

Attitude Determination and Control System Simulation for Low Earth Orbit CubeSats

An Implementation of Reference Generation, Control Strategies, and Data Management Layers

1st Vinícius Álvares Pereira

*Electrical Engineering of Escola Politécnica
Universidade de São Paulo
São Paulo, Brazil
vinicius.alvaresp@usp.br*

2nd Fábio de Oliveira Fialho

*Electrical Engineering of Escola Politécnica
Universidade de São Paulo
São Paulo, Brazil
fabio.fialho@usp.br*

Abstract—This paper presents the development of a high-fidelity, configurable Attitude Determination and Control System (ADCS) simulator designed to address the challenges of ground testing and validation for CubeSat missions. The simulator employs quaternion-based kinematics and rigid body dynamics, which are validated against in-orbit telemetry data.

The system supports multiple reference frames and attitude targeting modes, including Nadir, Earth Surface Tracking, and Inertial Pointing. Regarding control strategies, the work details the implementation of a PD controller for safe mode Sun pointing and a nonlinear Sliding Mode Controller for rough pointing maneuvers.

Furthermore, the paper describes a robust data management workflow utilizing HDF5 to integrate a user-friendly interface with MATLAB/Simulink. Simulation results demonstrate the effectiveness of the control algorithms and the simulator's capability to facilitate personalized mission analysis in an academic environment.

Index Terms—ADCS, Attitude Control, CubeSat, Quaternions, Simulation

I. INTRODUCTION

The CubeSat satellite model has been gaining increased popularity in recent decades due to its easier-to-build structures and cheaper fabrication costs, when compared to earlier classical models [1]. Universities across the globe have embraced this opportunity, starting their own CubeSat projects and study groups, expanding academic interest and scientific progress in the topic. However, approximately 41.3% of the nanosatellites launched between 2000 and 2016 have reported total or partial mission failure [2]. As CubeSat complexity increases beyond passive stabilization, statistical analyses show that the Attitude Determination and Control System (ADCS) becomes an increasingly cause of anomalies alongside power, on-board computer (OBC), and communication subsystems [3].

The ADCS can be considered a critical subsystem for CubeSats that need effective and precise attitude control in order to meet mission requirements. Universities and academic institutions frequently operate within limited budgets, tight schedules, and a shortage of specialized personnel, making it difficult to conduct tests and validate these systems on the ground [4]. Given these constraints, high-fidelity simulations

are crucial, as they provide an effective alternative for testing and validating the ADCS in specific mission scenarios, without the need for costly hardware and testing facilities.

Within this context, this paper discusses the development of a high-fidelity ADCS simulator at the University of São Paulo's Department of Telecommunications and Control Engineering. The project aims to enhance ADCS testing and validation in academia by creating a detailed simulator with configurable variables and mission parameters.

The simulator's foundational design is documented in [5], serving as a basis and a starting point for the work on this paper and other systems within the simulation. Initially developed in MATLAB/Simulink for engineers familiar with control systems, it evolved to include an integrated interface, allowing users to easily adjust simulator parameters and adapt CubeSat subsystems and algorithms.

The current content included in the interface and simulation is illustrated in the diagram in Fig. 1. It shows the recommended configuration sequence preset in the interface. The first tab is dedicated to the Mission Profile and Parameters configuration, where you can set up general mission components and definitions. This includes configuring the CubeSat's orbit, the overall satellite size, its coordinate systems, the desired attitude references, and the control modes used within a state machine framework.

After defining the general mission profile, the user establishes the CubeSat's inertia matrix within the Satellite Dynamics tab. The total body inertia matrix can be defined in two ways: either by manually inputting its values or by estimating them based on the satellite's mass, dimensions, and center of mass. Additionally, the inertia matrices for other subsystems within the CubeSat, such as the OBC and Payload subsystems, can be defined similarly, and then the overall body inertia matrix is calculated based on these individual subsystem inertia matrices and their positions on the satellite.

Next, the actuators and sensors are categorized in their respective tabs. The available actuators are magnetorquers and reaction wheels, each defined as a set of three actuators. Each individual actuator can have its defining mathematical

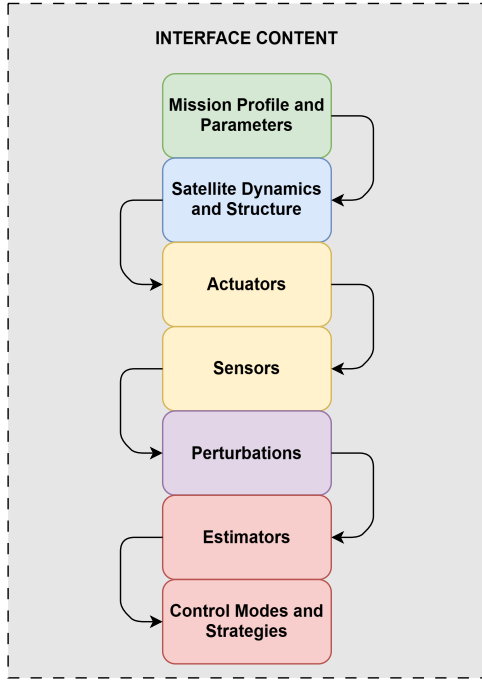


Fig. 1. Interface content representation with suggested configuration sequence

model parameters configured, along with sample time and initial conditions. The sensors can be customized in a similar way, with available options including star trackers, gyroscopes, magnetometers, and sun sensors.

In the Perturbations tab, both attitude and orbital disturbances are identified and configured. The simulator encompasses the primary disturbances that occur in Low Earth Orbit (LEO). These disturbances include magnetic field interactions, gravitational gradients, atmospheric drag, solar radiation pressure, third-body influences, and gravitational non-uniformities.

Lastly, the estimators, control modes, and control strategies are defined in their respective tabs. Available attitude estimators include Kalman Filters, Triad and Q-Method algorithms, and a simplified angular rate estimator. Control modes include open loop, safe mode, rough pointing, fine pointing, and science pointing. Each mode offers various control strategies, allowing users to select the active controller and corresponding estimator.

The presentation on this paper will focus on the development of the following topics: a description and implementation of different reference frames and attitude reference types in Sections III and IV, respectively; strategies for attitude determination and control for different operating modes in Section V; data management workflow throughout the entire user interaction process within the interface and simulator, in Section VI; and conclusions in Section VII.

II. KINEMATICS AND DYNAMICS

A. Kinematics

Euler angles are the most commonly used representation for rotations of rigid bodies. In this work, they are employed

to define and visualize the rotation angles of satellites and reference frames. Typically, Euler angles are represented by three rotations: the Yaw angle ψ around the Z-axis, the Pitch angle θ around the Y-axis, and the Roll angle ϕ around the X-axis. A visual representation of these angles is provided in Fig. 2.

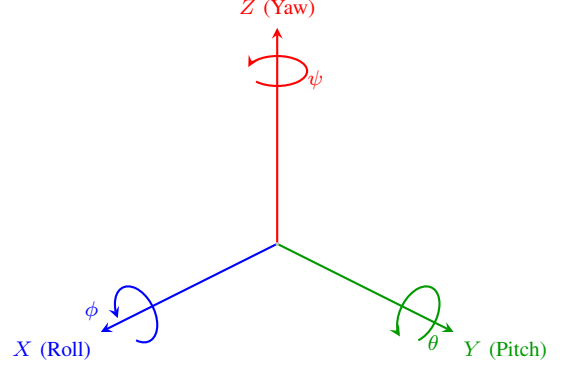


Fig. 2. Euler Angles Representation (Rotation at Axis Tips)

Frame rotations using Euler Angles can be calculated using cosine matrix computations; however, this type of representation can experience singularities known as gimbal locks, which result in a loss of reference along one of the axes. As a result, Euler angles are only desirable for visualization and defining angles, but not for attitude calculations and computations.

To address this limitation, we utilize the hypercomplex number known as a Quaternion. Quaternions were first introduced by William Rowan Hamilton in 1843 and have become widely adopted for representing rotations. They offer significant advantages, such as lower computational demands compared to the standard cosine matrix representation of Euler angles. Moreover, quaternions effectively eliminate the problem of gimbal lock singularities.

Quaternions are defined by:

$$\begin{aligned} \mathbf{q} &= q_1 + q_2\hat{i} + q_3\hat{j} + q_4\hat{k} \\ &= q_1 + \mathbf{q} \end{aligned} \quad (1)$$

where $\mathbf{q}$ represents the quaternion, q_1 is its scalar component, and $\mathbf{q}$ denotes its vectorial component.

To obtain and update the quaternion representing a rotation on a determined frame, the following calculation is used:

$$\dot{\mathbf{q}} = \frac{1}{2}[\Omega]\mathbf{q} \quad (2)$$

where $\dot{\mathbf{q}}$ represents the quaternion derivative, and Ω is the skew-symmetric matrix defined as:

$$\Omega = \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} \quad (3)$$

where ω_x , ω_y , and ω_z represent the angular velocities along each principal axis of the frame.

Also, an important quaternion related to control strategies is the error quaternion q_e . It represents the required rotation between the satellite's current attitude and the desired attitude and can be defined, together with its derivative $\dot{q}_e$, as shown in [7]:

$$q_e = q_d \otimes q_s^{-1} \quad (4)$$

$$\omega_e = \omega_s - \omega_d \quad (5)$$

$$\dot{q}_e = \frac{1}{2} [S(q_e) - q_{e4} \mathbf{I}_{3 \times 3}] \omega_e \quad (6)$$

where q_d and q_s are the desired and current satellite quaternion, respectively; $\otimes$ is the quaternion multiplication operator, ω_e is the angular velocity error vector between the satellite's angular velocity ω_s and the desired angular velocity ω_d , and $S(q_e)$ is a skew-symmetric matrix like the following:

$$S(q_e) = \begin{bmatrix} 0 & -q_{e4} & q_{e3} \\ q_{e4} & 0 & -q_{e2} \\ -q_{e3} & q_{e2} & 0 \end{bmatrix} \quad (7)$$

B. Dynamics

The rotation dynamics of the satellite follow Euler's Equations of Motion for a rigid body, given by the equation:

$$\mathbf{T} = \mathbf{J}\dot{\boldsymbol{\omega}} + (\boldsymbol{\omega} \times \mathbf{J}\boldsymbol{\omega}) \quad (8)$$

in this equation, $\mathbf{T}$ represents the resultant torque vector, $\mathbf{J}$ is the inertia matrix of the body, $\dot{\boldsymbol{\omega}}$ is the angular acceleration vector, and $\boldsymbol{\omega}$ is the angular velocity vector.

Equation (8) can be expanded in order to observe the coupling of the dynamics between each axis:

$$T_x = J_x \dot{\omega}_x + \omega_y \omega_z (J_z - J_y) \quad (9)$$

$$T_y = J_y \dot{\omega}_y + \omega_x \omega_z (J_x - J_z) \quad (10)$$

$$T_z = J_z \dot{\omega}_z + \omega_x \omega_y (J_x - J_y) \quad (11)$$

The final dynamics equation of the satellite incorporates both disturbance torques and control torques generated by the actuators. Specifically, in the context of torque produced by the reaction wheels, a gyroscopic effect is introduced as a result of the accumulation of angular momentum within the wheels. The dynamics using the magnetorquers is shown in (12), while the dynamics using the reaction wheels are shown in (13):

$$\dot{\boldsymbol{\omega}}_s = \mathbf{J}_s^{-1} [\mathbf{T}_d - \mathbf{T}_m - \boldsymbol{\omega}_s \times \mathbf{J}_s \boldsymbol{\omega}_s] \quad (12)$$

$$\dot{\boldsymbol{\omega}}_s = \mathbf{J}_s^{-1} [\mathbf{T}_d - \mathbf{T}_w - \boldsymbol{\omega}_s \times (\mathbf{J}_s \boldsymbol{\omega}_s + \mathbf{h}_w)] \quad (13)$$

where $\boldsymbol{\omega}_s$ and $\mathbf{J}_s$ are the satellite's angular velocity vector and moment of inertia matrix, respectively; $\mathbf{T}_d$ is the disturbances torque vector, $\mathbf{T}_m$ is the magnetorquer torque vector, $\mathbf{T}_w$ is the reaction wheels torque vector, and $\mathbf{h}_w$ is the reaction wheels angular momentum vector.

C. Dynamics validation

Menegaldo, at [5], uses the available Paris Observatory Pic-Sat satellite [6] in-orbit data to conduct a thorough validation process. It was only possible to validate the dynamics, given in (13), and the detumbling control, since the PicSat's remaining attitude control systems failed during mission deployment, preventing further ADCS validations.

After reviewing the satellite's attitude telemetry, Menegaldo found appropriate time windows with enough coherent data, capable of meeting the validation process requirements. Two different instances of this validation can be seen in Fig. 3 and Fig. 4, where the graphics on the left compare the telemetry angular velocity against the simulated angular velocity, while the graphics on the right show the error values (difference) between the real and simulated angular velocities.

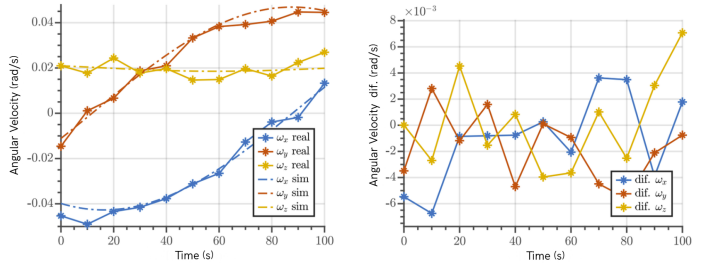


Fig. 3. First dynamics validation (Adapted from [5])

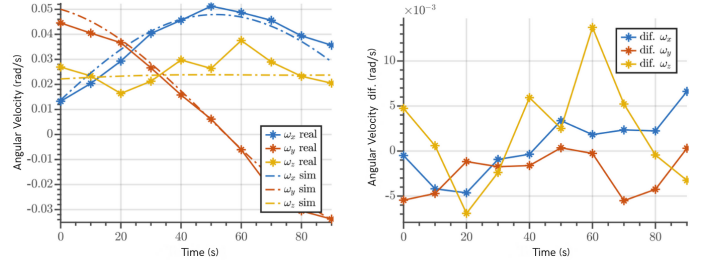


Fig. 4. Second dynamics validation (Adapted from [5])

In both cases, the error stayed within the range of 10^{-3} , which is more than adequate to conclude that the simulator's dynamics are validated. While it is currently not possible to validate the rest of the ADCS for the time being, all the work presented in this paper can be deemed reliable, as it is based on a simulator with confirmed validated dynamics.

III. REFERENCE FRAMES

In ADCS simulations, the first step is to define the reference frames, which serve as orthonormal three-dimensional bases for specifying and manipulating the spatial position and attitude angles of objects. These positions and angles can be expressed either within a single reference frame or in relation to another frame. The reference frames utilized in this work are the same as those defined in [5]. These include the Earth-Centered Inertial Frame (ECI), Earth-Centered Earth-Fixed Frame (ECEF), Local Vertical Local Horizontal

Frame (LVLH), Body Reference Frame (BRF), and Control Reference Frame (CRF).

A. ECI and ECEF frames

The Earth-Centered Inertial and Earth-Centered Earth-Fixed reference frames both have their origins located at the Earth's center. The ECI frame, shown in Fig. 5, is assumed to be non-rotating with respect to distant celestial objects (i.e., inertial for the purposes of this work), whereas the ECEF frame rotates with the Earth according to its spin movement.

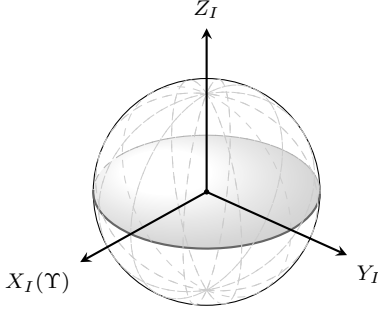


Fig. 5. ECI frame

In this study, the ECI frame is adopted as the primary reference frame, serving as the basis for expressing and transforming all other frames. The ECEF frame is employed specifically for representing reference attitudes associated with Earth-surface tracking.

To access the ECEF frame, it is necessary to have a quaternion that can transform coordinates from other frames into the ECEF frame, more specifically, from the ECI to the ECEF. The quaternion $q_{I/E}$, that attends this purpose, can be obtained through the following equation:

$$q_{I/E} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} \cos\left(\frac{\theta_G}{2}\right) \\ 0 \\ 0 \\ \sin\left(\frac{\theta_G}{2}\right) \end{bmatrix} \quad (14)$$

where θ_G is the Greenwich Sidereal Time (GST)

B. LVLH frame

The Local Vertical Local Horizontal frame, illustrated in Fig. 6, typically corresponds to the CubeSat's orbit, with its origin located at the satellite's geometric center. In this frame, the Z-axis points directly downwards towards the Earth (nadir), the X-axis points in the direction of the satellite's linear velocity, and the Y-axis completes the orthonormal basis in a right-handed (dextrorotatory) configuration.

The orbit's frame angular velocity in the inertial frame, $\omega_{O/I}$, can be calculated using the satellite's position and linear velocity:

$$\omega_{O/I} = \frac{\mathbf{r}_{s_I} \times \mathbf{v}_{s_I}}{\mathbf{v}_{s_I}^2} \quad (15)$$

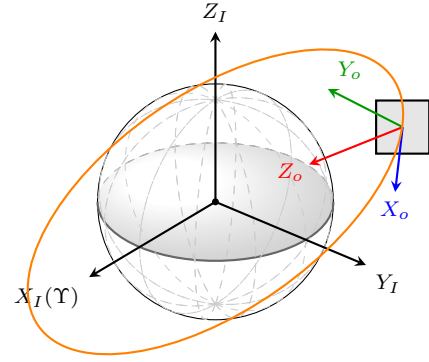


Fig. 6. LVLH frame

where $\mathbf{r}_{s_I}$ and $\mathbf{v}_{s_I}$ represent the CubeSat's position and linear velocity vectors in the inertial frame, respectively.

With (15), it is possible to retroactively calculate the quaternion that represents the frame rotation from the ECI frame to the LVLH frame, $q_{I/O}$. The derivative of this quaternion can be computed through (2) using the LVLH frame angular velocity ω_O , while ω_O is obtained through the rotation of $\omega_{O/I}$ to the LVLH frame by $q_{I/O}$, after integration of the quaternion derivative previously calculated:

$$\omega_O = q_{I/O} \otimes \omega_{O/I} \otimes q_{I/O}^* \quad (16)$$

$$\dot{q}_{I/O} = \frac{1}{2}[\Omega_O]q_{I/O} \quad (17)$$

where $q_{I/O}^*$ is the quaternion conjugate and Ω_O has the same structure as defined in (3), making use of the ω_O angular velocity vector.

C. BRF and CRF frames

The Body Reference frame is where the CubeSat's attitude angles are defined and represented. Its origin is at the CubeSat's geometric center, and its axes are oriented to better visually illustrate the defined attitude and pointing directions. Finally, the Control Reference frame is the frame viewed and used by the ADCS sensors, controllers, and actuators. This frame has its origin at the satellite's center of mass (CoM), with its pointing axes aligned with the vehicle's principal inertial axes. This alignment simplifies the calculations required by the control systems, approximating the CubeSat's inertia matrix as much as possible to a diagonal matrix. Both frames can be visualized in Fig. 7.

IV. REFERENCE ATTITUDE GENERATION

A crucial step in ADCS design is the selection of available attitude reference types and the specific reference attitudes that will be used as targets during the mission. The attitude reference types developed and included in the simulation are Nadir pointing, Earth's Surface Tracking, and Inertial Pointing.

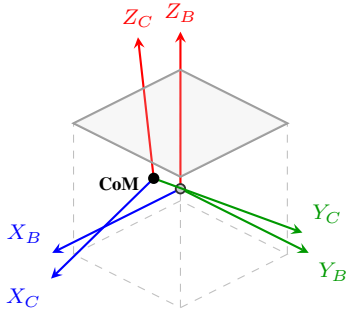


Fig. 7. BRF and CRF frame

A. Nadir Pointing

In aerospace terms, nadir refers to the vector pointing from an in-orbit satellite towards Earth, oriented perpendicular to the planet's surface. This usually means that the nadir vector will point towards Earth's center, as seen in Fig. 8. This mode is primarily used by Earth observation and remote sensing satellites, as well as by telecommunications and meteorological satellites.

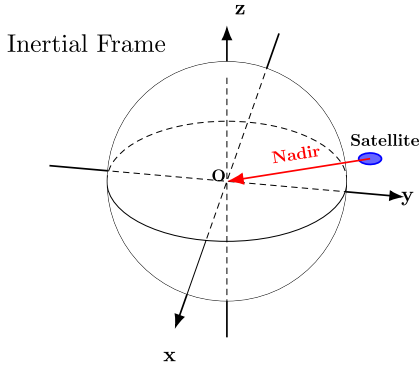


Fig. 8. Nadir attitude reference

In this reference, there is no need to convert angles to quaternion, as the quaternion $q_{I/O}$, obtained through (17), already represents the attitude desired by the nadir pointing, and can be directly designated as the desired attitude reference for the quaternion error calculation on (4).

B. Earth's Surface Tracking

The Earth's surface tracking attitude reference mode is employed for missions that require observation of specific areas of Earth for extended periods of time and greater amount of details. In this type of reference, presented in Fig. 9, the angles used to locate the target are specified as follows: the Longitude angle, measured from the Greenwich meridian; the Latitude angle, measured from the equator; and the Roll angle, which corresponds to the rotation of the satellite's pointing axis.

These angles need to be transformed into a reference quaternion in order to be used as an attitude reference on the quaternion-based ADCS controller logic. Defining the latitude

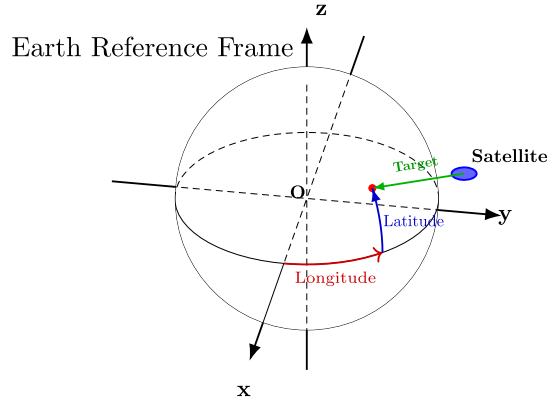


Fig. 9. Earth's surface tracking reference

angle as α and the longitude angle as β , the position vector of the target $\mathbf{r}_{target}$ in the ECEF system is expressed as follows:

$$\mathbf{r}_{target_E} = R_e \begin{bmatrix} \cos(\alpha) \cos(\beta) \\ \cos(\alpha) \sin(\beta) \\ \sin(\alpha) \end{bmatrix} \quad (18)$$

where R_e represents the radius of the Earth.

Next, the $\mathbf{r}_{target_E}$ vector is transformed into the ECI frame using the (14) quaternion. The normalized pointing axis $\hat{\mathbf{z}}_d$ of the BRF-centered desired attitude frame is then determined by subtracting the satellite's position vector in the ECI from the target's position vector:

$$\mathbf{r}_{target_I} = \mathbf{q}_{I/E}^* \otimes \mathbf{r}_{target_E} \quad (19)$$

$$\hat{\mathbf{z}}_d = \frac{\mathbf{r}_{target_I} - \mathbf{r}_{s_I}}{|\mathbf{r}_{target_I} - \mathbf{r}_{s_I}|} \quad (20)$$

To establish the two remaining axes of the desired attitude frame, $\hat{\mathbf{y}}_d$ is calculated through the cross product between $\hat{\mathbf{z}}_d$ and the satellite's linear velocity vector in the inertial frame $\mathbf{v}_{s_I}$. Next, $\hat{\mathbf{x}}_d$ is computed by the cross product between $\hat{\mathbf{z}}_d$ and $\hat{\mathbf{y}}_d$, completing the right-handed orthonormal frame:

$$\hat{\mathbf{y}}_d = \frac{\hat{\mathbf{z}}_d \times \mathbf{v}_{s_I}}{|\hat{\mathbf{z}}_d \times \mathbf{v}_{s_I}|} \quad (21)$$

$$\hat{\mathbf{x}}_d = \hat{\mathbf{y}}_d \times \hat{\mathbf{z}}_d \quad (22)$$

Finally, the direction cosine matrix representing the coordinate transformation to the desired attitude frame is generated by the previously calculated axes vectors, as:

$$DCM_{Tracking} = [\hat{\mathbf{x}}_d \ \hat{\mathbf{y}}_d \ \hat{\mathbf{z}}_d] \quad (23)$$

where $DCM_{Tracking}$ is the 3x3 direction cosine matrix for the BRF-centered desired attitude on the Earth's surface tracking reference mode. This matrix can be transformed in a quaternion reference for the (4) attitude error calculation using the standard transformation between direction cosine matrices and quaternions.

C. Inertial pointing

The inertial pointing attitude reference type refers to targets located anywhere within the ECI frame. This reference type, shown in Fig. 10, is usually employed by satellite missions aiming to observe other planets, stars, or any celestial objects that can be considered inertial regarding their positions on the ECI frame.

The angles that define the target's position are established in the inertial frame. These include the Right Ascension (RA) angle, which is measured from the vernal equinox axis; the Declination (Dec) angle, measured from the equator; and the Roll angle, as mentioned in the Earth's surface tracking example.

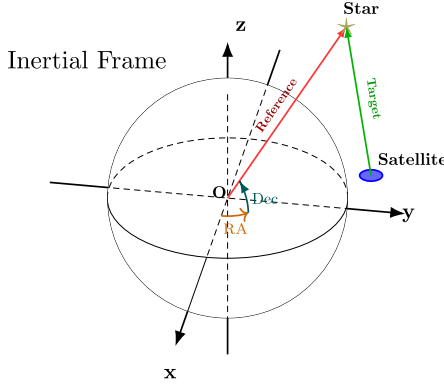


Fig. 10. Inertial pointing reference

To convert these angles into the desired reference quaternion, we can apply the direct transformation for the ZYX angle sequence. This is valid because the targets in this reference system can be regarded as being at an infinite distance from the satellite, and the desired attitude frame can be treated as equivalent to the Earth-centered attitude frame defined by the reference angles. Defining the Right Ascension angle as α , the Declination angle as β , and the Roll angle as ψ , we have:

$$\begin{aligned} c_x &= \cos(\psi/2), s_x = \sin(\psi/2) \\ c_y &= \cos(\beta/2), s_y = \sin(\beta/2) \\ c_z &= \cos(\alpha/2), s_z = \sin(\alpha/2) \end{aligned} \quad (24)$$

$$\mathbf{q}_{ref_{ECI}} = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} c_z c_y c_x + s_z s_y s_x \\ c_z c_y s_x - s_z s_y c_x \\ c_z s_y c_x + s_z c_y s_x \\ s_z c_y c_x - c_z s_y s_x \end{bmatrix} \quad (25)$$

where $\mathbf{q}_{ref_{ECI}}$ is the inertial pointing desired quaternion that will be used in the (4) error quaternion calculation.

V. ATTITUDE DETERMINATION AND CONTROL STRATEGIES

As mentioned previously, the attitude determination and control strategies present in the simulator are many, but the ones that will be presented and described next in this paper are the simplified angular rate estimator, the PD Sun pointing

controller for safe mode operation, and the Sliding Mode Controller for rough pointing maneuvers.

The simulation results presented in this section use the satellite structural parameters of the 16U Paris Observatory CubeSat CAASTOR, as listed in Table I, the initial conditions given in Table II, and the orbital definitions summarized in Table III.

TABLE I
SATELLITE PARAMETERS

Parameters	Values
Dimensions	16U (40cm × 10cm × 20cm)
Mass	21kg
Inertia Matrix $[J_s]$	$\begin{bmatrix} 0.366 & 0 & 0 \\ 0 & 0.313 & 0 \\ 0 & 0 & 0.088 \end{bmatrix} \text{ kg}\cdot\text{m}^2$
Center of Mass	$[0.04 \ 0.04 \ -0.04]^T \text{ m}$
Magnetorquers Max. Momentum	$[1.6 \ 1.6 \ 1.6]^T \text{ A}\cdot\text{m}^2$
Reaction Wheels Max. Torque	$[1.8 \ 1.8 \ 1.8]^T 10^{-3} \text{ N}\cdot\text{m}$
Reaction Wheels Max. Vel.	$[3 \ 3 \ 3]^T 10^3 \text{ rpm}$

TABLE II
INITIAL CONDITIONS

Parameters	Values
Angular Velocity $[\omega_s]$	$[0 \ 0 \ 0]^T \text{ rad/s}$
Quaternion $[\mathbf{q}_s]$	$[1 \ 0 \ 0 \ 0]^T$
Reaction Wheels Ang. Vel. $[\omega_w]$	$[0 \ 0 \ 0]^T \text{ rad/s}$
Sample time	0.01 s
Simulation time	1200 s

TABLE III
ORBIT PARAMETERS

Parameters	Values
Eccentricity	0.0009
Inclination	97.56°
Right Ascension	1.355°
Argument of Perigee	5.718°
True Anomaly	2.554°
Orbital Period	5664.6 s
Orbital Altitude	496.9 km

A. Simplified Angular Rate Estimator

When the CubeSat enters safe mode, it is essential to minimize energy usage across the entire system. As a result, only a

limited set of sensors and actuators are available, specifically the Sun sensors, magnetometers, and magnetorquer actuators.

Despite these limitations, the CubeSat still needs to perform maneuvers, such as orienting its solar panels toward the Sun or activating its detumbling control while in safe mode. However, due to the restricted resources, some standard algorithms for determining attitude and angular velocity cannot be used. To address this limitation, a simplified angular rate estimator is implemented using the sensors and actuators that remain operational in safe mode.

The algorithm described in [8] operates under the assumption that the Sun's position is nearly inertial and that the variation in the direction of the orbit's magnetic field can be ignored regarding the satellite's angular rates. Using both of these factors, it estimates the CubeSat's angular velocity as follows:

$$\hat{\omega}_k^B = \frac{\mathbf{b}_k \times \mathbf{b}_{k-1}}{\Delta T} \quad (26)$$

$$\hat{\omega}_k^S = \frac{\mathbf{s}_k \times \mathbf{s}_{k-1}}{\Delta T} \quad (27)$$

where $\hat{\omega}_k^B$ represent the satellite's angular velocity estimated with the magnetic field data vectors $\mathbf{b}$, measured by the magnetometer, and $\hat{\omega}_k^S$ represent the satellite's angular velocity estimated with the Sun's position data vector $\mathbf{s}$, measured by the Sun sensors. k and $k-1$ define the time instants of the measured data and ΔT denotes the sampling period of the ADCS.

In the case of Detumbling, where the data measured by the magnetometer is considered more reliable, the previous estimates can be combined to generate a more robust and complete angular rate estimation, in the following manner:

$$\hat{\omega}^{B+S} = \hat{\omega}^B + (\hat{\omega}^S \cdot \mathbf{b})\mathbf{b} \quad (28)$$

Similarly, the Sun sensors measurements are considered more reliable when the Sun pointing controller is active, leading to the following recommended combined estimation of angular velocity:

$$\hat{\omega}^{S+B} = \hat{\omega}^S + (\hat{\omega}^B \cdot \mathbf{s})\mathbf{s} \quad (29)$$

B. Sun pointing PD controller

The CubeSat enters safe mode when its battery reaches a low limit, or some critical error or emergency occurs. In this mode, one of the main tasks that needs to be performed in order to guarantee the stabilization and safe operation of the satellite is pointing its solar panels towards the Sun's direction, thereby preventing the battery from running out and the whole system from shutting down.

The control technique chosen to perform this task in this work was also introduced and explained in [8]. It consists of a classical PD control applied by magnetic actuators (due to the restrictions in safe mode) solely on the two remaining axes that won't be pointing to the Sun, and its equations are given as follows:

$$\mathbf{T}_{Sun} = K_p(-\mathbf{z}_s \times \mathbf{S}_{BRF}) + K_d(\boldsymbol{\omega}_d - \hat{\boldsymbol{\omega}}_s) \quad (30)$$

where $\mathbf{T}_{Sun}$ is the generated Sun pointing control torque signal, K_p is a proportional gain, $\mathbf{z}_s$ is the designated satellite pointing vector, $\mathbf{S}_{BRF}$ is the Sun's position normalized vector on the BRF, K_d is a derivative gain, and $\boldsymbol{\omega}_s$ is the satellite's angular velocity vector given by the simplified angular rate estimator on (29).

Considering the configurability and variability of the simulator's interface, which allows users to modify the structure of the CubeSat and its inertia matrix, a genetic algorithm-based PD tuning method was chosen to be implemented in the interface. This approach aims to assist users who have little to no knowledge of control systems in effectively tuning their parameters.

In magnetic control techniques, the acting torque is generated by the interaction between the magnetic dipole moment created by the magnetorquer actuators and the Earth's magnetic field, given by:

$$\mathbf{T}_m = \mathbf{m} \times \mathbf{B} \quad (31)$$

where $\mathbf{m}$ denotes the actuator's magnetic dipole moment vector and $\mathbf{B}$ the Earth's magnetic field vector acting on the satellite.

The graphics on Fig. 11 expose the safe mode sun pointing simulation results based on the parameters from Tables I, II and III. The constants $K_p = 0.063$ and $K_d = 1.042$ were utilized on the PD controller, optimized by the genetic algorithm for these specific parameters. First-order filters were applied on the output from the partial estimates (26) and (27), as it was recommended in [8], attenuating the measurement noise coming from the sensors.

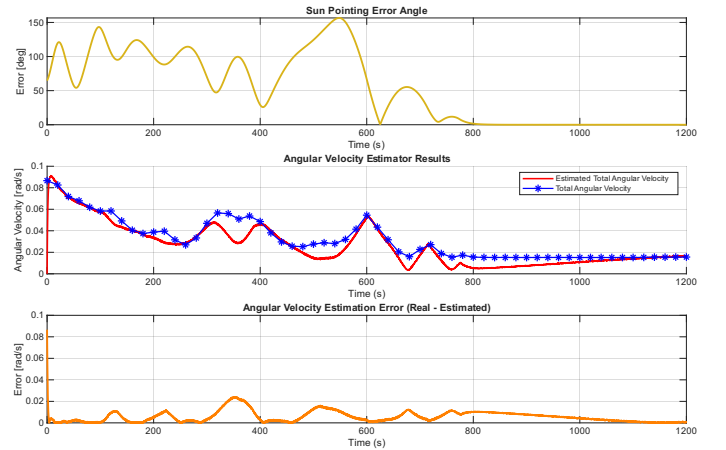


Fig. 11. Sun Pointing and Angular Velocity Estimation Simulation Results

The acting control torque can be considered of small magnitude due to being restricted by the interaction between the Earth's magnetic field and the field generated by the magnetic actuators in (31). Nevertheless, around the 800s mark, the control converges and successfully aligns the satellite's axis towards the Sun. The steady state error stays around 0.01° degrees after convergence.

The performance of the angular rate estimator is significantly worse compared to other, more sophisticated estimators,

with a maximum error of approximately 0.022 rad/s during the simulation. Despite its limitations, the simple angular rate estimator was sufficient for the effective convergence of the PD Sun Pointing control algorithm.

C. Sliding Mode Control

The rough pointing mode is designed to maneuver the satellite through large angles and approach the desired attitude. In the nonlinear dynamics of the system, as seen in (13), the required large-angle maneuvers in rough pointing prevent the option of linearizing the dynamics. Therefore, a nonlinear controller should be implemented for this mode.

Nonlinear Sliding Mode Control (SMC) is an effective technique for managing uncertainties in system parameters and external disturbances, making it ideal for space environments. It allows for precise control even with imperfect models or changing conditions. Developed by Soviet engineer Vladimir Utkin in the 1950s and 1960s, SMC quickly gained recognition and found diverse applications in control systems engineering.

In the paper by Yeh [9], the equations for the sliding surface and control torque input for attitude control using quaternions are thoroughly discussed. According to the Sliding Mode Control theory [10], the system error dynamics tend toward zero as the sliding surface is approached. Hence, it is crucial to have an accurate equation for the sliding surface. Specifically for attitude control, this equation can be expressed as:

$$s = P\bar{q}_e + \omega_e = 0 \quad (32)$$

In the equation, $s = [s_x \ s_y \ s_z]^T$ represents the sliding surface for each control axis of the satellite, P is a positive diagonal matrix, and $\bar{q}_e$ is the vectorial part of the error quaternion.

According to [9], it was proven using a Lyapunov function that equation (32) is asymptotically stable. This means that the system error dynamics tend towards zero in a finite time when it reaches the sliding surface.

In order to obtain a suitable control torque from the defined sliding surface, we first define a positive definite Lyapunov function candidate as follows:

$$V_s = \frac{1}{2} s^T J_s s \quad (33)$$

and it's derivative:

$$\begin{aligned} \dot{V}_s &= \frac{1}{2} s^T J_s \dot{s} + \frac{1}{2} \dot{s}^T J_s s \\ &= s^T J_s \dot{s} \end{aligned} \quad (34)$$

where $\dot{s}$ is given as:

$$\dot{s} = P\dot{\bar{q}}_e + \dot{\omega}_s - \dot{\omega}_d \quad (35)$$

Replacing equation (13) on (35) and then (35) on (34), we have the final derivative of the Lyapunov equation:

$$\begin{aligned} \dot{V}_s &= s^T J_s \dot{s} \\ &= s^T [J_s P\dot{\bar{q}}_e + J_s \dot{\omega}_s - J_s \dot{\omega}_d] \\ &= s^T [J_s P\dot{\bar{q}}_e + T_d - T_w \\ &\quad - \omega_s \times (J_s \omega_s + h_w) - J_s \dot{\omega}_d] \end{aligned} \quad (36)$$

The control torque signal T_{smc} , designated as the desired output torque from the reaction wheels, is based on (36) and written as:

$$\begin{aligned} T_{smc} &= Ks + J_s P\dot{\bar{q}}_e - \omega_s \times (J_s \omega_s + h_w) \\ &\quad - J_s \dot{\omega}_d + \Lambda_s \end{aligned} \quad (37)$$

where K is a positive definite diagonal matrix and $\Lambda_s = [\lambda_{s1} \ \lambda_{s2} \ \lambda_{s3}]^T$ is the discontinuous term responsible for leading the system to the sliding surface.

The terms λ_{si} , where $i = 1, 2, 3$, are defined as:

$$\lambda_{si} = c_{si} \text{Sat} \left(\frac{s_i}{\phi} \right) \quad (38)$$

Here, the saturation function resolves the control input chattering problem. ϕ represents the thickness of the boundary layer that surrounds the sliding surface area, and c_{si} is the discontinuous control variable that ensures that the system reaches the sliding surface from any initial value, based on the following condition:

$$c_{si} \geq |D + F - \omega_s \times \Delta J_s \omega_s + \Delta J_s P\dot{\bar{q}}_e| \quad (39)$$

where D is the total sum of attitude disturbance torques acting on the CubeSat, F is the nonlinearities present in the reaction wheel models, and ΔJ_s denote the difference between the assumed and actual values of the satellite's inertia matrix.

An SMC simulation was made for every attitude reference type presented in section IV, in order to evaluate the controller performance on different attitude control cases. All of them use a Q-Method plus Kalman filter combination as attitude estimator, as well as the parameters of Tables I, II, III, and IV. Table V reference angles were used in both Inertial Pointing and Earth Surface Tracking cases. The results are shown in Figs. 12, 13, and 14.

TABLE IV
SLIDING MODE CONTROL PARAMETERS

Parameters	Values
Sliding surface gain matrix $[P]$	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$
Control torque gain matrix $[K]$	$\begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{bmatrix}$
Discontinuous term constant $[c_{si}]$	$[1 \ 1 \ 1]^T 10^{-4}$
Sliding surface layer boundary $[\phi]$	0.5

In each scenario, the SMC controller effectively stabilized and converged to the desired attitude within a similar time frame, approximately around the 100-second mark. For both Earth Surface Tracking and Inertial Pointing modes, the steady-state angle error remained below 1 degree. In the Nadir

TABLE V
ATTITUDE REFERENCE ANGLES

Longitude/RA	Latitude/Dec	Roll	Switch Time
-45°	60°	0	0
120	-60°	0	600s

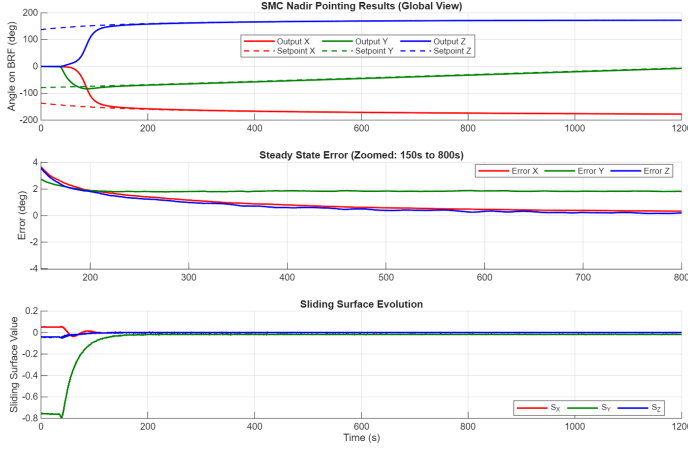


Fig. 12. Sliding Mode Control Nadir Pointing Simulation Results

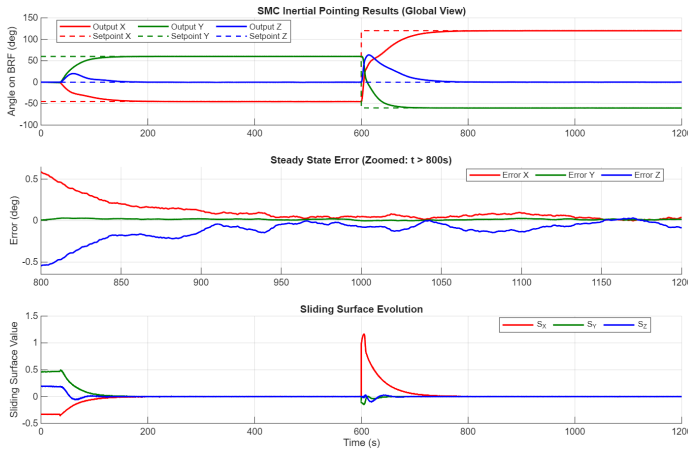


Fig. 13. Sliding Mode Control Inertial Pointing Simulation Results

Pointing simulation, the maximum steady-state error reached 2 degrees. In all cases, the behavior of the sliding surfaces aligned with theoretical expectations.

VI. DATA MANAGEMENT AND VISUALIZATION

A fundamental aspect of this project is the configurable user interface. This interface enables users to simulate examples of ADCS with customized subsystems, controllers, sensors, actuators, and structures. It is specifically designed to facilitate the testing of CubeSat missions in a secure and controlled simulation environment prior to actual launch.

A powerful and user-friendly database had to be carefully selected in order to manage the large volume of configurable data generated by the interface. The Hierarchical Data Format version 5 (HDF5) was chosen for its widespread use in

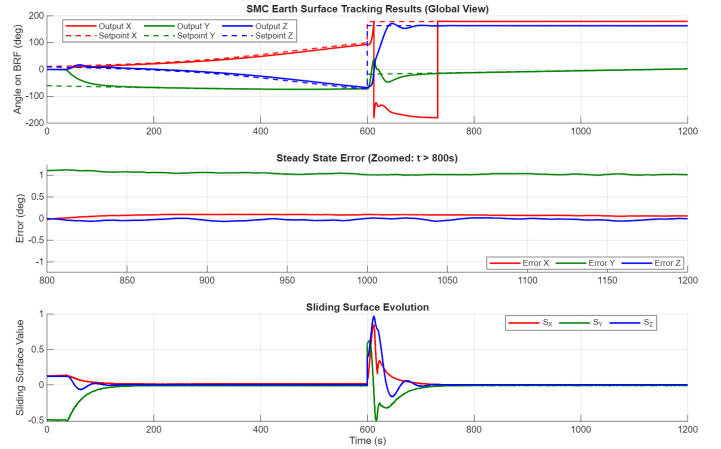


Fig. 14. Sliding Mode Control Earth Surface Tracking Simulation Results

high-performance projects, strong reliability, and capability to visualize complex data through its hierarchical structure. The HDF5 database is an essential component of the overall simulation workflow, providing data storage for every mission created within the interface and supplying the necessary information to both the interface and MATLAB/Simulink. A diagram that illustrates this workflow can be seen in Fig. 15.

After initializing the interface software, users can create a new mission or load an existing one from the HDF5 database. When creating a mission, all interface aspects can be modified. If loading a mission, users select from existing datasets, which populate the interface. Users can edit the content as needed, and when saving, the mission data is created or overwritten in the HDF5 database.

Once the mission's configuration is complete, the user can choose to run the simulation. When the run button is pressed, the first step is to save and store the current interface data in the database, regardless of whether it has been saved previously. After that, the corresponding mission data is transferred from the database to the MATLAB workspace in a structured and organized manner, allowing it to be accessed and used by the simulation in Simulink.

After the simulation is completed, all the designated output data is logged back into the MATLAB workspace. This allows for a programmatic structuring and organization of the data, which can later be effectively visualized using the MATLAB-native visualization tool, Simulink Data Inspector (SDI). The visualization signals are categorized into attitude outputs, actuator outputs, sensor outputs, estimator outputs, controller outputs, and disturbance/environment data.

VII. CONCLUSIONS AND FUTURE WORK

The Sun Pointing PD control algorithm, combined with the simple angular rate estimator, effectively aligned one of the satellite's axes towards the direction of the Sun during safe mode operation. Despite the significant limitations of the estimator due to the operational constraints, the steady-state pointing error was remarkably low.

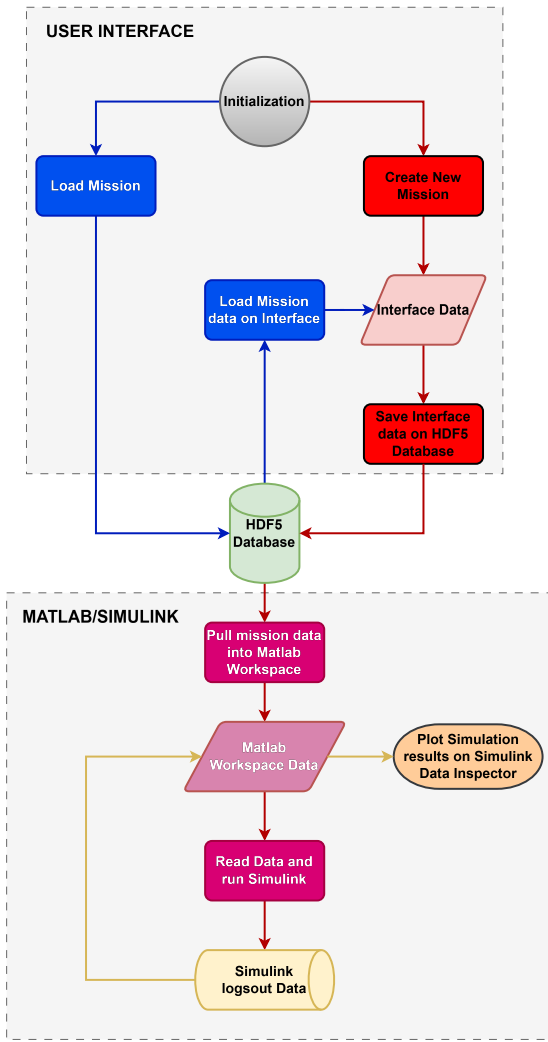


Fig. 15. Simulation data management workflow diagram

The proposed Sliding Mode Controller effectively controls CubeSat attitude in Low Earth Orbit, achieving desired orientations across various reference cases with acceptable convergence times and minimal steady-state error. Each attitude reference mode accurately defined targets in their respective frames, providing reliable inputs for the simulations.

The data management workflow effectively translates the user's mission configuration into simulations on MATLAB/Simulink, allowing for full customization. The HDF5 database maintains data integrity, serving as the link between the user interface and MATLAB/Simulink, simplifying data handling and storage. At the end, the SDI provides a user-friendly visualizer that displays clear, modular signals from the simulation results.

For future projects, an air bearing testbed is being developed at the laboratory at USP. This testbed aims to allow ground testing of Attitude Determination and Control Systems in a near-frictionless environment, thereby validating both hardware and control strategies. The first test subject will be the Cubli CubeSat, which was designed and documented by Fabio

Bobrow at [11].

Moreover, a specialized team at USP is developing an ADCS printed circuit board (PCB) to conduct ground tests on hardware and implement hardware-in-the-loop simulations using the simulator introduced in this work, complementing the necessary validation tests before mission launch.

ACKNOWLEDGMENTS

This study was financed by the Conselho Nacional de Desenvolvimento Científico e Tecnológico – Brasil (CNPq) – Process Number 407381/2022-8 and 350091/2024-2.

The authors would like to thank all current and former members of the project team whose contributions supported the development of this work, including Karoline Carvalho Burger, Cauê Garcia Menegaldo, David Arruda de Sousa, Lucas Wu Jiajun, Luca Augusto Paniago, Thallysson Maldonado Ramos, and Lucas Franco da Silva.

REFERENCES

- [1] The CubeSat Program, Cal Poly SLO, "CubeSat Design Specification (1U – 12U)," Rev. 14.1, San Luis Obispo, CA, Feb. 2022. [Online]. Available: <https://www.cubesat.org/>
- [2] S. A. Jacklin, "Small-satellite mission failure rates," NASA Technical Memorandum NASA/TM–2018–220034, 2018.
- [3] M. Langer and J. Bouwmeester, "Reliability of CubeSats – Statistical Data, Developers' Beliefs and the Way Forward," in Proc. 30th Annu. AIAA/USU Conf. Small Satellites, Logan, UT, USA, 2016.
- [4] M. Swartwout, "The first one hundred CubeSats: a statistical look," Journal of Small Satellites, vol. 2, no. 2, pp. 213–233, 2013.
- [5] C. G. Menegaldo, "Simulador de controle de atitude e propagação de órbita aplicado a nanossatélites em órbita baixa terrestre: desenvolvimento e validação com dados de voo do nanossatélite PicSat," M.S. thesis, Escola Politécnica, Univ. de São Paulo, São Paulo, Brazil, 2020.
- [6] V. Lapeyrière et al., "PicSat: a CubeSat mission for exoplanetary transit detection in 2017," in Proc. 31st Annu. AIAA/USU Conf. Small Satellites, Logan, UT, USA, Aug. 2017, paper SSC17-III-09.
- [7] B. G. Thomsen and J. Nielsen, "CubeSat Sliding Mode Attitude Control: Developing Testbed for Verification of Attitude Control Algorithms," M.S. thesis, Aalborg Univ., Aalborg, Denmark, 2016.
- [8] S. Debois and F. Viaud, "A Comparative Analysis of All-Magnetic Attitude Control Techniques for Nanosatellites Sun Acquisition," in Proc. 31st Annu. AIAA/USU Conf. Small Satellites, Logan, UT, USA, Aug. 2017, paper SSC17-P1-10.
- [9] F.-K. Yeh, "Sliding-mode adaptive attitude controller design for spacecrafts with thrusters," IET Control Theory & Applications, vol. 4, no. 7, pp. 1254–1264, Jul. 2010.
- [10] J.-J. E. Slotine and W. Li, Applied Nonlinear Control. Englewood Cliffs, NJ, USA: Prentice Hall, 1991.
- [11] F. Bobrow, "The Cubli: modeling and nonlinear attitude control utilizing quaternions," Ph.D. dissertation, Esc. Politécnica, Univ. de São Paulo, São Paulo, Brazil, 2022.