


	<div><div>UNIVERSITY OF SURREY</div></div>
	PROJECT
	EEEM009 coursework
EEEM009 CubeSat Detumbling Final Report	
	REVISION
	1
	REVISION OR RELEASE DATE
	04/03/2024
	STATUS
	For Validation
PREPARED BY	
Jorge Antonio Chavarin Montoya 6759311	
This document is the property of the University of Surrey and must not be copied or used for any purpose other than that for which it has been supplied.	


 UNIVERSITY OF SURREY	CubeSat Detumbling Final Report	Doc No:	
		Revision: 1	Status: Draft
EEEM009		Revision Release Date: 04/03/24	
		Page 2 of 28	

Table of Contents

ABSTRACT..... 3

ATTITUDE MODEL DESCRIPTION..... 4

DETUMBLING ALGORITHMS OVERVIEW..... 7

RESULTS AND DISCUSSION 9

CONCLUSION..... 14


BIBLIOGRAPHY 15

APPENDIX 16

 Control Algorithm Function 16


 Attitude Model Function..... 18

 Main file Solver.m 21

	CubeSat Detumbling Final Report	Doc No:	
		Revision: 1	Status: Draft
EEM009		Revision Release Date: 04/03/24	
		Page 3 of 28	

ABSTRACT

This paper addresses the modelling and control of satellite attitude dynamics for detumbling, a process aimed at stabilizing the satellite's orientation in space by reducing its rotational velocity to achieve a desired stable orientation. The study examines the utilization of magnetorquers and reaction wheels, acting along each axis (X,Y,Z), as actuators for attitude control. Formulating the problem entails understanding the satellite's rotational dynamics, including knowledge of its moment of inertia, the forces and torques exerted by magnetorquers and reaction wheels, and their physical limitations, as well as the satellite's initial conditions such as rotational velocity and orientation. While sensors for obtaining real-time measurements of the satellite's current orientation and actuator status are crucial for a practical system, they are not considered in this study, as its purpose is to describe state and control equations. The constraints considered for this work include the torque, ± 0.001 Nm, and maximum angular momentum, ± 0.1 Nms, that the reaction wheels can provide, as well as the maximum dipole moment of the magnetorquers, ± 1 Am². The objective of this project is to comprehend the dynamic behaviour of a satellite in orbit, develop detumbling algorithms using magnetorquers and reaction wheels with their respective constraints, and evaluate their performance accordingly.

 UNIVERSITY OF SURREY	CubeSat Detumbling Final Report		Doc No:
			Revision: 1 Status: Draft
EEEM009			Revision Release Date: 04/03/24
			Page 4 of 28

ATTITUDE MODEL DESCRIPTION

Dynamic and kinematic equations are fundamental tools in describing the motion of objects in space, such as satellites. While dynamic equations focus on the relationship between forces and torques applied to an object and how these affect its motion, kinematic equations describe motion without considering the causes that generate it. By looking at how forces influence motion, allows for a detailed and quantitative understanding of the dynamic behaviour of satellites and other physical systems in space. This section explains how dynamic and kinematic equations are used to build an attitude model.

Initially, the dynamic model of a satellite, affected by external torques, is represented by the equation:

$$I\dot{\omega} + \omega \times I\omega = T \quad (1)$$

where, T denotes the cumulative effect of all external torques, including those exerted by magnetorquers and reaction wheels ($T = N_{mt} - T_w$), where T_w is the torque generated by the reaction wheels.

The inertia matrix of the satellite I is defined as:

$$I = \begin{bmatrix} 1.67 \times 10^{-3} & 0 & 0 \\ 0 & 1.67 \times 10^{-3} & 0 \\ 0 & 0 & 1.67 \times 10^{-3} \end{bmatrix} kg \cdot m^2 \quad (2)$$

ω is the three-dimensional vector representing angular velocities of the satellite in the body frame with respect to a fixed inertial frame and $I\omega$ is the angular momentum of the satellite, also represented as h .

Magnetorquers create a magnetic dipole moment (m) by aligning electromagnetic coils the external magnetic field (B), creating a torque used to adjust the orientation of the satellite:

$$N_{mt} = m \times B \quad (3)$$

While reaction wheels spin at varying speeds to generate torque; they operate based on the principle of conservation of angular momentum between the spinning wheels and the body of the satellite. For a satellite with reaction wheels in each axis, the satellite angular momentum can be described as:

$$h = I\omega + h_w, \quad (4)$$

being $h_w = [h_{wx} \ h_{wy} \ h_{wz}]$ is the angular momentum of the reaction wheels.

Thus, substituting equations (3) and (4) in (1), and factorizing the angular velocity derivative, a second-degree equation describing the rotational motion of the satellite can be obtained:

$$\dot{\omega} = I^{-1}((N_{mt} - T_w) - \omega \times (I\omega + h_w)) \quad (5)$$

On the other hand, the kinetic equation describing the quaternion rate of change of the satellite in the body axes is defined as:


$$\dot{\bar{q}} = \frac{1}{2} \begin{bmatrix} 0 & \omega_{Oz} & -\omega_{Oy} & \omega_{Ox} \\ -\omega_{Oz} & 0 & \omega_{Ox} & \omega_{Oy} \\ \omega_{Oy} & -\omega_{Ox} & 0 & \omega_{Oz} \\ -\omega_{Ox} & -\omega_{Oy} & -\omega_{Oz} & 0 \end{bmatrix} \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} \quad (6)$$

Where $\omega_o = [\omega_{ox} \ \omega_{oy} \ \omega_{oz}]$, are the angular velocities of the satellite with respect to the orbital frame.

Using equations (5) and (6)(7) the state vector derivative representation of the system can be written as:

$$\dot{X} = \begin{pmatrix} \dot{\bar{q}} \\ \dot{\omega} \end{pmatrix} = f(X, t) = \begin{bmatrix} \frac{1}{2} \Omega(\omega_o) \bar{q} \\ I^{-1}[T - \omega \times I\omega] \end{bmatrix} \quad (7)$$

Being $\Omega(\omega_o)$ the four-by-four angular velocity matrix used at equation (6).

	CubeSat Detumbling Final Report	Doc No:	
		Revision: 1	Status: Draft
EEM009		Revision Release Date: 04/03/24	
		Page 5 of 28	

The provided MATLAB code is structured to simulate the attitude dynamics of a tumbling satellite. The **attitude_model()** function receives a time (t) and a state vectors (x) as inputs. The latter includes quaternion components, angular velocities, and angular momentum in the three axes, $x = [q_1 \ q_2 \ q_3 \ q_4 \ \omega_x \ \omega_y \ \omega_z \ h_x \ h_y \ h_z]$. The program employs the **ode45** integrator to simulate the satellite's attitude over a specified time duration (t_{max}).

The attitude model function first defines the external parameters, the inertia matrix of the satellite (2) and the external magnetic field in the orbit frame (B_o):

$$B_o = \begin{bmatrix} B_{ox} \\ B_{oy} \\ B_{oz} \end{bmatrix} = \begin{bmatrix} 3 \times 10^{-5} \\ 3 \times 10^{-5} \\ 3 \times 10^{-5} \end{bmatrix} \text{ Tesla}, \quad (8)$$

Next, the state space parameters are defined, which include the quaternion components (q_1, q_2, q_3, q_4), angular velocities of the satellite in the body frame ($\omega_x, \omega_y, \omega_z$), and the angular momentum of the reaction wheel (h_x, h_y, h_z).

Then, the quaternion vector is rescaled to a unit vector, to ensure numerical stability:

$$\bar{q} = \frac{q_1}{\|q\|} \hat{i}, \quad \frac{q_2}{\|q\|} \hat{j}, \quad \frac{q_3}{\|q\|} \hat{k}, \quad \frac{q_4}{\|q\|} \quad (9)$$

Where q_i are the quaternion components and,

$$\|q\| = \sqrt{q_1^2 + q_2^2 + q_3^2 + q_4^2} \quad (10)$$

In order to avoid singularities, the code proceeds to calculate the rotation matrix $A_{O \rightarrow B}$, a 2-1-3 Euler rotation sequence Direction Cosine Matrix that transforms from the orbit frame to the body frame relative can be modelled using a unit quaternion representation (9):

$$A_{O \rightarrow B} = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \quad (11)$$


Once the rotation matrix is determined, the function calculates the magnetic field in the body frame (B_B) by multiplying the rotation matrix with the external magnetic field in the orbit frame.

$$B_B = A_{O \rightarrow B} B_o \quad (12)$$

$$\begin{bmatrix} B_{Bx} \\ B_{By} \\ B_{Bz} \end{bmatrix} = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \begin{bmatrix} B_{ox} \\ B_{oy} \\ B_{oz} \end{bmatrix} \quad (13)$$

The Detumbling **control_algorithm()** function -described in the next section, obtain the magnetic dipole moment (m) from the magnetorquers and the torque (T_w) of the three reaction wheels based in the parameters of time (t), quaternions (q), angular velocities (ω), and the magnetic field in the body field (B_B). The variables obtained by this function are crucial for the stabilization of the satellite.

After calculating the control inputs, the code imposes constraints to ensure they stay within predefined bounds. In the first loop, the magnetic dipole moment (m) is restricted to $\pm 1 \text{ Atm}^2$. In the subsequent loop, the reaction wheel torque is constrained to $\pm 0.001 \text{ Nm}$, with an additional limit of $\pm 0.1 \text{ Nms}$ for the maximum angular momentum to prevent saturation. When the angular momentum nears this threshold and aligns with the torque (T_w), the torque is set to zero. These constraints serve as both design specifications and safety measures, preventing excessive magnetic disturbances from the magnetorquers and avoiding saturation of the reaction wheels. Adjusting these limits provides a safety margin to ensure a stable satellite operation.

	CubeSat Detumbling Final Report	Doc No:	
		Revision: 1	Status: Draft
		Revision Release Date: 04/03/24	
EEEM009		Page 6 of 28	

The code proceeds to calculate the torque generated by the magnetorquer (N_{mt}) using the cross-product operation with the magnetic field (Bb), as in (3).

The satellite exhibits synchronous rotation, which means that the angular velocity along the y-axis of the body frame equals the angular orbital velocity, or mean motion (n), resulting in the z-axis of the body always pointing towards nadir. Therefore, the satellite's angular velocities in the body frame with respect to the orbital frame are equal to the angular velocities of the body with respect to the inertial frame plus the orbital velocities:

$$\omega_O = \omega - A_{O \rightarrow B} \begin{bmatrix} 0 \\ -n \\ 0 \end{bmatrix} \quad (14)$$

With these angular rates obtained, the governing state space equations, dynamic (15) and kinetic (16), are solved:

$$\dot{w} = \text{inv}(I) * (N_{mt} - T - \text{cross}(w', (I * w' + h'))); \quad (15)$$

$$\begin{aligned} \dot{q} = 0.5 * [& \theta \text{wo}(3) - \text{wo}(2) \text{wo}(1); \\ & -\text{wo}(3) \theta \text{wo}(1) \text{wo}(2); \\ & \text{wo}(2) - \text{wo}(1) \theta \text{wo}(3); \\ & -\text{wo}(1) - \text{wo}(2) - \text{wo}(3) \theta] * q'; \end{aligned} \quad (16)$$

The torque is redefined as the angular momentum derivative, to be integrated in the function:

$$\dot{h} = T'; \quad (17)$$

The outputs of the function will be then:


$$\dot{X} = \begin{pmatrix} \dot{q} \\ \dot{\omega} \\ \dot{h}_w \end{pmatrix} \quad (18)$$

Finally, the resulting state vector rates of change (\dot{x}) are returned to the MATLAB environment, completing the simulation cycle. These rates of change capture the dynamic evolution of the satellite's orientation and motion over time, providing valuable insights for analysis and control purposes.

Understanding and applying this theoretical framework are crucial for predicting and controlling satellite behaviour in real-world scenarios. By optimizing orientation and trajectory through accurate modelling and control algorithms, operational risks can be mitigated, ensuring satellite stability in the dynamic space environment.

Magnetorquers and reaction wheels are essential components in satellite and spacecraft attitude (and detumbling) control systems. Magnetorquers, which harness the magnetic field to generate a magnetic moment and influence the satellite's orientation, are effective in low orbits but may be less effective in higher orbits or in the presence of unknown external magnetic fields. On the other hand, reaction wheels, which control the satellite's orientation by adjusting the rotational speed of the wheels, are highly accurate and efficient, but have limitations in the amount of angular momentum they can store and may require periodic desaturation manoeuvres, in addition they could generate extra vibrations to the system. A combination of the two devices is commonly used to exploit their respective advantages and compensate for their limitations, allowing effective orientation control in a variety of orbital situations.

This combination allows the satellite to regain control and restore its operational stability, which is crucial for successful space missions. Magnetorquers slow down the rotation speed by applying an opposing magnetic moment, while reaction wheels restore the desired orientation by precisely controlling the rotation speed.

 UNIVERSITY OF SURREY	CubeSat Detumbling Final Report		Doc No:
			Revision: 1 Status: Draft
EEEM009			Revision Release Date: 04/03/24
			Page 7 of 28

DETUMBLING ALGORITHMS OVERVIEW

Detumbling algorithms are essential for satellites, especially during key moments like deployment or after manoeuvres when the satellite might start spinning unexpectedly. These algorithms are designed to fix the uncontrolled spinning and keep the satellite pointed in the right direction compared to its orbit when possible.

Within the provided function **control_algorithm**, three detumbling strategies are meticulously implemented: two magnetorquer-based controls and a reaction wheel-based control. Each approach offers distinct advantages and is tailored to address specific challenges associated with satellite detumbling, ensuring robust and reliable performance in diverse operational scenarios.

The function accepts the current time (t), the attitude quaternion (q) representing the satellite's orientation on the body frame with respect to the orbital frame, (z-axis pointing towards nadir and y-axis aligned with the orbit anti-normal), rotation rates in the body frame (ω), the accumulated momentum in the reaction wheels (h), and the magnetic field in the body frame (B_b).

Next, the function initializes global variables, sets default outputs for the magnetic dipole moment (M) and torque for momentum wheels (T), and calculates the orbital velocity (n) to be used later in the same function. To activate each algorithm an if/else case has been implemented. The global variable named "control", called at the **solver.m** file, implements the corresponding algorithm.

In the first strategy, the program implements a B-dot controller using a cross-product operation between the angular velocities (ω) and the measured magnetic field vector (B_b), $\dot{B} \propto (\omega \times B)$, resulting in the magnetic dipole moment (m). Lastly, the resulting magnetic dipole moment (M) is determined by a gain factor (k). Adjusting the value of k allows tuning the controller behaviour.

$$M = k * (\omega \times B_b) \quad (19)$$

A second magnetorquer-based B_{dot} strategy is implemented. The algorithm initiates assigning the components of the magnetic field (B_b) in the body frame to the variables B_x , B_y and B_z and normalizes it

into a new vector $B = \|B_b\| = \sqrt{B_x^2 + B_y^2 + B_z^2}$.


Then, the function calculates the orientation of the satellite relative to the external magnetic field. This is achieved by computing the angle (β) between the satellite's Y-axis and the magnetic field vector:

$$\beta = \text{acos}\left(\frac{B_y}{B}\right) \quad (20)$$

Furthermore, the time derivative of β is computed to quantify the rate of change in orientation over time. This is done by evaluating the difference of β between the current and previous time steps, then the actual value of β is stored in a global variable so it can be used in the next iteration. Same steps are done for the time difference. The objective is to obtain $\dot{\beta}$, the time derivative of β :

```
% derivative is equal to difference of Betha at actual time t
% and Betha at previous time t.
dB = (Betha-v1);
v1 = Betha;
% Time difference.
dT = t - v2;
v2 = t;
% Time derivative of Betha
Bdot = dB/dT;
```

(21)

 UNIVERSITY OF SURREY	CubeSat Detumbling Final Report		Doc No:
			Revision: 1 Status: Draft
			Revision Release Date: 04/03/24
EEEM009			Page 8 of 28

The algorithm then proceeds to calculate the corrective magnetic dipole moment (**M**) required to stabilize the satellite. This is achieved through a proportional-derivative (PD) control scheme. The proportional and derivative gains (**K_s** and **K_d**) are predefined parameters that influence the response of the control system. Based on the current state of the satellite and the magnetic field, the PD controller computes the required magnetic dipole moment components (**M_x**, **M_y**, and **M_z**). The **M_y** component conforms the Derivative part of the controller and commands the angular rates of the X and Z-axis to align the satellite Y-axis to the orbit normal. While the **M_x** and **M_y** components control the Y-axis rate of the satellite to a certain angular velocity reference value. The algorithm to obtain the complete magnetic dipole moment on each axis is:

$$M_y = K_D * \dot{\beta} \quad (22)$$

$$M_x = K_s(\omega_y - \omega_{ref}) * \text{sign}(B_z) \text{ for } |B_z| > |B_x| \quad (23)$$

$$M_z = -K_s(\omega_y - \omega_{ref}) * \text{sign}(B_x) \text{ for } |B_x| > |B_z| \quad (24)$$

Where ω_y is the angular velocity in the Y-axis of the satellite in the body frame with respect to the inertial frame and, ω_{ref} is the reference velocity, equal to the orbital mean motion, but in opposite direction to achieve a synchronous rotation.

In contrast, the reaction wheel-based control strategy relies on the dynamic manipulation of angular momentum stored within onboard reaction wheels to stabilize the satellite's attitude. When the **control** parameter is set to "wheels", a target quaternion (**q_t**) is defined, representing the desired orientation of the satellite with respect to the orbital frame. Setting it to **q_t** = [0 0 0 1], indicates that it is aligned to the orbital frame with no rotation, and the z-axis is pointing to the nadir. This quaternion target comes from a global variable (v3) so it could be modified externally. This quaternion then is escalated to a unit quaternion.

The algorithm then computes the error quaternion (**q_e**) by determining the difference between the desired and actual orientations of the satellite. This error measure is key for figuring out what torque is needed to be applied to get the satellite pointing exactly to the desired orientation. The error quaternion is defined by:

$$\mathbf{q}_e = A(\mathbf{q}_t)\mathbf{q}^{-1} = \begin{bmatrix} q_{t4} & q_{t3} & -q_{t2} & q_{t1} \\ -q_{t3} & q_{t4} & q_{t1} & q_{t2} \\ q_{t2} & -q_{t1} & q_{t4} & q_{t3} \\ -q_{t1} & -q_{t2} & -q_{t3} & q_{t4} \end{bmatrix} \begin{bmatrix} -q_1 \\ -q_2 \\ -q_3 \\ q_4 \end{bmatrix} \quad (25)$$

In the next step the matrix of inertia (2) and angular velocities in the body frame with respect to the orbital frame from eq. (14) are calculated.

The torque commands for the reaction wheels, as described in equation (26) are computed using proportional and derivative control gains (**K_p** and **K_d**, respectively) applied to the quaternion error and angular velocities. These control gains are meticulously fine-tuned to modulate the magnitude and direction of the torque applied to the wheels, thereby facilitating precise and effective attitude adjustments while minimizing undesirable oscillations and overshoot.

$$\mathbf{T} = K_p \mathbf{I} \begin{bmatrix} q_{e1} \\ q_{e2} \\ q_{e3} \end{bmatrix} + K_d \mathbf{I} \boldsymbol{\omega}_o - \boldsymbol{\omega} \times (\mathbf{I} \boldsymbol{\omega} + \mathbf{h}) \quad (26)$$

To determine optimal gain values for the three algorithms, initially a trial-and-error approach was utilized to achieve an acceptable system response. This involved attempting to induce a Y-spinning motion in the satellite, with a velocity equal in magnitude but opposite in direction to the mean motion. Moreover, for the reaction wheel, the objective was to establish a final quaternion with all imaginary components set to zero, aligning both body and orbital frames. Subsequently, gains were fine-tuned by selecting those yielding the shortest settling time for the system.

RESULTS AND DISCUSSION

The gains chosen for each controller were initially determined through trial and error. Subsequently, a tuning process was conducted, prioritizing the minimization of transient and settling times while trying to get a system response as smooth as feasible. In other words, these gains were chosen to facilitate the fastest possible detumbling of the satellite, ensuring that it reaches a stable state where the rotational rates are lowered to an accepted and controllable level.

For all controllers, the settling times were found for a settling band of 2%. The Detumbling Time of the system was considered as the highest settling time of the 3-axis angular velocities for the B-dots controllers, whereas the real part of the quaternion stabilization was also included to calculate the settling time of the Reaction Wheels controller.

The first B-dot controller algorithm, the cross-product approach $\dot{\beta} \propto (\omega \times \beta)$, was less accurate than the one calculating B-dot as $\Delta\beta$ and took more time to reach stability, around 5 hours and 40 minutes, (~3.5 orbits). However, the computer resources used were lower. A single for loop was used to simulate a wide range of Gains as shown in Figure 1, where K=146 resulted to give the lower Settling Time of all. Gains out the range of [125,164] didn't detumble the satellite in the simulations.

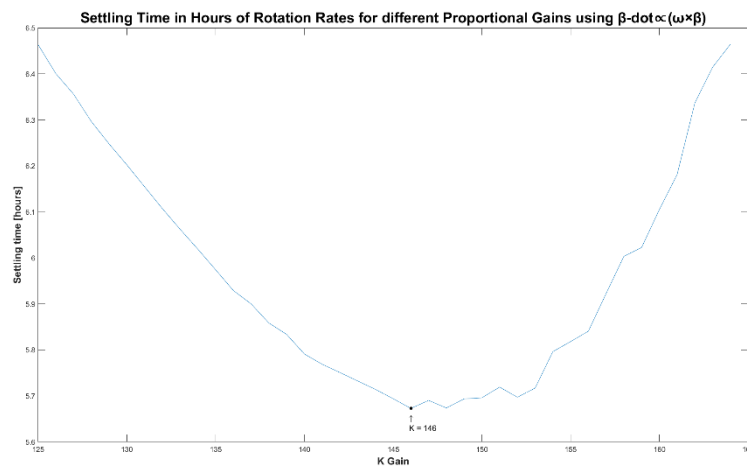


Figure 1. Settling Times using different Proportional Gains using B-dot (Bxw) Controller.

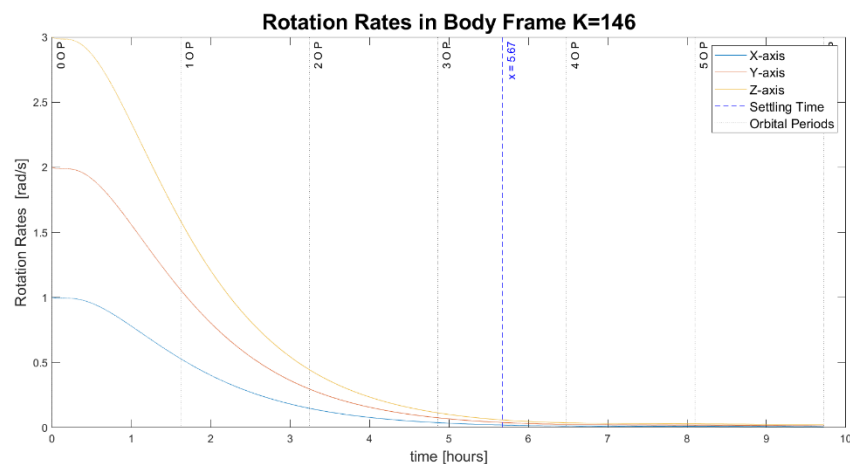


Figure 2. Rotation rates of the satellite in the body frame along orbits using B-dot (Bxw) Controller.

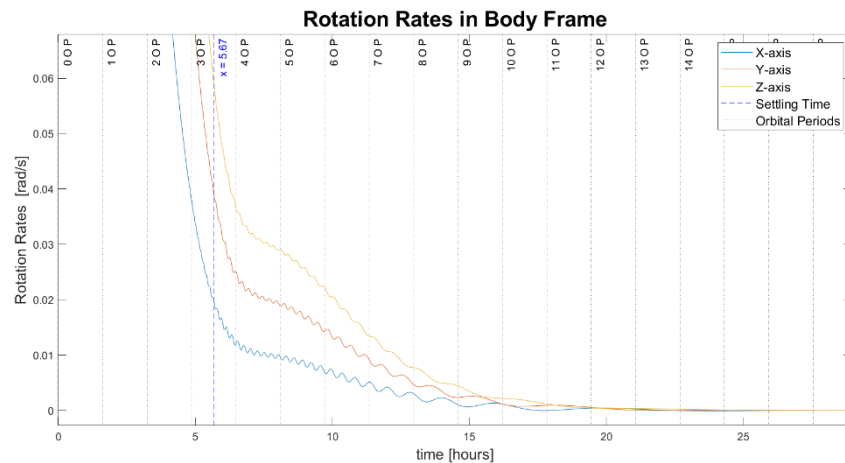


Figure 3. Zoom-in. Rotation rates of the satellite in the body frame along orbits using B-dot (**Bxw**) Controller.

A Gaussian noise was implemented using the **awgn()** MATLAB function to the magnetic field measured in the order of 40 nanoTeslas, with no significant changes in the response of the controller, the average settling time (detumbling) of the system was 20442.6 seconds, after 250 iterations. This response is only 20 seconds slower than the response with ideal measurements. In other words, taking in consideration the overall settling time, the controller could be said is not affected by typical noises on the magnetic field.

For the second algorithm, $\langle \dot{\beta}_y = \frac{\Delta \beta}{\Delta t} \rangle$, the best proportional and derivative gains were obtained through a simulation of nested loops, the Figure 4 shows the settling times (in hours) for different gains values. Best

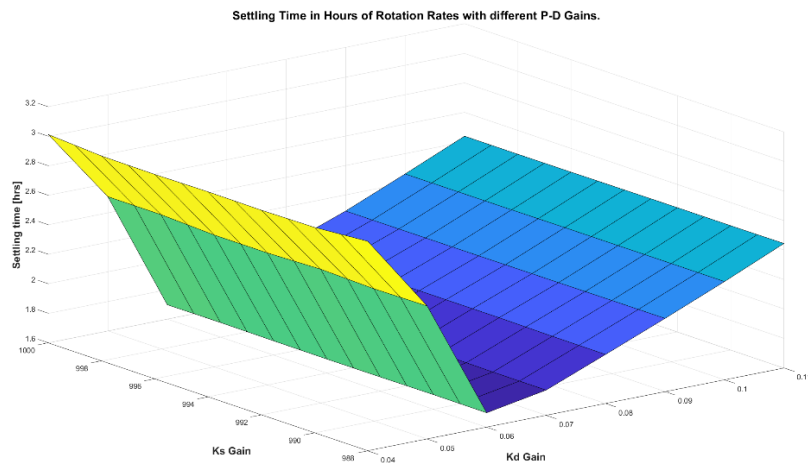


Figure 4. B-dot controller Settling Times of Rotation Rates for different Proportional & Derivative Gains.

values for the proportional gain vary between 0.06 and 0.07; values that increases the influence of rapid changes in error, which help to stabilize the system and reduce response time without causing unwanted oscillations or instability in the system.

The settling time of the satellite detumbling system is approximately 1 hour and 41 minutes (6094 seconds) with a settling band of 2%, as shown in Figure 5, and using a proportional gain $K_s = 990$ and a derivative $K_d = 0.06$ gains. In comparison, the satellite takes approximately 1 hour and 37 minutes (5833 seconds) to complete one orbit. The overall settling time was determined by selecting the highest value from each angular velocity axis.

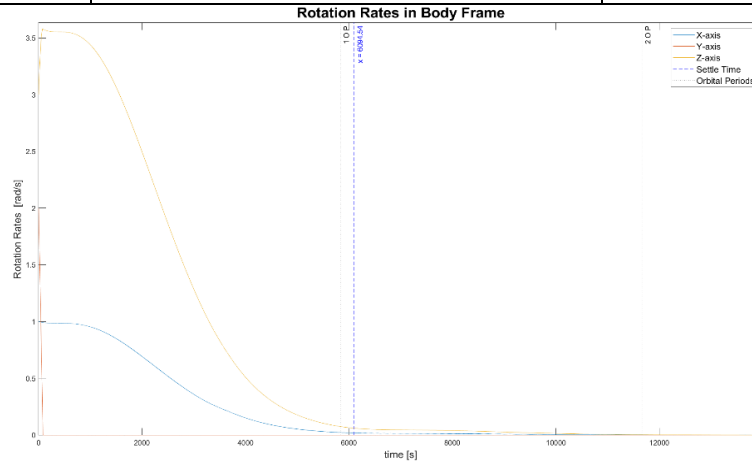


Figure 5. Rotation Rates of Detumbling Satellite in Body Frame. $K_s = 990$ & $K_d = 0.06$. Settling Time = 6094.5 seconds.

Table 1. Settling Times for each pair of Gains of PD controller.

Gains		Settling Times [hours]			
K_s	K_p	ω_x	ω_y	ω_z	Max. Time
990	0.06	1.669	0.023	1.693	1.693
990	0.07	1.703	0.023	1.771	1.771
994	0.06	1.664	0.023	1.703	1.703
994	0.07	1.701	0.023	1.776	1.776
996	0.06	1.668	0.023	1.698	1.698
996	0.07	1.701	0.023	1.776	1.776
1000	0.06	1.665	0.023	1.698	1.698
1000	0.07	1.719	0.023	1.777	1.777

The reaction wheel controller detumbled the satellite a lot faster than the other within only some seconds, Although the initially chosen gains promptly detumbled the satellite (nearly in 4 seconds), they did so abruptly, potentially resulting in undesired oscillations or vibrations in the system, which in turn could induce stress on the satellite components and, in extreme cases, lead to damage.

Therefore, the same strategy as in the second magnetorquer algorithm was used to tune both gains, as seen in Figure 6. The specific pair of gains was selected taking in consideration their Settling Times and smoothness reaction, Figure 7.

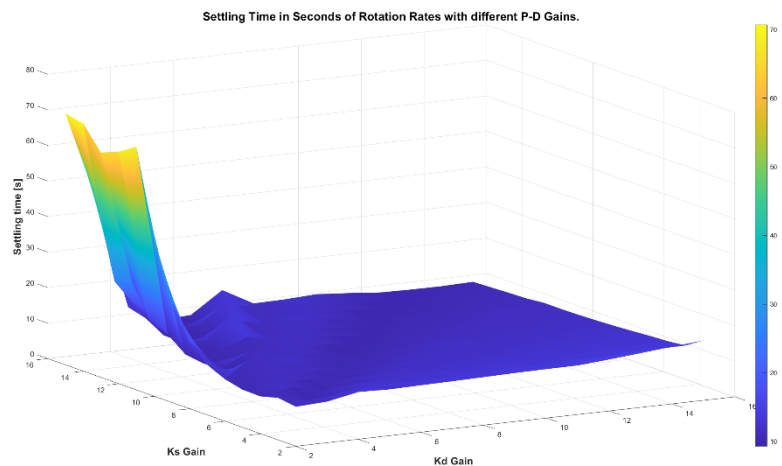


Figure 6. Reaction Wheel Controller Settling Times for Rotation Rates with different Proportional & Derivative Gains.

With this controller, the orientation of the body frame with respect to the orbital frame can be controlled, so the satellite could precisely point to any direction. The Quaternions in Figure 8 demonstrate that the body frame has no rotation with respect to the orbital frame.

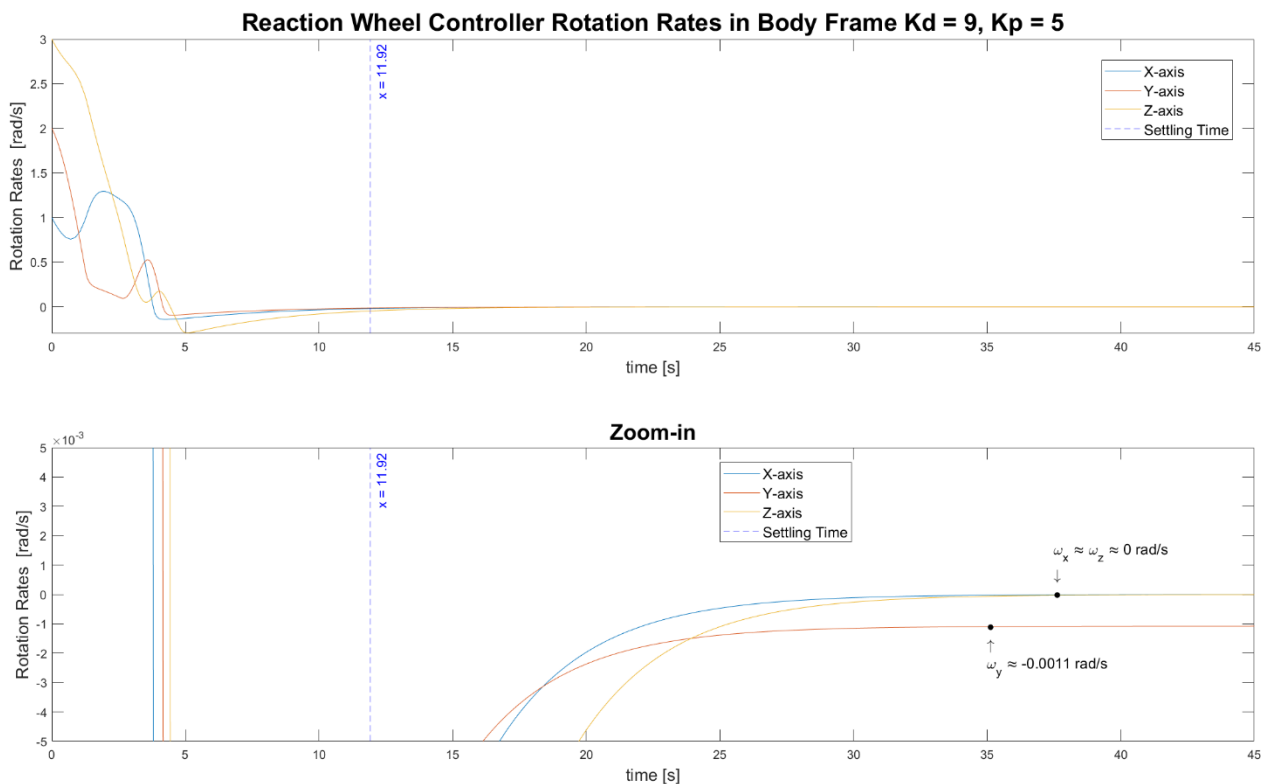


Figure 7. Rotation Rates Stabilization with Reaction Wheels PD Controller

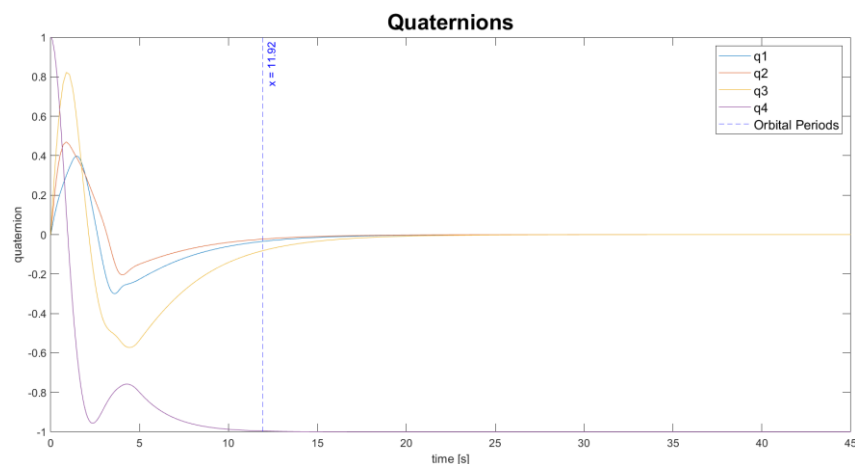


Figure 8. Orientation changes in time of the body axes with respect to the orbital frame described in Quaternions, using the Reaction Wheel controller.

However, when using a single proportional and derivative gain for all three axes, priority is given to the Y-spin. This can be observed in the momentums of the reaction wheels, where the Y-axis momentum remains constant, whereas the momentums, and consequently the angular velocities of the reaction wheels of the other two axes, oscillate to maintain the satellite stable.

When noise was applied to the momentum of the reaction wheels, the detumbling time was not affected, even though in some simulations the stabilization was faster. However, more oscillations in the angular velocities and momentum were registered.

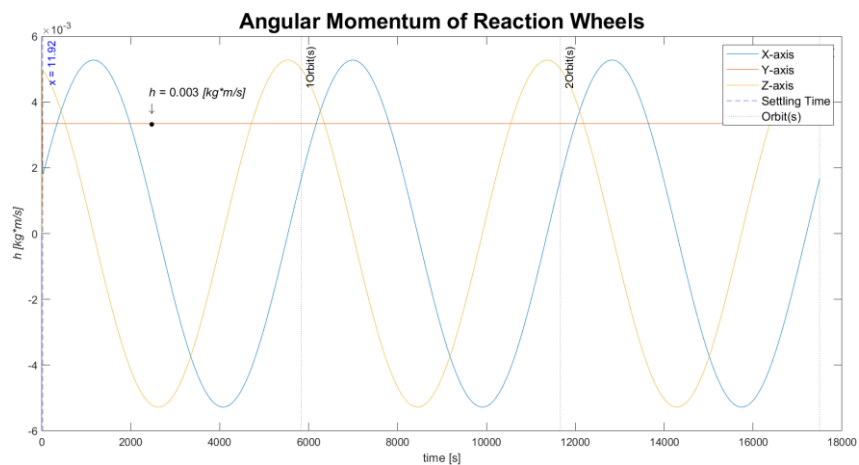


Figure 9. Angular Momentum of Reaction Wheels.

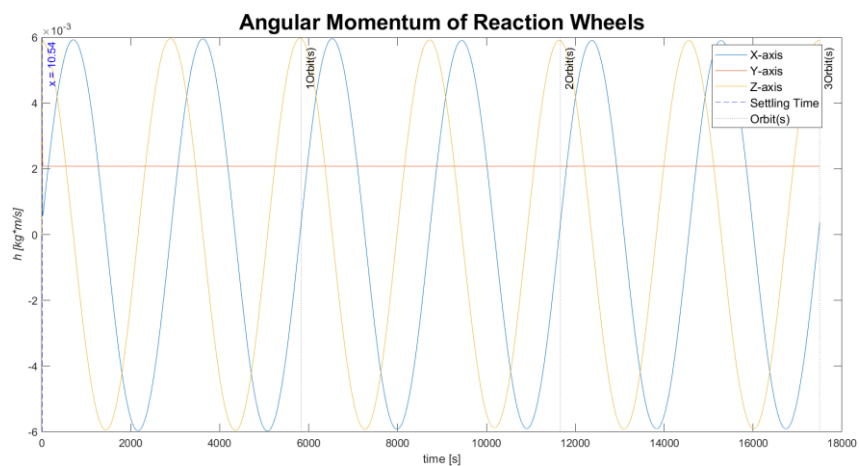


Figure 10. Angular Momentum of Reaction Wheels with noise applied.

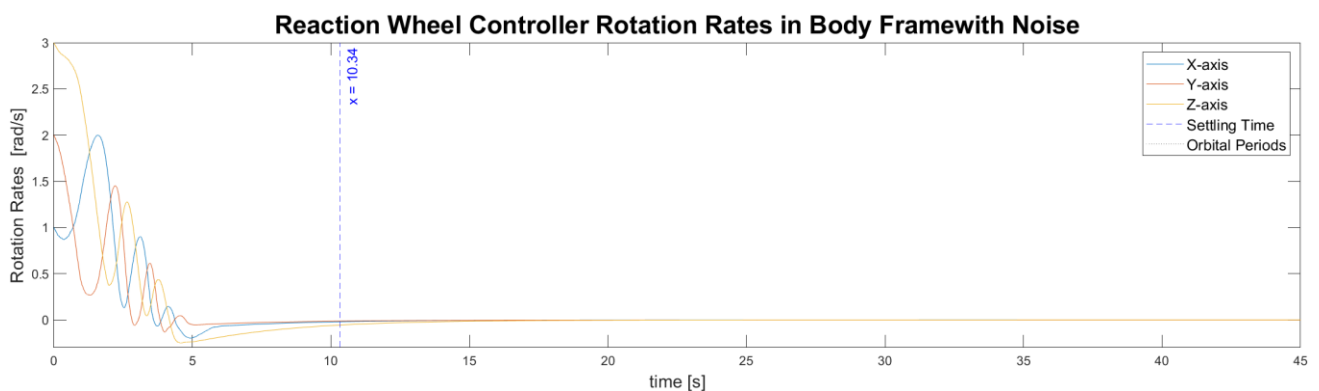




Figure 11. Rotation Rates of the Satellite using Reaction Wheels PD controller with noise applied.

	CubeSat Detumbling Final Report	Doc No:	
		Revision: 1	Status: Draft
EEM009		Revision Release Date: 04/03/24	
		Page 14 of 28	

CONCLUSION


The three detumbling strategies offer unique advantages and trade-offs, making them well-suited for distinct mission scenarios and operational environments. The magnetorquer-based control strategy, with its simplicity and energy efficiency, serves as a reliable option for detumbling in environments with moderate magnetic fields or minimal external disturbances. On the other hand, the reaction wheel-based control strategy, with its superior precision and adaptability, is crucial in scenarios demanding precise control and dynamic responsiveness. However, this comes with a trade-off of higher power consumption and potential mechanical wear over extended mission durations.

In conclusion, the detumbling algorithms presented in this paper offer robust and reliable solutions for stabilizing satellite orientation across a diverse array of operational scenarios. By carefully improving and adjusting these algorithms, they are ready to achieve outstanding results and ensure mission success. This solidifies their crucial role in pushing the boundaries of space exploration and satellite technology.

	CubeSat Detumbling Final Report	Doc No:	
		Revision: 1	Status: Draft
EEEM009		Revision Release Date: 04/03/24	
		Page 15 of 28	

BIBLIOGRAPHY

- Hausman, B. M. (2006). Measuring noise in magnetometers: An example using the Mars Global Surveyor magnetometers. *Journal of Geophysical Research*. doi::10.1029/2005JA011470
- Juchnikowski, G. B. (2013). Optimal control gain for satellite detumbling using B-dot algorithm. *Specialist Conference on Guidance, Navigation & Control, Delf University of Technology*. Retrieved from <https://aerospace-europe.eu/media/books/delft-0025.pdf>
- Liu, K. M. (n.d.). Reaction Wheel Disturbance Modeling, Jitter Analysis, and Validation Tests for Solar Dynamics Observatory. NASA.
- Sidi, M. (1997). *Spacecraft Dynamics and Control: S Practical Engineering Approach*. New York: Cambridge University Press.
- Steyn, W. H. (n.d.). *An Attitude Control Sustem for a Low-Cost Earth Observation Satellite with Orbit Maintenance Cpability*. United Kingdom: Surrey Space Center.
- Visagie, L. S. (2010). Modular Simulation and Visualisation Applivation for Satellite Attitude Control. *62nd Iternational Astronautical Congress*.

	CubeSat Detumbling Final Report	Doc No:	
		Revision: 1	Status: Draft
EEM009		Revision Release Date: 04/03/24	
		Page 16 of 28	

APPENDIX

Control Algorithm Function

% Function: [M T] = control_algorithm(t, q, w, h)

% Programmed by: Jorge Chavarin

% Date: 09/04/2024

function [M T] = control_algorithm(t, q, w, h, Bb)

% Inputs:

% t = time

% q = attitude quaternion with respect to the orbital frame (4-element vector)

% w = rotation rates in body frame (3-element vector)

% h = accumulated momentum in the wheels (3-element vector)

% Bb = magnetic field in body frame (3-element vector)

% Outputs:

% M = commanded magnetic dipole moment (3-element vector)

% T = commanded torque for the momentum wheels (3-element vector)

% Global variable: user defined...

global v1 v2 v3 control ms md wp wd qtg

% Default output

M = [0 0 0]';

T = [0 0 0]';

n = sqrt(3.98e5/7000^3); % orbital angular velocity

% control = "magnetorquer1";

% control = "magnetorquer2";

% control = "wheels";

if control == "magnetorquer1"

% % Bdot ~ k (w x B)

m=cross(w,Bb);

% Controller Gain

k=v3; % Global variable used for tuning

% k=146;

M = (k*m)';

elseif control == "magnetorquer2"

% Activate both algorithm depending on the angular velocity of the satellite

% **if** abs(w(1)) > .1 || abs(w(2)) > 1.05*n || abs(w(3)) > .1

% % Magnetic field in body frame

Bx = Bb(1);

By = Bb(2);

Bz = Bb(3);

% B = sqrt(Bx^2 + By^2 + Bz^2);


B = norm(Bb);

% % Angle between satellite Y axis and magnetic field vector

Betha = acos(By/B);

% % Derivative of B is equal to difference of Betha at actual time t

% % and Betha at previous time t.

	CubeSat Detumbling Final Report	Doc No:	
		Revision: 1	Status: Draft
EEM009		Revision Release Date: 04/03/24	
		Page 17 of 28	

```
dB = (Betha-v1);
```

```
v1 = Betha;
```

```
% % Time difference.
```

```
dT = t - v2;
```

```
v2 = t;
```

```
% % Derivative of Betha with respect to time
```

```
Bdot = dB/dT;
```

```
% % Angular velocities
```

```
wy = w(2);
```

```
wref = -n;
```

```
My=0; Mx =0; Mz = 0;
```

```
% % System Gain Variables
```

```
% % Using global variables for tuning
```

```
Ks=ms; Kd=md;
```

```
% % First approach
```

```
% Ks = 1000; Kd = 0.1;
```

```
% Ks=990;
```

```
% Kd=0.06;
```

```
if dT>0
```

```
    % % B-dot Controller
```

```
    My =Kd * Bdot;
```

```
    % if abs(Bz) > abs(Bx)
```

```
        Mx = Ks*(wy-wref)*sign(Bz);
```

```
    % end
```

```
    % if abs(Bx) > abs(Bz)
```

```
        Mz = -Ks*(wy-wref)*sign(Bx);
```

```
    % end
```

```
end
```

```
M = [Mx, My, Mz]';
```

```
elseif control == "wheels"
```

```
% else
```

```
% % Target Quaternion
```

```
% qt = qtg; % Using global variable qtg for simulation purposes
```

```
qt = [0,0,0,1]';
```

```
% % Target Quaternion Normalized
```

```
scale = sqrt(qt(1)^2 + qt(2)^2 + qt(3)^2 + qt(4)^2);
```

```
qt(1) = qt(1)/scale;
```

```
qt(2) = qt(2)/scale;
```

```
qt(3) = qt(3)/scale;
```


```
qt(4) = qt(4)/scale;
```

```
% % Rotation Matrix from Actual orientation to desired one.
```

```
Aqt = [ qt(4), qt(3), -qt(2), qt(1);
```

```
        -qt(3), qt(4), qt(1), qt(2);
```

```
        qt(2), -qt(1), qt(4), qt(3);
```


	CubeSat Detumbling Final Report	Doc No:	
		Revision: 1	Status: Draft
EEM009		Revision Release Date: 04/03/24	
		Page 20 of 28	

```

% Get the control inputs
[M T] = control_algorithm(t, q, w, h, Bb);

% Place constraints on the control inputs
for i=[1:3]
    if( M(i) > 1 )
        M(i) = 1;
    end
    if( M(i) < -1 )
        M(i) = -1;
    end
end
for i=[1:3]
    if( T(i) > 0.001 )
        T(i) = 0.001;
    end
    if( T(i) < -0.001 )
        T(i) = -0.001;
    end
    if( h(i) > 0.1 && T(i) > 0 )
        T(i) = 0;
    end
    if( h(i) < -0.1 && T(i) < 0 )
        T(i) = 0;
    end
end

% Calculate the torque generated by the magnetorquer
Nmt = cross(M, Bb);


% Calculate the rotation rates in the orbital frame
% wo = inv(Aob)*w';
n = sqrt(3.98e5/7000^3); % orbital angular velocity
wo = w' - Aob*[0;-n;0];

% Solve the governing state space equations
w_dot = inv(I)*(Nmt - T - cross(w', (I*w' + h')));
q_dot = 0.5*[0 wo(3) -wo(2) wo(1);
             -wo(3) 0 wo(1) wo(2);
             wo(2) -wo(1) 0 wo(3);
             -wo(1) -wo(2) -wo(3) 0]*q';
h_dot = T';

% % Concat torque to plot it in solver.m
% torque = cat(1, torque,[t,T(1), T(2), T(3)] );

% Return the result to matlab
dx(1) = q_dot(1);
dx(2) = q_dot(2);
dx(3) = q_dot(3);
dx(4) = q_dot(4);
dx(5) = w_dot(1);
dx(6) = w_dot(2);
dx(7) = w_dot(3);

```

	CubeSat Detumbling Final Report	Doc No:	
		Revision: 1	Status: Draft
EEM009		Revision Release Date: 04/03/24	
		Page 21 of 28	

```
dx(8) = h_dot(1);
dx(9) = h_dot(2);
dx(10) = h_dot(3);
dx = dx';
```

Main file Solver.m

```
% Script: solver(t)
% 2023_24
clear
clc

% Setup the global variables
global v1 v2 v3 control ms md wd wp qtg,

fprintf("Starting...\n");
% Set the simulation duration
% t_max = 100;
orbits = 1;
t_max = orbits*5833;

% % Set an array of time of each orbit period
n = sqrt(3.98e5/7000^3);
orbitPeriod = 2*pi()/n;
orbitPeriods= 1:1:orbits;


% % Select Algorithm
% control = "magnetorquer1";
% control = "magnetorquer2";
control = "wheels";

% % Parameters for calculate Settling Time
% % Parameters [wx wy wz wnorm qt qt]
yinit = [1, 2, 3, norm([1,2,3]), 1, 1]'; % Initial Values
yfinal = [0, -n, 0, n, -1, 1]'; % Stabilized Values

% % Gains
% % Magnetometer #1
v3 = 146;
% % Magnetometer #2
ms=990;
md=0.06;
% % Reaction Wheels
wd= 9;
wp = 5;

% % The next loops are used to test the controllers with several gains to
% % find the best combination of gains.
% % Just one algorithm can be used at a time
% % Each gain global variable in control_algorihm.m file must be
% % uncommented

% % Array to Store Settling times
```

	CubeSat Detumbling Final Report	Doc No:	
		Revision: 1	Status: Draft
EEEM009		Revision Release Date: 04/03/24	
		Page 22 of 28	

ST=[];

```

fprintf("Starting loop...\n");
% % % % M A G N E T O R Q U E R # 1
% % Loop to try different B-dot ~ k (w x B)
% for i=1:250
%     v3=i;
%     disp(i);

% % % % M A G N E T O R Q U E R # 2   Ks   Kd
% % Loop to try different Magnetorquers PD Gains
% for ms=980:1:1010
%     for md=0.01:0.01:.15
%         fprintf('%d, %.2f)\n', ms, md);

% % % R E A C T I O N   W H E E L S
% % Loop to try different PD Gain for Reaction Wheels
% for wd = 9:1:12
%     for wp = 5:1:10
%         fprintf('%d, %d)\n', wd, wp);

% Simulate the attitude of the spacecraft for t seconds duration
[t y] = ode45(@attitude_model, [0 t_max], [0 0 0 1 1 2 3 0 0 0]);

% % Array to store Normalized Angular Velocities of all iterations
norms=[];
for j = 1:size(y,1)
    norms(j) = norm(y(j,5:7));
end

% Array to store values of y from different iterations
% vel = [wx wy wz wnorm qt qt]
vel=[];
vel = cat(2, y(:,5:7), norms',y(:,4), y(:,4));


% Calculate Settling Time
S = lsiminfo(vel(2:end,:),t(2:end), yfinal, yinit);
% S = lsiminfo(vel,t, yfinal, yinit,'SettlingTimeThreshold',0.05);
% S = lsiminfo(vel,t, 'SettlingTimeThreshold',0.02);
STx = S(1).SettlingTime;
STy = S(2).SettlingTime;
STz = S(3).SettlingTime;
STnorm = S(4).SettlingTime;
STq = S(5).SettlingTime;
STq2 = S(6).SettlingTime;
STtotal= max([STx, STy, STz]);
% STtotal= max([STx, STy, STz, STq]);

% % Settling times of Magnetometer #1 B-dot Controller
% ST = cat(1,ST,[v3, STx, STy, STz, STnorm, STq, STq2, STtotal]);

% % Settling times of Magnetometer #2 PD (B-dot) Controller
% ST = cat(1,ST,[md, ms, STx, STy, STz, STnorm, STq, STq2, STtotal]);

% % Settling times of Reaction Wheel PD Controller
ST = cat(1,ST,[wd, wp, STx, STy, STz, STnorm, STq, STq2,STtotal]);

```


	CubeSat Detumbling Final Report	Doc No:	
		Revision: 1	Status: Draft
EEEM009		Revision Release Date: 04/03/24	
		Page 24 of 28	


```

ylabel('rad/s');
xline(S(3).TransientTime, 'r--');
text(S(3).TransientTime+0.1, 0.8, ['x = ', num2str(S(3).TransientTime)], 'Color', 'red');
xline(STz, 'b--');
text(STz+0.5, 0.5, ['x = ', num2str(STz)], 'Color', 'blue');
for i=1:length(orbitPeriods)
    xstr = num2str(orbitPeriods(i))+ " 0 P";
    xline(orbitPeriods(i)*orbitPeriod, 'k:', xstr);
end
legend('Wz', 'Transient Time', 'Settling Time', "Orbital Periods",'FontSize', 12);
title('Z-axis rotation');
hold off;

%% % Three axis
figure(2);
% Plot normal and Zoom version
% subplot(2,1,1);
% xlim([0,45]); ylim([-0.3, 3]);
plot(t, y(:,5));
hold on;
plot(t, y(:,6));
plot(t, y(:,7));
xlabel('time [s]', 'FontSize', 14);
ylabel('Rotation Rates [rad/s]', 'FontSize', 14);
xline(settleTime, 'b--', ['x = ', num2str(round(settleTime,2))], 'FontSize', 11);
for i=1:length(orbitPeriods)
    xstr = num2str(orbitPeriods(i))+ " 0 P";
    xline(orbitPeriods(i)*orbitPeriod, 'k:', xstr);
end
%% % Generic Title
title('Rotation Rates in Body Frame', 'FontSize', 20);
%% % Title when selecting Reaction Wheels
% title({'Reaction Wheel Controller Rotation Rates in Body Frame Kd = 9, Kp = 5'}, 'FontSize', 20);
legend("X-axis", "Y-axis", "Z-axis", "Settling Time", "Orbital Periods", 'FontSize', 12);
% legend("\omega_{x}", "\omega_{y}", "\omega_{z}", "Settling Time", "Orbital Periods", 'FontSize', 14);
hold off

%% % Plot Zoom Verison of Angular Velocities
% subplot(2,1,2);
% plot(t, y(:,5));
% hold on;
% plot(t, y(:,6));
% plot(t, y(:,7));
% xlabel('time [s]', 'FontSize', 14);
% ylabel('Rotation Rates [rad/s]', 'FontSize', 14);
% xline(settleTime, 'b--', ['x = ', num2str(round(settleTime,2))], 'FontSize', 11);
% for i=1:length(orbitPeriods)
%     xstr = num2str(orbitPeriods(i))+ " 0 P";
%     xline(orbitPeriods(i)*orbitPeriod, 'k:', xstr);
% end
% ylim([-0.005, .005]);
% xlim([0,45]);
% txt = {'\rm\uparrow', '\rm\uparrow', '\omega_{y} \approx -0.0011 rad/s', };
% text(35, -.0018, txt, 'FontSize', 12);

```


	CubeSat Detumbling Final Report	Doc No:	
		Revision: 1	Status: Draft
EEEM009		Revision Release Date: 04/03/24	
		Page 25 of 28	

```

% txt1 = {'\omega_{x} \approx \omega_{z} \approx 0 rad/s', '\rm\downarrow', '\rm\bullet',};
% text(37.5, 8e-4,txt1,'FontSize', 12);
% title('Rotation Rates in Body Frame Zoom-in','FontSize', 18);
% legend("X-axis", "Y-axis","Z-axis", "Settling Time","Orbital Periods",'FontSize', 12);
% % legend("\omega_{x}", "\omega_{y}", "\omega_{z}", "Settling Time","Orbital
Periods",'FontSize', 14);
% hold off


% % Quaternions
figure(3);
plot(t, y(:,1))
hold on
plot(t, y(:,2))
plot(t, y(:,3))
plot(t, y(:,4))
xlabel('time [s]');
ylabel('quaternion');
% yline(0, 'k');
xline(settleTime, 'b--', ['x = ', num2str(round(settleTime,2))]);
for i=1:length(orbitPeriods)
    xstr = num2str(orbitPeriods(i))+ " 0 P";
    xline(orbitPeriods(i)*orbitPeriod, 'k:', xstr);
end
legend("q1", "q2", "q3", "q4", "Orbital Periods", 'fontsize', 12)
title('Quaternions','FontSize', 20);
hold off

figure(4);
plot(t, y(:,8));
hold on
plot(t, y(:,9));
plot(t, y(:,10));
xlabel('time [s]');
ylabel('\lith [kg*m/s]', 'FontSize', 11);
% txt = {'\lith \rm= 0.003 \it[kg*m/s]', '\rm\downarrow', '\rm\bullet'};
% text(2400, 0.00385,txt,'FontSize', 11);
% text(2000, 0.0028,txt,'FontSize', 11);
title('Angular Momentum of Reaction Wheels','FontSize', 20);
xline(settleTime, 'b--', ['x = ', num2str(round(settleTime,2))]);
for i=1:length(orbitPeriods)
    xstr = num2str(orbitPeriods(i))+ "Orbit(s) ";
    xline(orbitPeriods(i)*orbitPeriod, 'k:', xstr);
end
legend("X-axis", "Y-axis", "Z-axis", 'Settling Time', 'Orbit(s)', 'FontSize', 11);
hold off

% % 3D Plots to compare Gains responses

% % mat = [ST(:,1:4), ST(:,end)]; % Magnetorquer #1
% % mat = [ST(:,1:5), ST(:,end)]; % Magnetorquer #2
% mat = [ST(:,1:5), ST(:,7),ST(:,end)]; % Reaction Wheels
% ST_NaN = mat(~any(isnan(mat), 2), :);
% if control == "magnetorquer2"
%     kd=ST_NaN(:,1);
%     ks=ST_NaN(:,2);
%     wx=ST_NaN(:,3);


```

	CubeSat Detumbling Final Report	Doc No:	
		Revision: 1	Status: Draft
		Revision Release Date: 04/03/24	
EEEM009		Page 26 of 28	

```

% wy=ST_NaN(:,4);
% wz=ST_NaN(:,5);
% wt=ST_NaN(:,6);
% wtnorm=ST_NaN(:,6)/3600;
% wtorbit=ST_NaN(:,6)/(2*pi()/n);
% wtotal=ST_NaN(:,end);
% wtotalhrs=ST_NaN(:,end)/3600;
% wtotalorbit=ST_NaN(:,end)/(2*pi()/n);
% [X, Y] = meshgrid(min(kd):0.01:max(kd), min(ks):1:max(ks));
% Z = griddata(kd, ks, wtotalhrs, X, Y);
%
% figure(101);
% surf(X,Y,Z);
% xlabel('Kd Gain','FontSize', 14, 'FontWeight', 'bold');
% ylabel('Ks Gain','FontSize', 14, 'FontWeight', 'bold');
% zlabel('Settling time [hrs]','FontSize', 14, 'FontWeight', 'bold');
% title('Settling Time in Hours of Rotation Rates with different P-D
Gains.','FontSize', 20);
%
% Z = griddata(kd, ks, wtotal, X, Y);
% figure(102);
% surf(X,Y,Z);
% xlabel('Kd Gain','fontsize', 14, 'FontWeight', 'bold');
% ylabel('Ks Gain','fontsize', 14, 'FontWeight', 'bold');
% zlabel('Settling time [s]','fontsize', 14, 'FontWeight', 'bold');
% title('Settling Time in Seconds of Rotation Rates with different P-D
Gains.','fontsize', 16);
%
% Z = griddata(kd, ks, wtotalorbit, X, Y);
% figure(103);
% surf(X,Y,Z);
% xlabel('Kd Gain','fontsize', 14, 'FontWeight', 'bold');
% ylabel('Ks Gain','fontsize', 14, 'FontWeight', 'bold');
% zlabel('Settling time [Orbital Periods]','fontsize', 14, 'FontWeight', 'bold');
% title('Settling Time in Orbital Periods of Rotation Rates with different P-D
Gains.','fontsize', 16);
%
% elseif control == "magnetorquer1"
% % [v3, STx, STy, STz, STnorm, STq, STq2]
% kp=ST_NaN(:,1);
% wx=ST_NaN(:,2); wy=ST_NaN(:,3); wz=ST_NaN(:,4);
% wt=ST_NaN(:,end); wthrs=ST_NaN(:,end)/3600; wtorbit=ST_NaN(:,end)/(2*pi()/n);
% STmean=mean(wt);
%
% figure(201);
% plot(kp,wt)
% xlabel('K Gain','fontsize', 14, 'FontWeight', 'bold');
% ylabel('Settling time [seconds]','fontsize', 14, 'FontWeight', 'bold');
% title('Settling Time in seconds of Rotation Rates for different Proportional
Gains.','fontsize', 14);
%
%
% figure(202);
% plot(kp,wthrs)
% xlabel('K Gain','FontSize', 14, 'FontWeight', 'bold');
% ylabel('Settling time [hours]','FontSize', 14, 'FontWeight', 'bold');


```

	CubeSat Detumbling Final Report		Doc No:	
			Revision: 1	Status: Draft
			Revision Release Date: 04/03/24	
EEEM009			Page 27 of 28	

```

%      txt = {'\bullet', '\uparrow', 'K = 146'};
%      text(145.9, 5.653,txt,'FontSize', 11);
%      title('Settling Time in Hours of Rotation Rates for different Proportional
Gains.','FontSize', 20);
%
%
%      figure(203);
%      plot(kp,wtorbit)
%      xlabel('K Gain','fontsize', 14, 'FontWeight', 'bold');
%      ylabel('Settling time [Orbital Periods]','fontsize', 14, 'FontWeight', 'bold');
%      title('Settling Time in Orbital Periods of Rotation Rates for different
Proportional Gains.','fontsize', 14);
%
%
%      figure(204);
%      plot(kp,wy)
%      xlabel('K Gain','fontsize', 14, 'FontWeight', 'bold');
%      ylabel('Settling time in hours','fontsize', 14, 'FontWeight', 'bold');
%      title('Settling Time in Hours of Y-Axis Rotation Rate for different Proportional
Gains.','fontsize', 14);
%
%
% elseif control == "wheels"
%      fprintf("Executing IF_ELSE Wheel...");
%      % k=1;
%      % [wd, wp, STx, STy, STz, STnorm, STq, STq2,STtotal]
%      kd=ST_NaN(:,1);
%      ks=ST_NaN(:,2);
%      wx=ST_NaN(:,3);
%      wy=ST_NaN(:,4);
%      wz=ST_NaN(:,5);
%      wnorm=ST_NaN(:,6);
%      qt=ST_NaN(:,7);
%      STtotal=ST_NaN(:,end);
%      STtotalmin=ST_NaN(:,end)/60;
%      [X, Y] = meshgrid(min(kd):0.01:max(kd), min(ks):1:max(ks));
%      Z = griddata(kd, ks, STtotalmin, X, Y);
%
%      figure(301);
%      ss1= surf(X,Y,Z);
%      ss1.EdgeColor = 'none';
%      colorbar
%      xlabel('Kd Gain','fontsize', 14, 'FontWeight', 'bold');
%      ylabel('Ks Gain','fontsize', 14, 'FontWeight', 'bold');
%      zlabel('Settling time [min]','fontsize', 14, 'FontWeight', 'bold');
%      title('Settling Time in Minutes of Rotation Rates with different P-D
Gains.','fontsize', 16);
%
%
%      Z = griddata(kd, ks, STtotal, X, Y);
%      figure(302);
%      ss2 = surf(X,Y,Z);
%      ss2.EdgeColor = 'none';
%      colorbar
%      xlabel('Kd Gain','fontsize', 14, 'FontWeight', 'bold');
%      ylabel('Ks Gain','fontsize', 14, 'FontWeight', 'bold');

```

	CubeSat Detumbling Final Report	Doc No:	
		Revision: 1	Status: Draft
EEEM009		Revision Release Date: 04/03/24	
		Page 28 of 28	

```
%      xlabel('Settling time [s]','fontsize', 14, 'FontWeight', 'bold');
%      title('Settling Time in Seconds of Rotation Rates with different P-D
Gains.','fontsize', 16);
%
%
% end

fprintf("THE END.\n")
```