Design, Analysis, and Simulation of Attitude Controllers for the MicroNimbus Mission

Author:
Jian H. Li

Advisor:
Dr. Glenn Ligthsey

Space Systems Design Laboratory (SSDL)
Daniel Guggenheim School of Aerospace Engineering
Georgia Institute of Technology

April 20, 2018
This project is presented to the Academic Faculty by Jian H. Li under the advisement of Dr. Glenn Lightsey, in partial fulfillment of the requirements for the Degree Master of Sciences in Aerospace Engineering at Daniel Guggenheim School of Aerospace Engineering, Georgia Institute of Technology.
Contents

1 Introduction 3

2 ADC Hardware Overview 4
   2.1 Sensors .................................................. 5
      2.1.1 Sun Sensors ........................................... 5
      2.1.2 IMU .................................................. 6
      2.1.3 Magnetometer ......................................... 6
      2.1.4 GPS Receiver ......................................... 7
   2.2 Actuators ................................................. 8
      2.2.1 Reaction Wheel ........................................ 8
      2.2.2 Magnetic Torquer ..................................... 9

3 Attitude Controllers 9
   3.1 Attitude Controllers Overview ................................ 9
   3.2 Control Law & Stability Analysis ................................ 10
   3.3 Attitude Controllers ....................................... 15
      3.3.1 Nadir-Pointing Controller ............................. 16
      3.3.2 Sun-Pointing Controller ............................... 18
      3.3.3 Slew Controller ....................................... 19
      3.3.4 Axis-Command Controller .............................. 20
      3.3.5 B-Dot Controller ...................................... 21

4 Simulation & Results 22
   4.1 B-dot Controller ........................................... 22
   4.2 Nadir-Pointing Controller ................................... 24
   4.3 Sun-Pointing Controller ................................... 27
   4.4 Slew Controller ............................................ 29
   4.5 Axis-Command Controller ................................... 31

5 Pseudocode 33
   5.1 B-Dot Controller ........................................... 33
   5.2 Nadir-Pointing Controller ................................... 34
   5.3 Sun-Pointing Controller ................................... 36
   5.4 Slew Controller ............................................ 38
   5.5 Axis-Command Controller ................................... 40

6 Conclusion and Future Work 42

7 Acknowledgments 42
Design, Analysis, and Simulation of Attitude Controllers for the MicroNimbus Mission

Jian H. Li* and Glenn Lightsey†

Abstract

MicroNimbus is a small satellite mission currently under development at Georgia Tech Space System Design Lab. Its mission is to measure the vertical temperature profile of the atmosphere. To achieve mission success, the satellite needs to control its attitude in different modes throughout the mission, which has motivated a development of several attitude controllers. These controllers include a B-dot controller for detumble mode, a Nadir-Pointing controller for science mode, a Slew controller for ground station down-link mode, a Sun-Pointing controller for battery charging mode, and a User-Defined Pointing controller for calibration mode. These controllers are designed based on their functionality, and the stability characteristics in the closed-loop system. This project includes adapting, designing, and implementing the controllers on a NASA GSFC open source software 42 for a full attitude control simulation. Additionally, the project report discusses the theory behind each of controllers and how it is implemented in the simulation tool. These controllers serve as a baseline design for eventual implementation as flight controllers on MicroNimbus and other similar CubeSat mission in the Space System Design Lab.

Nomenclature

\( ADC \) = Attitude Determination and Control
\( F_N \) = Inertia Frame
\( F_L \) = LVLH Frame
\( F_B \) = Spacecraft Body Frame
\( F_R \) = Commanded Frame
\( F_S \) = Sun Pointing Frame
\( F_G \) = Ground Station Pointing Frame
\( F_C \) = Axis Commanded Frame
\( IGRF \) = International Geomagnetic Reference Field
\( LVLH \) = Local-Vertical, Local-Horizontal
\( ECI \) = Earth Centered Inertial
\( IMU \) = Inertial Measurement Unit
\( PD \) = Proportional-Derivative

*Graduate Student, Guggenheim School of Aerospace Engineering, Georgia Institute of Technology.
†Professor, Guggenheim School of Aerospace Engineering, Georgia Institute of Technology.
1 Introduction

Most spacecraft missions have specific pointing requirements, which are driven by the operation modes of the mission. These pointing requirement can range from Earth pointing, inertial pointing, sun pointing, and other target pointing directions. Communications, broadcast, and Earth science related satellites are examples of Earth pointing satellite. They point their payload instruments nadir (towards the center of earth) for most portion of their missions. The Hubble Space Telescope[1] (HST) is an example of inertial pointing satellite. The HST points its main mirror to a commanded location in the sky for observations. The Solar & Heliospheric Observatory[1] (SOHO) is an example of a sun pointing spacecraft whose mission is to study the sun. All of these pointing modes are achieved by the on-board attitude determination and control system.

MicroNimbus is an Earth remote sensing spacecraft, whose mission is to collect global, near real-time atmospheric temperature measurements. The spacecraft requires several attitude controllers to complete its mission. These include a Nadir-Pointing controller, a Sun-Pointing controller, an Inertial-Pointing controller, and a Slew controller. Each of these controllers are designed based on specific payload and subsystem requirements. The Nadir-Pointing controller is used to point the aperture of the payload instruments towards the surface of the earth. The Sun-Pointing controller aligns the satellite’s solar panel directly
towards the sun for battery charging. The Inertial-Pointing controller allows operators to point the spacecraft in any inertial direction for sensor calibration or sensor health check out. The Slew controller is used to maximize the downlink time during a ground station pass.

A thesis by Margaret Tam, *An Attitude Determination and Control System for Small Satellites* [2], details Attitude Determination and Control System (ADCS) designs from previous CubeSat missions by Texas Spacecraft Lab (TSL). These missions include, ARMADILLO, BEVO-2, and RACE ¹. The thesis provides hardware ADCS configurations for each mission mentioned above as well as attitude controller designs and simulations. The thesis outlines controller designs which are not only driven by the respective mission requirements but also constrained by the capability of the configuration and capability of the ADCS hardware. MicroNimbus has a CubeSat design that is influenced by and improves upon previous TSL CubeSats. More specifically, it has a different ADCS hardware configuration and mission science requirements from its predecessors. Thus, a new set of attitude controllers must be designed tailored to its new hardware setup.

The purpose of this project is to develop functional attitude controllers for the MicroNimbus mission. This paper discusses theory behind each controller design, the performance of each controller, and how it is implemented in a simulation.

## 2 ADC Hardware Overview

The key factors that drive the controller design are the ADCS hardware capabilities. This section provides the specifications of each ADCS component. The hardware specifications are used as input parameters for the simulation. The ADCS is shown in Figure 1 and Figure 2. The ADCS is divided into top module and bottom module. Reaction wheels, magnetic torquers, one of the sun sensor modules, and the IMU are located at the top module. Another sun sensor module and magnetometers are located at the bottom module. Fig.3 shows the GPS receiver, which is located on a 3U side of the spacecraft.

![Figure 1: Top ADC Subsystem Module Exploded View](image1)

![Figure 2: Bottom ADC Subsystem Module Exploded View](image2)

¹[http://sites.utexas.edu/tsl/past-missions/](http://sites.utexas.edu/tsl/past-missions/)
2.1 Sensors

This section provides an overview of the sensors used in MicroNimbus. The technical specifications and measurements of each type of sensors are also provided in individual section. The ADCS sensors include sun sensor, magnetometer, IMU and GPS receiver. For a given time, the sun sensor measures the sun vector in $F_B$, and the magnetometer measures the local magnetic field vector in $F_B$. The GPS receiver provides the location of the spacecraft in $F_N$. With an accurate magnetic field model and sun propagator model, one can predict the magnetic field vector and the sun vector based on the given location provided by the GPS receiver. Combine with the measured vectors and the predicted vectors, a MEKF can be used to calculate the spacecraft attitude with respect to the inertia frame.

2.1.1 Sun Sensors

The sun sensor measures the sun vector in body frame. The selected sun sensor is nanoSSOC-A60 that produced by SOLARMEMS. It is shown in Figure 4. It is a two orthogonal axes sun sensor with a field of view of $\pm 60^\circ$. Its accuracy is less than 0.5$^\circ$. The raw outputs of each sun sensor are analog voltage signals range between 0V and 3.3V. Low-pass filters and Analog-to-Digital converters are implemented to filter and convert the output signals. There are 8 units of sun sensors on the spacecraft, which are divided into two modules, four units per module. The module is located on each end of the spacecraft. The sun sensor modules are shown in Figure 5. This configuration provides full-sky coverage for the spacecraft.
2.1.2 IMU

The M-G364PDCA is a small factor IMU with 6 degrees of freedom that produced by EPSON. Figure 6\textsuperscript{1} shows an image of the IMU. The features of the IMU are listed in Table 1.

![EPSON M-G364PDCA IMU](image)

The IMU consists two sensors, the triple gyroscopes and the accelerometer. The gyro sensor measures the speed or angular of rotation about each body axis of the spacecraft. Similarly, the accelerometer measures the gravity acceleration of each body axis. The IMU operates without any knowledge of the external inertial frame, $F_N$. The attitude determination is done using the IMU combined with external references such as sun sensors and magnetometers.

\textsuperscript{1}https://global.epson.com/products/ndrivers/sensing/system/imu/g364/
Table 1: M-G364PDCA EPSON IMU Features[4]

<table>
<thead>
<tr>
<th>Features</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dimension</td>
<td>24×24×10 mm</td>
</tr>
<tr>
<td>Weight</td>
<td>10 grams</td>
</tr>
<tr>
<td>Triple Gyroscopes Dynamic Range</td>
<td>± 100°/s</td>
</tr>
<tr>
<td>Gyro Bias Instability</td>
<td>2.2°/h</td>
</tr>
<tr>
<td>Gyro Angular Random Walk</td>
<td>0.09°/√hr</td>
</tr>
<tr>
<td>Tri-Axis Accelerometer Dynamic Range</td>
<td>± 3 G</td>
</tr>
<tr>
<td>Accelerometer Bias Instability</td>
<td>0.05 mG</td>
</tr>
<tr>
<td>Digital Serial Interface</td>
<td>SPI/UART</td>
</tr>
<tr>
<td>Operating Temperature Range</td>
<td>−40°C to +85°C</td>
</tr>
</tbody>
</table>

2.1.3 Magnetometer

The magnetometer measures the strength and the direction of the local magnetic field. The measurements are recorded in $F_B$. Its measurement helps to determine the spacecraft attitude relative to the local magnetic field. The measurements combined with earth magnetic field model and orbit information provide spacecraft attitude relative to $F_N$.

Figure 7: Honeywell Magnetic Sensor HMC 1053

Fig.7 shows the Honeywell magnetometer HMC1053. Two magnetometers are used in the ADCS.
Table 2: Honeywell HMC1053 Features[5]

<table>
<thead>
<tr>
<th>Features</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field Range</td>
<td>±6 gauss</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>0.8 – 1.2 mV/V/gauss</td>
</tr>
<tr>
<td>Resolution</td>
<td>120 µ gauss</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>5 MHz</td>
</tr>
</tbody>
</table>

2.1.4 GPS Receiver

The GPS receiver used in MicroNimbus is the dual-frequency OEM615 unit produced by NovAtel. It provides precise position measurement for the ADCS. The OEM615 tracks all current GNSS constellation including GPS, GLONASS, Galileo, and QZSS[6]. Some technical specification are highlighted in Table3.

Table 3: NovAtel OEM615 Features[6]

<table>
<thead>
<tr>
<th>Features</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel Configuration</td>
<td>120 Channels</td>
</tr>
<tr>
<td>Signal Tracking</td>
<td>GPS L1, L2</td>
</tr>
<tr>
<td>Position Accuracy (RMS)</td>
<td>L1:1.5m; L1/L2:1.2m</td>
</tr>
<tr>
<td>Measurement Precision (RMS)</td>
<td>L1:4cm</td>
</tr>
<tr>
<td>Maximum Data Rate</td>
<td>50Hz</td>
</tr>
<tr>
<td>Velocity Accuracy</td>
<td>0.03 m/s RMS</td>
</tr>
<tr>
<td>Velocity Limit</td>
<td>515 m/s</td>
</tr>
</tbody>
</table>

2.2 Actuators

There two types of actuators used in MicroNimbus, reaction wheels and magnetic torquers. Reaction wheels are used for fine attitude pointing. Magnetic torquers are used for detumble mode and reaction wheel angular momentum unloading.

2.2.1 Reaction Wheel

There are three reaction wheels in the spacecraft, one per each axis. The selected reaction wheel is the RW-0.01 by Sinclair. They are used as momentum wheel throughout the mission, which means that the reaction wheels operates at a nominal spin rate above zero to provide a nearly constant momentum on each axis. The reaction wheel specification is shown in Table4.
Table 4: Sinclair Picosatellite Reaction Wheels RW-0.01[7]

<table>
<thead>
<tr>
<th>Features</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal Momentum</td>
<td>0.01 Nms</td>
</tr>
<tr>
<td>Peak Momentum</td>
<td>0.018 Nms</td>
</tr>
<tr>
<td>Torque</td>
<td>±1 mNm</td>
</tr>
<tr>
<td>Dimension</td>
<td>50mm × 50mm × 30mm</td>
</tr>
<tr>
<td>Telemetry</td>
<td>UART or I²C</td>
</tr>
</tbody>
</table>

Figure 8: Sinclair Interplanetary Reaction Wheel

2.2.2 Magnetic Torquer

The magnetic torquers are used mainly for spacecraft detumble and reaction wheel momentum unloading. The magnetic torquers can compensate attitude drift by minor disturbance torques, but the actuation is slow. Thus, they are not used for fine attitude pointing. The spacecraft includes three magnetic torquers, one per each axis. The magnetic torquers are made in-house. The designed is driven by many factors including size, power consumption, magnetic moment dipole, and material. The desired magnetic moment dipole is $0.2Am^2$ for each torquer. However, the actual magnetic dipoles turned out to be much higher than expected, $0.8Am^2$ for X-axis and Y-axis and $0.36Am^2$ for Z-axis. Some design modifications are needed in order to achieve $0.2Am^2$ magnetic moment dipole. Fig.9 shows a magnetic torquers during the manufacturing progress.
3 Attitude Controllers

3.1 Attitude Controllers Overview

This section discusses the attitude controllers design for the MicroNimbus mission. There are several attitude controllers designed for the mission. Each controller governs a mission mode. These controllers are Nadir-Pointing controller, Axis-Command controller, Sun-Pointing Controller, Slew coontroller, and B-Dot controller.

A control flow diagram is shown in Fig.10. The logic transitions between each control mode are indicated in figure. Note that the De-Saturation module is not an attitude controller. It is a mechanism that helps to de-saturate the angular momentum of each reaction wheel. Additionally, an Axis-Command controller is not listed in this flow diagram.
3.2 Control Law & Stability Analysis

The control law that used for most of the attitude controls except the B-dot control is a quaternion-based nonlinear PD type tracking controller. The PD controller is shown in Eq.1

\[ \tau_{cmd} = -K_p \times \delta q_{1:3} - K_d \times \omega \]  

Each component in Eq.1 is shown from Eq.2 to Eq.6,

\[ \tau_{cmd} = \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \end{bmatrix} \]  

\[ K_p = \begin{bmatrix} K_{p,x} & 0 & 0 \\ 0 & K_{p,y} & 0 \\ 0 & 0 & K_{p,z} \end{bmatrix} \]  

\[ K_d = \begin{bmatrix} K_{d,x} & 0 & 0 \\ 0 & K_{d,y} & 0 \\ 0 & 0 & K_{d,z} \end{bmatrix} \]  

\[ \delta q = \begin{bmatrix} \delta q_{1:3} \\ \delta q_4 \end{bmatrix} = q \otimes q_{cmd}^{-1} \]
The PD controller is suggested by Crassidis and Markley in [1].

The closed-loop dynamics system with the PD controller implemented can be described by Eq.7, Eq.8, and Eq.9 shown below. For simplicity of the analysis, the dynamics used here assume perfect attitude knowledge and no perturbation forces.

\[ \omega = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \]  

(6)

\[ \delta \dot{q}_{1:3} = \frac{1}{2} [\delta q_{1:3} \times] \omega + \frac{1}{2} \delta q_4 \omega \]  

(7)

\[ \delta \dot{q}_4 = -\frac{1}{2} \delta q_{1:3}^T \omega \]  

(8)

\[ \dot{\omega} = -J^{-1}([\omega \times]J \omega + K_p \times \delta q_{1:3} + K_d \times \omega) \]  

(9)

The closed-loop system is an error dynamics system with the only equilibrium point of \([\delta q_{1:3}^T, \omega^T] = 0\). The stability property of the equilibrium point of the closed-loop system can found using Lyapunov’s direct method. Define a Lyapunov candidate function as follow,

\[ V = \frac{1}{4} \omega^T J \omega + \frac{1}{2} K_p \delta q_{1:3}^T \delta q_{1:3} + \frac{1}{2} K_p (1 - \delta q_4)^2 \]  

(10)

For simplicity of the Lyapunov analysis, \(K_p\) in Eq.10 and \(K_d\) in Eq.12 are taken as scalars. Note that the Lyapunov function satisfies the following conditions, a). \(V\) is energy-like which means that \(V = 0\) only at the equilibrium point and \(V > 0\) for all \(\omega \neq 0\) and \(\delta q \neq I_q\). b). \(V \leq 0\). Condition b can be shown in the following,

\[ \dot{V} = \frac{1}{2} \omega^T J \dot{\omega} + K_p \delta q_{1:3}^T \delta \dot{q}_{1:3} - K_p (1 - \delta q_4) \delta \dot{q}_4 \]  

(11)

Substitute Eq. 7, Eq. 8, and Eq. 9 into Eq 11 gives

\[ \dot{V} = -\frac{1}{2} (\omega^T \delta q_{1:3}) [K_p + K_p \delta q_4 - K_p (1 + \delta q_4)] - \frac{1}{2} K_d \omega^T \omega = -\frac{1}{2} K_d \omega^T \omega \]  

(12)

Eq.12 shows that \(\dot{V} \leq 0\) as long as \(K_d > 0\). This result only shows the equilibrium point being simply stable. To further prove asymptotic stability of the equilibrium point, LaSalle Theorem is used as followed, let the set \(V_0 = \{\delta q, \omega | \dot{V} = 0\}\). Find the largest invariant set, \(M\), in \(V_0\). According to Eq.12, the largest invariant set \(M\) can be found as followed,

\[ V_0 = \{\delta q, \omega | \dot{V} = 0\} = \{\omega(t) = 0\}, \forall t > 0. \]  

(13)

\[ \Rightarrow \dot{\omega}(t) = 0 \]  

(14)

which according to Eq.9,

\[ \dot{\omega}(t) = 0 \Rightarrow \delta q_{1:3}(t) = 0 \]  

(15)
Note that the largest invariant set $M$ in $V_0$ is the equilibrium point itself. According to Corollary 4.1 by Khalil [8], the equilibrium point of the closed-loop system is asymptotically stable.

The next step is to determine the control gains, $K_p$ and $K_d$. There is no systematic to obtain the gains using the full nonlinear model directly. Hence, a simplified process is used here to obtain the control gain. Fig.11 shows the spacecraft rotation along its body y-axis with a commanded angle, $\theta_c$.

The rotation dynamics can be describe by Eq.16 below,

$$\tau_{cmd} = J_y \ddot{\theta} \quad (16)$$

Now, let the $\tau_{cmd}$ be the PD control effort. It has the following form in Eq.17

$$\tau_{cmd} = K_p(\theta_{cmd} - \theta) + K_d(\dot{\theta}_{cmd} - \dot{\theta}) \quad (17)$$

The closed-loop system can be modeled as the system diagram shown in Fig.12,
The closed-loop transfer function of the system is

\[ G(s) = \frac{K_p + K_ds}{J_y s^2 + K_d s + k_p} \] (18)

The denominator of Eq.18 is the characteristic equation of the system. It has a similar form as the characteristic equation of a second-order spring-damper system, which is shown in Eq.19.

\[ s^2 + 2\zeta\omega_n s + \omega_n^2 = 0 \] (19)

\[ s^2 + \frac{1}{J_y}K_d s + \frac{K_p}{J_y} = 0 \] (20)

Hence, one can relate \( K_p \) and \( K_d \) gain to the damping ratio, \( \zeta \) and natural frequency, \( \omega_n \) of the system. Step response properties of the system such as rise time, maximum percent overshoot, and settling time can be adjusted by choices of \( \zeta \) and \( \omega_n \). Eq.21 and Eq.22 show the calculation of \( K_p \) and \( K_d \) gain.

\[ K_p = J_y\omega_n^2 \] (21)

\[ K_d = 2J_y\zeta\omega_n \] (22)

The same method can be applied to different body axes of the spacecraft which results in Eq.3 and Eq.4.

A step response analysis is performed using MATLAB. For the analysis, damping ratio is set to be 0.7 and the natural frequency is set to be 0.1. The PD compensator is shown in Eq.23.

\[ C = 0.00059 \times \frac{1 + 12s}{1} \] (23)
Fig. 13 shows a unit step response of the closed-loop system. The rise time is 8.05 seconds, and the settling time is 44.2 seconds. The maximum percent overshoot is 23.1%.

Fig. 14 shows the gain margin and phase margin of the system. The simulation setting yields a phase margin of 122°. The gain margin is infinity since the system phase does not cross $-180^\circ$. 
Fig. 15 shows the root locus of the system. The closed-loop poles are stable, and the zero is in the right hand side of the plane. It implies that the system is minimum-phase.

3.3 Attitude Controllers

This section discusses each mission mode and its corresponding attitude controller. The procedure and method to construct specific pointing controller are heavily based on the requirements that set by the mission mode. Pointing controllers are constructed using a PD controller as the core control law. On the other hand, the B-dot controller uses a bang-bang control scheme.

Fig. 16 shows isometric view of the MicroNimbus spacecraft. The body axes configuration is also shown in the figure.
A series of Cartesian Frames express the dynamics of the spacecraft. $F_B$ represents the spacecraft body frame in current time. $F_R$ represents the command frame. The Commanded frame is the body frame at desired position at final time. $F_L$ represents the LVLH frame. $F_N$ represents the inertial frame which is the ECI frame.

At initial state, the spacecraft is at a random attitude. Its body attitude $F_B$ with respect to $F_N$ is found be on-board sensors measurements. The pointing controllers shall drive the spacecraft to some commanded attitudes, $F_R$. $F_R$ is related to some reference frames. These reference frames are designed based on different pointing mode requirements. For example, during science mode, the reference frame is $F_L$. During sun charging mode, the reference frame is $F_S$ which has an axis constantly pointing to the sun. Once $F_R$ and its reference frame are established, the quaternion between these two frames is obtained and used as the control command input. Additionally, the angular rate between these two frames shall be zero to maintain spacecraft’s constant attitude.

### 3.3.1 Nadir-Pointing Controller

The Nadir-Pointing controller is used for the science mode of the mission. The science mode requires the spacecraft to point its payload sensing axis directly down toward the earth surface (nadir). The science payload is a frequency-agile microwave radiometer that installed in the payload module[9]. Fig.16 shows the radiometer horn antenna right below the spacecraft body y-axis. The payload sensing axis is parallel to the spacecraft body y-axis. During the science mode, the Nadir-Pointing controller shall point the sensing axis nadir. Fig.17 shows the spacecraft attitude during the science mode.
Figure 17: Nadir Pointing Scheme during Science Mode

In Fig.17, $F_R$ represents the command frame. It is the spacecraft body frame at desired attitude with control action. The $F_R$ align with $F_L$ in a specific manner. $R_Z$ axis aligns with $L_1$ axis, $R_X$ axis aligns with $L_2$ axis, and $R_Y$ axis aligns with $L_3$ axis. $L_1$ points toward local velocity vector, $L_3$ points local nadir, and $L_2$ points negative orbital angular momentum direction.

The advantage of using $F_L$ as reference frame for $F_R$ is that $L_3$ axis always points nadir during the flight. $R_Y$ and $L_3$ axis alignment ensures sensing axis nadir pointing.

Eq.24, Eq.25, and Eq.26 show the frame transformation between $F_L$ and $F_R$.

$$F_L = T_X(-\frac{\pi}{2})T_Y(-\frac{\pi}{2})F_R \quad (24)$$

$$T_X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\theta) & \sin(\theta) & 0 \\ 0 & -\sin(\theta) & \cos(\theta) & 0 \end{bmatrix} \quad (25)$$

$$T_y = \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\theta) & 0 & \cos(\theta) & 0 \end{bmatrix} \quad (26)$$

Furthermore, the quaternion from $F_R$ to $F_L$ is,

$$q_{LR,cmd} = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \quad (27)$$
\( q_{LR, cmd} \) is the command quaternion for the Nadir-Pointing controller. It is also the step input for the dynamics system. When the controller reduces the quaternion error, \( \delta q_{LR} \) down to 0, the spacecraft will achieve the desired attitude shown in Fig.17. Eq.5 provides the calculation of \( \delta q_{LR} \).

After achieving the desired attitude, the spacecraft needs to maintain its attitude which means that the angular rate between \( F_R \) and \( F_L \), \( \omega_{RL} \) need to be 0. Hence, \( \omega_{RL} \) is the derivative error term in the PD controller. Eq.28 shows the \( \tau_{cmd} \) for Nadir-Pointing controller.

\[
\tau_{cmd, Nadir-Points} = -K_p \times \delta q_{1:3LR} - K_d \times \omega_{RL}
\]

3.3.2 Sun-Pointing Controller

During the sun charging mode, the spacecraft need to point its maximum solar panel covered body surface directly toward the sun. The maximum solar panel covered surface is directly opposite of the body y-axis. Fig.18 shows the spacecraft attitude during sun charging mode.

Figure 18: Sun Pointing Scheme during Charging Mode

\( F_S \) represents the sun pointing frame in space. \( S_1 \) axis points directly toward the sun. Direction of \( S_1 \) is provided by the sun vector measured by sun sensors. \( S_2 \) is an arbitrary axis in space that perpendicular to \( S_1 \), \( S_3 \) is the axis that complete the Cartesian frame. Components of \( F_S \) are expressed the in inertial frame, \( F_N \). At desired attitude, the spacecraft shall align the \(-R_Y \) body axis with \( S_1 \), and \( R_X \) with \( S_2 \).

The key to solve the problem is to obtain the frame transformation matrix between \( F_S \) and \( F_R \). Fig.18 shows that at the desire state, \(-R_Y \) aligns with \( S_1 \) and \( R_X \) aligns with \( S_2 \).
Essentially, $R_X$ and $S_2$ can be viewed as the same vector but with different representations. The same idea applies to $-R_Y$ and $S_1$ as well. With the knowledge of these two axes with expressions in two frames, one can obtain the frame transformation matrix between these two frames by using the Triad method [1]. The quaternion between these two frames, $q_{SR}$, can be obtained from the matrix as well.

After achieving the desired attitude, the spacecraft need to maintain zero angular rate with respect to $F_S$ which means $\omega_{RS}$ should be zero. Eq.29 shows calculation to obtain $\omega_{RS}$.

$$\dot{C}_{SR} = C_{SR}[\omega_{RS} \times] \quad (29)$$

$C_{SR}$ is the frame transformation matrix from $F_R$ to $F_S$. $\dot{C}_{SR}$ is the time rate change of $C_{SR}$ which is obtained through the Triad method at every time step. Eq.30 shows the $\dot{C}_{SR}$ calculation.

$$\dot{C}_{SR} = \frac{C_{SR}(t_k) - C_{SR}(t_{k-1})}{dt} \quad (30)$$

Eq.31 shows the $\tau_{cmd}$ for Sun-Pointing controller.

$$\tau_{cmd, Sun-Points} = -K_p \times \delta q_{1:3SR} - K_d \times \omega_{RS} \quad (31)$$

### 3.3.3 Slew Controller

A Slew controller is used to maximize the downlink time during ground pass. Fig.19 shows the slew control scheme during data downlink mode. The S-band patch antenna is located on the surface that perpendicular to $R_Y$ axis. To maximize the data downlink time, spacecraft shall its $R_Y$ axis directly toward the ground station once the spacecraft appears in the local horizon.

$\rho$ represents the range vector from the ground station to the spacecraft. $G_1$ axis from $F_G$ points directly opposite $\rho$. $G_2$ is an arbitrary axis that perpendicular to $G_1$. $G_3$ is the third axis that complete the Cartesian frame. Similar to $F_S$, $F_G$ is expressed in inertia frame, $F_N$.

Note that $F_G$ rotates in inertial space as $\rho$ changes when spacecraft passes over the ground station. The Slew controller shall maintain $F_R$'s orientation constant with respect to $F_G$. The relation between $F_R$ and $F_G$ is shown in Fig.19.
Since \( G_2 \) and \( R_X \) point to the same direction during slew control, as well as \( G_1 \) and \( R_Y \), one can use Triad method to obtain the frame transformation matrix, \( C_{GR} \), between \( F_G \) and \( F_R \) as well as \( q_{GR} \). \( q_{GR} \) is the command quaternion for the Slew controller. The angular rate, \( \omega_{RG} \), can be found from Eq.30 and Eq.29. Eq.32 shows the \( \tau_{cmd} \) for Slew controller.

\[
\tau_{cmd,Slew} = -K_p \times \delta q_{1:3GR} - K_d \times \omega_{RG}
\]

### 3.3.4 Axis-Command Controller

Axis-Command controller allows operators to command the spacecraft to any desired attitude. It is mainly used for sensor calibrations or health check. Fig.20 shows an arbitrary Axis-Command control scheme.
Figure 20: Axis-Command Control Scheme

$F_C$ is the reference frame for $F_R$. $C_1$ and $C_2$ axes as well as $F_R$ attitude with respect to $F_C$ are defined by user command. The quaternion between $F_C$ and $F_R$ can be found by using Traid method. Eq. 33 shows the $\tau_{\text{cmd}}$ for control input. Additionally, the angular rate between these two frames, $\omega_{RC}$, shall be zero.

$$\tau_{\text{cmd}, \text{Axis-Command}} = -K_p \times \delta q_{1:3CR} - K_d \times \omega_{RC} \quad (33)$$

### 3.3.5 B-Dot Controller

B-dot controller is a bang-bang controller that used for detumble mode. It uses three-axes magnetorquer to reduce the spacecraft random angular rate. The controller uses the torque generated by local magnetic filed vector and magnetorquer moment dipole to damp out spacecraft tumble motion.

The B-dot controller is easy to implement since it does not involve any frames quaternion calculations like the pointing controllers do. However, its actuation is slow. Its pointing accuracy is much less than that of the regular pointing controllers. In some orbits, spacecraft may not be fully controllable by B-dot control due to local magnetic field direction.

Eq.34 shows the mathematical model of the B-dot controller. $i$ represents the body axes. $M_{\text{max},i}$ is the maximum moment dipole can be generated by the $i_{th}$ body axis.

$$M_i = \begin{cases} -M_{\text{max},i} & \text{if } \dot{B}_i > 0 \\ M_{\text{max},i} & \text{if } \dot{B}_i < 0 \end{cases}, i = x, y, z \quad (34)$$
\[ \dot{B} = \frac{B(t_k) - B(t_{k-1})}{dt} \] (35)

The derivative of measured magnetic field, \( B \), is calculated in a discrete time manner.

4 Simulation & Results

The simulation is performed using software, 42, by NASA Goddard. To accurately depict the performance of each controller, the high fidelity orbit dynamics model includes two-body motion, aerodynamic forces and torques, spherical harmonic earth gravity gradient torque, solar pressure forces and torques, third body effect from moon. The simulation assumes perfect attitude knowledge.

The magnetic field model used for the simulation is the International Geomagnetic Reference Field (IGRF) with a degree of 8 and order of 8. Earth’s gravity model has a degree of 4 and order of 4. Moon’s gravity model has a degree of 2 and order of 0.

The simulation orbit has a periapsis of 401 km and an apoapsis of 408 km. Its eccentricity is 0.0002316. The orbit inclination is 51.6°, and its right ascension of ascending node is 50°. The orbit period is about 92 minutes.

Simulation time varies for each controller. Simulation time step is 0.2 second. The simulation date is set at April, 4th, 2018. The data output is set at 1-second interval.

The mass moment of inertia, natural frequency, damping ratio, and control gains are listed in Table.5

Due to the lack of sensor model and actuator model in current simulation, the attitude knowledge is assumed to be perfect, and the command actuations are assumed to be perfect and instantaneous.

<table>
<thead>
<tr>
<th>Mass Moment of Inertia [kg ( \cdot ) m²]</th>
<th>( \zeta )</th>
<th>( \omega_n ) [rad/sec]</th>
<th>( K_p )</th>
<th>( K_d )</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>0.05071</td>
<td>0.7</td>
<td>0.1</td>
<td>0.0005071</td>
</tr>
<tr>
<td>Y</td>
<td>0.04604</td>
<td>0.7</td>
<td>0.1</td>
<td>0.0004604</td>
</tr>
<tr>
<td>Z</td>
<td>0.02985</td>
<td>1</td>
<td>0.5</td>
<td>0.0074625</td>
</tr>
</tbody>
</table>

4.1 B-dot Controller

The B-dot controller is used for detumble mode. The controller does not complete eliminate the spacecraft’s angular rate. It can only reduce the spacecraft’s body rotation to certain value. When the spacecraft body angular rate reduces down to a predefined threshold, the detumble mode is considered complete. Three initial angular rate conditions are used for simulation. Table.6 shows the initial conditions for the detumble simulation. An angular rate is preset on each body axis after deployment. The simulation duration is 16000 second.
Table 6: B-dot Controller Detumble Simulation Initial Conditions

<table>
<thead>
<tr>
<th>Case</th>
<th>$\omega_{X,0} \ [^\circ/sec]$</th>
<th>$\omega_{Y,0} \ [^\circ/sec]$</th>
<th>$\omega_{Z,0} \ [^\circ/sec]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>-5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Case 2</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Case 3</td>
<td>-4</td>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>

Figure 21: Detumble Simulation Case 1  
Figure 22: Detumble Simulation Case 2  
Figure 23: Detumble Simulation Case 3  
Figure 24: Angular Rate Magnitude

Fig.21 to Fig.24 show the spacecraft angular rates during detumble. Fig.21, Fig.22, and Fig.23 show the body axis angular rate history during detumble. Fig.24 shows the norms of total angular rate for all three cases. The figures show that the angular rate of each body axis reaches a steady state at around 3000 second with B-dot detumble control. Each simulation case show a different steady state result. These indicate that the detumble control performance is not only driven by the controller but also driven by initial conditions. Case 1 and Case 3 both show that the spacecraft reaches steady state after 2000 seconds, and
the biggest angular rate residuals remain in body z-axis. This means that the body z-axis is parallel to the local magnetic field vector at steady state. No effective torque can be produced to slow down the rotation. The condition is shown in Fig.25. The red color axis is the local magnetic field vector which is parallel to the body z-axis, \( \hat{b}_3 \). Case 2 shows a worst case. A large angular rate residual remains at body x-axis. The spacecraft is rotating around its body x-axis with a \(-2^0/sec\) rate. Again, the x-axis is parallel to the local magnetic field vector. No torque can be generated to slow down the rotation. Fig.24 shows the magnitude of angular rate, \( ||\omega|| \), for each detumble case. At steady state, the minimum \( ||\omega|| \) of case 2 is \(1.6^0/sec\), and the minimum \( ||\omega|| \) for case 1 and case 3 are \(0.078^0/sec\) and \(0.117^0/sec\) respectively.

One of the reasons the bang-bang controller is selected for detumble is physical hardware communication constraint. If a PWM functionality is available for torque rod control, a proportional controller can be used. The proportional control requires to vary the magnetorquer moment dipole. According to Leomanni[10], a proportional controller can further reduce steady state angular rate.

![Figure 25: B-dot Controller Detumble](image)

4.2 Nadir-Pointing Controller

The Nadir-Pointing controller is used for science mode. The quaternion command is given by Eq.27. Pointing axis error is defined as the angle between body y-axis, \( R_Y \) and the \( l_3 \) axis. Fig.17 shows the axes configuration. The simulation duration is 6000 seconds. The control command starts at 5 second. The simulation assumes perfect attitude knowledge during the whole orbit. Fig.26 shows the spacecraft at science mode during the simulation.
Figure 26: Nadir-Pointing Attitude in Simulation

Figure 27: Nadir-Pointing Axis Error
Fig. 27 and Fig. 29 show the pointing error and the quaternion error. The maximum pointing error is about 0.06° in steady state. The quaternion error of each axis is close to zero. Fig. 28 shows the angular rate between $F_R$ and $F_L$. The Nadir-Pointing controller preforms well in terms of pointing accuracy.

Fig. 30 and Fig. 31 show the reaction wheels command torques and angular momenta. Even the required torque is small, the angular momenta build up quickly. The current simulation uses 0 Nms nominal momentum for each reaction wheels. Technically, the reaction wheel should operates at 0.01 Nms nominal momentum. Reaction wheel saturation will occur with 0.01 Nms nominal momentum. The spacecraft loss control authority during wheel saturation. A desaturation mechanism is needed to unload the reaction wheels’ momenta. The desaturation mechanism shall be implemented using the three-axes magnetorquers.
4.3 Sun-Pointing Controller

The Sun-Pointing controller is used for sun charging mode. Fig.18 shows that the controller helps the spacecraft to point its negative y-axis toward the sun vector to maximize charging area. Fig.32 and Fig.33 show the spacecraft at sun pointing attitude during the simulation. $-\hat{b}_2$ axis is parallel to sun vector, $\odot$.

The simulation duration is 4000 seconds. The control command starts at 5 seconds. The simulation assumes perfect knowledge of spacecraft’s attitude and sun vector. The pointing error is defined as the angle between body -y-axis, $R_Y$, and the sun vector, $S_1$. 

Figure 32: Sun Pointing Attitude Simulation in 42

Figure 33: Sun Pointing Controller Simulation in 42
Fig. 34 shows the pointing error during sun charging mode. The maximum error is 0.19° at steady state. Fig. 36 shows the quaternion errors. They are close to zero at steady state. The three-axes angular rates are also close to zero at steady state.
Fig.37 and Fig.38 show the angular momentum and torque of each reaction wheel. The nominal angular momentum of each reaction wheel is currently set as zero. Fig.37 shows that the angular momenta required for sun charging mode are less than those for science mode for the current designed $F_S$. However, the reaction wheel will saturate if the nominal angular momentum is set to be $0.01Nms$. A desaturation mechanism is highly recommended.

### 4.4 Slew Controller

Slew controller is used to maximize data downlink time during ground station passing. The simulation time is 1500 seconds. The orbit right ascension of ascending node is adjusted to $160^\circ$ to provide a ground track that passes Georgia Tech ground station. The location of Georgia Tech ground station is $33.77^\circ N$ and $84.49^\circ W$ with 300 meters above sea level. These information is used to calculate the location vector with respect to $F_N$ in 42. Fig.39 shows the ground track coverage of the slew control simulation. The ground track passes over Georgia Tech ground station. Fig.40 shows the spacecraft slew control when passing over the ground station. The $F_b$ aligns with $F_G$ instead of $F_L$. However, $F_G$ is not showed in the figure.
Fig. 39 shows the pointing error when using the Slew controller. The error angle is defined as the angle between $-R_y$ axis and $G_1$ axis that shown in Fig.19. The maximum error is $0.28^\circ$ at steady state. Fig.43 and Fig.42 show that the quaternion error and angular rate are relatively zero during slew control.
Fig. 44 and Fig. 45 show the angular momenta and torques of the reaction wheel during slew. The angular momenta required for slew control is smaller than those required for Nadir-Pointing control.

4.5 Axis-Command Controller

The purpose of the Axis-Command controller is to give the ability to operators to orient the spacecraft to any desired attitude. It can be used for sensor calibration or spacecraft health check. The current design uses $F_L$ for reference frame. Operators can control the spacecraft attitude with respect to LVLH frame. Fig. 46 shows the Axis-Command control for the simulation. The simulation time is 6000 seconds. Pointing error is not provided since the current command attitude is arbitrary. It is up to users to define the pointing error based on the choice of primary vector.
Figure 46: Axis-Command Pointing Sample

Figure 47: Axis Command Pointing Angular Rate Error

Figure 48: Axis Command Pointing Quaternion Error
Fig. 49 and Fig. 50 show the angular momenta and commanded torques of the reaction wheels. They are similar to the results provided the Nadir-Pointing controller. This is expected since the Nadir-Pointing controller is a special case of the Axis-Command controller.

5 Pseudocode

This section details the algorithm and variables used for each attitude controller. The simulation version of the algorithm is written in C, and it is located on the mission server drive1.

5.1 B-Dot Controller

<table>
<thead>
<tr>
<th>Variables</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$dt$</td>
<td>Simulation time step size, scalar.</td>
</tr>
<tr>
<td>$b_{vb}$</td>
<td>Local magnetic field vector measured in $F_b$, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>$b_{vbold}$</td>
<td>Local magnetic field vector measured in $F_b$ from previous step, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>$B_{dot}$</td>
<td>Time rate change of local magnetic field vector, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>$M_{mtbcmd}$</td>
<td>Commanded magnetic moment dipoles of the magnetorquers, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>$M_{mtbmax}$</td>
<td>Maximum magnetic moment dipoles of the magnetorquers, $3 \times 1$ Vector</td>
</tr>
</tbody>
</table>

1Z:/16043048-MicroNimbus/Documents/ADC/sim/controller
Algorithm 1 B-dot Controller

1: **procedure** B-dot Controller for Detumble
2: \hspace{1em} set bvb, bvbold, Bdot
3: \hspace{1em} Compute Bdot for each body axis as follow:
4: \hspace{2em} for \( i = 1 : 3 \) do
5: \hspace{3em} Bdot[\( i \)] = ( bvb[\( i \)] - bvbold[\( i \)] ) / dt
6: \hspace{1em} Compute control effort using bang-bang control as follow:
7: \hspace{2em} for \( i = 1 : 3 \) do
8: \hspace{3em} if Bdot[\( i \)] > 0 then
9: \hspace{4em} Mmtbcmd[\( i \)] = -1 \times Mmtbmax[\( i \)]
10: \hspace{3em} else if Bdot[\( i \)] < 0 then
11: \hspace{4em} Mmtbcmd[\( i \)] = 1 \times Mmtbmax[\( i \)]
12: \hspace{2em} Update bvbold for the next time step as follow:
13: \hspace{2em} for \( i = 1 : 3 \) do
14: \hspace{3em} bvbold[\( i \)] = bvb[\( i \)]

5.2 Nadir-Pointing Controller

Table 8: Variables for Nadir-Pointing Controller Variables

<table>
<thead>
<tr>
<th>Variables</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>( dt )</td>
<td>Simulation time step size, scalar.</td>
</tr>
<tr>
<td>CLN</td>
<td>Directional cosine matrix from ( F_N ) to ( F_L ), 3\times3 Matrix</td>
</tr>
<tr>
<td>qln</td>
<td>Quaternion from ( F_N ) to ( F_L ), 4\times1 Vector</td>
</tr>
<tr>
<td>qrl</td>
<td>Quaternion from ( F_R ) to ( F_L ), 4\times1 Vector</td>
</tr>
<tr>
<td>qrn</td>
<td>Quaternion for ( F_N ) to ( F_L ), 4\times1 Vector</td>
</tr>
<tr>
<td>qbr</td>
<td>Quaternion for ( F_R ) to ( F_B ), 4\times1 Vector</td>
</tr>
<tr>
<td>( \omega_{ln} )</td>
<td>Angular rate of ( F_L ) w.r.t ( F_N ), 3\times1 Vector</td>
</tr>
<tr>
<td>( \omega_{rn} )</td>
<td>Angular rate of ( F_R ) w.r.t ( F_L ), 3\times1 Vector</td>
</tr>
<tr>
<td>( \omega_{err} )</td>
<td>Angular rate error, 3\times1 Vector</td>
</tr>
<tr>
<td>therr</td>
<td>Quaternion error in angular form, 3\times1 Vector</td>
</tr>
<tr>
<td>Tcmd</td>
<td>Control torques, 3\times1 Vector</td>
</tr>
<tr>
<td>Twhlcmd</td>
<td>Reaction wheel command torques, 3\times1 Vector</td>
</tr>
<tr>
<td>( Kp )</td>
<td>Proportional control gains, 3\times1 Vector</td>
</tr>
<tr>
<td>( Kr )</td>
<td>Derivative control gains, 3\times1 Vector</td>
</tr>
</tbody>
</table>
Algorithm 2 Nadir-Pointing Controller

1: `procedure Nadir-Pointing Controller`
2:   set $qln$
3:   set command quaternion $qrl = [0.5, 0.5, 0.5, 0.5]$
4:   Compute $qln$ from the CLN
5:   Compute command quaternion $qrn$ using $qln$ and $qrl$
6:   Compute command $\omega rn$ using $\omega ln$ and $qrn$
7:   Compute error quaternion $qbr$ using $qrn$ and $qbn$
8:   Convert $qbr$ into angular form $therr$
9:   Compute angular rate error $\omega err$ as follow:
10:   for $i = 1 : 3$
11:      $\omega err[i] = \omega ln[i] - \omega rn[i]$
12:   Compute the command torque as follow:
13:   for $i = 1 : 3$
14:      $Tcmd[i] = - Kp[i] \times therr[i] - Kr[i] \times \omega err[i]$
15:   Input command torque to reaction wheels as follow:
16:   for $i = 1 : 3$
17:      $Twhlcmd[i] = - Tcmd[i]$
### 5.3 Sun-Pointing Controller

Table 9: Variables for Sun-Pointing Controller Variables

<table>
<thead>
<tr>
<th>Variables</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$dt$</td>
<td>Simulation time step size, scalar.</td>
</tr>
<tr>
<td>CRN</td>
<td>Directional cosine matrix from $F_N$ to $F_R$, 3×3 Matrix</td>
</tr>
<tr>
<td>OldCRN</td>
<td>CRN from previous time step, 3×3 Matrix</td>
</tr>
<tr>
<td>Cdot</td>
<td>Time rate change of CRN, 3×3 Matrix</td>
</tr>
<tr>
<td>KN</td>
<td>An inertial vector in $F_N$, 3×1 Vector</td>
</tr>
<tr>
<td>R2N</td>
<td>Normal vector to svn in $F_N$, 3×1 Vector</td>
</tr>
<tr>
<td>svn</td>
<td>Sun vector in $F_N$, 3×1 Vector</td>
</tr>
<tr>
<td>PriVR</td>
<td>Primary vector in $F_R$, 3×1 Vector</td>
</tr>
<tr>
<td>SecVR</td>
<td>Secondary vector in $F_R$, 3×1 Vector</td>
</tr>
<tr>
<td>PriVN</td>
<td>Primary vector in $F_N$, 3×1 Vector</td>
</tr>
<tr>
<td>SecVN</td>
<td>Secondary vector in $F_N$, 3×1 Vector</td>
</tr>
<tr>
<td>qbn</td>
<td>Quaternion from $F_N$ to $F_B$, 4×1 Vector</td>
</tr>
<tr>
<td>qrn</td>
<td>Quaternion for $F_N$ to $F_L$, 4×1 Vector</td>
</tr>
<tr>
<td>qbr</td>
<td>Quaternion for $F_R$ to $F_B$, 4×1 Vector</td>
</tr>
<tr>
<td>$\omega bn$</td>
<td>Angular rate of $F_B$ w.r.t $F_N$, 3×1 Vector</td>
</tr>
<tr>
<td>$\omega rn$</td>
<td>Angular rate of $F_R$ w.r.t $F_L$, 3×1 Vector</td>
</tr>
<tr>
<td>$\omega \text{err}$</td>
<td>Angular rate error, 3×1 Vector</td>
</tr>
<tr>
<td>therr</td>
<td>Quaternion error in angular form, 3×1 Vector</td>
</tr>
<tr>
<td>Tcmd</td>
<td>Control torques, 3×1 Vector</td>
</tr>
<tr>
<td>Twhlcmd</td>
<td>Reaction wheel command torques, 3×1 Vector</td>
</tr>
<tr>
<td>$Kp$</td>
<td>Proportional control gains, 3×1 Vector</td>
</tr>
<tr>
<td>$Kr$</td>
<td>Derivative control gains, 3×1 Vector</td>
</tr>
</tbody>
</table>
Algorithm 3 Sun Pointing Controller

1: procedure Sun-Pointing Controller
2: set CRN, Cdot, R2N
3: set inertial vector $KN = [0, 0, 1]$
4: set primary vector $PriVR = [0, -1, 0]$
5: set secondary vector $SecVR = [0, 0, 1]$
6: Compute $R2N$ by cross product $KN \times svn$
7: Assign primary vector $PriVN$ as follow:
8: \[ for \ i = 1:3 \ do \]
9: \[ PriVN[i] = svn[i] \]
10: Assign secondary vector $SecVN$ as follow:
11: \[ for \ i = 1:3 \ do \]
12: \[ SecVN[i] = R2N[i] \]
13: Compute CRN by TRIAD Method using $PriVR, SecVR, PriVN, SecVN$
14: Compute command quaternion $qrn$ from CRN
15: Compute $Cdot$ as follow:
16: \[ for \ i = 1:3 \ do \]
17: \[ for \ j = 1:3 \ do \]
18: \[ Cdot[i][j] = (CRN[i][j] - OldCRN[i][j]) / dt \]
19: Compute $\omega rn$ using $Cdot$ and CRN
20: Update $OldCRN$ as follow:
21: \[ for \ i = 1:3 \ do \]
22: \[ for \ j = 1:3 \ do \]
23: \[ OldCRN[i][j] = CRN[i][j] \]
24: Compute quaternion error $qbr$ using $qbn$ and $qrn$
25: Convert $qbr$ into angular form $therr$
26: Compute angular rate error $\omega err$ as follow:
27: \[ for \ i = 1:3 \ do \]
28: \[ \omega err[i] = \omega bn[i] - \omega rn[i] \]
29: Compute the command torque as follow:
30: \[ for \ i = 1:3 \ do \]
31: \[ Tcmd[i] = -Kp[i] \times therr[i] - Kr[i] \times \omega err[i] \]
32: Input command torque to reaction wheels as follow:
33: \[ for \ i = 1:3 \ do \]
34: \[ Twhlcmd[i] = -Tcmd[i] \]

## 5.4 Slew Controller

Table 10: Variables for Slew Controller Variables

<table>
<thead>
<tr>
<th>Variables</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$dt$</td>
<td>Simulation time step size, scalar.</td>
</tr>
<tr>
<td>CRN</td>
<td>Directional cosine matrix from $F_N$ to $F_R$, $3 \times 3$ Matrix</td>
</tr>
<tr>
<td>CWN</td>
<td>Directional cosine matrix from $F_N$ to $F_W$, $3 \times 3$ Matrix</td>
</tr>
<tr>
<td>OldCRN</td>
<td>CRN from previous time step, $3 \times 3$ Matrix</td>
</tr>
<tr>
<td>Cdot</td>
<td>Time rate change of CRN, $3 \times 3$ Matrix</td>
</tr>
<tr>
<td>gsECEF</td>
<td>Ground station location in $F_W$, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>gsECI</td>
<td>Ground station location in $F_N$, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>Lat</td>
<td>Ground station latitude in radian, Scalar</td>
</tr>
<tr>
<td>Lon</td>
<td>Ground station longitude in radian, Scalar</td>
</tr>
<tr>
<td>Alt</td>
<td>Ground station sea level altitude in meter, Scalar</td>
</tr>
<tr>
<td>range</td>
<td>$\rho$ in $F_N$, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>range_unit</td>
<td>$\rho$ unit vector in $F_N$, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>PosN</td>
<td>Spacecraft location in $F_N$, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>KN</td>
<td>An inertial vector in $F_N$, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>R2N</td>
<td>Normal vector to $svn$ in $F_N$, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>PriVR</td>
<td>Primary vector in $F_R$, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>SecVR</td>
<td>Secondary vector in $F_R$, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>PriVN</td>
<td>Primary vector in $F_N$, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>SecVN</td>
<td>Secondary vector in $F_N$, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>qbn</td>
<td>Quaternion from $F_N$ to $F_B$, $4 \times 1$ Vector</td>
</tr>
<tr>
<td>qrn</td>
<td>Quaternion for $F_N$ to $F_L$, $4 \times 1$ Vector</td>
</tr>
<tr>
<td>qbr</td>
<td>Quaternion for $F_R$ to $F_B$, $4 \times 1$ Vector</td>
</tr>
<tr>
<td>$\omega bn$</td>
<td>Angular rate of $F_B$ w.r.t $F_N$, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>$\omega rn$</td>
<td>Angular rate of $F_R$ w.r.t $F_L$, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>$\omega err$</td>
<td>Angular rate error, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>therr</td>
<td>Quaternion error in angular form, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>Tcmd</td>
<td>Control torques, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>Twhlcmd</td>
<td>Reaction wheel command torques, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>$K_P$</td>
<td>Proportional control gains, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>$K_r$</td>
<td>Derivative control gains, $3 \times 1$ Vector</td>
</tr>
</tbody>
</table>
Algorithm 4 Slew Controller

1: procedure SLEW CONTROLLER
2: set CRN, Cdot, OldCRN
3: set inertial vector KN = [0, 0, 1]
4: set R2N, range, range_unit, gsECEF, gsECI
5: set Lat = 0.5895, Lon = −1.473, Alt = 300
6: set primary vector PriVR = [0, −1, 0]
7: set secondary vector SecVR = [0, 0, 1]
8: Compute gsECEF using function WGS84ToECEF(Lat, Lon, Alt)
9: Compute gsECI using CWN and gsECEF
10: Compute range as follow:
11: for i=1:3 do
12: range[i] = PosN[i] - gsECI[i]
13: Calculate range_unit
14: Compute R2N by cross product KN × range_unit
15: Assign primary vector PriVN as follow:
16: for i=1:3 do
17: PriVN[i] = range_unit[i]
18: Assign secondary vector SecVN as follow:
19: for i=1:3 do
20: SecVN[i] = R2N[i]
21: Compute CRN by TRIAD Method using PriVR, SecVR, PriVN, SecVN
22: Compute command quaternion qrn from CRN
23: Compute Cdot as follow:
24: for i=1:3 do
25: for j=1:3 do
26: Cdot[i][j] = (CRN[i][j] - OldCRN[i][j]) / dt
27: Compute ωrn using Cdot and CRN
28: Update OldCRN as follow:
29: for i=1:3 do
30: for j=1:3 do
31: OldCRN[i][j] = CRN[i][j]
32: Compute quaternion error qbr using qbn and qrn
33: Convert qbr into angular form therr
34: Compute angular rate error ωerr as follow:
35: for i = 1 : 3 do
36: ωerr[i] = ωbn[i] - ωrn[i]
37: Compute the command torque as follow:
38: for i = 1 : 3 do
40: Input command torque to reaction wheels as follow:
41: for i = 1 : 3 do
42: Twhlcmd[i] = −Tcmd[i]
43: end for
44: end for
45: end for
46: end for
47: end for
48: end for
49: end for
50: end for
51: end for
52: end for
53: end for
54: end for
55: end for
56: end for
57: end for
58: end for
59: end for
60: end for
61: end for
62: end for
63: end for
64: end for
65: end for
66: end for
67: end for
68: end for
69: end for
70: end for
71: end for
72: end for
73: end if
74: end procedure
### 5.5 Axis-Command Controller

Table 11: Axis-Command Controller Variables

<table>
<thead>
<tr>
<th>Variables</th>
<th>Descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>$dt$</td>
<td>Simulation time step size, scalar.</td>
</tr>
<tr>
<td>CRL</td>
<td>Directional cosine matrix from $F_L$ to $F_R$, $3 \times 3$ Matrix</td>
</tr>
<tr>
<td>CLN</td>
<td>Directional cosine matrix from $F_N$ to $F_L$, $3 \times 3$ Matrix</td>
</tr>
<tr>
<td>PriVR</td>
<td>Primary vector in $F_R$, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>SecVR</td>
<td>Secondary vector in $F_R$, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>PriVN</td>
<td>Primary vector in $F_N$, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>SecVN</td>
<td>Secondary vector in $F_N$, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>qbn</td>
<td>Quaternion from $F_N$ to $F_B$, $4 \times 1$ Vector</td>
</tr>
<tr>
<td>qrn</td>
<td>Quaternion for $F_N$ to $F_R$, $4 \times 1$ Vector</td>
</tr>
<tr>
<td>qln</td>
<td>Quaternion for $F_N$ to $F_L$, $4 \times 1$ Vector</td>
</tr>
<tr>
<td>qbr</td>
<td>Quaternion for $F_R$ to $F_B$, $4 \times 1$ Vector</td>
</tr>
<tr>
<td>$\omega_{ln}$</td>
<td>Angular rate of $F_L$ w.r.t $F_N$, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>$\omega_{rn}$</td>
<td>Angular rate of $F_R$ w.r.t $F_L$, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>$\omega_{err}$</td>
<td>Angular rate error, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>therr</td>
<td>Quaternion error in angular form, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>Tcmd</td>
<td>Control torques, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>Twhlcmd</td>
<td>Reaction wheel command torques, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>Kp</td>
<td>Proportional control gains, $3 \times 1$ Vector</td>
</tr>
<tr>
<td>Kr</td>
<td>Derivative control gains, $3 \times 1$ Vector</td>
</tr>
</tbody>
</table>
Algorithm 5 Axis Command Controller

1: \textbf{procedure} Axis-Command Controller
2: \hspace{1em} set CRL, qln
3: \hspace{1em} set primary vector PriVR = [0, 1, 0]
4: \hspace{1em} set secondary vector SecVR = [0, 0, 1]
5: \hspace{1em} set primary vector PriVN = [0, 1, 0]
6: \hspace{1em} set secondary vector SecVN = [0, 0, -1]
7: Compute CRL by TRIAD Method using PriVR, SecVR, PriVN, SecVN
8: Compute command quaternion qrl from CRL
9: Compute quaternion qln from CLN
10: Compute command quaternion qrn using qln and qrl
11: Compute command \( \omega_{rn} \) using \( \omega_{ln} \) and qrn
12: Compute error quaternion qbr using qrn and qbn
13: Convert qbr into angular form therr
14: Compute angular rate error \( \omega_{err} \) as follow:
15: \hspace{1em} for \( i = 1 : 3 \) do
16: \hspace{2em} \( \omega_{err}[i] = \omega_{bn}[i] - \omega_{rn}[i] \)
17: \hspace{1em} Compute the command torque as follow:
18: \hspace{2em} for \( i = 1 : 3 \) do
19: \hspace{3em} Tcmd[i] = \(-K_p[i] \times therr[i] - K_r[i] \times \omega_{err}[i]\)
20: Input command torque to reaction wheels as follow:
21: \hspace{2em} for \( i = 1 : 3 \) do
22: \hspace{3em} Twhlcmd[i] = \(-Tcmd[i]\)
6 Conclusion and Future Work

This paper presents the design of the attitude controllers for the Earth remote sensing CubeSat, MicroNimbus. These controllers are developed based on the pointing requirement of the mission. The design methods and theories are explained in this paper. A stability analysis for these controllers is also included in this paper. The commanded attitudes are proved to be stable using the current controller. These controllers are implemented on a NASA GSFC software for simulations. The simulation results show that the controllers are functional. These controllers serve as a baseline design for the flight version controllers on MicroNimbus and other similar CubeSat mission in the Space System Design Lab.

Future work includes sensors and actuators model implementation on the simulation, de-saturation mode implementation for each pointing controllers, controller testing and optimization. The current attitude controller simulation assumes perfect attitude knowledge and perfect control actuation. Sensor noise and biases can be incorporated into the simulation which makes the simulation more realistic. Actuators model will also enhance the fidelity of the simulation. Reaction wheel momentum imbalance, torque and momentum relation, and hardware actuation time are largely influence the control actuation performance. These hardware characteristics should be included in the future simulation model. The most important task is to create a desaturation mechanism for each pointing controller. This mechanism can be done using the three-axes magnetic torquers. Simulation results show that during the nadir pointing mode and the axis-command pointing mode, the angular momenta of reaction wheels change dramatically. If the reaction wheels operate as momentum wheels at a nominal momentum 0.01 Nms, all three wheels will saturate quickly. Thus, the spacecraft losses controllability. Magnetic torquers should be used to unload the momenta throughout any attitude pointing mode. Furthermore, the attitude controllers are not optimized. For example, the $F_R$ used in Sun-Pointing controller and Slew Controller is not unique. Users can optimize the $F_R$ to reduce angular momenta of reaction wheels.

7 Acknowledgments

The authors would also like to thank Dr. Glenn Lightsey, Tanish Himani, Julian Brew, Shez Virani, Andrew Fear and Chris Carter for supporting this work in various ways.

References


[7] Sinclair Interplanetary. Picosatellite Reaction Wheels (RW-0.01), 05 2013. Revision 1.3.

