative to a fixed point is also complex when several appendages are positioned in the satellite body, besides computationally heavy. The equations of motion with respect to a fixed point $O$ are:

$$\begin{aligned}
\dot{\mathbf{p}}^b &= \mathbf{f} - \boldsymbol{\omega}^{\times}\mathbf{p} \\
\dot{\mathbf{h}}_o^b &= \mathbf{g}_o - \mathbf{v}_o^{\times}\mathbf{p} - \boldsymbol{\omega}^{\times}\mathbf{h}_o \\
\dot{\mathbf{h}}_{j,n}^n &= \mathbf{g}_{j,n} + [\mathbf{h}_{j,n}^{\times} + m_n \mathbf{C}_{bn}^T (\mathbf{v}_o + \boldsymbol{\omega}^{\times}\mathbf{b}_n)^{\times} \mathbf{C}_{bn}\, \mathbf{d}_n^{\times}](\mathbf{C}_{bn}^T \boldsymbol{\omega} + \boldsymbol{\omega}_n)
\end{aligned} \quad , \tag{3}$$

where $\mathbf{f}$ are the resulting external forces, $\mathbf{g}_o$ is the external torques (relative to $O$), $\mathbf{p}$ is the total satellite momentum, $\mathbf{v}_o$ is satellite velocity relative to an inertial frame, $\mathbf{h}_o$ is the total angular momentum and $\mathbf{C}_{bn}$ is the rotation matrix between the main body axes and the $n^{\text{th}}$ appendage system coordinates, fixed in its center of mass, as seen in Fig. 1. $\mathbf{h}_{j,n}$ is the appendage angular momentum with respect to the joint point $J_n$, $\mathbf{b}_n$ is the positioning vector of the articulated joint, $\mathbf{d}_n$ is the position of the appendage center of mass relative to its joint, $\mathbf{g}_{j,n}$ is the net torque applied in the joint $n$ by the satellite (no external torque on appendage), $m_n$ is its mass and $\boldsymbol{\omega}_n$ is the appendage angular velocity relative to the main body. Superscript × indicates the matrix of the cross product [8], and superscript $b$ or $n$ means that the derivative is taken with respect to the main body or $n^{\text{th}}$ appendage, respectively.

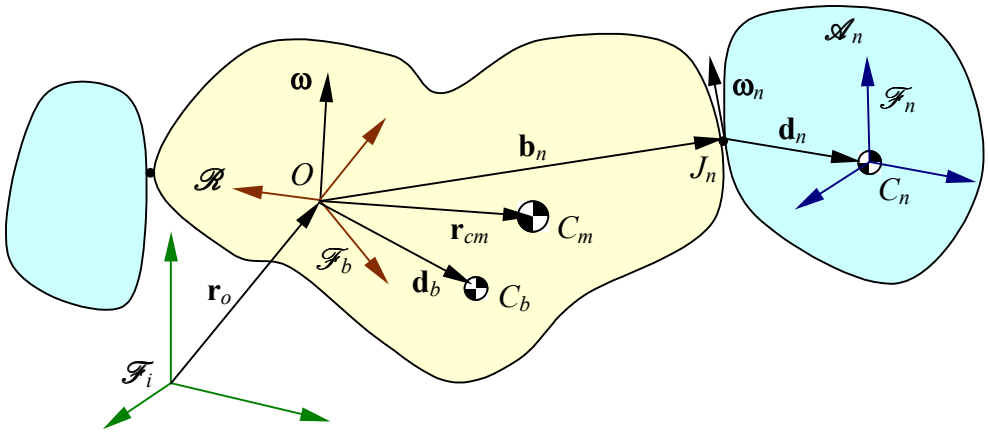


**Figure 1. Satellite main body $\mathscr{R}$ with articulated appendages $\mathscr{A}_n$.**

The linear and angular momentum of the whole satellite, and the angular momentum of the appendage $n$ can be written, respectively, as

$$\begin{aligned}
\mathbf{p} &= m\,\mathbf{v}_o - \mathbf{c}^{\times}\boldsymbol{\omega} - \sum_{n=1}^{N} m_n\, \mathbf{C}_{bn}\, \mathbf{d}_n^{\times}\, \boldsymbol{\omega}_n \\
\mathbf{h}_o &= \mathbf{c}^{\times}\,\mathbf{v}_o + \mathbf{J}\,\boldsymbol{\omega} + \sum_{n=1}^{N} \mathbf{J}_{on}\,\boldsymbol{\omega}_n \\
\mathbf{h}_{j,n} &= m_n\, \mathbf{d}_n^{\times}\, \mathbf{C}_{bn}^T\, \mathbf{v}_o + \mathbf{J}_{on}^T\, \boldsymbol{\omega} + (\mathbf{J}_{dd,n} + \mathbf{I}_n)\,\boldsymbol{\omega}_n
\end{aligned} \quad , \tag{4}$$

where $m$ is the satellite mass including appendages and the inertias are obtained from

$$\begin{aligned}
\mathbf{J} &= \mathbf{J}_b + \sum_{n=1}^{N} [\mathbf{J}_{bb,n} + \mathbf{J}_{bd,n} + \mathbf{J}_{on}\, \mathbf{C}_{bn}^T] \\
\mathbf{J}_{on} &= \mathbf{J}_{db,n}\, \mathbf{C}_{bn} + \mathbf{C}_{bn}\, (\mathbf{J}_{dd,n} + \mathbf{I}_n) \\
\mathbf{J}_{bb,n} &\triangleq m_n\, (\mathbf{b}_n^T\, \mathbf{b}_n\, \mathbf{1} - \mathbf{b}_n\, \mathbf{b}_n^T),
\end{aligned} \tag{5}$$

$$\mathbf{J}_{bd,n} \triangleq m_n (\mathbf{b}_n^T \mathbf{C}_{bn} \mathbf{d}_n \mathbf{1} - \mathbf{b}_n \mathbf{d}_n^T \mathbf{C}_{bn}^T)$$
$$\mathbf{J}_{db,n} \triangleq m_n (\mathbf{d}_n^T \mathbf{C}_{bn}^T \mathbf{b}_n \mathbf{1} - \mathbf{C}_{bn} \mathbf{d}_n \mathbf{b}_n^T) = \mathbf{J}_{bd,n}^T$$
$$\mathbf{J}_{dd,n} \triangleq m_n (\mathbf{d}_n^T \mathbf{d}_n \mathbf{1} - \mathbf{d}_n \mathbf{d}_n^T)$$

and $\mathbf{J}_b$ is the inertia matrix of the main body relative to $O$, and $\mathbf{I}_n$ is the inertia matrix of appendage $n$ relative to its center of mass $C_n$ in appendage fixed frame. $\mathbf{J}_{on}$, $\mathbf{J}_{bd,n}$ and $\mathbf{J}_{db,n}$ are named mixed inertia. The momentum equations can also be presented in the system inertia matrix form $\mathbf{P} = \mathbf{M}\,\mathbf{V}$ [8], or:

$$\begin{pmatrix} \mathbf{p} \\ \mathbf{h}_o \\ \mathbf{h}_{j,1} \\ \cdots \\ \mathbf{h}_{j,n} \end{pmatrix} = \begin{pmatrix} m\mathbf{1} & -\mathbf{c}^\times & -m_1 \mathbf{C}_{b1} \mathbf{d}_1^\times & \cdots & -m_N \mathbf{C}_{bN} \mathbf{d}_N^\times \\ \mathbf{c}^\times & \mathbf{J} & \mathbf{J}_{o1} & \cdots & \mathbf{J}_{oN} \\ m_1 \mathbf{d}_1^\times \mathbf{C}_{b1}^T & \mathbf{J}_{o1}^T & \mathbf{J}_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ m_N \mathbf{d}_N^\times \mathbf{C}_{bN}^T & \mathbf{J}_{oN}^T & 0 & \cdots & \mathbf{J}_N \end{pmatrix} \begin{pmatrix} \mathbf{v}_o \\ \boldsymbol{\omega} \\ \boldsymbol{\omega}_1 \\ \cdots \\ \boldsymbol{\omega}_n \end{pmatrix}. \tag{6}$$

The velocities $\mathbf{v}_o$, $\boldsymbol{\omega}$ and $\boldsymbol{\omega}_n$ need to be computed in order to solve the equations of motion, so the system inertia matrix $\mathbf{M}$ shall be inverted. This can be done by a recursive block wise inversion, since $\mathbf{M}$ is symmetric, with order of $6 + N$. Unfortunately this inversion must be evaluated at each time step because the rotation matrix $\mathbf{C}_{bn}$ is time dependent. The kinetic energy is simple given by

$$T = \frac{1}{2} \mathbf{V}^T \mathbf{M}\,\mathbf{V} \ . \tag{7}$$

A drawback in the above equations is the normally unknown torque $\mathbf{g}_{j,n}$ at each joint. The electrical and mechanical characteristics of the motors and gears at the joint are difficult to obtain, and linear models are far from realistic. To make things worse, the driving mechanism is frequently controlled in closed loop, as, for instance, in solar panels that follow the sun around the orbit. It is proposed then that the appendage angular velocity $\boldsymbol{\omega}_n$ or acceleration $\dot{\boldsymbol{\omega}}_n$ can be used instead the unknown torque. This procedure reduces the order of the whole system to 6 degree, since it is assumed that the angular velocity in each joint is already known. Within this approach the equations of motion can be derived for the angular momentum about the system center of mass, which is now uncoupled from the translation motion $\mathbf{v}_o$ resulting a 3 degree system:

$$\dot{\boldsymbol{\omega}}^b = \mathbf{J}^{-1} \left\{ \mathbf{g}_{cm} - \boldsymbol{\omega}^\times \mathbf{h}_{cm} - \sum_{n=1}^{N} \mathbf{C}_{bn} \{ \boldsymbol{\omega}_n^\times [\mathbf{J}_n (\mathbf{C}_{bn}^T \boldsymbol{\omega} + \boldsymbol{\omega}_n)] + \mathbf{J}_n (\dot{\boldsymbol{\omega}}_n + \mathbf{C}_{bn}^T \boldsymbol{\omega}^\times \mathbf{C}_{bn} \boldsymbol{\omega}_n) \} \right\} , \tag{8}$$

and now the inertia matrices are given by

$$\mathbf{J} = \mathbf{I}_b + \mathbf{J}_r + \sum_{n=1}^{N} [\mathbf{J}_{bb,n} + \mathbf{J}_{bd,n} + \mathbf{J}_{on} \mathbf{C}_{bn}^T - \mathbf{J}_{br,n} - \mathbf{J}_{rd,n} - \mathbf{J}_{dr,n}]$$
$$\mathbf{J}_n = \mathbf{I}_n + \mathbf{J}_{dd,n} + \mathbf{C}_{bn}^T (\mathbf{J}_{db,n} - \mathbf{J}_{dr,n}) \mathbf{C}_{bn}$$
$$\mathbf{J}_r \triangleq m (\mathbf{r}_{cm}^T \mathbf{r}_{cm} \mathbf{1} - \mathbf{r}_{cm} \mathbf{r}_{cm}^T) \quad , \tag{9}$$
$$\mathbf{J}_{br,n} \triangleq m_n (\mathbf{b}_n^T \mathbf{r}_{cm} \mathbf{1} - \mathbf{b}_n \mathbf{r}_{cm}^T)$$
$$\mathbf{J}_{rd,n} \triangleq m_n (\mathbf{r}_{cm}^T \mathbf{C}_{bn} \mathbf{d}_n \mathbf{1} - \mathbf{r}_{cm} \mathbf{d}_n^T \mathbf{C}_{bn}^T)$$

$$\mathbf{J}_{dr,n} \triangleq m_n (\mathbf{d}_n^T \mathbf{C}_{bn}^T \mathbf{r}_{cm} \mathbf{1} - \mathbf{C}_{bn} \mathbf{d}_n \mathbf{r}_{cm}^T) = \mathbf{J}_{rd,n}^T$$
$$\mathbf{J}_{dkdn} \triangleq m_n (\mathbf{d}_k^T \mathbf{C}_{kn} \mathbf{d}_n \mathbf{1} - \mathbf{C}_{kn}^T \mathbf{d}_k \mathbf{d}_n^T)$$

where $\mathbf{I}_b$ is the inertia matrix of the main body relative to its center of mass $C_b$, $\mathbf{J}_{dkdn}$ is a cross inertia between the appendages $n$ and $k$ and $\mathbf{r}_{cm}$ is the center of mass in main body frame, obtained from

$$\mathbf{r}_{cm} = \frac{1}{m} \sum_{n=1}^{N} m_n (\mathbf{b}_n + \mathbf{C}_{bn} \mathbf{d}_n) . \tag{10}$$

The angular momentum $\mathbf{h}_{cm}$ about the system center of mass $C_m$ is therefore given by

$$\mathbf{h}_{cm} = \mathbf{J}\, \boldsymbol{\omega} + \sum_{n=1}^{N} \mathbf{C}_{bn} \mathbf{J}_n\, \boldsymbol{\omega}_n . \tag{11}$$

The above equations can be changed in order to comply to a gyrostat, dual spin or to a satellite equipped with $K$ reaction or momentum wheels. The dynamic equations of motion result

$$\dot{\boldsymbol{\omega}}^b = \mathbf{J}^{-1} \left\{ \mathbf{g}_{cm} - \boldsymbol{\omega}^{\times} \mathbf{h}_{cm} - \sum_{n=1}^{N} \mathbf{C}_{bn} \{ \boldsymbol{\omega}_n^{\times} [\mathbf{J}_n (\mathbf{C}_{bn}^T \boldsymbol{\omega} + \boldsymbol{\omega}_n)] + \mathbf{J}_n (\dot{\boldsymbol{\omega}}_n + \mathbf{C}_{bn}^T \boldsymbol{\omega}^{\times} \mathbf{C}_{bn} \boldsymbol{\omega}_n) \} - \sum_{k=1}^{K} g_k \mathbf{a}_k \right\}, \tag{12}$$

where $g_k$ is the torque applied to wheel $k$, and $\mathbf{a}_k$ is the unit vector of its rotation axis in spacecraft coordinates. Since each wheel exchange momentum with the satellite, the dynamic equations shall include the angular momentum $h_{w,k}$ of $K$ wheels:

$$\dot{h}_{w,k} = g_k . \tag{13}$$

The total angular momentum of the satellite shall also be modified to include the wheels:

$$\mathbf{h}_{cm} = \mathbf{J}\, \boldsymbol{\omega} + \sum_{n=1}^{N} \mathbf{C}_{bn} \mathbf{J}_n\, \boldsymbol{\omega}_n + \sum_{k=1}^{K} h_{w,k} \mathbf{a}_k , \tag{14}$$

and the inertia matrix is computed by

$$\mathbf{J} = \mathbf{I}_b + \mathbf{J}_r + \sum_{n=1}^{N} [\mathbf{J}_{bb,n} + \mathbf{J}_{bd,n} + \mathbf{J}_{on} \mathbf{C}_{bn}^T - \mathbf{J}_{br,n} - \mathbf{J}_{rd,n} - \mathbf{J}_{dr,n}] - \sum_{k=1}^{k} I_{k,s} \mathbf{a}_k \mathbf{a}_k^T , \tag{15}$$

in which $I_{k,s}$ is the inertia of the wheel's rotor $k$ around its spin axis.

From the model presented above it is clear that the rotation matrix $\mathbf{C}_{bn}$ between the main body and each appendage frame shall be provided in order to compute the attitude. However, those matrices are time dependent and have only 3 independent parameters. The vectors $\mathbf{b}_n$ and $\mathbf{d}_n$ shall also be provided, and together with the angular velocity vector $\boldsymbol{\omega}_n$ sum 12 parameters for each appendage. A small reduction in this number can be achieve if the joint is supposed to be rotational instead of a spherical, which is very reasonable since most of satellites appendages rotates around a given axis. In this case each appendage shall have 10 independent parameters. In the simulation package it was adopted a Denavit-Hartemberg parameters, which allows the complete specification of the rotation

and translation between coordinate systems. They are largely used in robotics to perform coordinate transformation between robot joints and links [9], [10]. The 10 Denavit-Hartemberg parameters for coordinate transformation are shown in Fig. 2. They are three link angles $\theta_{n0}$, $\theta_{n1} + \theta_n(t)$ and $\theta_{n2}$; three link offsets $d_{n0}$, $d_{n1}$ and $d_{n2}$; two link lengths $a_{n0}$ and $a_{n1}$ and two link twists $t_{n0}$ and $t_{n1}$. It is worth to mention that the joint rotation angle, $\theta_n(t)$, comes from the double numeric integration of $\dot{\boldsymbol{\omega}}_n(t)$. The rotation matrix and vectors for appendage $n$ are then achieved by

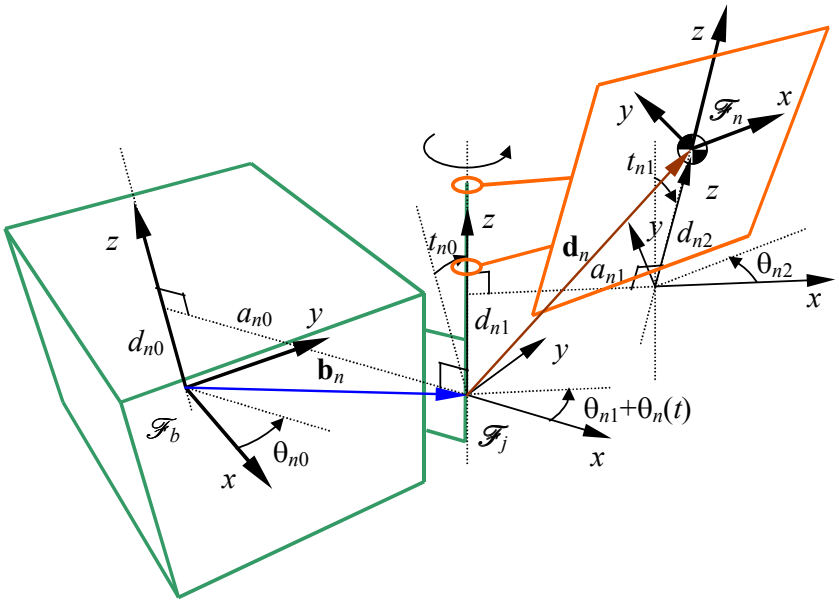


**Figure 2. Denavit-Hartemberg parameters for appendage rotation.**

$$
\begin{aligned}
\mathbf{C}_{bn} &= \mathbf{C}_{bj}\ \mathbf{C}_{jn} \\
\mathbf{C}_{bj} &= \mathbf{R}_z(\theta_{n0})\ \mathbf{R}_x(t_{n0})\ \mathbf{R}_z(\theta_{n1})\ \mathbf{R}_z(\theta_n(t)) \\
\mathbf{C}_{jn} &= \mathbf{R}_x(t_{n1})\ \mathbf{R}_z(\theta_{n2}) \\
\mathbf{b}_n &= \mathbf{R}_z(\theta_0)\ \mathbf{v}_{n0} \\
\mathbf{d}_n &= \mathbf{C}_{bj}\ [\mathbf{v}_{n1} + \mathbf{C}_{jn}\ \mathbf{v}_{n2}] = \mathbf{C}_{bj}\mathbf{v}_{n1} + \mathbf{C}_{bn}\ \mathbf{v}_{n2}
\end{aligned}
\quad , \tag{16}
$$

with $\mathbf{v}_{n0} = \begin{pmatrix} a_{n0} & 0 & d_{n0} \end{pmatrix}^T$, $\mathbf{v}_{n1} = \begin{pmatrix} a_{n1} & 0 & d_{n1} \end{pmatrix}^T$, $\mathbf{v}_{n2} = \begin{pmatrix} 0 & 0 & d_{n2} \end{pmatrix}^T$, and

$$
\theta_n(t) = \int\int \dot{\omega}(t)\, dt + \omega_{0n}\, t\,. \tag{17}
$$

It shall be noted that $\theta_{n1}$ is already the second integration constant. The elementary rotation matrices $\mathbf{R}_i$ around the $i$ axis are defined by

$$
\mathbf{R}_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\operatorname{sen}\theta \\ 0 & \operatorname{sen}\theta & \cos\theta \end{pmatrix},\ \mathbf{R}_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \operatorname{sen}\theta \\ 0 & 1 & 0 \\ -\operatorname{sen}\theta & 0 & \cos\theta \end{pmatrix},\ \mathbf{R}_z(\theta) = \begin{pmatrix} \cos\theta & -\operatorname{sen}\theta & 0 \\ \operatorname{sen}\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{18}
$$

## 4. Simulation results

Some tests were performed in order to assure the accuracy, reliability and correctness of the software package. Attitude simulation of single rigid body was carried out with expected nutation and precession behaviors. The results can be seen in video animation [11] made by the author.

Procedures to control the simulation in real time were developed and tested for clock synchronization. The real time control adjusts the numeric integration step so as to keep tracking between the simulation time and the computer clock. This kind of control is usually employed in computer games and car and airplane simulators. Results for step size adjustment are presented in Fig. 3 that shows the synchronization error as function of the propagation time. The tracking error remains below 5 ms most of the time. Step size for attitude propagation can be as low as 50 microseconds in a 2.4 GHz computer.

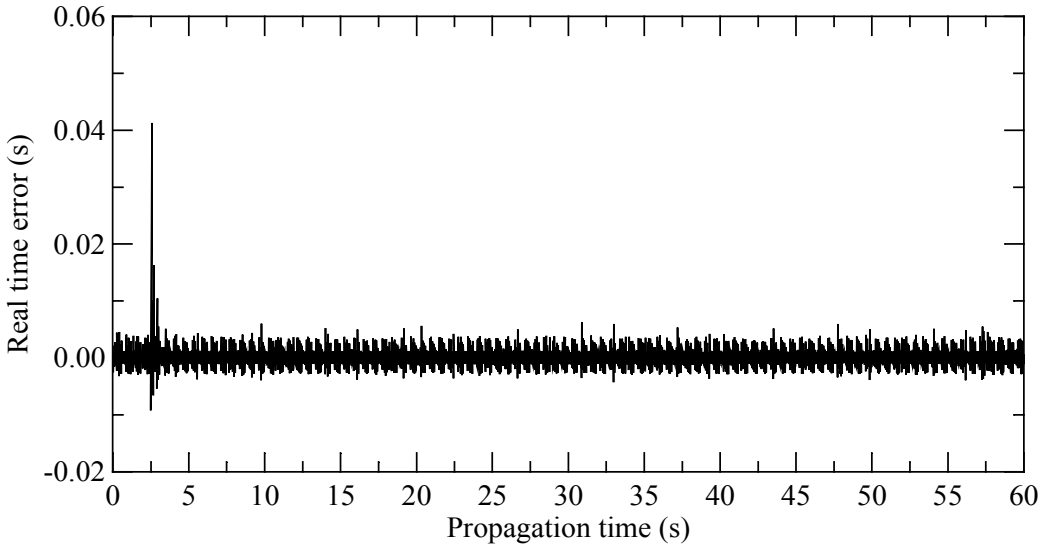


**Figure 3. Real time error during step size adjustment and synchronization control.**

Examples of simulation in sensors output are presented in Fig. 4 for a magnetometer (single axis), in Fig. 5 for a star sensor (single axis shown here), Fig. 6 for a GPS (showing only the semi-major axis), and Fig. 7 shows a single axis gyroscope. Models for sensors take into account standard deviation for random white noise, scale factor, bias and alignment. All these parameters are configurable by specific functions. For instance, the function `int set_inertial_unit (matrix3 axis_dir, vector3 scale_error, vector3 bias, matrix3 random_drift, matrix3 random_walk)` allows configuring a 3-axis gyroscope. In all these figures the initial attitude is $80^o$, $-20^o$ and $-150^o$ for a 313 Euler rotation angles, with no perturbation torque or control during simulation and no initial angular velocity. The keplerian elements for orbit were: semi-major axis of 6978 km, eccentricity of 0.06, inclination 1 rad, perigee argument of 0.5 rad and null values for right ascension of the ascending node and mean anomaly. Date and time were chosen as Jan. 1st 2001 at 10:40:00 hour UTC. The satellite is a rigid body with diagonal inertia matrix of 10, 15 and 20 kg $m^2$ in axes *x*, *y* and *z*. A noise with standard deviation of 2 nT and a bias of –2 nT was used in the magnetometer simulation of Fig. 4. The star sensor in Fig. 5 has a noise with 6 $10^{-5}$ rad standard deviation in *x*-axis, and was sampled at 1 Hz. Figure 6 shows the GPS receiver output converted to keplerian elements. The variation in the semi-major axis is due to the osculating elements coming from the SGP8 orbit propagation. Simulation of a single axis gyro in *z* direction shown in Fig. 7 took in consideration a null scale factor, bias of 5 $10^{-9}$ rad/sec, random drift of $10^{-8}$ rad/sec and random walk of 4 $10^{-10}$ rad/$sec^{3/2}$.

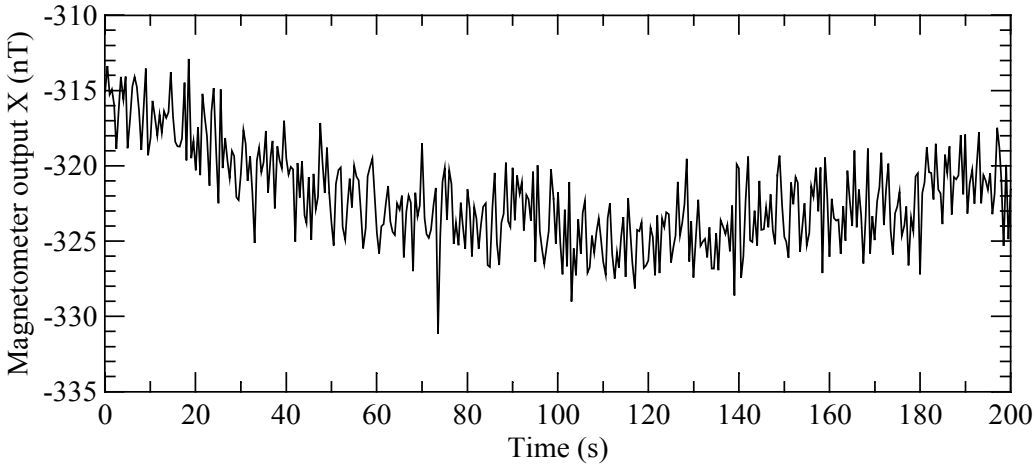


**Figure 4. Simulation of a magnetometer (X axis) output.**

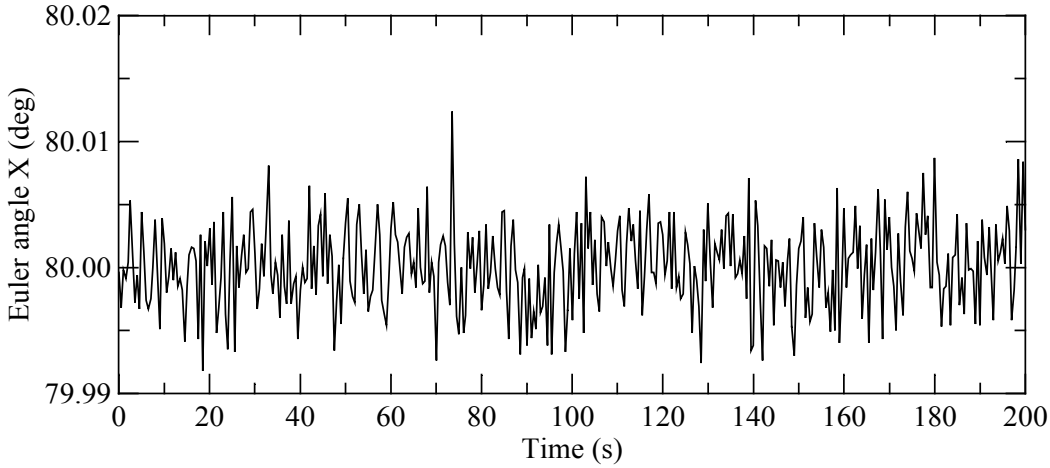


**Figure 5. Simulation of star sensor output in X axis.**

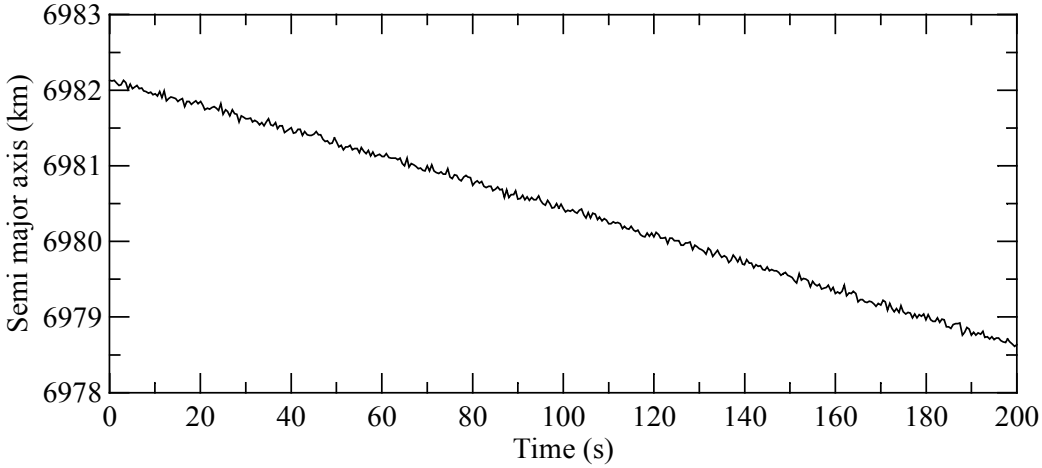


**Figure 6. Simulated output of a GPS receiver, converted to keplerian elements (only semi-major axis is shown)**

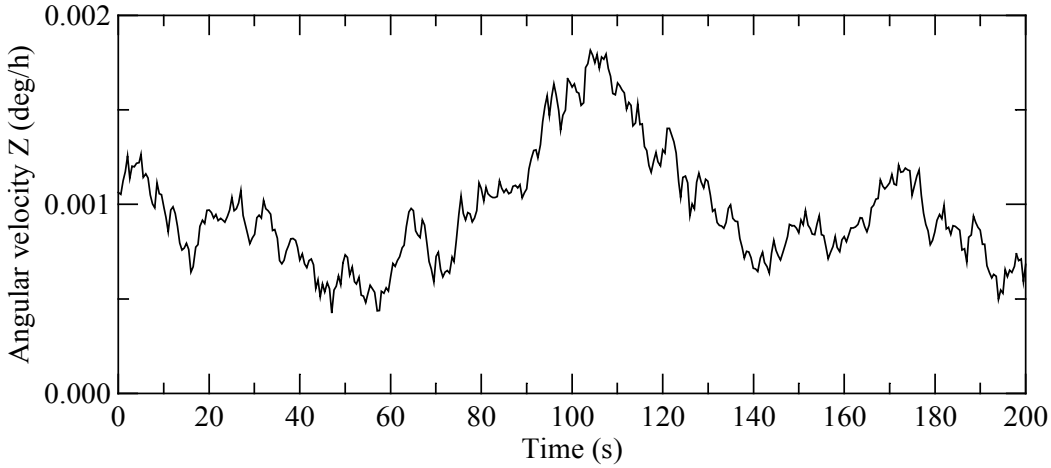


**Figure 7. Simulated gyro output in z axis, sampled at 1 Hz.**

In Fig. 8 the output for 8 analog solar sensors in octahedron configuration are shown as function of time. It can be seen in this figure that the satellite enters the Earth shadow around 190 sec. Transition in penumbra is clearly noted at that time. The sensors are modeled with a noise with 0.01 of standard deviation and a multiplicative noise proportional to sensor output with standard deviation of 0.02. A velocity of $3.6^{o}$ per second around the satellite *z*-axis was introduced in this simulation.

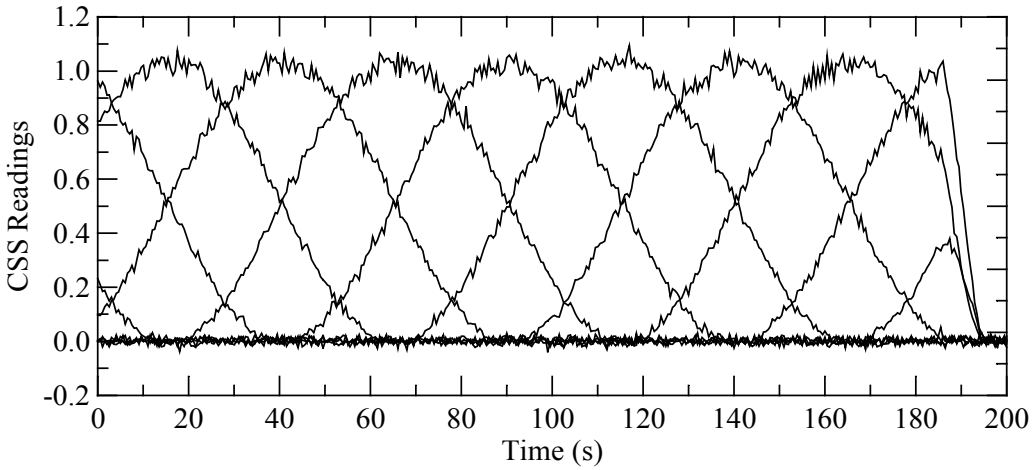


**Figure 8. Simulation of 8 analog sun sensors in octahedron configuration. Earth and Moon albedos were not considered**.

At the time of this writing the dynamic equations for a satellite with appendages, as reported in Section 3, was not yet coded in to the package, although a rigid body with reaction wheels can be simulated in the current version. Figure 9 shows a simulated attitude acquisition of a satellite with 6 thrusters of 2 N each. The thruster torques were 0.8 Nm in positive and negative directions around the satellite coordinate axes. The attitude control algorithm reduces the satellite spin ratio whenever the angular velocity is larger than 0.1 rad/sec, or, otherwise, uses a bang-bang control. The thrusters for each axis are selected by the sign of the attitude error, which is computed by a term proportional to a 1-2-3 attitude angles plus a term proportional to the angular velocity. The proportional constants were 1 and 6 respectively. Figure 10 presents the phase diagram of the acquisition maneuver. In the figures the red, green and blue curves were assigned to the *x*, *y* and *z* axes, respectively. A dead band based on thruster's minimum impulse of 0.05 sec (seen as two straight parallel lines in Fig. 10) was introduced in order to avoid unnecessary thruster activation.

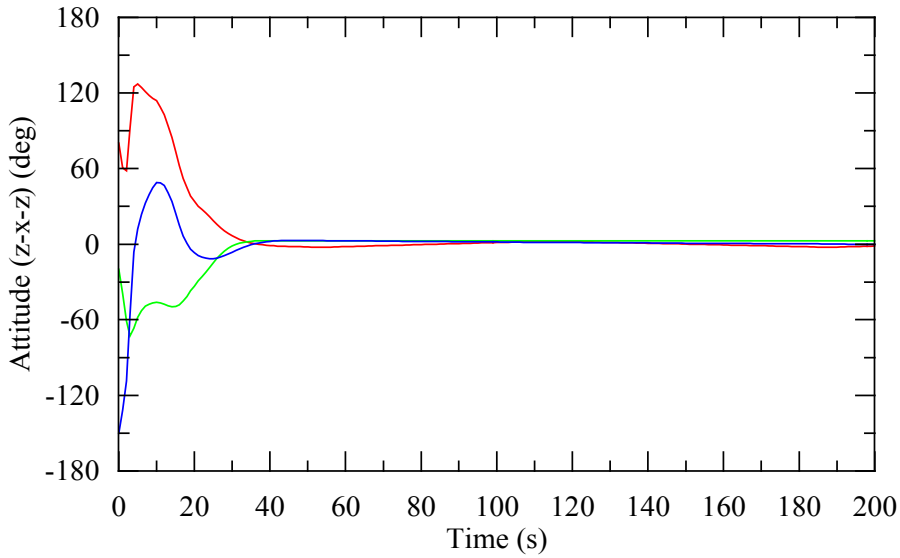


**Figure 9. Attitude angles of an attitude acquisition simulation by means of 6 thrusters**.

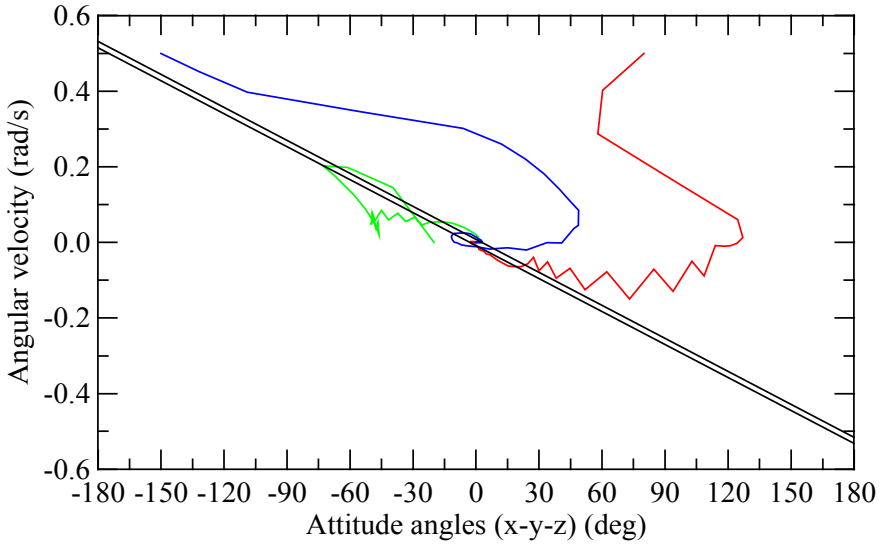


**Figure 10. Phase diagram of the simulated attitude acquisition maneuver**.

## 5. Conclusions

A first version of the attitude simulation software was presented here. The software package was written in C and is composed by several functions capable to configure the satellite properties, sensors, actuators, attitude and orbit. However, the control loop algorithm and its coding is still an attribution of the user. The package has 36 operators to perform vector-matrix operations in a Matlab like style, and sums something around 180 functions total, among coordinate transformations (orbit and attitude), sensor simulation, numeric integrator selection and configuration, satellite mass properties, actuators selection and switching, magnetic field computation, Sun position and real time processing control.

Software validation was carried out by means of the dynamic behavior of the satellite attitude, together with analysis of the angular momentum and kinetic energy. Some video animations of the dynamics produced with this package and POVray are available at [11].

Next steps to improve the software package are:

- To operate the attitude simulator and controller in separated computers communicating through serial interface;

- To implement and to test multi-body attitude dynamics
- To include some environmental perturbations – atmospheric drag, solar pressure, gravity gradient torque and residual magnetic torque
- To improve the sensor and actuator models

## 6. References


[1] Wertz, J. R. *Spacecraft attitude determination and control*, London: D. Reidel, 1978 (Astrophysics and Space Science Library).

[2] Brower, D.; Clemence, G. M. *Methods of celestial mechanics*. New York, NY, Academic, 1961.

[3] Hoots, F. R.; Roehrich, R. L. *Models for propagation of NORAD element sets*, Aerospace Defense Command, United States Air Force, Spacecraft Report No 3. Dec. 1980.

[4] 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-2189-RPE/392).

[5] Flandern, T. C.; Pulkkinen, K. F. Low precision formulae for planetary positions. *The Astrophysical Journal Supplement Series*, V. 41, n. 3, pg. 319-411, Nov, 1979.

[6] – International Geomagnetic Reference Field – International Association of Geomagnetism and Aeronomy. In http://www.ngdc.noaa.gov/IAGA/vmod/igrf.html. Access in 2009.

[7] Carrara, V. Manual do pacote computacional para simulação de atitude de satélites. São José dos Campos, INPE, 2007. In http://www2.dem.inpe.br/val/projetos/att_pro/index.htm, Access in 2010.

[8] Hughes, P. C. Spacecraft Attitude Dynamics. Dover, Mineola, NY, 1986.

[9] Craig, J. J. *Introduction to Robotics: Mechanics and Control* (2nd Edition). Addison-Wesley, 1989.

[10] Asada, H.; Slotine, J.-J. E. *Robot Analysis and Control*. John Wiley and Sons, New York, 1986.

[11] INPE, Divisão de Mecânica Espacial e Controle, Carrara, V. *Dinâmica de atitude*. In http://www2.dem.inpe.br/val/projetos/atdyn/index.html, Access in 2011.