ATTITUDE DETERMINATION AND CONTROL SUBSYSTEM (ADCS) FOR JNANAM

Table of Contents

ADCS Requirements 4
Physical Architecture 5
Hardware 5
Sensors 5
Actuators 6
Positions and Orientation 9
Sensors 10
Actuators 10
ADCS mode definitions 11
Mode descriptions 11
Safe Mode 11
Stand-by mode 11
Detumbling mode 11
Nominal mode 11
Attitude Profiles 12
Triggers 13
ADCS mode sequence 14
Error Metric 15
Pointing Error 15
Absolute Knowledge Error 15
Absolute Performance Error 15
Mean Knowledge Error 15
Mean Performance Error 15
Relative Knowledge Error 15
Relative Performance Error 16
Drift Performance Error 16
Sensor Error 16
Magnetometer detumbling 16
Magnetometer nominal 16
Coarse sun sensor 16
Gyroscope noise 17
Gyroscope bias 17
AOCS Control and determination algorithms 18
Orbit Propagator 18
On-board ephemeris 18

Sun position 18
Eclipse 19
Sensor processing algorithms 21
Coarse sun sensor filtering 21
Initial bias estimation 22
Sampling frequency 25
Inputs 25
Outputs 25
Estimation, guidance and control algorithms 25
Actuator mapping and command processing algorithms 27
Simulations 31
Introduction 31
Simulation model 31
Reference Frames 31
Transformation matrices 34
Environmental models and assumptions 36
Sensor model 39
Actuator model 41
Analyses 43
Gain Tuning 43
Simulation Performance campaign plan & results 47
Nominal scenarios 47
Variables Table 51

ADCS Requirements

Req ID	**Requirement**
ADC- 1	The ADCS shall provide knowledge of the orientation of the spacecraft relative to the Earth.
ADC- 2	The ADCS shall provide knowledge of the angular motion of the spacecraft about Centre of Mass.
ADC- 3	The ADCS shall provide rate and position control of all axes of the spacecraft.
ADC- 4	The ADCS should be capable of pointing the -Z face and the +Y face at the specified ground target.
ADC- 5	The ADCS shall be capable of recovering from a tip off rate of no more than 5°/sec within 6 hours
ADC- 6	The ADCS shall be capable of placing science targets within the field of view of the spectrometer and dosimeter while up to 25° off NADIR
ADC- 7	The ADCS shall be capable of tracking given an Earth based coordinate in latitude and longitude.
ADC-8	The ADCS shall be capable of rolling about the Z-Axis to point the solar panels at the sun
ADC- 9	The ADCS shall be capable of pointing during all points and times of an orbit (eclipse operations)
ADC- 10	The ADCS shall track the sun with an error less than 15 deg

Physical Architecture

Hardware

The ADCS Physical Architecture involves Magnetometers, Gyroscope, Sun sensors, Temperature sensor and Magnetorquers as actuators.

Sensors

Magnetometers

The BMX160 has a 3-axis geomagnetic sensor which is used in our ADCS unit.

Key parameters	
Device Resolution (at T_A=25∘C)	0.3 µT
Sensitivity Temperature Drift	±0.01 %/K
Zero-B Offset (at T_A=25∘C)	±40 µT
Output Noise (Regular Preset)	0.6 µT
Output Data Rate (ODR)	12.5 Hz
Current Consumption	660 µA

Gyroscope

The BMX160's 16-bit digital triaxial gyroscope offers selectable ranges (±125°/s to ±2000°/s), low power consumption (850 µA in full operation mode), and a fast start-up time of 10 ms.

Key parameters	
Range (Selectable via serial digital interface)	±125°/s to ±2000°/s
Nominal Sensitivity	262.4 LSB /°/s
Sensitivity Temperature Drift	±0.02 %/K
Zero-Rate Offset	±3 °/s
Zero-Rate Offset change over Temperature	**±0.05 °/s/K**
Output Noise	0.007 (°/s/√Hz
Output Data Rate (ODR)	25 to 3200 Hz
Current Consumption	850 µA
Cross Axis Sensitivity	2 %
Bias stability	3 °/h

Sun sensors

The coarse sun sensor is comprised of photodiodes integrated on each solar panel.

Key parameters	
Field of View	± 60°
Dimensions	4 x 2 x 1.05
Radiant sensitive area (in mm2)	0.27
Breakdown voltage	Min 16 V

Power dissipation	100 mW

Temperature sensor

The BMX160 includes a temperature sensor with data accessible via registers 0x20-0x21

Key parameters	
Temperature Range	-40 to 85 °C
Temperature Sensor slope	0.002 K/LSB
Temperature Sensor offset	±2 K
Output Data Rate (ODR) (Gyro active)	100 Hz
Resolution (Gyro active)	16 bit

Actuators

Magnetorquers

Magnetorquers are actuators that generate a magnetic moment that interacts with the Earth's magnetic field creating a torque enabling satellite in-orbit maneuvering. There are several types of magnetic torquers available, Torquer and Air coil are used for the mission.

Air Coil Magnetic Torquer:

Air coil is a type of actuator which is made from self-bonding wire and is very small height but large outer dimensions in comparison to magnetorquers. They don't feature iron cores and are therefore less complex in means of attitude control algorithms. These Air coils are made of multiple loops of self-bonding wire without a core. Based on coil, and a set of geometric constraints, it is the attitude determination and control system (ADCS) engineer's responsibility to find a coil that fits the specific requirements of a CubeSat mission. Air coil comprises dimensions of 95.86x90.14x20mm. Air coil helps in stabilization of the CubeSat which achieves detumbling of the CubeSat after injection into the orbit. The dipole moment of the rod is 0.33Am².

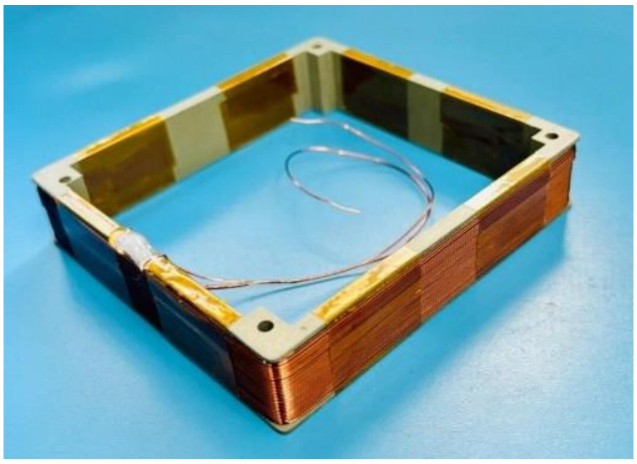


Figure: Air Coil Magnetic Torquer

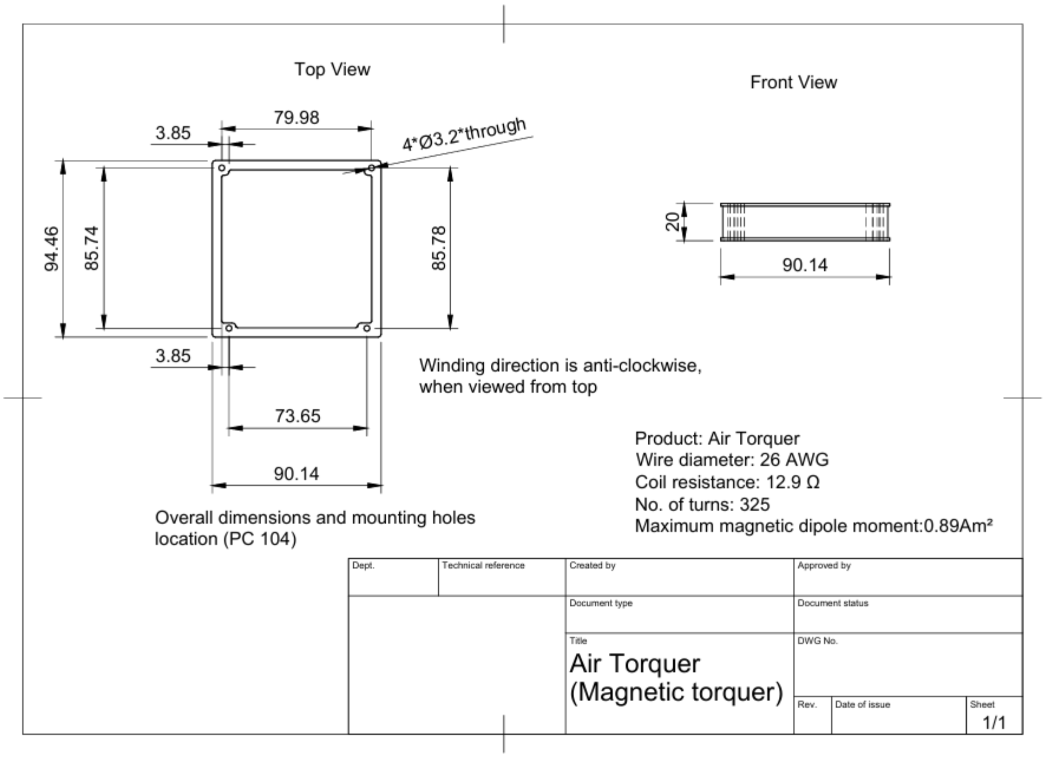


Figure: Air Coil Magnetic Torquer dimension

Properties	Torquer
Dimension of MT	94.46 x 90.14 x 20
Magnetic Permeability core	1200
Max current rating	400mA
Max magnetic dipole moment	0.33Am^2
Voltage	3.3V
Weight	123 grams
Operational Temperature	-40 to 120C

Magnetic Torquer:

Magnetorquers are used in controlling the attitude of a spacecraft either directly, by interacting with the local Earth's magnetic field. Magnetorquer use a magnetic alloy rod which produces an amplification effect over an air-cored magnetorquer. Each rod is typically bifilar wound for redundancy, or the windings can be activated together to increase the torque produced. While drive circuits for the rods can be supplied if required, they typically run directly between a switched power output and the on-board power control system. The Magnetic torquer dimensions comprises of 16x16x61.5mm. These torquers utilize a core with high magnetic permeability of 1200 with maximum current rating of 400mA. The dipole moment of the rod is 0.3Am^2.

Figure: Magnetic Torquer

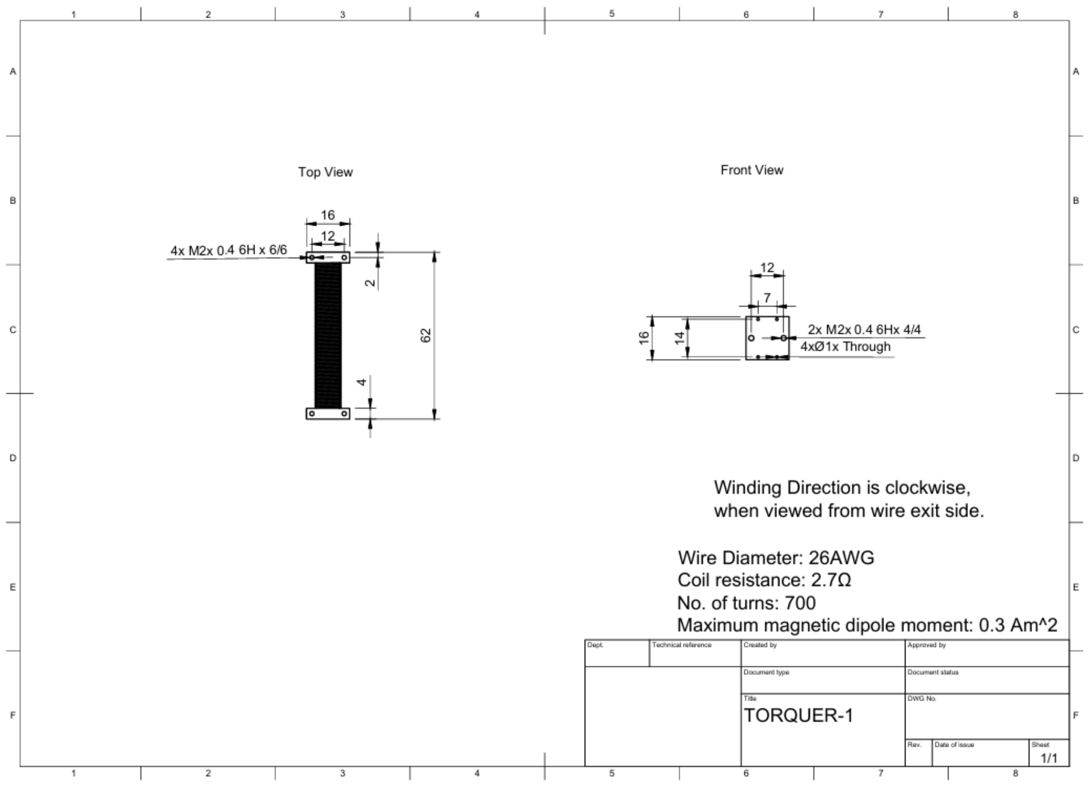


Figure: Magnetic Torquer dimension

Properties	Torquer
Length of MT	60mm
Diameter of MT	13.89/14.63
Dimension of MT	62 x 16 x 16
Magnetic Permeability core	1200
Max current rating	400mA
Max magnetic dipole moment	0.3 Am^2

Voltage	3.3V
Weight	80gms
Connector	2 Pin molex pico blade
Operational Temperature	-40 to 120C
End Cap Material	PEEK

Positions and Orientation

The placement position of all sensors and actuators is defined relative to the satellite body reference frame, as shown in Fig.

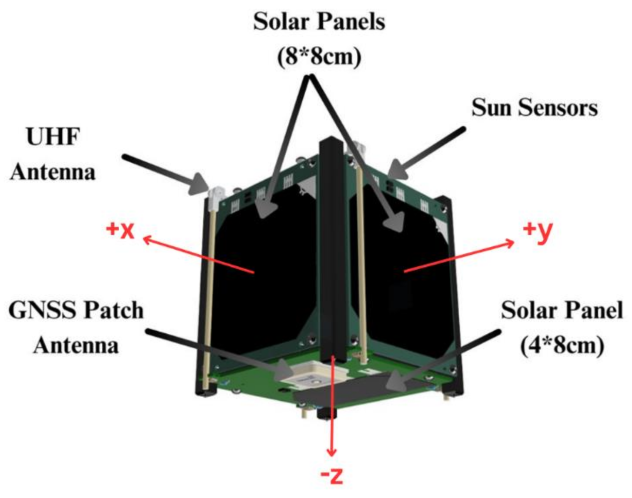


Figure: Satellite body reference frame

The position of the ADCS/OBC board and Magnetorquers Board is presented in Figure and the place where the GNSS path antenna is placed is presented in Figure .

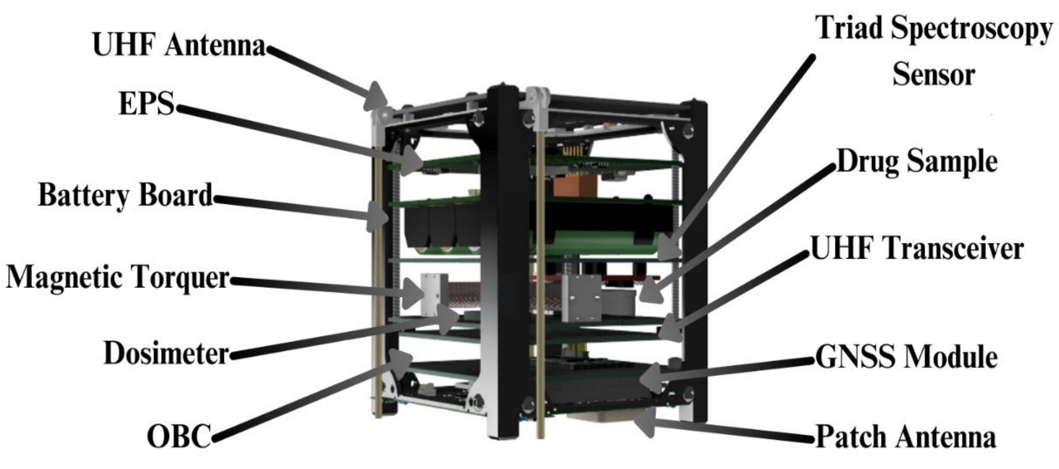


Figure : JNANAM Subsystem

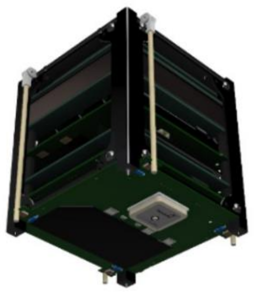


Figure 4: View of GNSS path antenna position in JNANAM

Sensors

Magnetometers

The BMX160 is placed on the OBC PCB. The Coil Sensors X, Y, Z of the magnetometer are aligned to the corresponding axis of the satellite body frame.

Gyroscope

The BMX160 is placed on the OBC PCB. Its X, Y, Z axes are aligned to the corresponding ones of the satellite body frame.

Temperature Sensors

The BMX160 (temperature sensors) are installed on the OBC PCB. Their X,Y,Z axes are aligned to the corresponding ones of the satellite body frame.

Coarse Sun Sensor

The coarse sun sensor is comprised of photodiodes integrated on each solar panel PCB as shown in figure. There are two photodiodes on the top part of the PCB one for redundancy.

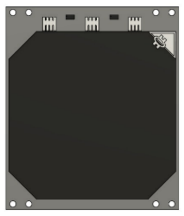


Figure: Solar panel PCB

Actuators

Magnetorquers

The Magnetorquers are placed on the ADCS board. The X, Y, Z axes of the Magnetorquers are aligned to the corresponding one of the satellite body frame.

ADCS mode definitions

In order to satisfy the ADCS requirements the following operating modes were agreed upon:

Table : *ADCS modes*

ADCS modes
Safe Mode
Stand-by mode
Detumbling mode
Nominal mode

Mode descriptions

In this section each ADCS mode's purpose is outlined. Descriptions are accompanied by the respective mode's control cycle.

Safe Mode

Both sensors will be off. System will go into safe mode when the battery level is below a set threshold.

Stand - by mode

The purpose of ADCS stand-by mode is to be a stable state for ADCS to fall back on in case of failure or whenever ADCS needs to be disabled. The attitude is estimated based on sensor's data, but no attitude controls is performed.

Detumbling mode

The purpose of the ADCS detumbling mode is to autonomously dissipate and control the satellite's angular velocity. This is achieved with a B-dot controller directly using measurements from the high precision 3-axis magnetometer. Actuation is performed solely by the magnetorquers.

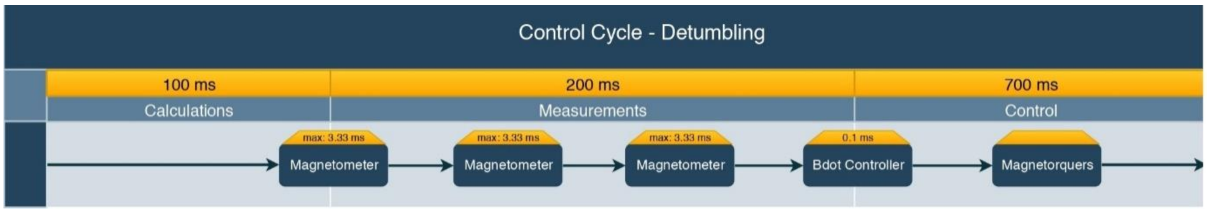


Figure : Detumbling mode control cycle

Nominal mode

The purpose of the ADCS nominal mode is nadir-pointing according to the requirements. Nadir-pointing enables the reliable and continuous data transmission via the UHF Antenna to the GS. Nadir-pointing is achieved by determining and controlling the satellite's attitude.

Attitude is determined by combining sensor measurements from the magnetometer, the sun sensors and the gyroscope. Attitude control is carried out by the magnetorquers.

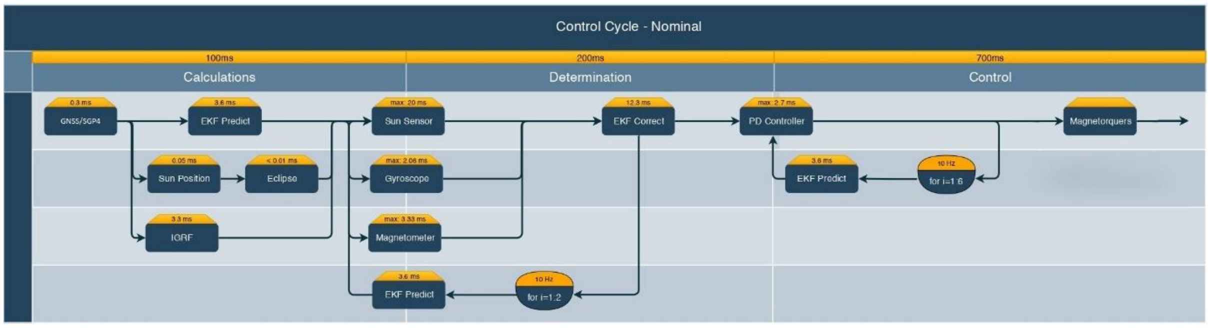


Figure : Nominal mode control cycle

Attitude Profiles

ADCS MODE	Attitude Profile
Safe Mode	O-01
Stand-by mode	S-01
Nominal mode	N-01
Detumbling mode	D-01

E-01

This is the default attitude profile of ADCS safe mode. No Sensor measurements is performed. No actuation is performed.

S-01

This is the default attitude profile of ADCS stand-by mode. Sensor measurements is done. No actuation is performed.

N-01

This is the default attitude profile of ADCS nominal mode. Attitude determination is accomplished combining measurements from the high-precision magnetometer, the sun sensor and the gyroscope. The satellite orbit is propagated using Simplified General Perturbations 4 (SGP4), which is frequently initialized by a Two-Line Element Sets (TLEs) file. Actuation is carried out using the magnetorquers. The controller used is a PD with a set of 1 constant gains.

D - 01

This is the default attitude profile of ADCS detumbling mode. No attitude determination is taking place. The only sensor measurements are from the high-precision magnetometer. Actuation is performed by the magnetorquers. ADCS makes use of a B-dot controller, which is directly fed the magnetometer measurements.

Triggers

ADCS can switch modes based on the system-level state of JNANAM, and also on interrupts, and watch dog timers.

All transitions are implemented as follows:

Detumbling Mode → Nominal Mode: ADCS switches to ADCS nominal mode when the angular rate of the satellite, which is approximated by the B-dot metrics, decreases to a rate lower than the 0.035 rad/sec on each satellite body frame axes threshold and all corresponding diagnostic checks have finished successfully.

Nominal Mode → Detumbling Mode: ADCS switches to ADCS detumbling mode when the angular rate of the satellite exceeds the threshold rate of 0.087 rad/sec on any satellite body frame axis.

Detumbling Mode → Stand - by Mode: ADCS switches to ADCS stand-by mode either when the corresponding diagnostic checks have failed.

Nominal Mode → Stand - by Mode: ADCS switches to ADCS stand-by mode either when the corresponding diagnostic checks have failed.

Stand - by Mode → Detumbling Mode: ADCS switches to ADCS detumbling mode when the corresponding command from the OBC unit is received, or if the magnitude of the angular rate of the satellite is higher than the 2.72 rad/s threshold, provided that no fault in detumbling mode has been detected.

(Detumbling Mode / Stand - by Mode/ Nominal Mode) →Safe Mode: ADCS switches from any mode to safe mode when the battery level is below a set threshold.

Approximation of Angular Velocity using B - dot metrics

Additionally, during ADCS detumbling mode, where B-dot is enabled, the angular velocity of the satellite needs to be defined in order for the angular velocity threshold, which triggers the transition from ADCS detumbling mode to ADCS nominal mode, to be determined regularly. However, since no sensors, apart from magnetometers, are active in ADCS detumbling mode, the angular velocity of the satellite needs to be approximated utilizing the B-dot metrics. The aforementioned calculation is performed according to Equation (1).

$$\boldsymbol{b} \times \boldsymbol{\omega} = \dot{\boldsymbol{b}} \longrightarrow \text{skew}(\boldsymbol{b}) \cdot \boldsymbol{\omega} = \dot{\boldsymbol{b}} \qquad \textbf{(1)}$$

where $\boldsymbol{b}$ is the magnetic field expressed on the body frame of the satellite and $\boldsymbol{\omega}$ is the angular velocity of the satellite body from the orbit to the body frame, expressed in the body frame. Therefore, if the skew($\boldsymbol{b}$) is an invertible matrix:

$$\boldsymbol{\omega} = \text{skew}(\boldsymbol{b})^{-1} \cdot \dot{\boldsymbol{b}} \qquad \textbf{(2)}$$

However, the skew($\boldsymbol{b}$) is close to singular, thus MATLAB is unable of performing the operation described in Equation (2). Therefore, the angular velocity can only be approximated, as shown in Equation (3)

$$\boldsymbol{\omega} = \frac{\text{skew}(\boldsymbol{b}) \cdot (-\dot{\boldsymbol{b}})}{\boldsymbol{b}T\boldsymbol{b}} \qquad (3)$$

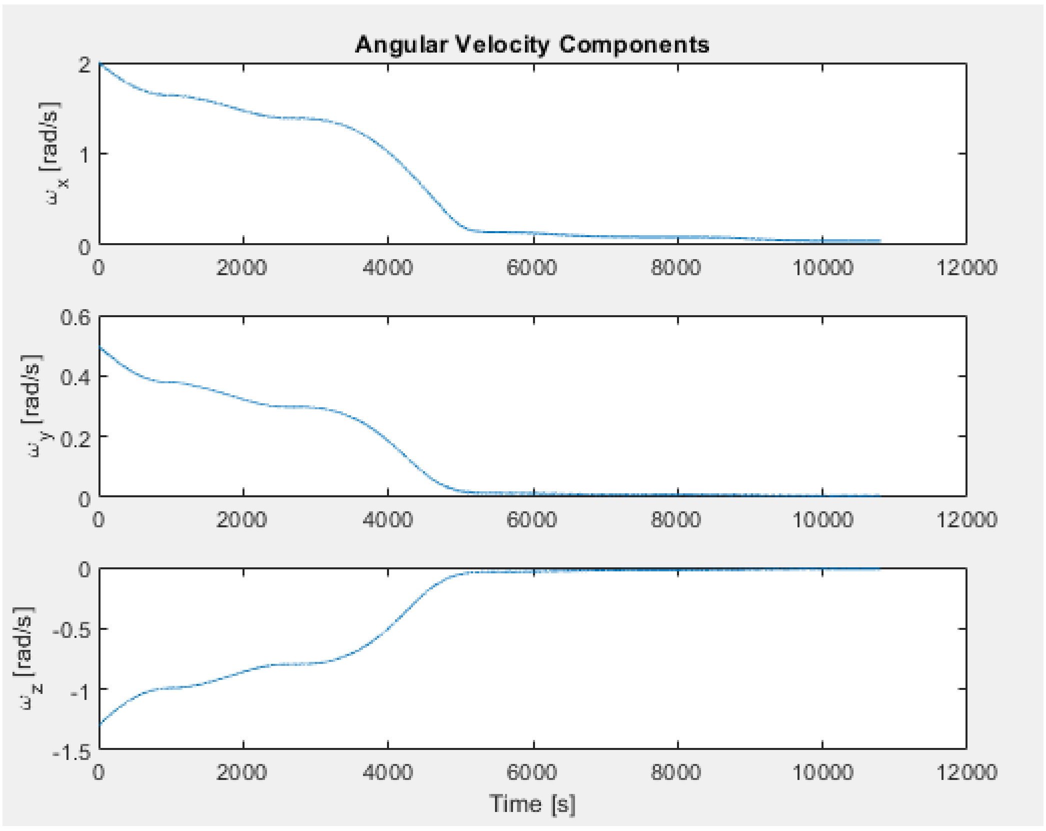


Figure : *Angular velocity during AOCS Detumbling Mode*

ADCS mode sequence

Spacecraft Commissioning mode: ADCS initially operates in stand-by mode. Upon command from OBC, ADCS switches to detumbling mode.

Spacecraft Nominal mode: ADCS initially operates in detumbling mode. If Detumbling → Nominal trigger is activated ADCS switches to nominal mode. If Nominal → Detumbling trigger is activated ADCS switches to detumbling mode.

Error Metric

Pointing Error

In order to derive the AOCS pointing budget, the following errors are calculated:

- Absolute Knowledge Error
- Absolute Performance Error
- Mean Knowledge Error
- Mean Performance Error
- Relative Knowledge Error
- Relative Performance Error
- Drift Performance Error

Absolute Knowledge Error

Absolute knowledge error is the instantaneous value of the knowledge error at any given time. It is expressed as ek(t). Confidence intervals were calculated by the bootstrap method.

Absolute Performance Error

Absolute performance error is the instantaneous value of the performance error at any given time. It is expressed as ep(t). Confidence intervals were calculated by the bootstrap.

Mean Knowledge Error

Mean knowledge error is the mean value of the knowledge error over a specified time interval. The time intervals are chosen one after the other and are equal to Δt = 500 sec.

$$MKE(\Delta_t) = \frac{1}{\Delta_t}\int_{\Delta_t} e_k(t)dt \tag{13}$$

Mean Performance Error

Mean performance error is the mean value of the performance error over a specified time interval. The time intervals are chosen one after the other and are equal to Δt = 500 sec.

$$MPE(\Delta_t) = \frac{1}{\Delta_t}\int_{\Delta_t} e_p(t)dt \tag{14}$$

Relative Knowledge Error

Relative knowledge error is the difference between the instantaneous knowledge error at a given time, and its mean value over a time interval containing that time. The time intervals chosen are the ones chosen in the mean knowledge error, Δt = 500 sec.

$$RKE(t, \Delta_t) = e_k(t) - \frac{1}{\Delta_t}\int_{\Delta_t} e_k(t)dt \qquad (15)$$

Relative Performance Error

Relative performance error is the difference between the instantaneous performance error at a given time, and its mean value over a time interval containing that time. The time intervals chosen are the ones chosen in the mean performance error, Δt = 500 sec.

$$RPE(t, \Delta_t) = e_p(t) - \frac{1}{\Delta_t}\int_{\Delta_t} e_p(t)dt \qquad (16)$$

Drift Performance Error

Drift performance error is the difference between the means of the performance error taken over two time intervals within a single observation period.

$$DPE(\Delta_{t_1}, \Delta_{t_2}) = \frac{1}{\Delta_{t_1}}\int_{\Delta_{t_1}} e_p(t)dt - \frac{1}{\Delta_{t_2}}\int_{\Delta_{t_2}} e_p(t)dt \qquad (17)$$

where ep(t) is the performance error. For SSO the time periods were selected as Δt1 = Δt2 = 250 sec.

We can observe that during eclipse significantly large knowledge errors occur, due to the lack of measurements from the sun sensors. Consequently, the performance errors are skewed for the duration of each eclipse and for a short time period immediately after.

Sensor Error

Magnetometer detumbling

During detumbling AOCS mode, magnetometer measurements are acquired and utilized by the controller in order to create the desired torque and dissipate the satellite's angular velocity. It is evident that the larger the noise to the magnetometer measurements, the more difficult it is to command the desired torque.

Magnetometer nominal

During nominal AOCS mode, magnetometer measurements are acquired and utilized by the MEKF in order to estimate the satellite's attitude. The larger the noise to the magnetometer measurements, the more difficult it is to estimate correctly the satellite's attitude.

Coarse sun sensor

During nominal AOCS mode, sun sensor measurements are acquired and utilized by the MEKF in order to estimate the satellite's attitude. It is evident that the larger the noise to the sun sensor measurements, the more difficult it is to estimate correctly the satellite's attitude.

Gyroscope noise

During nominal AOCS mode, gyroscope measurements are acquired and utilized by the MEKF in order to estimate the satellite's attitude. It is evident that the larger the noise to the gyroscope measurements, the more difficult it is to estimate correctly the satellite's attitude.

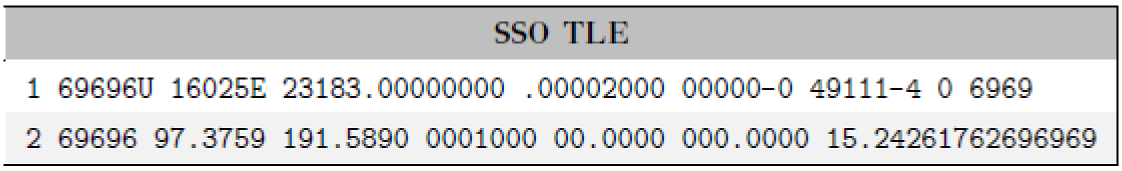
Gyroscope bias

During nominal AOCS mode, gyroscope measurements are acquired and utilized by the MEKF in order to estimate the satellite's attitude. It is evident that the larger the bias drift to the gyroscope measurements, the more difficult it is to estimate correctly the satellite's attitude.

AOCS Control and determination algorithms

Orbit Propagator

JNANAM AOCS is required to determine the satellite's position and velocity autonomously. The GNSS (NOVATEL OEM 7700) has been chosen to accomplish this task. In order to use the SGP4 propagator in the simulation’s environment, TLE files were created for simulation purposes based on the potential orbit of JNANAM.



SSO TLE
1 69696U 16025E 23183.00000000 .00002000 00000-0 49111-4 0 6969
2 69696 97.3759 191.5890 0001000 00.0000 000.0000 15.24261762696969

Figure : *SSO TLE, 6 LTAN, 500km*

On-board ephemeris

Sun position

.AcubeSAT's AOCS is required to determine the sun's position relative to the spacecraft autonomously. The ephemeris modeling as detailed in [bib] was used for the AOCS. This modeling is selected due to its relatively high precision and low computational load. Software for the ephemeris modeling can be directly accessed https://www.celestrak.com/software/vallado-sw.php.

Analytically, the corresponding algorithm in order to calculate sun's position is:

```
Algorithm 1: Sun position
$T_{UT1} = \frac{JD_{UT1} - 2451545.0}{36525}$;
$\lambda_{M_\odot} = 280.460^\circ + 36000.771 \cdot T_{UT1}$;
LET $T_{TDB} \cong T_{UT1}$;
$M_\odot = 357.5291092^\circ + 35999.05034 \cdot T_{TDB}$;
$\lambda_{ecliptic} = \lambda_{M_\odot} + 1.914666471^\circ \cdot \sin(M_\odot) + 0.019994643 \cdot \sin(2 \cdot M_\odot)$;
$r_\odot = 1.000140612 - 0.016708617 \cdot \cos(M_\odot) - 0.000139589 \cdot \cos(2 \cdot M_\odot)$;
$\epsilon = 23.439291^\circ - 0.0130042 \cdot T_{TDB}$;
$r_\odot^{ECI} = \begin{bmatrix} r_\odot \cdot \cos(\lambda_{ecliptic}) \\ r_\odot \cdot \cos(\epsilon) \cdot \sin(\lambda_{ecliptic}) \\ r_\odot \cdot \sin(\epsilon) \cdot \sin(\lambda_{ecliptic}) \end{bmatrix} AU$;
$\alpha_\odot = \arctan(\cos(\epsilon) \cdot \tan(\lambda_{ecliptic}))$;
$\delta_\odot = \arcsin(\sin(\epsilon) \cdot \sin(\lambda_{ecliptic}))$;
```

Sampling frequency

The sun position model will be operated during nominal AOCS mode once every 1 second during the calculations part.

Inputs

The sun ephemeris requires the following inputs:

- **Time** in JD format for which the calculation will take place.

Outputs

The sun ephemeris returns the following outputs:

- **Sun position:** in the ECI frame for the specified epoch.

Eclipse

While in orbit, the satellite may enter eclipse, this phenomenon has to be calculated in order to define whether or not sun sensor measurements are credible at any moment. The eclipse modelling as developed by Vallado in [bib] and [bib] was used for the AOCS. This is implemented in the simulation’s environment and, in the future, may be implemented on-board for further sanity checks to the sensor's measurements.

Analytically the corresponding algorithm in order to calculate eclipse is:

Algorithm 2: Eclipse

```
$x_1 = \frac{R_e AU}{R_s + R_e}$;
$\alpha_1 = \pi - \arccos\left(\frac{R_e}{x_1}\right) - \arccos\left(\frac{R_e}{|\mathbf{r}|}\right)$;
$x_2 = \frac{R_e AU}{R_s - R_e}$;
$\alpha_2 = \arccos\left(\frac{R_e}{x_2}\right) - \arccos\left(\frac{R_e}{|\mathbf{r}|}\right)$;
$\alpha = \pi - \arccos\left(\frac{\mathbf{r}_s \cdot \mathbf{r}}{|\mathbf{r}_s| \cdot |\mathbf{r}|}\right)$;
if $\alpha_2 < \alpha < \alpha_1$ then
    $S = Penumbral$;
if $\alpha < \alpha_2$ then
    $S = Umbral$;
else
    $S = No \quad Eclipse$;
```

Sampling frequency

The eclipse calculation model will be operated during nominal AOCS mode once every 1 second during the calculations part.

Inputs

The eclipse model requires the following inputs:

- **Satellite position:** in the ECI frame.
- **Sun position:** in the ECI frame.

Outputs

The eclipse model returns the following outputs:

- **Eclipse:** which defined whether the satellite is in umbral, penumbral or non-eclipse conditions.

5.3 Magnetic field model

JNANAM's AOCS is required to determine the magnetic field's vector autonomously. To determine this, the International Geomagnetic Reference Field (IGRF) model is used.

Sampling frequency

The magnetic field model will be operated during nominal AOCS mode once every 1 second during the calculations part.

Inputs

The magnetic field requires the following inputs:

- **Time:** for which the calculation will take place.
- **Latitude:** of the satellite for the specified epoch.
- **Longitude:** of the satellite for the specified epoch.
- **Altitude:** of the satellite for the specified epoch.
- **Coordinate system used:** for our integration it is set to 'geodetic'.

Outputs

The magnetic field returns the following outputs:

- **Magnetic field:** in the NED frame for the specified epoch.

Sensor processing algorithms

The control cycle of each mode requires measurements with a frequency of 10 Hz. However, each sensor has its own internal sampling frequency which is much higher than 10Hz. Thus, an algorithm for the handling of measurements must be chosen. Two options are considered here, the final decision being left for a later stage:

- Measurements are used as-is whenever required by the active mode's control cycle. The rest are discarded.
- Between two consecutive control cycle timesteps, all measurements are kept. The median value of some measurements closest to the next timestep is calculated and fed to the active controller as the measurement for the next timestep. This calculation filters out random spikes in sensor measurements.

Coarse sun sensor filtering

The CSS output consists of photodiode currents on each side of the satellite. Currents

on opposite sides are added signed (+/-) and the sums are then used to create the sun position vector. The vector is then normalized.

Wahba’s problem

MEKF requires an initial state estimation in order to start attitude determination. The state vector is defined as follows:

$$\boldsymbol{x} = (\boldsymbol{q}_{1:4}, \boldsymbol{\omega}, \boldsymbol{\beta}) \qquad (18)$$

where $\boldsymbol{q}_{1:4}$ is the ECI to body quaternion, $\boldsymbol{\omega}$ is the angular velocity of the body frame relative to the ECI frame, expressed in the body frame, and $\boldsymbol{\beta}$ is the bias.

A random initial estimation would be inefficient and risky. Thus, a better initial estimate is acquired by solving Wahba's problem, which is stated as such:

Find the orthogonal matrix A with determinant +1 that minimizes the loss function

$$L(A) = 1\frac{1}{2}\sum_{vi=1}^{N} a_i \,||\boldsymbol{b}_i - A\boldsymbol{r}_i||\text{^}2 \qquad (19)$$

where $\boldsymbol{b}_i$ is a set of N unit vectors measured in a spacecraft’s body frame, $\boldsymbol{r}_i$ are the corresponding unit vectors in a reference frame, and a_i are nonnegative weights. In this specific case, the unit vectors are the sun and magnetic field unit vectors, in the body and ECI frames respectively. Both weights are equal to 1. A is the attitude matrix.

To solve Wahba's problem, the singular value decomposition (SVD) algorithm was selected. The entire procedure is presented below:

```
Algorithm 3: Attitude estimation with SVD
Result: Attitude matrix
1 $B = a_1 \cdot sun_{\text{body}} \cdot sun_{\text{eci}} + a_2 \cdot mgn_{\text{body}} \cdot mgn_{\text{eci}}$ ;
2 $[U, S, V] = svd(B)$ ;
3 $d = \det(U) \cdot \det(V)$ ;
4 if $d < 0$ then
5     for $i = 1 : 3$ do
6         $U(i, 3) = -U(i, 3)$;
7     end
8 $A = U \cdot V^T$;
```

Initial bias estimation

To increase the robustness of the bias estimation, an initial bias estimation is implemented which is fed to the MEKF. In order to calculate this, Wahba's problem is solved in *n* successive time-steps. By combining these, we can calculate $n - 1$ angular rates $\boldsymbol{\omega}_{k-1,k}$ with $k = 2, 3, ..., n$ using the following equation:

$$\omega(t) = 2\frac{dq(t)}{dt}q^{-1}(t) \qquad (20)$$

where $\boldsymbol{q}$ is the attitude quaternion. This is a continuous-time equation, however it can be discretized as such:

$$\boldsymbol{\omega_{k-1,k} = 2\frac{\Delta q}{\Delta t}q_k^{-1}} \qquad (21)$$

The quaternion derivative is calculated numerically on each time-step. The angular rates $\boldsymbol{\omega}_{k-1,k}$ are then time-averaged in order to obtain a more robust estimate:

$$\boldsymbol{\omega}_{mean} = \frac{\sum_{k=2}^{n}\boldsymbol{\omega_{k-1,k}}}{n-1} \qquad (22)$$

Finally the bias is calculated:

$$\hat{\boldsymbol{\beta}}_{\text{initial}} = \boldsymbol{\omega}_{\textbf{mean}} - \boldsymbol{\omega}_{\textbf{measured}} \qquad (23)$$

Multiplicative Extended Kalman Filter

For the online estimation of attitude and angular velocity, the Multiplicative Extended Kalman Filter (MEKF) has been selected, it is a modification of the standard Extended Kalman Filter. Instead of directly estimating the global attitude quaternion, it estimates the error between the real and estimated global attitude quaternions, and updates the latter accordingly. Two functions are needed for its operation:

· **State Function:** The state function calculates the next global state, consisting of the attitude quaternion and the gyroscope bias. Angular velocity is not propagated, as the satellite's dynamic model accuracy cannot be accurately characterized. Instead, the ground-truth bias-corrected gyro measurement is used (26).

· **Measurement Function:** The measurement function calculates measurements based on the estimated state. Magnetometer and sun sensor measurements are implemented as transformations of the reference vectors from ECI to body frame, which is calculated based on the aforementioned estimated state. More specifically:

$$\hat{b}_{meas} = \hat{q}^{-1} \otimes b \otimes \hat{q} \qquad (24)$$

where b is the magnetic induction in ECI, $\hat{q}$ is the estimated ECI to body attitude quaternion, $\hat{b}meas$ is the estimated magnetic induction in the body frame. The expected sun sensor measurement is acquired by transforming the ECI vector to body frame, and then applying the procedure mentioned paragraph 6.2.5.3, without including random noise.

MEKF uses of the following variables:

· **Covariance Matrix [*P*]:** The covariance matrix expresses the correlation of each state element w.r.t every state element.

· ***Q* & *R* Matrices:** The ***Q*** and ***R*** matrices express the uncertainties in the kinematic/ dynamic models and sensor measurements respectively.

· **Jacobian:** The state transition and measurement functions' Jacobian matrices are calculated analytically as per the following:

$$F(\hat{x}(t), t) = \begin{bmatrix} -[\hat{\omega}(t)\times] & -I_{3\times 3} \\ 0_{3\times 3} & 0_{3\times 3} \end{bmatrix}$$

$$H_k(\hat{x}_k^-) = \begin{bmatrix} [A(\hat{q}^-)r_1\times] & 0_{3\times 3} \\ \vdots & \vdots \\ [A(\hat{q}^-)r_n\times] & 0_{3\times 3} \end{bmatrix}$$

MEKF requires an initial state estimation before starting its operation. After initialization, it works in a constant cycle, outlined below. $f(x, u)$ is the transition function and Fk is its Jacobian, while in $h(x)$ is the measurement function and Hk its Jacobian.

Algorithm 4: Multiplicative Extended Kalman Filter Predict

```
1 while True do
2     $\hat{x}_{k+1|k} = f(\hat{x}_{k|k}, u_k)$ ;                // Predict the next state
3     $A = \begin{bmatrix} -F_k & Q_{6\times 6} \\ 0_{6\times 6} & F_k^T \end{bmatrix}$
4     $B = e^{A \cdot dt}$
5     $\Phi_k = B_{7:12,7:12}$
6     $Q_s = \Phi_k \cdot B_{1:6,7:12}$
7     $P_{k+1|k} = \Phi_k P_{k|k} \Phi_k^T + Q_s$ ;      // Predict the next error covariance
```

Algorithm 5: Multiplicative Extended Kalman Filter Correct

```
1 while True do
2     $K_k = P_{k|k-1} H_k (H_k P_{k|k-1} H_k^T + R_k)^{-1}$ ;     // Compute the Kalman gain
3     $\epsilon_k = K_k [z_k - h_k(\hat{x}_{k|k-1})]$ ;              // Calculate error state
4     $q_{error} = \begin{bmatrix} 1 \\ 0.5 \cdot \epsilon_k(1) \\ 0.5 \cdot \epsilon_k(2) \\ 0.5 \cdot \epsilon_k(3) \end{bmatrix}$ ;    // Calculate error quaternion assuming that
                                                         it is close to identity
5     $q_{k|k} = q_{k|k-1} \otimes q_{error}$ ;    // Update global quaternion with error state
6     $b_{k|k} = b_{k|k-1} + \epsilon_k(4:6)$ ;              // Update bias with error state
7     $q_{k|k} = q_{k|k} / norm(q_{k|k})$ ;                   // Normalize quaternion
8     $P_{k|k} = (I - K_k H_k) P_{k|k-1}$ ;                  // Correct error covariance
```

Bias model in MEKF

Bias is assumed to be constant during the predict phase of MEKF:

$$\dot{\hat{\beta}} = 0 \qquad (26)$$

The measurement function is modified to include bias when calculating expected measurements. Bias estimation is automatically updated on the correct phase of MEKF, based on the measurement residual.

$$\tilde{y}_k = z_k - h(\hat{X}_{k|k-1}) \qquad (27)$$

MEKF seeks to minimize the difference between the measurement function (which includes bias) and the ground truth measurements. When these match up, bias has been accurately estimated, thus the above calculation result is sufficient.

Albedo compensation

Provided that the Albedo model is sufficiently accurate, Albedo calculations are integrated inside the MEKF correct step.

Sampling frequency

The sensor processing algorithms are operated during nominal AOCS mode once every 0.1 second during the determination part.

Inputs

The sensor processing algorithms require the following inputs:

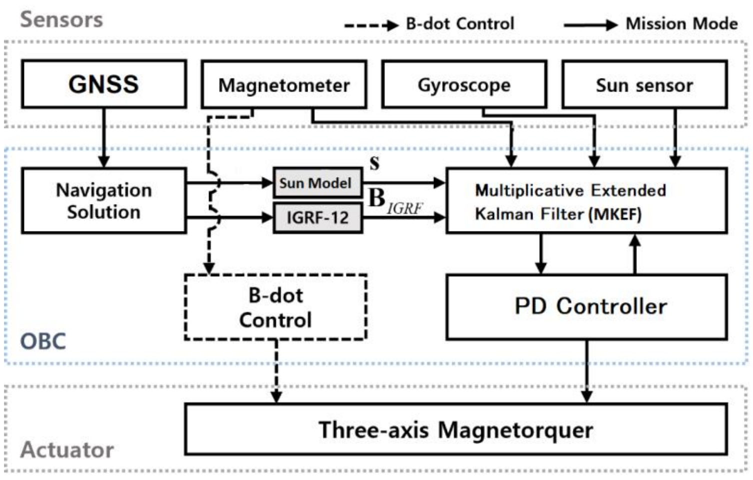
- **Magnetometer measurements**
- **Gyroscope measurements**
- **Sun sensor measurements**

Outputs

The sensor processing algorithms return the following outputs:

- **Quaternion:** ECI to body.
- **Angular velocity:** of the body frame with respect to the ECI frame, expressed in the satellite body frame.

Estimation, guidance and control algorithms



Detumbling mode

During AOCS detumbling mode, the angular velocity of the body frame relatively to the ECI frame, expressed in the satellite body frame, is dissipated. Since attitude determination, and specifically MEKF, is disabled during this mode, angular velocity cannot be estimated directly. In order to solve this issue, the derivative of the magnetic field, ˙ b, is used. This metric is considered to be a sufficiently proportional to the angular velocity. The controller used is calculated as:

$$m_i = -K.\dot{b}_i \qquad (28)$$

where *i* denotes as the index of each of the three axes and m is the desired magnetic moment applied to the magnetorquers.

On-board magnetic field measurements are acquired through the magnetometers, which subsequently are used to calculate ˙ b.

Nominal mode

During AOCS nominal mode, the satellite is performing **nadir pointing**. Initially, the GNSS is used to find the satellite position. These position data are used by the on-board ephemeris and IGRF models to calculate the sun position and magnetic field vectors respectively.

In order to transition from AOCS detumbling to nominal mode, an initial state estimation is required. This is implemented by solving Wahba's problem for sensor fusion, which, by acquiring magnetic field, sun position values and magnetometer and sun sensor measurements, can estimate the state quaternion. In addition to this, by acquiring consecutive quaternion estimations, we can estimate the state angular velocity. Finally, by combining this estimation with the gyroscope measurements, we can estimate the state bias, thus constructing the required initial state estimation for MEKF, entering nominal AOCS mode.

In nominal mode, the MEKF is operating normally both by correcting and predicting the state estimation. During attitude determination, magnetometer, sun sensor and gyroscope measurements are acquired. These measurements are used by the MEKF to improve its state estimation. Simultaneously, MEKF predicts the next state estimation, with a frequency of 10 Hz, which is used throughout the entirety of the control cycle. During actuation, based on the predicted state estimation, the desired control torque is calculated by a PD controller, with a frequency of 10 Hz. The commanded torque is subsequently applied to the actuators, achieving nadir pointing.

The PD controller requires the quaternion from the orbit to the satellite body frames and the angular velocity of the body frame with respect to the orbit frame expressed in the body frame. In order to acquire these parameters, the ECI to body quaternion is rotated with the Orbit to ECI quaternion, as produced by the corresponding rotation matrix:

$$q_{ob} = q_{oi} \otimes q_{ib} \qquad (29)$$

where ob, oi, ib depict the orbit to body, orbit to ECI, ECI to body frame rotations. And the required angular velocity is calculated as:

$$\omega_{b,ob} = \omega_{b,ib} - \omega_{b,io} \qquad (30)$$

where $\omega_{b,io}$ is $\begin{matrix} 0 \\ \omega_0 \\ 0 \end{matrix}$

Actuator mapping and command processing algorithms

Controllers

Our control design includes two controllers, one B-dot and one PD controller. The first one is used during the AOCS Detumbling Mode to dissipate the kinetic energy of the satellite and the second one during the AOCS Nominal Mode to perform position tracking.

B-dot Controller

B-dot control uses only magnetorquers, as actuators, and only magnetometer, as sensor. The controller is used to create, through magnetorquers, a magnetic dipole in the opposite direction to the derivative of the magnetic field vector *b*, estimated with magnetometers data:

$$m_i = -K \cdot \dot{b}_i \qquad (31)$$

where *i* denotes as the index of each of the three axes and $\dot{b}_i$ the temporal derivative of the geomagnetic field (***b***) with respect to this axis.

In order to calculate the $\dot{b}$, we subtract the temporal ***b***2 estimation with the previous ***b***1 estimation, given by the magnetometer, and the result is divided by the time-period elapsed between those two estimations, which is equal to 0.1 sec.

$$\dot{b} = \frac{b_2 - b_1}{0.1} \qquad (32)$$

Afterwards, the magnetic dipole ***m*** is scaled through the algorithm 6 .

```
Algorithm 6: Magnetic dipole scaling
Result: Final dipole given
1 for $i$ = 1:3 do
2     if $m_i > m_{max}$ then
3         $m_i = m_{max}$;
4     else if $m_i < -m_{max}$ then
5         $m_i = -m_{max}$;
6     else
7         $m_i = m_{initial}$
8 end
```

Finally, the torque given to be produced by the magnetorquers is calculated using the equation

$$\boldsymbol{\tau_m} = \boldsymbol{m} \times \boldsymbol{b} \qquad \textit{(33)}$$

PD Controller

PD control uses magnetorquers actuators and all sensors. The goal is to drive the actual quaternion, which depicts our current orientation, to the commanded and constant quaternion denoted by $\boldsymbol{q}_c$, as well as stabilizing the satellite upon reaching the desired orientation. The error quaternion is given by

$$\boldsymbol{\delta q} = \begin{bmatrix} \boldsymbol{\delta q} \\ \boldsymbol{\delta q 2:4} \end{bmatrix} = q_c^{-1} \otimes q \qquad (34)$$

The total $\boldsymbol{T}_{\text{commanded}}$ to be given through our actuators is calculated using the equation below:

$$\boldsymbol{T}_{\text{commanded}} = -\text{sign}\ (\delta q_1)\ K\text{p}\boldsymbol{\delta q}_{\boldsymbol{2:4}} - k_{\text{d}}\boldsymbol{\omega} \qquad \textbf{\textit{(35)}}$$

where k_{p} and k_{d} are gains matrices and $\boldsymbol{\omega}$ is the angular velocity from the satellite body frame to the ECI frame expressed on the body frame.

Sampling frequency

The B-dot controller will be operated during detumbling AOCS mode once every 0.1 sec during the controller B-dot part. The PD controller will be operated during nominal AOCS mode once every 0.1 sec during the controller PD part.

Inputs

The controllers require the following inputs:

- **Orbit to body quaternion**
- **Angular velocity of the satellite**: between orbit frame and body frame, expressed

  in the Body frame.
- **Magnetic field vector**: measured by the magnetometers, expressed in the body frame

Outputs

The controllers return the following outputs:

- **Torques**: to be produced by the magnetorquers, when the B-dot controller is enabled, or when the PD controller is enabled.

Control Torque allocation

The satellite is equipped with three magnetorquers, each one controlling an axis of the body frame, and a control system that provides the desired control torque, **u**, to be applied to the spacecraft.

The control vector, u, is decomposed into orthogonal and parallel components with respect to the orientation of the instantaneous magnetic field vector expressed in the satellite body frame:

$$u = u_{||} + u_{\perp} \tag{38}$$

The control torque $u_{||}$ cannot be applied to the system through the magnetorquers, since they produce torques that are perpendicular to the geomagnetic field vector.

The magnetic field vector is normalized to obtain b_hat = B_body / norm(B_body) , ensuring a unit vector for torque calculations.

Magnetic Torque (T_magnetic): The torque perpendicular to the magnetic field is assigned to the magnetorquers. The torque perpendicular to the magnetic field is computed using the projection matrix:

T_magnetic = skew(b_hat)' * skew(b_hat) * (T_commanded);

- skew(b_hat) is the skew-symmetric matrix of b_hat, used to compute the cross product.
- skew(b_hat)' * skew(b_hat) is a projection matrix that extracts the component of (T_commanded) orthogonal to b_hat, since magnetorquers can only produce torque perpendicular to the magnetic field (T_magnetic = M x B).

Magnetic Dipole (M): The magnetic dipole required to produce T_magnetic is calculated:

M = -cross(T_magnetic, B_body) / (norm(B_body))^2

- M = -cross(T_magnetic, B_body) / (norm(B_body))^2 solves for the dipole that produces T_magnetic given B_body.
- Subtracting known_rm' corrects for the satellite's residual magnetic dipole.

Magnetic Dipole Scaling: The computed Magnetic Dipole (M) is scaled to ensure it does not exceed the maximum capability of the magnetorquers

```
for $i$ = $1{:}3$ do
    if $m_i > m_{max}$ then
        $m_i = m_{max}$;
    else if $m_i < -m_{max}$ then
        $m_i = -m_{max}$;
    else
        $m_i = m_{initial}$
end
```

The effective magnetic torque is calculated using the real magnetic field:

```
T_magnetic_effective = cross(M, B_body_real);
```

- This ensures the torque reflects the actual magnetic field (B_body_real) rather than the estimated field (B_body).

Simulations

Introduction

For the purpose of presenting the most detailed and precise results possible, 1 distinct simulation is being used:

Total AOCS Simulation

With reference to the Nominal Simulation, the orbit propagator, IGRF and sun position models are utilized, in order to produce and combine sensor measurements, which are in turn used by MEKF. The results are fed to a PD controller to control the satellite.

Also, regarding the detumbling simulation, magnetometer measurements are acquired and utilized by the B-dot controller to calculate the desired torque. The torque is then commanded to the magnetorquers to dissipate the satellite angular velocity.

Results in this total simulation seem correct and sufficient. One issues has emerged from this simulation:

- **Significant error during eclipse:** During eclipse, due to not acquiring reliable coarse sun sensor data, the knowledge error is greatly increased. As a result, pointing during eclipse significantly deteriorates.

Simulation model

Reference Frames

In order to represent the kinematic and dynamic attitude of the satellite in a mathematical manner, while in orbit around the Earth, we need first to define a number of reference frames [Landis Markley].

Earth Centered Inertial

The Earth Centered Inertial Frame is an inertial reference frame with the axis origin placed at the center of Earth. The Z axis is aligned with the axis of rotation of Earth. The X axis follows the vernal equinox, while the Y axis completes the orthogonal coordinate system, using the right hand rule.

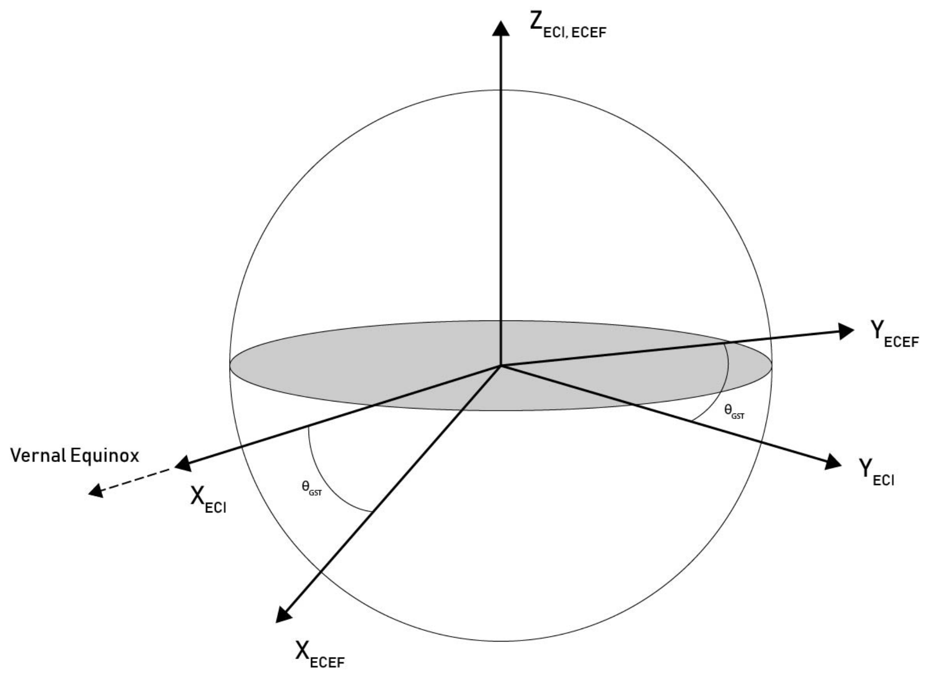


Figure: ECI and ECEF Frames

Earth - Centered Earth - Fixed

The Earth-Centered Earth-Fixed reference frame is based on the ECI frame principles. The difference lies upon the fact that the X and Y axes rotate along with the rotation of Earth, as shown in previous figure. The Z axis is aligned with ECI Z axis. The X axis points where the the first meridian intersects with the equator's plane. Lastly, the Y axis completes the orthogonal coordinate system, using the right-hand rule.

Orbit Frame

The origin of the orbit frame is located on the satellite body. The Z axis follows the velocity vector, tangent to the orbit's trajectory. The X axis points towards nadir, while the Y axis completes the orthogonal coordinate system, using the right hand rule, as shown in Figure

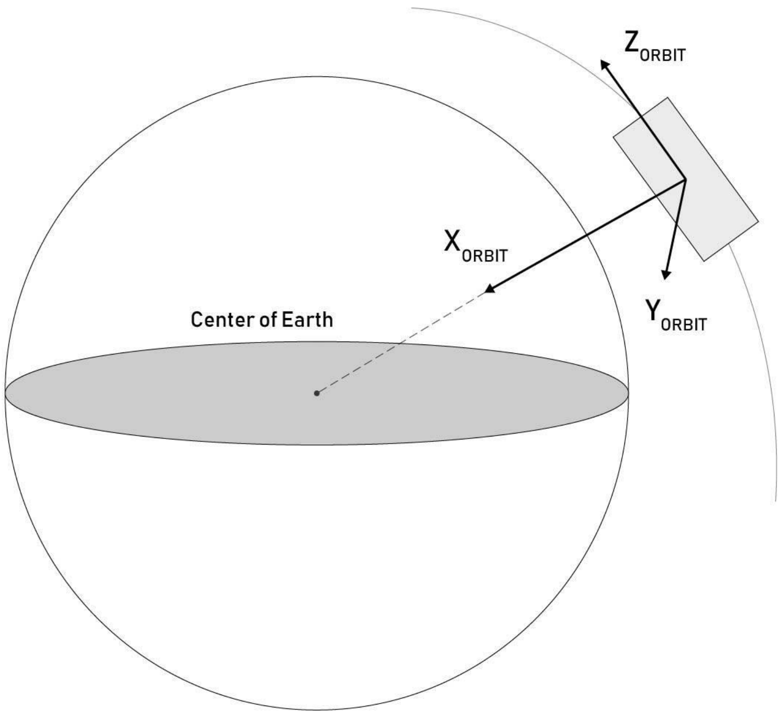


Figure: *Orbit Frame*

NED Frame

The NED reference frame is used mainly to represent the geomagnetic field from the IGRF model. The X axis of this frame points at the true North, while the Z axis points towards nadir. The Y axis completes the orthogonal coordinate system, using the right-hand rule, as shown in the Figure.

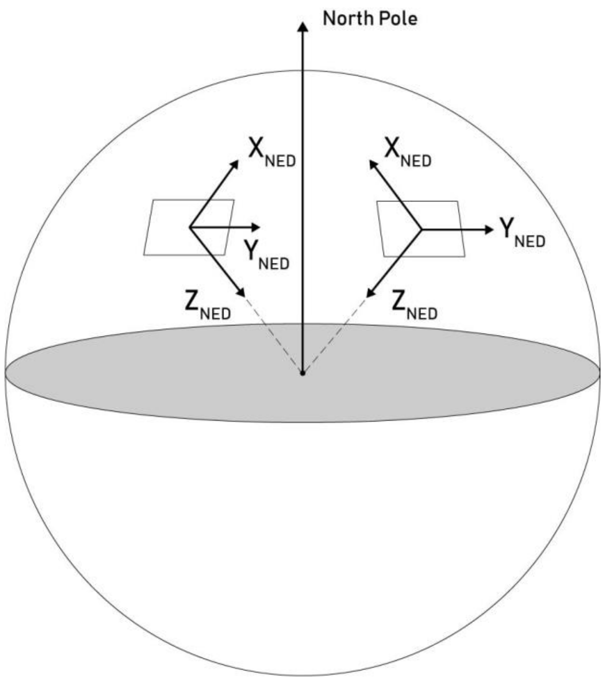


Figure : *NED Frame*

Body Frame

The origin of the body frame is located on the geometric center of the satellite. The X+ faces where the patch antenna is placed. The Z+ axis faces towards the deployable antenna. The Y+ axis completes the orthogonal coordinate system, using the right hand rule, as shown in the Figure .

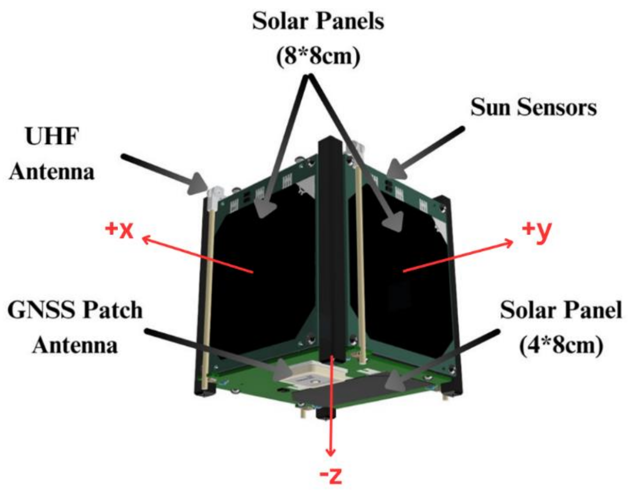


Figure: *Body Frame*

Transformation matrices

Having defined all the necessary reference frames which are to be used for the satellite's orientation representation, it is necessary to define the respective transformation matrices between them.

ECI to ECEF

The vectors which are defined in the ECI reference frame can simply be transformed to the ECEF reference frame via a turn with respect to the Z axis. Specifically:

$$\boldsymbol{R}_{\mathrm{IE}} = \begin{bmatrix} \cos(\theta_{\mathrm{GST}}) & \sin(\theta_{\mathrm{GST}}) & 0 \\ -\sin(\theta_{\mathrm{GST}}) & \cos(\theta_{\mathrm{GST}}) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

where θ_{GST} parameter denotes the rotation angle between the two reference frames, which is going to be utilized in the following equation:

$$\theta_{GST} = \lambda_0 + \omega \cdot t$$

where the $\lambda 0$ is the longitude, ω is the rotation rate of the Earth, which equals to ω = 7.29211586 · 10^{-5} rad/sec, and t is the elapsed time.

ECI to Orbit

The transformation matrix from the ECI reference frame to the orbit frame requires some orbital elements. Namely:

- Right ascension of the ascending node
- Inclination
- Argument of perigee
- Mean anomaly

All of the above mentioned orbital elements are provided by the SGP4 orbit propagator. Then, the transformation matrix is defined as:[*James Tsui*]

$$\boldsymbol{R}_{\mathrm{IO}} = \begin{bmatrix} cos(\Omega)cos(\omega) - sin(\Omega)cos(i)sin(\omega) & sin(\Omega)cos(\omega) + cos(\Omega)cos(i)sin(\omega) & sin(i)sin(\omega) \\ -cos(\Omega)sin(\omega) - sin(\Omega)cos(i)cos(\omega) & -sin(\Omega)sin(\omega) + cos(\Omega)cos(i)cos(\omega) & sin(i)cos(\omega) \\ sin(\Omega)sin(i) & -cos(\Omega)sin(i) & cos(i) \end{bmatrix}$$

where Ω is the right ascension of the ascending node, *ω* is the sum of the argument of perigee and the mean anomaly, whereas *i* is the inclination. A visual representation of this transformation matrix is:

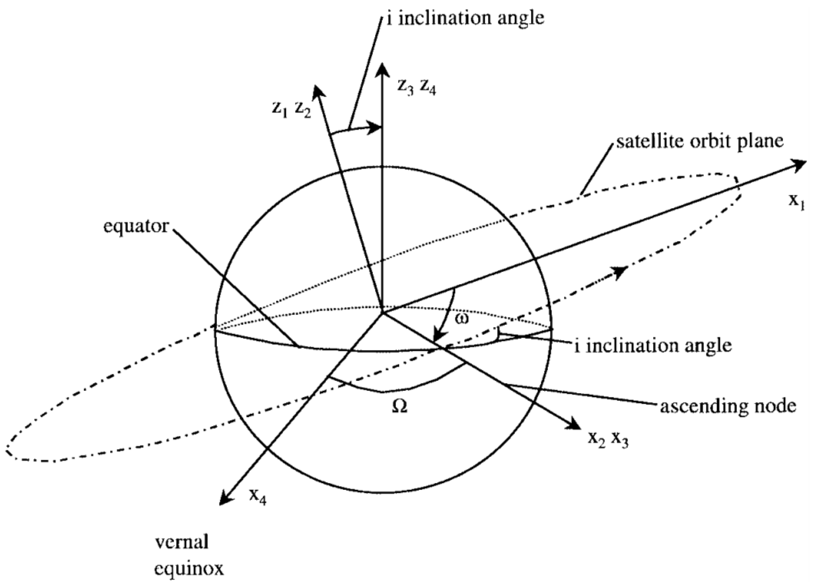


Figure : *ECI to orbit transformation matrix*

Due to the defined orbit frame, the transformation matrix rows have to be adjusted. Specifically, the X row sign is changed and the Y and Z rows are swapped.

$$\boldsymbol{R}_{\mathrm{IO}} = \begin{bmatrix} -\boldsymbol{R}_{\mathrm{IO},1} \\ \boldsymbol{R}_{\mathrm{IO},3} \\ \boldsymbol{R}_{\mathrm{IO},2} \end{bmatrix}$$

NED to ECEF

The transformation matrix between NED and ECEF reference frames is required in order to translate the geomagnetic field vector to an inertial reference frame. The transformation matrix requires the following variables:

- Latitude
- Longitude

Then, the transformation matrix is defined as:

$$\boldsymbol{R}_{\mathrm{NE}} = \begin{bmatrix} -\sin(\phi)\cos(\lambda) & -\sin(\lambda) & -\cos(\phi)\cos(\lambda) \\ -\sin(\phi)\sin(\lambda) & \cos(\lambda) & -\cos(\phi)\sin(\lambda) \\ \cos(\phi) & 0 & -\sin(\phi) \end{bmatrix}$$

where λ is the latitude and ϕ is the longitude.

Orbit to Body

The transformation matrix between any reference frame and the body frame is:

$$\boldsymbol{R}_{\mathrm{OB}} = \boldsymbol{I} + 2\eta \boldsymbol{S}(\epsilon) + 2\boldsymbol{S}^2(\epsilon)$$

where $\boldsymbol{R}_{\mathrm{BO}} = (\boldsymbol{R}_{\mathrm{OB}})^{\mathrm{T}}$, η is the quaternion scalar part, $\boldsymbol{\epsilon}$ is the quaternion vector part,

S() is the skew matrix and the **I** is the identity matrix.

Environmental models and assumptions

While AcubeSAT will orbit around the Earth in LEO, several environmental phenomena will influence both the orientation and the attitude determination system. The most significant torques generated in LEO, as described in [*James R.*], [Landis Markley] and [*Samir A. Rawashdeh*], are :

- Gravitational torque
- Aerodynamic drag
- Solar radiation pressure
- Residual magnetic dipole moment

Gravitational torque

All non-symmetric, rigid bodies that are inside a gravitational field are subject to gravitational torque [*James R. Wertz*]. The gravitational torque is calculated as follows:

$$\boldsymbol{\tau}_{\mathrm{g}} = 3 \cdot (\boldsymbol{\omega}_0)^2 \cdot R_{\mathrm{OB,x}} \times \boldsymbol{I} \cdot R_{\mathrm{OB,x}} \tag{56}$$

where $R_{OB,x}$ is the first column of the rotation matrix from the orbit to the body reference frame, which is the body frame representation of a nadir-pointing unit vector, $\omega 0$ is the angular velocity of the orbit frame relative to the ECI expressed in the orbit frame, I is the inertia tensor of the satellite. The aforementioned angular velocity is calculated from:

$$\omega_{\mathrm{o}} = \sqrt{\frac{\mu}{R_{\mathrm{c}}^3}} \tag{57}$$

where $\mu = G \cdot M_E$ is the standard gravitational constant of Earth, $G = 6.674 \cdot 10^{-11}$ m^{3}kg^{-1}s^{-2} is the gravitational constant and $ME = 5.972 \cdot 10^{24}$ kg is the total mass of Earth, R_C is the distance between the satellite and the center of Earth (6371km + 500km).

Aerodynamic drag

In LEO the external torque produced from the aerodynamic drag is significant and comparable to the gravity gradient disturbance. As the altitude of the orbit decreases, the aerodynamic torque becomes the dominant disturbance torque as the atmospheric density and the satellite velocity increase. From [4] the aerodynamic torque is calculated as

$$\boldsymbol{\tau}_{\mathbf{ad}} = \frac{1}{2}\, \rho\, C_{\mathrm{d}}\, A\, V^2\, (\boldsymbol{u}_{\mathbf{v}} \times (\boldsymbol{C}_{\mathbf{pa}} - \boldsymbol{Cm})) \tag{58}$$

where $C_d \in [2, 2.5]$ is the drag coefficient. According to the FYS design specification, the drag coefficient that should be used in for this design is equal to 2, ρ is the atmospheric density, V is the linear satellite velocity and $\boldsymbol{u}_v$ is the unit linear velocity vector of the satellite, which in this case is the third column of the rotation matrix as stated in the gravity gradient torque. The affected area A is defined as the area projected to the plane perpendicular to $\boldsymbol{u}_v$. The center of atmospheric pressure $\boldsymbol{C}_{pa}$ is calculated as:

$$\boldsymbol{C}_{\mathrm{pa}} = \frac{\int x P(x)\, dx}{\int P(x)\, dx} \tag{59}$$

where x is the distance from the origin of the body frame and P is the atmospheric pressure. Assuming that P is constant at every point along the cubesat's surface, the center of atmospheric pressure is taken at the center of the affected area. In Equation (58) only the torque generated due to the displacement of the atmospheric pressure center relative to the center of mass Cm is taken into account. The dissipation torque produced by the satellites spin is considered several orders of magnitude smaller than the first and therefore is being neglected.

Solar Radiation Pressure

The solar pressure depends on the angle of incidence of the sun rays on the satellite body frame and the reflectance of the cubesat's surface. Since the surface reflectance is unknown, the total torque can be calculated, as a worst-case scenario, by [*James R. Wertz*]:

$$\boldsymbol{\tau}_{\mathrm{srp}} = \frac{F_{\mathrm{s}}}{c} A_{\mathrm{s}} \left(1+q\right) \cos(i) \cdot (\boldsymbol{u}_{\mathrm{s}} \boldsymbol{C}_{\mathrm{ps}}) \qquad (60)$$

where *Fs* is the solar constant (or solar flux density), *c* is the speed of light, *q* is the reflection coefficient, which ranges from 0 to 1 and in this case is assumed 0.6, *i* is the sun angle of incidence and ***u***s is the sun unit vector in body frame, derived from orbit propagator. The affected area *A*s is defined as the projected area of the cubesat to the plane perpendicular to ***u***s. The center of solar pressure ***C***ps is assumed to be at the center of the affected area.

Residual Magnetic Torque

All electronics, especially the conductive elements, that the satellite uses, produce a magnetic dipole that interacts with the geomagnetic field, producing a disturbance torque. The torque created by the residual magnetic moment is given as:

$$\boldsymbol{\tau}_{\mathrm{rm}} = \boldsymbol{m} \times \boldsymbol{b} \qquad (61)$$

where ***m*** is the magnetic moment of the satellite and ***b*** the geomagnetic field. Note that the total magnetic moment that the satellite produces, excluding the effect of the MTQ, can only be accurately measured after the assembly of the satellite. The residual magnetic torque is modeled as

$$\boldsymbol{m} = \boldsymbol{m}_{\mathrm{const}} + \boldsymbol{m}_{\mathrm{var}} \qquad (62)$$

where ***m***const and ***m***var are constant and variable values correspondingly of the residual magnetic dipole moment of the satellite, with

$$m_{\mathrm{var}} < \frac{m_{\mathrm{const}}}{9} \qquad (63)$$

Albedo effect

The sun sensors designed for CubeSat missions generally expect only one light source to exist. In practice, though, this is not the case due to the earth's albedo. The albedo effect is the diffused reflection of solar radiation from a body out of the total solar radiation it receives. The diffused reflected light from the Earth affects the attitude determination system, due to the additional error in the sun sensors. To model the impact of this effect, Earth Albedo Toolbox for MATLAB [5] is utilized. For it to work, reflectivity data from Earth is required.

Currently, data from the NASA TOMS project [6] is used. Albedo compensation is included inside Simulation.

Sensor model

Magnetometer model

Magnetometer noise can be modelled as a combination of bias, scale error and white noise terms. The RM3100 magnetometer has no inherent bias drift Section 2. Assuming the magnetometer is calibrated, the bias and scale errors are eliminated. Under these assumptions, magnetometer noise is modeled solely as additive Gaussian white noise.

$$\hat{\boldsymbol{b}} = (I + A) \cdot \boldsymbol{b} + \boldsymbol{\eta} \tag{71}$$

Where b is the real value of the magnetic induction, A is the misalignment rotation Matrix, I is the identity matrix and η is a Gaussian white noise process. η consists of magnetic and thermal noise. Based on [16], the RM3100 sensor was tested by the manufacturer through various scenarios, concluding that the sensor's mean noise value equals to 0 and its standard deviation being close to 20 nT. As a result the magnetometer sensor noise can be modelled as Gaussian white noise with a standard deviation of 20 nT. In our approach the standard deviation value was set to 30 nT as indicated by the component datasheet.

Gyroscope model

According to Markley-Crassidis [1], the gyroscope measurement model is given by:

$$\boldsymbol{\omega}(t) = (1 + A)\boldsymbol{\omega}^{\text{true}}(t) + \boldsymbol{\beta}^{\text{true}}(t) + \boldsymbol{\eta}_{\upsilon} \tag{72}$$

$$\dot{\boldsymbol{\beta}}^{\text{true}} = \boldsymbol{\eta}_{\text{u}} \tag{73}$$

where ω is the measured angular velocity in the gyroscope reference frame, ω true the real angular velocity in the same frame, A is the misalignment rotation Matrix, I is the identity matrix, β true the drifting bias of the gyroscope, ηυ and ηu are independent zeromean Gaussian white-noise processes. A more general model could be used including scale factors and misalignments, however they have not yet been modeled, as there is no way of knowing them considering the satellite has not been assembled yet. Discretizing the above model yields the following equations [1]:

$$\boldsymbol{\omega}_{\text{k+1}} = (I + A)\boldsymbol{\omega}^{\text{true}}_{\text{k+1}} + \frac{1}{2}(\boldsymbol{\beta}^{\text{true}}_{\text{k+1}} + \boldsymbol{\beta}^{\text{true}}_{\text{k}}) + \left(\frac{\sigma_{\upsilon}^2}{\Delta t} + \frac{1}{12}\sigma_{\text{u}}^2\Delta t\right)^{\frac{1}{2}} \boldsymbol{N}_{\upsilon_{\text{k}}} \tag{74}$$

$$\boldsymbol{\beta}^{\text{true}}_{\text{k+1}} = \boldsymbol{\beta}^{\text{true}}_{\text{k}} + \sigma_{\text{u}}\Delta t^{\frac{1}{2}}\boldsymbol{N}_{\text{u}_{\text{k}}} \tag{75}$$

where the subscript k denotes the k-th time-step, Nυk and Nuk are zero-mean Gaussian white-noise processes with covariance each given by the identity matrix while σ_{υ}^2 and σ_{u}^2 are the covariances associated with white-noise and bias respectively.

Coarse sun sensor

Noise due to Earth's Albedo is included by utilizing Earth Albedo Toolbox for MATLAB [5], which, given reflectivity data, returns the percentage of sunlight diffused from Earth. The details of implementation are described below:

- Each of the six coarse sun sensors needs to be modeled separately. To that end, six frames are defined, one for every sensor. Four frames are defined by rotating the body frame around the Y axis (0,90,-90,180)◦ . The other two are defined by rotating the body frame around the Z axis (90,-90)◦
- Albedo radiation is assumed to come from nadir. Nadir vector is acquired in the body frame by rotating the satellite position vector in the ECI frame with the ECI to Body quaternion, and then negating it, to yield the vector pointing from the Body frame to Earth, instead of away from Earth.
- Current induced in the coarse sun sensors is proportional to the cosine of the angle between the vector normal to the sensor and the light source in the sensor frame. Symbolically:

$$i = I_s \cos(\theta) \tag{76}$$

$$\cos(\theta) = \boldsymbol{source}_{\text{sensor}} \cdot \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^{\mathrm{T}} \tag{77}$$

  If cos(θ) < 0 then the sensor does not detect light from this source

- Currents are calculated for each sensor by adding the currents induced directly by sunlight and Earth Albedo. Assuming Is = 1 for direct sunlight, current induced by Earth's Albedo is scaled by the Albedo percentage as calculated from the MATLAB toolbox. Shot noise as described above is added here.

$$i_{\text{sun}} = \boldsymbol{sun}_{\text{sensor}} \cdot \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^{\mathrm{T}} \tag{78}$$

$$i_{\text{albedo}} = albedo\boldsymbol{nadir}_{\text{sensor}} \cdot \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^{\mathrm{T}} \tag{79}$$

$$i_{\text{total}} = (i_{\text{sun}} + i_{\text{albedo}})(1 \pm 0.01 w_{\text{k}}) \tag{80}$$

  where w is a Poisson process with λ = 1.

- Finally, currents in opposite faces are subtracted and the resulting vector normalized, yielding the final sun sensor measurement in the body frame.

$$\boldsymbol{sun}_{\text{meas}} = \frac{(i_1 - i_4, i_6 - i_5, i_2 - i_3)}{\|(i_1 - i_4, i_6 - i_5, i_2 - i_3)\|} \tag{81}$$

Actuator model

With reference to the models of the actuators, functions, which simulate the operation of the actuators, are generated.

Magnetorquers

The magnetorquers generate magnetic dipole moment, m, which interacts with the earth's geomagnetic field, b, to induce a torque acting on the satellite τ_m, as described in Equation (33). Consequently the magnetic moment generated by the magnetorquers is:

$$\boldsymbol{m} = \frac{\boldsymbol{b} \times \boldsymbol{\tau}_\mathbf{m}}{\boldsymbol{b}^\mathbf{T}\boldsymbol{b}} \tag{82}$$

Saturation of the Magnetorquers

This paragraph is based on the work of Forbes and Damaren [9].

In a practical context, current limitations associated with the magnetic torque rods impose a saturation limit on m, such that |mj | ≤ mmax, where mmax is the saturation limit of each identical torque rod. We extend our geometric method to handle saturation issues associated with magnetic actuation. The actuator configuration on AcubeSAT disallows the desired control, u, to be fully realized. However, a control torque that is reduced in magnitude as a result of saturation, u s , is permissible. Given that u s can be applied, it follows that the direction of each control vector be equivalent, that is, uˆ Tuˆ s = 1. Upon saturation of any one of the torque rods, the total magnetic torque to be applied to the spacecraft is scaled accordingly. Consequently, according to the Equation (49):

$$\boldsymbol{\tau}^{\mathrm{s}}_{\mathrm{m}} = k^{\mathrm{s}}_{\mathrm{ma}}\hat{\boldsymbol{\tau}}_{\mathrm{m}} = \boldsymbol{b}^{\times T}\boldsymbol{m}^{\mathrm{s}} \tag{83}$$

where $\boldsymbol{m}^{\mathrm{s}} = k^{\mathrm{s}}_{\mathrm{ma}}\boldsymbol{m}'$, $\boldsymbol{m}' = \frac{\boldsymbol{m}}{\|\boldsymbol{\tau}_{\mathrm{m}}\|}$ and $k^{\mathrm{s}}_{\mathrm{ma}} = \min(|\frac{m_{\mathrm{max}}}{m_1}|, |\frac{m_{\mathrm{max}}}{m_2}|, |\frac{m_{\mathrm{max}}}{m_3}|)$.

Both τ m and τ s m point in the same direction, but have different magnitudes. It follows that

$$\boldsymbol{u}^{\mathrm{s}} = k^{\mathrm{s}}_{\mathrm{ma}}\hat{\boldsymbol{\tau}}_{\mathrm{m}} + k^{\mathrm{s}}_{\mathrm{wa}}\hat{\boldsymbol{\tau}}_{\mathrm{w}} \tag{84}$$

where $k^{\mathrm{s}}_{\mathrm{wa}} = \frac{k^{\mathrm{s}}_{\mathrm{ma}} \cdot k_{\mathrm{wa}}}{k_{\mathrm{ma}}}$, owing to the requirement $\hat{\boldsymbol{u}}^{\mathrm{T}}\hat{\boldsymbol{u}}^{\mathrm{s}} = 1$. It follows that the torques to be applied by the magnetic actuators and the reaction wheel are

$$\boldsymbol{\tau}_{\mathrm{m}} = k^{\mathrm{s}}_{\mathrm{ma}}\hat{\boldsymbol{\tau}}_{\mathrm{m}} \quad \text{and} \quad \boldsymbol{\tau}_{\mathrm{w}} = k^{\mathrm{s}}_{\mathrm{wa}}\hat{\boldsymbol{\tau}}_{\mathrm{w}} \tag{85}$$

Keep into consideration that the magnitude of the wheel torques scale so that uˆ and uˆ ^{s} are collinear.

Residual Magnetic Dipole Compensation

The main disturbance induced in the satellite is considered to be the magnetic disturbance, generated by the residual magnetic dipole of the satellite electronics and magnetized components. In order to compensate for this disturbance, the constant part of the residual magnetic dipole is estimated as mˆ and set to [0.048 0.051 0.047] , which is then subtracted from the magnetic dipole value corresponding to the commanded torque, as

$$\boldsymbol{m}_{\textbf{applied}} = \boldsymbol{m}_{\textbf{commanded}} - \hat{\boldsymbol{m}} \tag{86}$$

Magnetorquers Power

The magnetic moment m of each magnetorquer can also be calculated as:

$$m = \mu_t N I A \tag{87}$$

where the units of $\boldsymbol{m}$ is Am^2, N is the number of windings, I is the flowing current in the coil and A is the area enclosed by the coil. For torquer rods, we calculate μ_t as $\mu_t = 0.3 \cdot \left(\frac{\text{length}}{\text{diameter}}\right)^{1.5}$. For air core magnetorquers $\mu_t = 1$.

Therefore, the current in each coil is equal to:

$$I_i = \frac{(\boldsymbol{b} \times \boldsymbol{\tau_m})_\mathrm{i}}{\mu_{t_\mathrm{i}} \mathrm{N_i} A_\mathrm{i} \cdot \boldsymbol{b}^\mathrm{T} \boldsymbol{b}} \tag{88}$$

where $i = [x, y, z]$ and refers to each magnetorquer.

The current I is calculated using the equation:

$$V_\mathrm{i} = I_\mathrm{i} \cdot R_\mathrm{i} \tag{89}$$

where Ri is the resistance of each magnetorquer. The power P is calculated using the equation:

$$P_\mathrm{i} = V_\mathrm{i} \cdot I_\mathrm{i} \tag{90}$$

Analyses

The performance of the satellite in nominal conditions and for the selected parameters' values is shown in Simulation Performance campaign plan & results

Parameters	Values	Units
Orbits	1	
q_desired	[1, 0, 0, 0]	
Satellite's Initial Angular Velocity	(varying)	rad/sec
B-dot gain	1	
PD Kp gain	$1 \cdot 10^{-5} \cdot$diag([17, 161, 139])	
PD Kd gain	$1 \cdot 10^{-4} \cdot$diag([152, 117, 142])	
PD Kp gain during eclipse	1.2·Kp	
PD Kd gain during eclipse	2·Kd	
Q Variance	$0.5 \cdot 10^{-5} \cdot$eye(6, 6)	
Q Variance during eclipse	6×6 matrix	
R Variance	6×6 diagonal matrix	
R Variance during eclipse	$10^{5} \cdot$R_variance	
Magnetorquers Maximum Magnetic Dipole	0.3	Am^2

Gain Tuning

Gain tunning is done in iteration

```
% Clear workspace and initialize

close all;

clc;

clear variables;

% Define base scaling factors

Kp_base = 1e-05;

Kd_base = 1e-04;

% Define bounds for Kp and Kd for each axis (X, Y, Z)

% Based on original code: 1 to 500 for each axis
```

```
lb = [1, 1, 1, 0.01, 1, 1]; % Lower bounds: [Kp_x, Kp_y, Kp_z, Kd_x, Kd_y,
Kd_z]

ub = [500, 500, 500, 1000, 500, 500]; % Upper bounds

% Initial guess (midpoint of bounds)

x0 = (lb + ub) / 2;


% Define optimization options

options = optimoptions('fmincon', ...

    'Display', 'iter', ... % Show iteration progress

    'MaxIterations', 100, ... % Limit iterations for efficiency

    'MaxFunctionEvaluations', 1000, ... % Limit function evaluations

    'Algorithm', 'interior-point', ... % Suitable for constrained problems

    'StepTolerance', 1e-6, ... % Convergence tolerance

    'ConstraintTolerance', 1e-6);


% Run optimization

try

    [optimal_gains, min_APE, exitflag, output] = fmincon(...

        @(x) objective_function(x, Kp_base, Kd_base), ...

        x0, [], [], [], [], lb, ub, ...

        @(x) nonlinear_constraints(x, Kp_base, Kd_base), ...

        options);


    % Extract optimal gains

    Kp_x = optimal_gains(1);

    Kp_y = optimal_gains(2);

    Kp_z = optimal_gains(3);

    Kd_x = optimal_gains(4);

    Kd_y = optimal_gains(5);

    Kd_z = optimal_gains(6);

    fprintf('Testing Kp_x = %d, Kd_x = %d ,Kp_y = %d, Kd_y = %d ,Kp_z = %d,
Kd_z = %d ,\n', Kp_x, Kd_x, Kp_y, Kd_y,Kp_z, Kd_z);

   % Run simulation with optimal gains to get per-axis APE

    Kp_gain = Kp_base * diag([Kp_x, Kp_y, Kp_z]);
    Kd_gain = Kd_base * diag([Kd_x, Kd_y, Kd_z]);

    [instant_error_perform, ~, ~] = Nadir_Pointing_function(Kp_gain,
Kd_gain);


    % Calculate mean APE per axis

    if size(instant_error_perform, 1) >= 1000

        APE_subset = abs(instant_error_perform(1001:end, :));

        mean_APE_per_axis = mean(APE_subset, 1);

        fprintf('1mean_APE_per_axis = %d \n', mean_APE_per_axis);

    else

        error('Simulation returned insufficient data (%d rows).',
size(instant_error_perform, 1));

    end

    % Display results

    fprintf('Optimal Gains (scaled by Kp_base=%.1e, Kd_base=%.1e):\n',
Kp_base, Kd_base);

    fprintf('Kp_x = %.2f, Kd_x = %.2f\n', Kp_x, Kd_x);

    fprintf('Kp_y = %.2f, Kd_y = %.2f\n', Kp_y, Kd_y);

    fprintf('Kp_z = %.2f, Kd_z = %.2f\n', Kp_z, Kd_z);

    fprintf('Overall Mean APE: %.4f degrees\n', min_APE);

    fprintf('Mean APE per axis:\n');

    fprintf('X-axis: %.4f degrees\n', mean_APE_per_axis(1));

    fprintf('Y-axis: %.4f degrees\n', mean_APE_per_axis(2));

    fprintf('Z-axis: %.4f degrees\n', mean_APE_per_axis(3));

    fprintf('Optimization Exit Flag: %d (%s)\n', exitflag, output.message);

catch err

    fprintf('Optimization failed: %s\n', err.message);

    return;

end

% Objective function: Compute overall mean APE

function mean_APE = objective_function(x, Kp_base, Kd_base)

    Kp_x = x(1); Kp_y = x(2); Kp_z = x(3);

    Kd_x = x(4); Kd_y = x(5); Kd_z = x(6);

    try
```

```
        Kp_gain = Kp_base * diag([Kp_x, Kp_y, Kp_z]);

        Kd_gain = Kd_base * diag([Kd_x, Kd_y, Kd_z]);

        fprintf('Testing Kp_x = %d, Kd_x = %d ,Kp_y = %d, Kd_y = %d ,Kp_z =
%d, Kd_z = %d ,\n', Kp_x, Kd_x, Kp_y, Kd_y,Kp_z, Kd_z);

        [instant_error_perform, ~, ~] = Nadir_Pointing_function(Kp_gain,
Kd_gain);

        if size(instant_error_perform, 1) < 1000

            mean_APE = 1e6; % Large value for invalid simulations

            return;

        end

        APE_subset = abs(instant_error_perform(1001:end, :));

        mean_APE = mean(APE_subset(:));

        fprintf('2mean_APE = %d \n', mean_APE);

    catch

        mean_APE = 1e6; % Large value for failed simulations

    end

end

% Nonlinear constraints: Ensure per-axis APE < 20 degrees

function [c, ceq] = nonlinear_constraints(x, Kp_base, Kd_base)

    Kp_x = x(1); Kp_y = x(2); Kp_z = x(3);

    Kd_x = x(4); Kd_y = x(5); Kd_z = x(6);

    try

        Kp_gain = Kp_base * diag([Kp_x, Kp_y, Kp_z]);

        Kd_gain = Kd_base * diag([Kd_x, Kd_y, Kd_z]);

        fprintf('Testing Kp_x = %d, Kd_x = %d ,Kp_y = %d, Kd_y = %d ,Kp_z =
%d, Kd_z = %d ,\n', Kp_x, Kd_x, Kp_y, Kd_y,Kp_z, Kd_z);

        [instant_error_perform, ~, ~] = Nadir_Pointing_function(Kp_gain,
Kd_gain);

        if size(instant_error_perform, 1) < 1000

            c = [1e6; 1e6; 1e6]; % Large values for invalid simulations

            ceq = [];

            return;

        end

        APE_subset = abs(instant_error_perform(1001:end, :));

        mean_APE_per_axis = mean(APE_subset, 1);
        fprintf('3mean_APE_per_axis = %d \n', mean_APE_per_axis);

        c = mean_APE_per_axis - 20; % mean_APE_per_axis <= 20

        ceq = []; % No equality constraints

    catch

        c = [1e6; 1e6; 1e6]; % Large values for failed simulations

        ceq = [];

    end

end
```

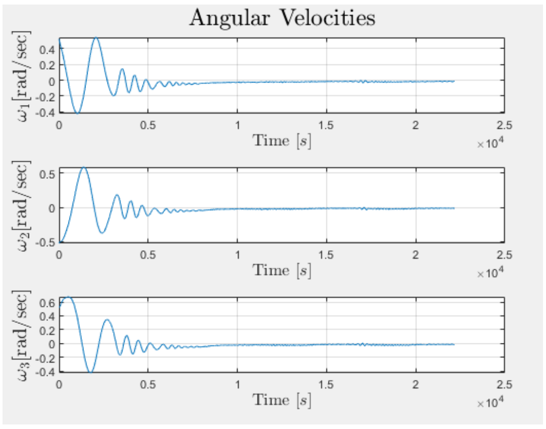
Simulation Performance campaign plan & results

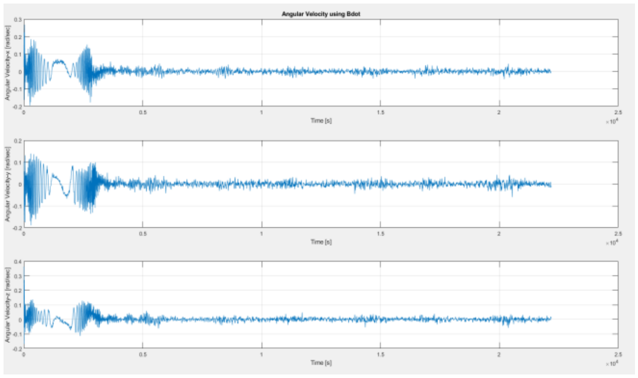
Nominal scenarios

In order to demonstrate the attitude control stability and feasibility of the mission with regards to AOCS, the performance of the satellite in nominal conditions is simulated.

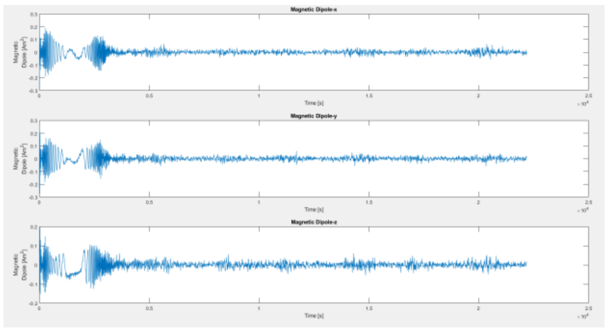
Detumbling

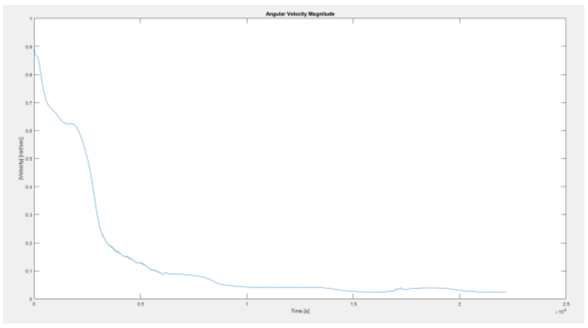
The detumbling simulation is carried out in MATLAB Simulation, from an initial angular velocity of $[\,\pi/6 \quad -\pi/6 \quad \pi/6]$ using the SSO TLE and a duration of 22180 sec, namely 4 orbits.





(a)Velocities *(b) Angular Velocity approximation*



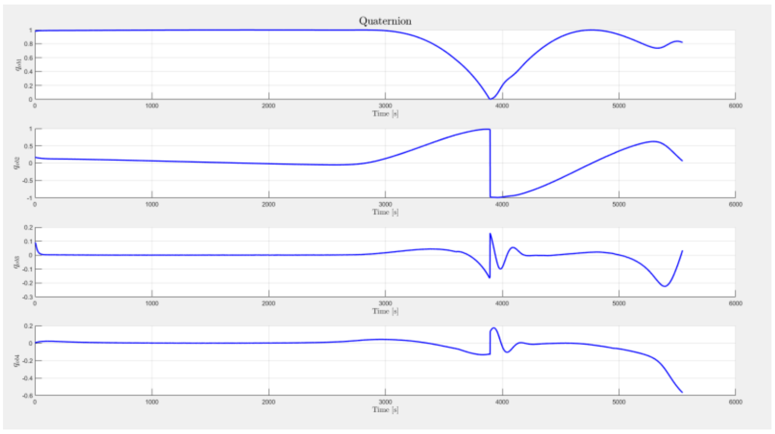


(c)Magnetic Dipole *(d) Angular Velocity Magnitude*

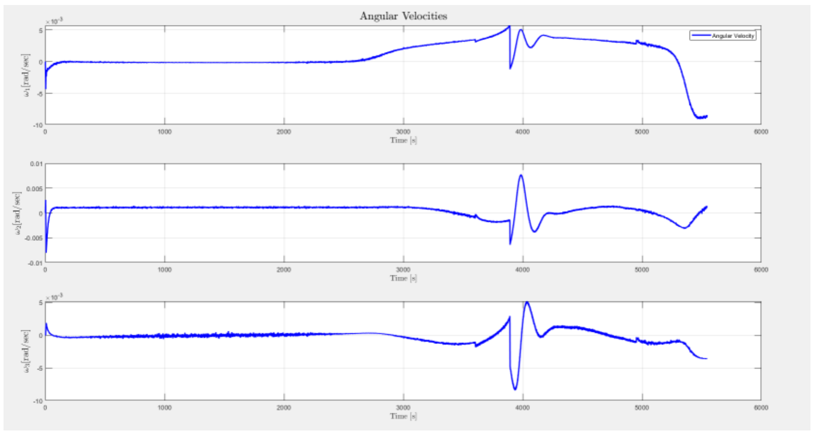
Figure : Nominal Detumbling

Nominal

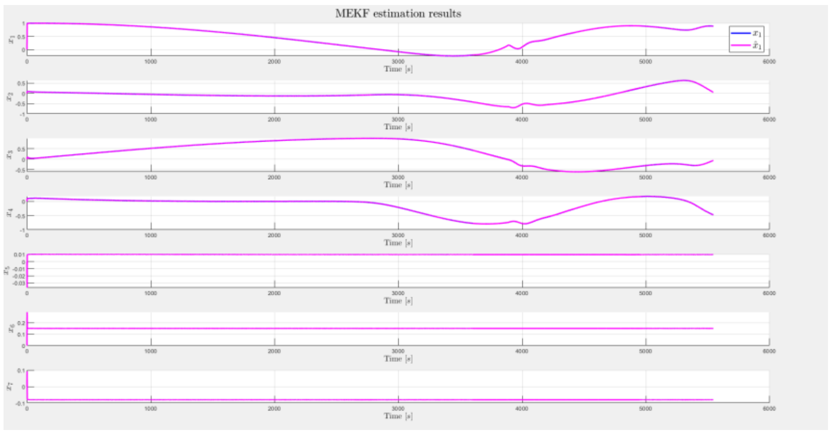
The Nominal simulation is carried out in MATLAB Simulation. The simulation's duration is 5545 sec, namely 1 orbits.



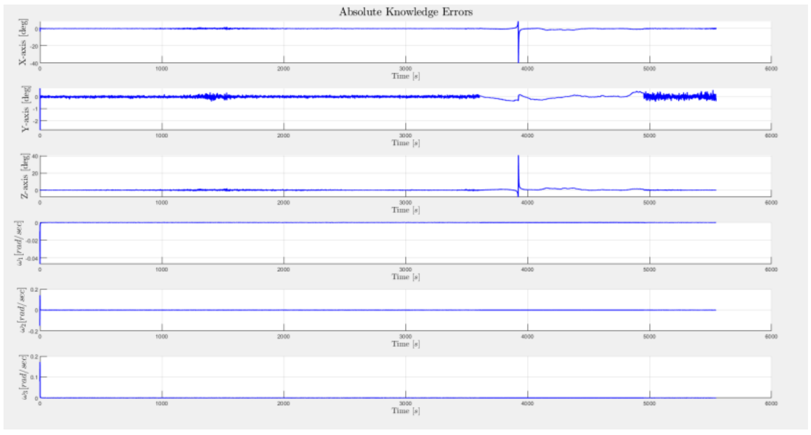
(a) Quaternion



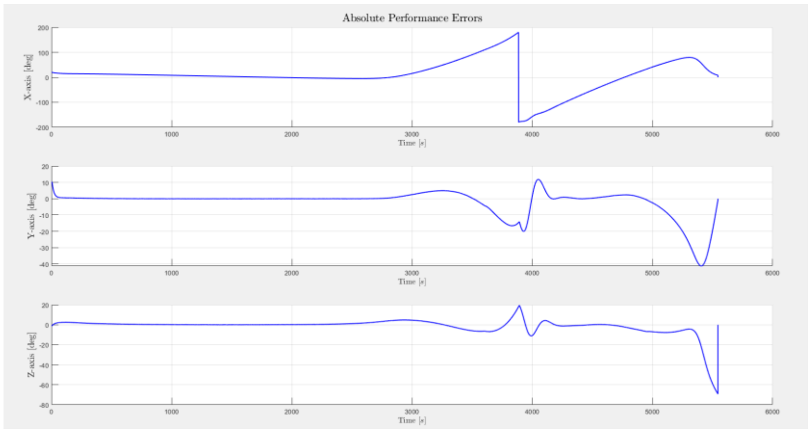
(b)Velocities



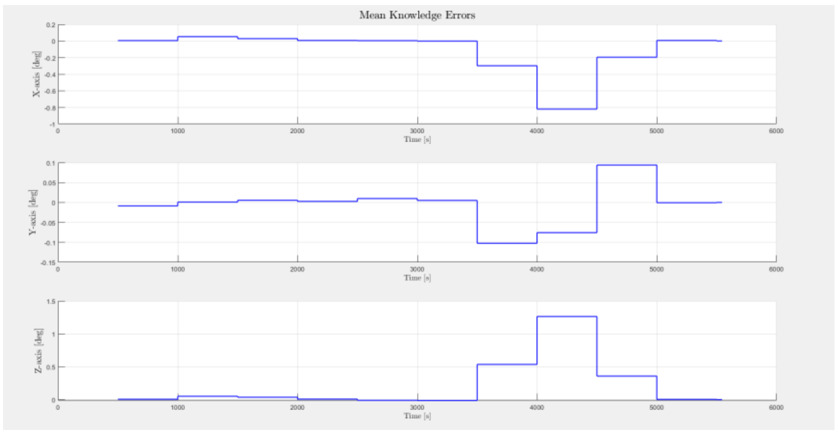
(c)MEKF Estimation Results



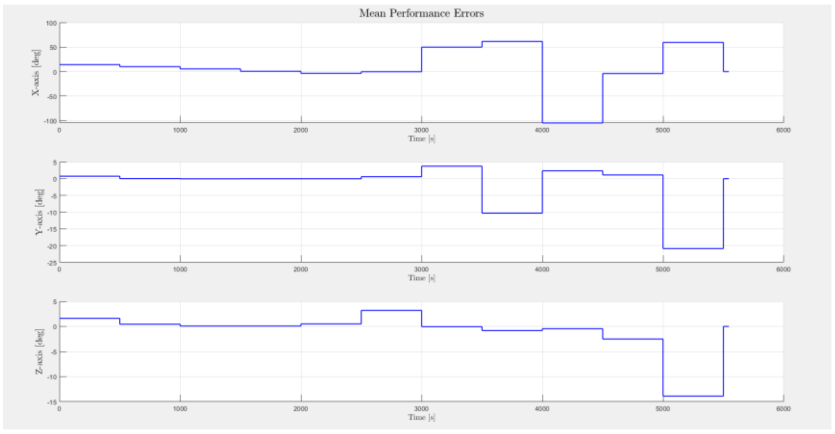
(d)AKE



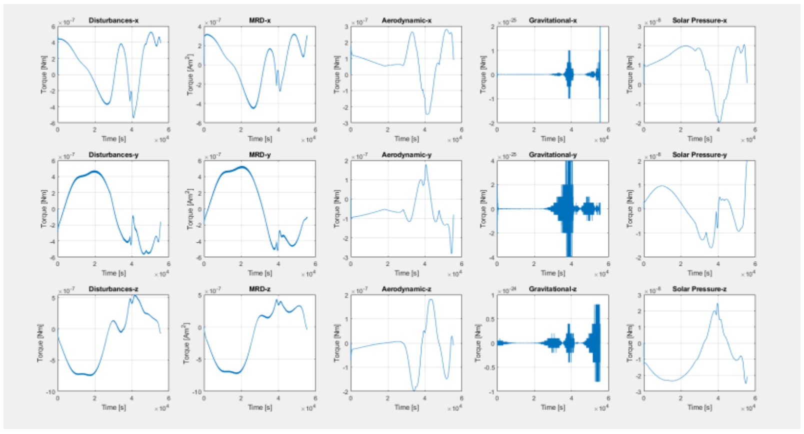
(e)APE



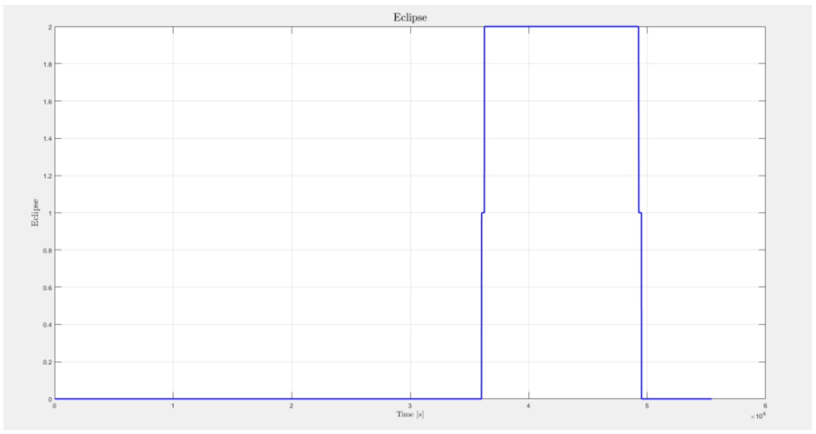
(f)MKE



(g)MPE



(h)Disturbances



(i)Eclipse

Figure: Nominal Mode

Condition	X-axis	Y-axis	Z-axis	Overall Mean APE
Non-Eclipse	3.8487	0.1719	0.96	1.6602
Eclipse	88.4395	4.9232	3.6361	32.333

Variables Table

General Parameters				
Variable	**Variable name**	**Value**	**Units**	**Comments**
Satellite mass	m	1.3	Kg	-
Inertia matrix	I		kg·m²	-
Center of mass	Cm	[0.05, 0.05, 0.05]	m	-
Earth radius	Re	6371200	m	-
Satellite altitude	Rs	500	km	-
Gravitational constant	G	$6.67428 \cdot 10^{-11}$	$m^3 \cdot kg^{-1} \cdot s^{-2}$	-
Earth mass	M	$5.972 \cdot 10^{24}$	kg	-
Satellite angular velocity relative to Earth	w_o	$1.108 \cdot 10^{-3}$	rad/sec	$\sqrt{}(G \cdot M / Radius^3)$
Satellite linear velocity in orbit	v_satellite	7616.33	m/sec	$\sqrt{}(G \cdot M / Radius)$

Orbit Period	orbit_period	5545	sec	(2·π) / (w_o)
Initial Angular Velocity on each axis in Body Frame	vRot0	(varying)	rad/sec	-

Determination Parameters				
Parameter	Variable Name	Value	Units	Comments
Process Covariance Matrix	Q	6×6 matrix	-	-
Noise Covariance Matrix	R	6×6 diagonal matrix	-	-
Magnetometer Noise	m_noise	sqrt([$1.83 \cdot 10^{-6}$, $1.83 \cdot 10^{-6}$, $1.83 \cdot 10^{-6}$])	-	-
Coarse Sun Sensor Noise	css_noise	0.01	-	-
Gyroscope Noise	gyro_noise	0.0026	-	-
Gyroscope Bias Drift	gyro_bias	$7.7570 \cdot 10^{-4}$	-	-

Control Parameters				
Parameter	Variable Name	Value	Units	Comments
Kp gain for PD controller	Kp_gain	$[17, 161, 139] \cdot 10^{-5}$	-	-
Kd gain for PD controller	Kd_gain	$[152, 117, 142] \cdot 10^{-4}$	-	-
Kp gain during eclipse for PD controller	-	1.2 · Kp_gain	-	-
Kd gain during eclipse for PD controller	-	2 · Kd_gain	-	-
Kp gain for B-dot controller	Kp_gain_bdot	1	-	-
Desired quaternion	x_desired	[1, 0, 0, 0]	-	-

Magnetic field data from magnetometers	B_body	(varying)	T	-
Initial orientation in Quaternion	q	(varying)	-	-

Magnetorquer parameters				
Parameter	Variable Name	Value	Units	Comments
Core Diameter	-	[13.89, 13.89, -]	mm	-
Length	-	[60, 60, 94.46]	mm	-
Width	-	[-, -, 90.14]	mm	-
Height	-	[-, -, 20]	mm	-
Area enclosed by the coil	A_coils	[0.000151, 0.000151, 0.0085]	mm^2	-
Maximum magnetic dipole	m	[0.3, 0.3, 0.33]	Am^2	-
Wire turns	N_coils	[700, 700, 325]	-	-
Resistance	R_coils	[2.7, 2.7, 12.9]	Ohm	-

Reference Documents

[1] Landis Markley and John Crassidis. *Fundamentals of Spacecraft Attitude Determination and Control*. Jan. 2014. ISBN: ISBN: 978-1-4939-0802-8. DOI: 10.1007/978-1- 4939-0802-8.

[2] James Tsui. "Fundamentals of Global Positioning System Receivers: A Software Approach." In: Oct. 2001, pp. i–xvii. ISBN: 9780471200543. DOI: 10.1002/0471200549. fmatter_indsub.

[3] James R. Wertz Wiley J. Larson. Space Mission Analysis and Design. Space Technology Library. Microcosm, 1999. ISBN: 978-1881883104.

[4] Samir A. Rawashdeh. "CubeSat Aerodynamic Stability at ISS Altitude and Inclination." In: 2012.

[5]MATLAB Earth Albedo Toolbo.

[6] Total Ozone Mapping Spectrometer-Earth Probe.