DEGREE PROJECT

Simulations of Satellite Attitude Maneuvers - Detumbling and Pointing

Lisa Jonsson

Space Engineering, master's level
2019

Luleå University of Technology
Department of Computer Science, Electrical and Space Engineering

LULEÅ UNIVERSITY OF TECHNOLOGY

Acknowledgements

I wish to thank various people for their contribution to this project; my supervisors Dr. Leonard Felicetti and Mr. Jakob Viketoft for their expert advice and proffesional guidance; Mr. Stefan Strålsjö for the possibility of writing my Master's thesis at ÅAC; my family and friends for their encouragement.

Abstract

For all satellites, attitude, positioning and orbit propagation calculations are vital for the success of a mission. The attitude control system usually have different modes for different parts of the mission. Two of these modes, detumbling and pointing, will be considered here. The aim of this thesis is to build a simulation environment and implement the attitude controllers for a small satellite in low earth orbit. The simulation background is set up in MATLAB, the wellknown B-dot algorithm is used as detumbling controller and it is successfully implemented in C. A literature review on pointing controllers available today is carried out, and a linear quaternion feedback controller, a nonlinear quaternion feedback controller, and a sliding-mode controller are chosen to be tested for a nadir pointing maneuver in terms of time efficiency, stability and robustness. These tests are carried out for two cases, the first case simulating a 3u CubeSat and the second case simulating a small satellite of 50 kg. For the CubeSat, the linear quaternion feedback controller was the best one, in terms of time efficiency, stability and robustness, the nonlinear quaternion feedback controller almost as good, and the performance of the sliding-mode controller was not satisfactory due to chattering. For the small satellite case the sliding-mode controller gave the most satisfactory result, neither of the quaternion feedback controllers could yield as good pointing accuracy as the sliding-mode controller.

Contents

Acronyms

AOCS - Attitude and Orbital Control System

DCM - Direction Cosine Matrix

ECI - Earth Centered Inertial Coordinate System

LEO - Low Earth orbit

NED - North East Down Coordinate System

OBC - On Board Computer

SBRF - Satellite Body Reference Frame

1 Introduction

For all satellites, attitude, positioning and orbit propagation calculations are vital for the success of a mission. Attitude and orbital control systems usually have different modes for different parts of the mission. Two of these modes, detumbling and pointing, will be considered here. When a satellite is separated from its launcher, it tumbles freely in an uncontrolled way. The solar panels now need to align roughly with the sun so the on-board batteries can charge. By controlling the satellite such that it follows the magnetic field around the Earth, it can be detumbled. Once the on-board batteries are charged, a more complex attitude control system is used for operation. It is often desirable for the satellite to point in a certain direction.

The aim of this thesis is to build a simulation environment and implement the controllers to test their behavior for a small satellite in low earth orbit. The simulation background is set up in MATLAB and the modeling of the environment is described in Ch.2. Because detumbilng and pointing are two separate maneuvers they are also separated into two different chapters. Ch.3 contains a background, a description of the chosen detumbling controller, design specification, software module specifications, simulation results and a conclusion. In Ch. 4 a literature review of pointing algorithms is presented, the chosen pointing controllers are then described more in detail and the simulation results are shown.

2 Modelling

2.1 Reference Frames

For describing the attitude and the orbit of the satellite, different reference frames are needed.

2.1.1 Earth Centered Inertial Coordinate System (ECI)

The ECI reference frame is defined as a right handed Cartesian coordinate system having its origo in the center of the Earth. The x-axis is aligned with the vernal equinox, the z-axis is orthogonal to the equatorial plane with the positive direction passing though the North Pole. The y-axis is the cross product of the z-axis and the x-axis. The ECI coordinate system is illustrated in Fig.1

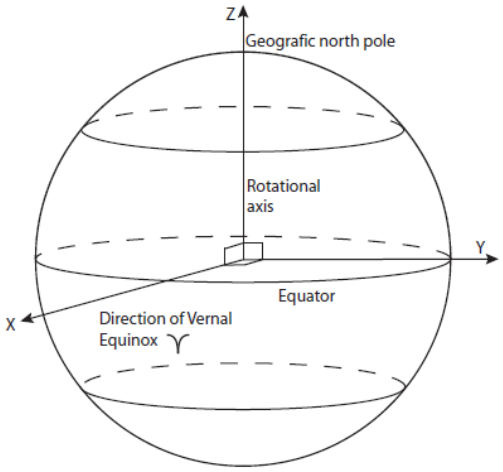

Figure 1: The Earth Centered Inertial Reference Frame (Attitude Control Sysyem for AAUSAT-II, Aalborg University, 2005, p.22)

2.1.2 Satellite Body Reference Frame (SBRF)

The Satellite Body Reference Frame is a right handed Cartesian coordinate system where the axes are fixed to the body axes of the satellite, and it is used to determine the orientation of the satellite. Here, origo is located in the center of mass of the satellite.

2.1.3 North East Down Coordinate System (NED)

Given the ECI position of the satellite $\vec{r}_{ECI}$, the downward (nadir) direction in the NED coordinate system can be defined as

$$\hat{\mathbf{D}} = -\frac{\vec{r}_{ECI}}{|\vec{r}_{ECI}|} \tag{1}$$

The east direction,$\hat{\mathbf{E}}$, can be found by calculating the angle λ from the four-quadrant-inverse-tangent of the first and second components of the position vector $\vec{r}_{ECI}$=$[r_1\ r_2\ r_3]$.

$$\lambda = atan2(r_2, r_1) \tag{2}$$

$$\hat{\mathbf{E}} = \begin{bmatrix} -sin(\lambda) \\ cos(\lambda) \\ 0 \end{bmatrix} \tag{3}$$

The north direction, $\hat{\mathbf{N}}$, is given by the cross product

$$\hat{\mathbf{N}} = \hat{\mathbf{E}} \times \hat{\mathbf{D}} \tag{4}$$

The rotation matrix, $\mathbf{R}_{ECI}^{NED}$, which rotates a vector from NED to the ECI coordinate system, can be formed

$$\mathbf{R}_{ECI}^{NED} = \begin{bmatrix} \hat{\mathbf{N}} & | & \hat{\mathbf{E}} & | & \hat{\mathbf{D}} \end{bmatrix} \tag{5}$$

2.2 Environmental Disturbances

The environmental disturbances considered here are the aerodynamic effects, the gravity gradient effects and the solar pressure effects. The total external disturbance torque in the Satellite Body Reference Frame is given by

$$\mathbf{N}_{ext} = \mathbf{N}_{drag} + \mathbf{N}_{GG} + \mathbf{N}_R \tag{6}$$

where $\mathbf{N}_{drag}$ denotes the torque produced from aerodynamic effects, $\mathbf{N}_{GG}$ denotes the torque produced from gravity gradient effects and $\mathbf{N}_R$ denoted the torque produced by radiation effects [1]. The calculation of these disturbance torques are described below.

2.2.1 Aerodynamic Disturbance Torque

For a satellite below 400 km, the aerodynamic torque is the dominant environmental disturbance torque. The interaction between the upper atmosphere and a satellite produces a torque around the center of mass. The aerodynamic disturbance force on a surface element A is given by

$$f_{drag} = -\frac{1}{2}\rho C_d A \left\|\mathbf{V}\right\|^2 \hat{\mathbf{V}} \tag{7}$$

where ρ is the atmospheric pressure (here MATLAB'S built-in function atmosnrlmsise00 is used to calculate ρ) and C_d is the drag coefficient. When no measured value of C_d is available it may be set to 2 according to [2, Ch. 17.2.3]. $\mathbf{V}$ is the magnitude of the velocity of the spacecraft in the ECI coordinate system. The aerodynamic torque is given by

$$N_{drag} = C_p \times f_{drag} \tag{8}$$

where C_p is a vector from the center mass to the center of pressure of the satellite.[2, Ch.17.2.3]

2.2.2 Gravity Gradient Torque

Because of the variation in the Earth's gravitational filed, any non-symmetrical object in orbit is subject to a gravitational torque. This gravity-gradient torque results from the inverse square of the gravitational force field:

$$N_{GG} = \frac{3\mu_e}{\left\|\mathbf{R}_{SA}\right\|^3}\left(\hat{\mathbf{R}}_{SA} \times (I \cdot \hat{\mathbf{R}}_{SA})\right) \tag{9}$$

where μ_e is the gravitational constant multiplied with the mass of Earth, $mathbf{R}_{SA}$ is a vector from the center of the Earth to the satellite and I is the moment-of-inertia tensor of the satellite.[2, Ch. 17.2.1]

2.2.3 Radiation Disturbance Torque

Radiation hitting the surface of the satellite produces a force which results in a torque about the satellite's center of mass. The solar radiation varies as the inverse square of the distance from the sun, and therefore the solar radiation pressure is altitude independent. Major sources of electromagnetic radiation pressure are solar illumination, solar radiation reflected from the earth and radiation emitted from the Earth and its atmosphere. Since the radiation from the Earth is small compared to the solar radiation, the radiation emitted from the Earth will not be considered. The solar radiation force is given by

$$F_R = -KAP\hat{\mathbf{R}}_{SA\rightarrow S} \tag{10}$$

where K is a dimensionless constant in the range of $0 \leqslant K \leqslant 2$, depending on the amount of sunlight reflected by the satellite. $\hat{\mathbf{R}}_{SA\to S}$ is a vector from the satellite to the sun, A is the cross sectional area of the satellite perpendicular to $\hat{\mathbf{R}}_{SA\to S}$ and P is the momentum flux from the sun given by $P = 4.4 \cdot 10^{-6}$ $[kg/ms^2]$.
The disturbance torque in SBRF is given by

$$N_R = F_R \times R_{COM} \tag{11}$$

where R_{COM} is a vector from the center of mass to the geometrical center of the satellite.[1, Chapter 2.3.4]

2.3 Modelling Eclipse

The satellite is only affected by solar radiation torques when the satellite illuminated by the sun. Determining if the satellite is in eclipse can be done by comparing the angles $\pm\Theta$ and Ψ.

$$\Theta = \pm arcsin\left(\frac{R_\oplus}{\|\mathbf{R}_{s/c}\|}\right) \tag{12}$$

$R_{s/c}$ is a vector from the center of the Earth to the satellite and $R_\oplus$ is the radius of the Earth.

$$\Psi = arccos(\hat{\mathbf{R}}_{\oplus\to\odot} \cdot \hat{\mathbf{R}}_{s/c}) \tag{13}$$

$R_{\oplus\to\odot}$ is a vector from the center of the Earth to the center of the sun. The satellite is in eclipse if $|\Psi| \leqslant |\theta|$. [2, Ch 3.5]

2.4 Orbit Propagator

For modeling of the position and velocity of the satellite, an orbit propagator must be used. A Kepler orbit model is used here, not taking into account effects of Earth's oblateness or the satellite drag.

2.5 Magnetic field model

In order to simulate the behavior of magnetometers and magnetotorquers a model of Earth's magnetic field is needed. Here MATLAB's built in function *wrldmagm* from the Aerospace Toolbox is used. The function is based on the World Magnetic Model for 2015-2020 from NOAA (North Oceanic and Atmospheric Administration) [3].

2.6 Attitude Transformations

2.6.1 Direction Cosine Matrix

The attitude of a three-dimensional body is conveniently described with a set of axes fixed to the body. The basic three-axis attitude transformation is based on the direction cosine matrix (DCM). Here, the 3-2-1, or ψ-θ-ϕ, Euler angle rotation sequence is used, which gives the following direction cosine matrix [4].

$$[A_{321}] = \begin{bmatrix} c\theta c\psi & c\theta s\psi & -s\theta \\ -c\phi s\psi + s\phi s\theta c\psi & c\phi c\psi + s\phi s\theta s\psi & s\phi c\theta \\ s\phi s\psi + c\phi s\theta c\psi & -s\phi c\psi + c\phi s\theta s\psi & c\phi c\theta \end{bmatrix} \tag{14}$$

2.6.2 Quaternions

The quaternion is a number system that extends the real numbers into a four-dimensional domain similarly as the complex numbers extends the real numbers into a two-dimensional domain. Quaternions can be used to describe rotations of coordinate systems, and they provide a singularity-free representation of kinematics. The quaternion is defined in the following way

$$\mathbf{q} = q_4 + \hat{\mathbf{i}}q_1 + \hat{\mathbf{j}}q_2 + \hat{\mathbf{i}}q_3 \tag{15}$$

where q_1,q_2,q_3 and q_4 are real numbers and $\hat{\mathbf{i}}$,$\hat{\mathbf{j}}$ and $\hat{\mathbf{k}}$ are unit vectors. These unit vectors satisfy the following conditions [4].

$$\begin{aligned} &\hat{\mathbf{i}}^2 = \hat{\mathbf{j}}^2 = \hat{\mathbf{k}}^2 = -1 \\ &\hat{\mathbf{i}}\hat{\mathbf{j}} = -\hat{\mathbf{j}}\hat{\mathbf{i}} = \hat{\mathbf{k}} \\ &\hat{\mathbf{j}}\hat{\mathbf{k}} = -\hat{\mathbf{k}}\hat{\mathbf{j}} = \hat{\mathbf{i}} \\ &\hat{\mathbf{k}}\hat{\mathbf{i}} = -\hat{\mathbf{i}}\hat{\mathbf{k}} = \hat{\mathbf{j}} \end{aligned} \tag{16}$$

2.6.3 The DCM and quaternions

The DCM can be expressed in quaternions,

$$A_{\mathbf{q}} = \begin{bmatrix} q_1^2 - q_2^2 - q_3^2 + q_4^2 & 2(q_1q_2 + q_3q_4) & 2(q_1q_3 - q_2q_4) \\ 2(q_1q_2 - q_3q_4) & -q_1^2 + q_2^2 - q_3^2 + q_4^2 & 2(q_2q_3 + q_1q_4) \\ 2(q_1q_3 + q_2q_4) & 2(q_2q_3 - q_1q_4) & -q_1^2 - q_2^2 + q_3^2 + q_4^2 \end{bmatrix} \tag{17}$$

If the DCM is expressed as

$$A_{rw} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{18}$$

the quaternions can also be found from from the elements of the DCM.

$$\begin{aligned} q_4 &= \pm\sqrt{1 + a_{11} + a_{22} + a_{33}} \\ q_1 &= 0.25(a_{23} - a_{32})/q_4 \\ q_2 &= 0.25(a_{31} - a_{13})/q_4 \\ q_3 &= 0.25(a_{12} - a_{21})/q_4 \end{aligned} \tag{19}$$

When q_4 is very small, numerical inaccuracies may occur when calculating the remaining components. This problem can be solved by adding three other equation sets, and always using the set for which the divisor component is maximal. All of the equation sets are mathematically equivalent [4].

$$\begin{aligned} q_1 &= \pm\sqrt{1 + a_{11} - a_{22} - a_{33}} \\ q_2 &= 0.25(a_{12} - a_{21})/q_4 \\ q_3 &= 0.25(a_{13} - a_{31})/q_4 \\ q_4 &= 0.25(a_{23} - a_{32})/q_4 \end{aligned} \tag{20}$$

$$\begin{aligned} q_3 &= \pm\sqrt{1 - a_{11} - a_{22} + a_{33}} \\ q_1 &= 0.25(a_{13} - a_{31})/q_4 \\ q_2 &= 0.25(a_{23} - a_{32})/q_4 \\ q_4 &= 0.25(a_{12} - a_{21})/q_4 \end{aligned} \tag{21}$$

$$\begin{aligned} q_2 &= \pm\sqrt{1 - a_{11} + a_{22} - a_{33}} \\ q_1 &= 0.25(a_{12} - a_{21})/q_4 \\ q_3 &= 0.25(a_{23} - a_{32})/q_4 \\ q_4 &= 0.25(a_{31} - a_{13})/q_4 \end{aligned} \tag{22}$$

2.7 Dynamics & Kinematics

2.7.1 Magnetic actuation

When using magnetic actuation, such as magnetotorquers, the attitude dynamics of the satellite can be described as

$$\mathbf{J}\dot{\omega} = \omega \times \big(\mathbf{J}\omega\big) + \mathbf{T}_{coils} + \mathbf{T}_{ext} \tag{23}$$

where ω is the angular rate of the satellite, expressed in SBRF. $\mathbf{J}$ is the total inertia matrix of the spacecraft, $\mathbf{T}_{coils}$ is the vector of magnetic torques produced by magnetotorquers and $\mathbf{T}_{ext}$ is the vector of disturbance torques [2]. The magnetic moment of the magnetotorquers can be described as

$$m_{mt} = nIA \tag{24}$$

where n is the number windings in the coil, I is the current and A area enclosed by the magnetotorquers.
The attitude kinematics is described by

$$\dot{\mathbf{q}} = \frac{1}{2}\Omega(\omega)\mathbf{q} \tag{25}$$

where $\Omega(\omega)$ is a 4x4 skew symmetric matrix defined in below:

$$\Omega(\omega) = \begin{bmatrix} 0 & \omega_3 & -\omega_2 & \omega_1 \\ -\omega_3 & 0 & \omega_1 & \omega_2 \\ \omega_2 & -\omega_1 & 0 & \omega_3 \\ -\omega_1 & -\omega_2 & -\omega_3 & 0 \end{bmatrix}, \omega \in \mathbb{R}^3 \tag{26}$$

2.7.2 Reaction Wheel actuation

The attitude dynamics, of a satellite with reaction wheels, can be modeled as

$$\dot{\omega} = -\mathbf{J}^{-1}(\omega \times \mathbf{J}\omega) - \mathbf{J}^{-1}(\omega \times \mathbf{h}) + \mathbf{J}^{-1}\mathbf{T}_w + \mathbf{J}^{-1}\mathbf{T}_e \tag{27}$$

where **J**is the total inertia matrix of the spacecraft, now also including the reaction wheels, $\mathbf{h}$ is the angular momentum of the reaction wheels, $\mathbf{T}_w$ is the torque produced by the reaction wheels acceleration and $\mathbf{T}_{ext}$ is the external disturbance torque. The angular momentum of the reaction wheels is given below [5].

$$\dot{\mathbf{h}} = -\mathbf{T}_w \tag{28}$$

3 Detumbling

When a satellite separates from it's launcher, it tumbles in space in an uncontrolled manner. Before the satellite can begin its nominal operation, the angular velocity has to be reduced. The purpose of the detumbling controller is to dampen the angular velocity vector of the satellite. A satellite in LEO with an altitude of 400 km is considered detumbled when its angular rate has decreased to $0.13°/s$ or less [1].
To test the detumbling algorithm, a modeling environment is to be set up in MATLAB. The algorithm is to be written in C, and with some modifications they shall be able to be executed on ÅAC's target environment OBC.

There exists a number of different detumbling controllers and some of them are tested in [6] for a 2U CubeSat. Controllers based on magnetometer feedback, gyroscope feedback, sun sensor feedback and passive control are compared in terms of maneuver time, precision, control effort, robustness and implementation. The magnetometer feedback B-dot controller was concluded to be the best and easiest controller to implement for a 2U CubeSat. The B-dot controller is chosen in this project because it is the easiest and most common detumbling controller for small satellites.

3.1 The B-dot Controller

The torque produced by magnetic torquers is given by

$$\mathbf{L} = \mathbf{m} \times \mathbf{B} \tag{29}$$

where $\mathbf{m}$ is the commanded magnetic dipole moment generated by the torquers and $\mathbf{B}$ is the local geomagnetic field expressed in body-frame coordinates. Detumbling from arbitrary initial spinning of the satellite is accomplished by commanding the magnetic dipole moment of the magnetometers by

$$\mathbf{m} = -\frac{k}{\|\mathbf{B}\|}\dot{\mathbf{B}} \tag{30}$$

where K is a positive scalar gain [7]

3.2 Design Specification

A simulation framework in MATLAB, with the structure of the block diagram in Fig.2, shall be built. This framework shall include an orbital propagator, an attitude propagator, a function to calculate attitude disturbance torques and a function that acts as a magnetometer. Orbital, satellite and hardware parameters (such as magnetotorquer size) shall be easy to change. The control loop is written in C and it shall:

- get the magnetometer readings (from the simulation of magnetometers in the MATLAB workspace)
- calculate the control torque needed to detumble
- calculate the magnetotorquer current needed to detumble
- set the magnetotorquer current (put the current values into the MATLAB simulation workspace)

The control loop is a C-source function that shall be called from the main program in MATLAB. Therefore the control loop file must also contain a gateway function (mexFunction) between C and MATLAB. The control loop function can the be called like a regular MATLAB function from the MATLAB workspace. The four C-functions run by the control loop are regular C-source functions. (For more information on MEX-files, see the **MATLAB documentation on MEX-files.**)
All MATLAB variables are by default double-precision floating point variables and stored as mxArrays. The mxArray is the fundamental MATLAB type in C. Though MATLAB supports single-precision floating-point and 8-, 16-, 32-, and 64-bit integers, both signed and unsigned.

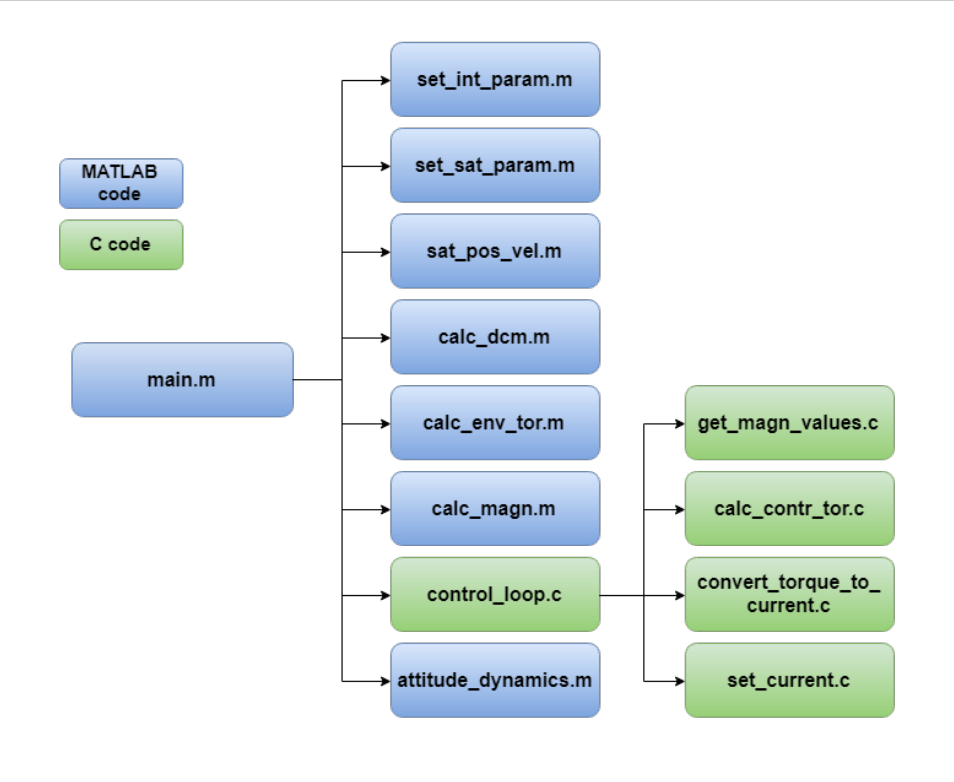

Figure 2: A block diagram of the structure of the simulation framework together with the detumbling controller.

3.3 Module specifications

Here, the function of all the modules from the block diagram in Fig. 2 are described and the input and output parameters of every block are shown.

main.m

Simulation main program: MATLAB script to run the simulation, the script should also plot the position, attitude, velocity, angular rate, control and disturbance torques.

set_int_param.m

MATLAB-function where the integration parameters for the simulation are defined and set. The parameters can easily be changed inside the function before running the simulation.
The function takes no input and the output is the following parameters:

set_int_param.m			
IN/OUT	Name	Description	Type
OUT	dt	Simulation time step [s]	double
OUT	t0	Simulation start time [s]	double
OUT	tf	Simulation end time [s]	double
OUT	time	Simulation time vector [s]	double

Table 1: IN/OUT parameters of set_int_param.m

set_sat_param.m

MATLAB-function where the satellite properties, orbital parameters and the hardware parameters are defined and set. The parameters can easily be changed inside the function before running the simulation. The function takes no input and the output is the following parameters:

set_sat_param.m			
IN/OUT	Name	Description	Type
Satellite parameters			
OUT	SAT.MASS	Total satellite mass [kg]	double
OUT	SAT.LENGTH	Satellite side length [m]	double
OUT	SAT.INERTIA	Satellite inertia [kg*mˆ2]	double
OUT	SAT.QUATERNION	Initial attitude [quaternions]	double
OUT	SAT.ANGULAR_RATE	Initial angular rate [rad/s]	double
OUT	SAT.COM	Satellite Center of Mass [m]	double
OUT	SAT.COP	Satellite Center of Pressure [m]	double
OUT	SAT.GEO_CENTR	Satellite geometrical center [m]	double
OUT	SAT.CIRC_EST	Radius of estimated spherical satellite [m]	double
Orbital parameters			
OUT	SAT.ECCENTRICITY	[rad]	double
OUT	SAT.ALTITUDE	[km]	double
OUT	SAT.SEMI_MAJOR_AXIS	[km]	double
OUT	SAT.INCLINATION	[rad]	double
OUT	SAT.RIGHT_ASCENSION_0	Initial argument of right ascension [rad]	double
OUT	SAT.ARGUMENT_OF_PERIGEE_0	Initial argument of perigee [rad]	double
OUT	SAT.MEAN_ANOMALY	[rad]	double
Magnetotorquer parameters			
OUT	SAT.MAGNETOTORQUER_AREA	[mˆ2]	double
OUT	SAT.MAGNETOTORQUER_N	Number of turns of magnetotorquer coil	double
OUT	SAT.RATE_HIGH_LIMIT	Angular rate limit for detumbling [rad/s]	double

Table 2: IN/OUT parameters of set_sat_param.m

sat_pos_vel.m

MATLAB-function to compute current satellite position and velocity in ECI coordinate system. The IN/OUT parameters are described in table 3.

sat_pos_vel.m			
IN/OUT	Name	Description	Type
IN	SAT	Array containing all satellite parameters.	double
IN	time	Current time [s]	double
OUT	pos_sat	Satellite position in ECI [km]	double
OUT	vel_sat	Satellite velocity in ECI [km]	double

Table 3: IN/OUT parameters of sat_pos_vel.m

calc_dcm.m

MATLAB-function to calculate the direction cosine matrix for coordinate system transformations from ECI to SBRF reference frame.

calc_dcm.m			
IN/OUT	Name	Description	Type
IN	quaternion	Attitude quaternion vector	double
OUT	dcm	Direction cosine matrix	double

Table 4: IN/OUT parameters of calc_dcm.m

calc_env_torque.m

MATLAB-function to calculate environmental disturbance torque affecting the satellite: torque produced by aerodynamic drag, torque produced by the gravitational gradient and torque produced by solar radiation. For calculating torque generated by atmospheric drag, a built in MATLAB function (atmosnrlmsise00 from MATLAB Aerospace Toolbox) for calculating the atmospheric pressure will be used here. The IN/OUT parameters are described in the table below.

calc_env_tor.m			
IN/OUT	Name	Description	Type
IN	SAT	Array containing all satellite parameters.	double
IN	pos_sat	Satellite position in ECI [km]	double
IN	vel_sat	Satellite velocity in ECI [km]	double
IN	dcm	Direction Cosine Matrix	double
OUT	tot_env_torques	Total environmental torques [Nm]	double
OUT	n_aero_sbrf	Torque produced by aerodynamic drag [Nm]	double
OUT	n_gg_sbrf	Torque produced by gravity gradient [Nm]	double
OUT	n_solar_sbrf	Torque produced by solar radiation [Nm]	double

Table 5: IN/OUT parameters of calc_env_torque.m

calc_magn.m

MATLAB-function to calculate the magnetic field strength. A MATLAB built-in function ,wrldmagm from MATLAB Aerospace Toolbox, for calculating the magnetic field strength will be used here. The values from wrldmagm are given in the north east down coordinate system so they have to be converted into the SBRF. The IN/OUT parameters are described in the table below.

calc_ magn.m			
IN/OUT	Name	Description	Type
IN	SAT	Array containing all satellite parameters.	double
IN	pos_sat	Satellite position in ECI [km]	double
IN	dcm	Direction Cosine Matrix	double
OUT	b_sbrf_x	Magnetic field in the x-direction [T]	float
OUT	b_sbrf_y	Magnetic field in the y-direction [T]	float
OUT	n_gg_sbrf	Magnetic field in the z-direction [T]	float

Table 6: IN/OUT parameters of calc_magn .m

control_loop.c

C-function, this is the control loop which shall run the following four C-functions. This function shall run once at every time step of the simulation. This function should also include a mexFunction, a gateway function from C to MATLAB. The IN/OUT parameters are described in table 7.

control_loop.c			
IN/OUT	Name	Description	Type
IN	dt	Timestep between magnetometer readings [s]	float
IN	gain	Controller gain	float
IN	number_of_turns	Number of turns of magnetotorquer coil	float
IN	mt_area	Magnetotorquer area [m^2]	float

Table 7: IN/OUT parameters of control_loop.c

get_magn_values.c

C-function to get readings from the magnetometer (magnetometer readings from MATLAB simulation workspace), and save these values until the next iteration of control loop. The IN/OUT parameters are described in table 8.

get_magn_values.c			
IN/OUT	Name	Description	Type
IN	time	Current time [s]	uint32_t
IN/OUT	magn_value_x	Magnetic field strength [T]	float
IN/OUT	magn_value_y	Magnetic field strength [T]	float
IN/OUT	magn_value_z	Magnetic field strength [T]	float
IN/OUT	last_value_x	Magnetic field strength [T]	float
IN/OUT	last_value_y	Magnetic field strength [T]	float
IN/OUT	last_value_z	Magnetic field strength [T]	float

Table 8: IN/OUT parameters of get_magn _values.c

calc_contr_torque.c

C-function to calculate the control torque needed to detumble the satellite using the B-dot detumbling algorithm. The IN/OUT parameters are described in table 9.

calc_contr_torque.c			
IN/OUT	Name	Description	Type
IN	timestep	Timestep between magnetometer readings [s]	float
IN	gain	Detumble controller gain	float
IN	mag_field_x	Magnetic field strength [T]	float
IN	mag_filed_y	Magnetic field strength [T]	float
IN	mag_filed_z	Magnetic field strength [T]	float
IN	last_mag_field_x	Magnetic field strength [T]	float
IN	last_mag_field_y	Magnetic field strength [T]	float
IN	last_mag_field_z	Magnetic field strength [T]	float
IN/OUT	contr_tor_x	Control torque [Nm]	float
IN/OUT	contr_tor_y	Control torque [Nm]	float
IN/OUT	contr_tor_z	Control torque [Nm]	float

Table 9: IN/OUT parameters of calc_contr_torque.c

convert_torque_to_current.c

C-function to calculate magnetotorquer current needed to detumble the satellite. The IN/OUT parameters are described in table 10.

convert_torque_to_current.c			
IN/OUT	Name	Description	Type
IN	contr_torque_x	Control torque [Nm]	float
IN	contr_torque_y	Control torque [Nm]	float
IN	contr_torque_z	Control torque [Nm]	float
IN	n_of_turns	Number of turns of magnetotorquer coil	float
IN	mt_area	Magnetotorquer area [mˆ2]	float
IN/OUT	current_mmt_x	Magnetotorquer current [A]	float
IN/OUT	current_mmt_y	Magnetotorquer current [A]	float
IN/OUT	current_mmt_z	Magnetotorquer current [A]	float

Table 10: IN/OUT parameters of convert_torque_to_current.c

set_current.c

C-function to set the magnetotorquer current, here putting the current values into the MATLAB simulation workspace. Because the MATLAB standard type is double, it is easiest to send double parameters to MATLAB. The IN/OUT parameters are described in table 11.

set_current.c			
IN/OUT	Name	Description	Type
IN/OUT	current_mmt_x	Magnetotorquer current [A]	float
IN/OUT	current_mmt_y	Magnetotorquer current [A]	float
IN/OUT	current_mmt_z	Magnetotorquer current [A]	float
OUT	current_x	Magnetotorquer current [A]	double
OUT	current_y	Magnetotorquer current [A]	double
OUT	current_z	Magnetotorquer current [A]	double

Table 11: IN/OUT parameters of set_current.c

attitude_dynamics.m

MATLAB-function to calculate the attitude and the angular velocity of the satellite, taking environmental disturbances and control torques into account. The IN/OUT parameters are described in table 12.

attitude_dynamics.m			
IN/OUT	Name	Description	Type
IN	time	Current time [s]	double
IN	x	Angular rate [rad/s] and attitude (quaternions)	double
IN	control_torque	Control torque vector [Nm]	double
IN	env_torque	Environmental torque vector [Nm]	double
IN	inertia	Inertia of satellite [kg*mˆ2]	double
OUT	dx	Change of attitude and angular velocity	double

Table 12: The IN/OUT parameters of attitude_dynamics.m

3.4 Simulation Results

The b-dot control law in Eq. (29) was simulated for a small satellite in low earth orbit, the altitude of the orbit is 400 km and the inclination is 45°. The initial angular rate is set to $\omega_0 =$[5.7 5.7 5.7] deg/s. The controller gain is set to $k = 1.5 * 10^3$. The angular velocities are shown in Fig. 3, and a zoom of the angular velocities in Fig. 4 shows that the controller has detumbled the satellite to $0.13°/s$ in 5500 s (92 min). Thus the controller manages to detumble the satellite in about one orbit. The associated magnetic control torques are shown in 5.

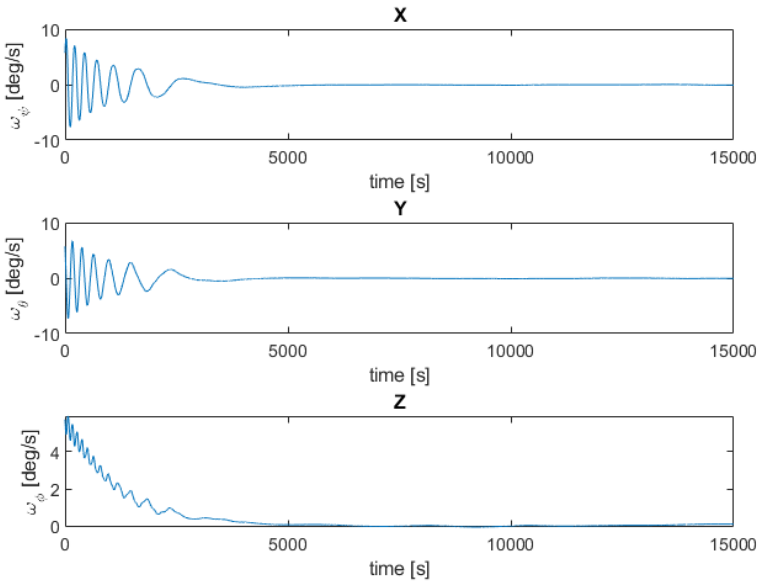

Figure 3: The angular rate for the detumbling simulation of the control law in Eq.(29).

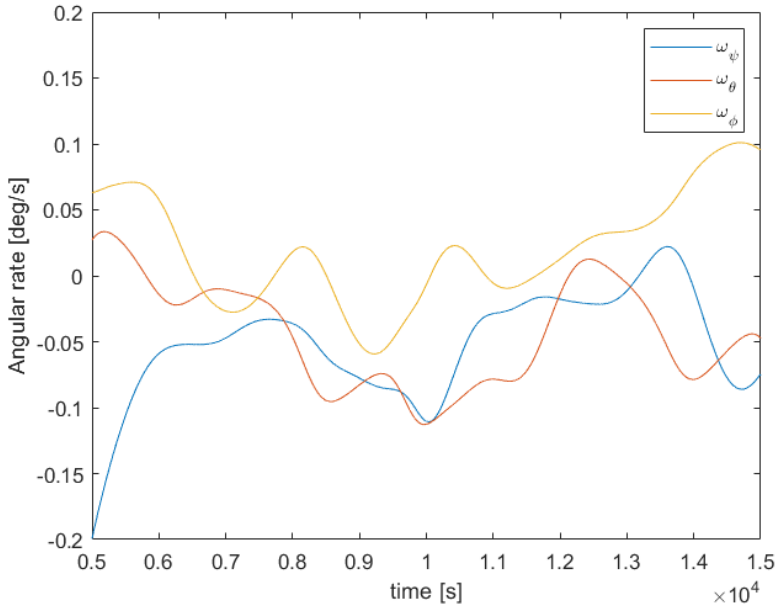

Figure 4: A zoom of the angular rate for the Detumbling simulation of the control law in Eq.(29)).

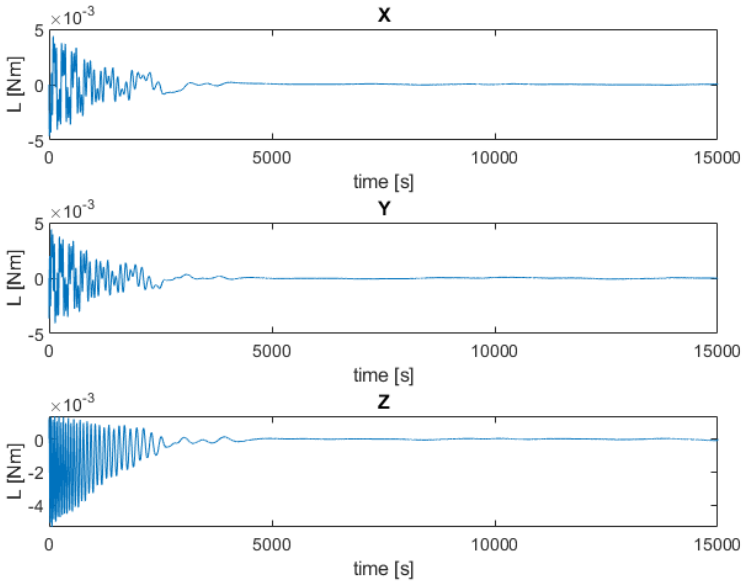

Figure 5: Control torques for the Detumbling simulation of the control law in Eq.(29).

4 Pointing

4.1 Literature Review

Research in attitude maneuvers has been carried out since the beginning of space missions. Emphasis has been put on the development of time-optimal maneuvering methods for spacecraft and robust control schemes to account for disturbances.
The aim of this text is to review time-efficient attitude maneuverer available today and compare these algorithms to the closed-loop quaternion feedback algorithm. The main focus is put on a rigid spacecraft without appendages, using reaction wheels as actuators. Two to three of these algorithms will be chosen for simulation in MATLAB to examine their performance. These algorithms will be reviewed in terms of:

- Maneuver time
- Robustness (ability to handle unexpected disturbances)
- Stability
- Applicability for implementation on an on-board computer

To put the current algorithms in context, a small overview of earlier advancements within the field of time-optimal control will first be presented. Then a current time-optimal algorithm suitable for attitude control will be presented. Because the time-optimal algorithm is not executable in real time, it is of interest to bring up other algorithms that are not necessarily time-optimal but time-efficient and executable in real time.

4.1.1 Background

In a survey of time-optimal attitude maneuvers [8] the literature covering the advancements within time-optimal control, from the sixties until the year of publication 1994, is reviewed. A closed-loop control for a rest-to-rest Louvre and a tracking maneuver is presented in [9]. In this paper, the proof of stability of the controller is performed by considering the Lyapunov stability criterion. A control strategy which takes thrusters and control moment gyros into account is presented in [10] . A passive control method that utilizes the torque due to the solar pressure is presented in [11]. A recurrent problem in most of these papers is the impracticality of implementing an optimal solution in real time due to the complexity of the problem and the time it took computing the control input. This lead to solutions that were not time-optimal but suboptimal [12, 13].
Most spacecraft today and back then made use of strong, lightweight materials which are also quite flexible. A number of researchers have therefore covered this topic. In [14], the equations of motion and closed-loop control

algorithms for a satellite which must target optical rays off mirrors mounted on flexible robotic arms are developed. Though the control algorithm is not time-optimal, this model is still relevant to current systems. In [15], it is concluded that there is a trade-off between maneuver time and complexity of the flexible spacecraft model. It is shown that, below a certain maneuver time, including more flexible modes results in a higher spillover energy at the end of the maneuver.
In addition to the rigid-body equations of motion, the flexibility effects add theoretically infinite degrees of freedom to the problem. This might result in an increase of the complexity of the time-optimal problem [8]. More recent advances in this field are available in [16] but these are not presented here as the focus of this thesis is control of rigid spacecraft.
The solution to the time-optimal Euler axis rotation for an asymmetric rigid spacecraft is presented in [17]. This approach is time-optimal but the motion is constrained to follow a specific path around a fixed Euler axis. However, in [18] it was shown that the rotation around a specific Euler axis does not necessarily lead to a general solution to the time-optimal problem. In [19], a near-minimum-time solution for the open-loop system was found by solving an LQR (Linear Quadratic Regulator) iteratively, reducing the final time until the desired control profile was obtained.

4.1.2 Time-Optimal Control

A suitable time-optimal algorithm is the Time-Optimal Control by Minimizing the Cost Function presented by [20]. Within the saturation limits of the reaction wheels, this algorithm should perform a large-angle reorientation where the spacecraft is to maneuver about an inertially fixed axis as fast as possible. The optimal control input is to be determined such that the system is transferred from its given initial state to a specified final state in minimum time. This is solved by constraining the control input and minimizing the cost function which gives a set of differential equations with specified boundary conditions for the time-optimal control. According to Ponytryagin's minimum principle, the Hamiltonian can be minimized such that the optimal control input can be determined. Though this algorithm is time-optimal, the solutions to the set of differential equations are usually obtained using numerical methods that are computationally extensive. [20]
For linear systems, integration of the differential equations can be done in real time, the solution can therefore be expressed in closed form and a closed control loop can be implemented. When a linear approach is not possible, there are two options: Either a suboptimal control algorithm is accepted, or the control action is calculated periodically at the ground station. If the control action is calculated at a ground station, this is done for longer periods of time requiring the spacecraft attitude controller to be of open-loop type. [16]

4.1.3 Quaternion Feedback Control

The Quaternion feedback controller is a well-known type of controller suitable for implementation on on-board computers though this type of controller is not time-optimal. In this closed-loop system the attitude error is fed back to the system as quaternion errors.
As described in [7, Ch. 7.2], regulation control is defined as bringing the attitude to some fixed location (usually the identity quaternion $\begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}^T$) and the angular velocity to zero. Three feedback controllers are presented in [7, Ch. 7.2]:

1. The first one is locally asymptotically stable, this means that the control law asymptotically reorients the spacecraft to the desired attitude from an arbitrary small angle.

2. The second one is globally asymptotically stable, this means that the control law asymptotically reorients the spacecraft to the desired attitude from a larger angular offset than above.

3. The third one is globally asymptotically stable and it also provides the shortest distance maneuver.

The last control law is proven to be globally asymptotically stable with reaction wheels in [7, Ch 7.2].

4.1.4 Sliding mode Control

Sliding mode control is a popular control approach for nonlinear systems, and the structure of the control law varies based on the position of the state trajectory. This type of control is robust to random model parameter errors, but comes at a price of possibly high control activity.
Sliding mode control essentially reduces the order of the system to a first-order system by feedback linearizion. However, if the dynamics of the system are not known exactly, this may lead to severe errors when a direct feedback linearizion control law is applied on an actual system. This means that robustness, in the presence of parametric and un-modeled uncertainties, cannot be ensured. Because the sliding mode control law is not continuous, closed-loop behavior cannot always be guaranteed for this method.
The performance of a sliding mode control law presented in [7] is suboptimal, but the control algorithm provides a good trade-off between robustness and practical compatibility [7, 21]. A sliding mode controller for both the case of external torques and internal reaction wheels is described in [7, Ch. 7.2] from the control model earlier presented by [22].

4.1.5 Robust $H - \infty$ Control

The nonlinear H_∞ state feedback attitude control method is designed for large spacecraft maneuvers, which are subject to moment of inertia uncertainties. The controller essentially determines feedback gains for the full state feedback control problem so that the spacecraft is stabilized in the presence of uncertainties and disturbances. This H_∞ controller is convenient for thruster-based control systems, since the moment of inertia will change when the amount of propellant is decreasing [23].

4.1.6 Adaptive and Terminal Sliding Mode Control

Adaptive controllers update the model during operation based on measured performances. A popular adaptive controller is the one developed in [24], because it can provide a finite-time convergence for the control system.
The closed-loop continuous controllers proposed in [25] are able to deal with system uncertainty and actuator saturation while simultaneously providing fast finite-time convergence speed for the control system.

4.1.7 Conclusion

I chose to consider a rigid rather than a flexible spacecraft due to the added complexity from considering flexibility effects.
The time-optimal control by minimizing the cost function method is superior when it comes to minimizing the maneuver time. But, open-loop control schemes, such as the time-optimal control by minimizing the cost function, are sensitive to spacecraft parameter uncertainties and unexpected disturbances. Closed-loop schemes, such as the quaternion feedback method and the terminal sliding mode controller, are better to account for uncertainties and disturbances, and can therefore provide a more robust design. Optimal algorithms, that minimize a user-defined cost function and provide the control trajectory over the desired time interval, cannot be executed in real time. Such algorithms are convenient for analysis purposes but less useful for real spacecraft applications. Feedback algorithms, on the other hand, are more suitable for actual application because they can be implemented in real time [20, 16, 7]. For analysis purposes, it is still interesting to compare the time-optimal algorithm to real-time controllers.
For the quaternion feedback algorithms, it would be interesting to see if implementing the third controller provides a faster maneuver than the second controller presented earlier. The question is if the third controller can compete with the other algorithms in terms of maneuver time and accuracy.
The sliding mode method has some nice features: it provides a good trade-off between robustness and practical compatibility. Though the sliding mode controller is not always guaranteed to be a closed-loop control law, it is presented in [7, Ch. 7.3] in a way such that it should be applicable for real-time

implementation. However, it would be interesting to see if the sliding mode controller for the case of both external torques and internal reaction wheels presented by [22] provides a faster maneuver time than the quaternion feedback method.
Since the $H - \infty$ controller is more convenient for thruster-based control systems, this method will not be discussed further.
The adaptive terminal sliding mode controller provides closed-loop characteristics while the system converges in finite-time which is attractive for real life application of the system. Therefore, it would be interesting to see if this type of controller could provide the same, or improved pointing accuracy, at a faster maneuver time than the quaternion feedback algorithm.
From this review I conclude that these attitude algorithms are interesting to study further in terms of maneuver time and performance:

- Sliding mode Control
- The third Quaternion Feedback Control Method

The Adaptive and Terminal Sliding Mode Control algorithm would indeed be interesting to examine. However, since time is limited and the controller proposed in [25] is advanced (and not applied to a simple control case with reaction wheels) so adapting this law may be outside the scope of this thesis.

4.2 Controllers & Simulation Results

The algorithms selected in the literature review are tested for a nadir-pointing maneuver, where the satellite has to point to nadir from a 180 degree angle offset. The tests are performed in MATLAB where a simulation background, including orbital and attitude propagation and environmental disturbances, is set up. The pointing algorithms are tested for two different cases. The first case simulates a bigger satellite of 50 kg with three reaction wheels and the second case simulates the behavior of a 3u CubeSat.

The result is considered stable when the angular rate does not differ more than $0.001°/s$ which is ideal for most space missions [26]. The initial angular offsets from nadir are 20°, 20° and 180° in the roll, pitch and yaw directions respectively.
This gives the initial quaternion $\mathbf{q} = [0.9698, 0.1710, -0.1710, -0.0302]$. The initial angular rate is $\omega = [0.13, 0.13, 0.13]$ deg/s representing a detumbled satellite.
For a valid comparison between the different algorithms, the controller gains of both algorithms have been selected in the same way. By minimizing the control effort needed to bring the quaternion error to zero and the angular rate to 0.001° or less, the gain parameters can be found.

4.2.1 Cost Function

The total cost of a control maneuver, the control effort, is used to select the gain parameters of the controllers. The cost function is also used for comparison of the different controllers in terms of control effort.

$$C = \int_{t_0}^{t_f} (\mathbf{T}_w \bullet \mathbf{T}_w)\, dt \tag{31}$$

The total cost, C, of the maneuver is calculated as the integral of the dot product of the reaction wheel torque vector $\mathbf{T_w}$.

4.2.2 Simulation case Cubesat

The CubeSat is a rectangular block with standard dimensions of 100x100x340.5 mm and a mass of 4 kg based on the upper limit of a 3u cubesat [27]. The nominal momentum of inertia without reaction wheels, J, is a diagonal matrix with $J_{11} = 0.0056\ kgm^2$, $J_{22} = 0.026\ kgm^2$ and $J_{33} = 0.0026\ kgm^2$ [28]. Here the Satellite Body Reference Frame Two commercially available reaction wheels, the Microwheel RWP015 [29] and the Cubewheel [30], suitable for a 3-axis stabilization of a 3u CubeSat are presented in the table below.

	Microwheel RWP015 Blue Canyon Technologies	**CubeWheel Medium CubeSpace**
Momentum	0.015 Nms	0.01082 Nms
Max Torque	0.004 Nm	0.001 Nm
Mass	0.130 kg	0.150 kg
Volume	42 x 42 x 19 mm	46 x 46 x 31.5 mm

Table 13: Two commercially available reaction wheels suitable for a 3u CubeSat.

Though the Microwheel from table 13 is smaller and more effective, the Medium CubeWheel is chosen to be used in this simulation representing a worst case scenario.
The moment of inertia of each reaction wheel can be calculated as $J_w = \frac{1}{2} m_w r_w^2$, here $m_w = 0.150$ kg and $r_w = 0.0183$ m gives $J_w = 2.5 * 10^{-5}$ kgm^2. The installation matrix of the three orthogonal reaction wheels is

$$A_{rw} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{32}$$

4.2.3 Simulation case Small Satellite

The small satellite is cubic with a side length of 0.7 m and a total mass of 50 kg. The nominal momentum of inertia without reaction wheels, J, is a diagonal matrix with $J_{11} = 24.5$ kgm^2, $J_{22} = 22$ kgm^2 and $J_{33} = 31.85$ kgm^2. The table below shows a commercially reaction wheels, the RWP500 wheel [31] and the RW1 wheel [32] from Blue Canyon Technologies, suitable for small satellites:

	Reaction Wheel RWP500 Blue Canyon Technologies	**Reaction Wheel RW1 Blue Canyon Technologies**
Momentum	0.5 Nms	1.5 Nms
Max Torque	0.025 Nm	0.1 Nm
Mass	0.75 kg	1.6 kg
Volume	11 x 11 x 3.8 cm	15 x 15 x 6.5 cm

Table 14: A commercially available reaction wheel suitable for a small satellite.

The RW1 wheel from Blue Canyon Technologies is chosen because the mass and size of the RW1 reaction wheel is reasonable for a 50 kg satellite. The moment of inertia of each reaction wheel can be calculated as above, and $m_w = 0.75$ kg and $r_w = 0.04$ m g $J_w = 6 * 10^{-4}\ kgm^2$. The installation matrix of the three orthogonal reaction wheels is the same as above in Eq.(32).

4.2.4 The Quaternion Feedback Controllers

The simplest quaternion feedback control law is given by

$$\mathbf{L} = -K_p \delta \mathbf{q}_{1:3} - K_d \boldsymbol{\omega} \tag{33}$$

where L is the torque command, K_p & K_d are positive scalar gains and ω is the angular velocity of the spacecraft. The error quaternion is given by

$$\delta \mathbf{q} \equiv \begin{bmatrix} \delta \mathbf{q_{1:3}} \\ \delta q_4 \end{bmatrix} = \mathbf{q} \otimes \mathbf{q}_c^{-1} \tag{34}$$

where $\mathbf{q}_c$ is the desired target quaternion, and $\mathbf{q}$ is the current spacecraft quaternion. The closed loop system can be described by the quaternion kinematic equation,

$$\delta \dot{\boldsymbol{q}} = \frac{1}{2} \omega \otimes \delta \mathbf{q} \tag{35}$$

together with Euler's rotational equation of motion,

$$\dot{\boldsymbol{\omega}} = -\mathbf{J}^{-1} \left([\boldsymbol{\omega} \times] \mathbf{J} \boldsymbol{\omega} + K_p \delta \mathbf{q}_{1:3} + K_d \boldsymbol{\omega} \right) \tag{36}$$

where $\mathbf{J}$ is the total inertia matrix of the spacecraft including the reaction wheels. (The cross product matrix ,$[\boldsymbol{\omega} \times]$, and the quaternion cross product matrix, $[\boldsymbol{q} \otimes]$ are both described in Appendix A). The only equilibrium point is $\begin{bmatrix} \delta \mathbf{q_{1:3}^T} & \boldsymbol{\omega}^T \end{bmatrix}^T = 0$, and stability is shown in [7] using Lyapunov's direct method. Asymptotic stability is also proven in [7] using LaSalle's theorem. This means that the control law asymptotically reorients the spacecraft to the desired attitude from an arbitrary orientation.
The term δq_4 can be both ± 1 but both values generate equal attitude. Though, the control law in Eq. (1) does not provide the shortest path to the final orientation. This is an issue when the fourth component of the initial error quaternion is negative. [7] provides two other control laws that overcome this problem. The first one is given by

$$\mathbf{L} = -K_p sgn(\delta q_4) \delta \boldsymbol{q}_{1:3} - K_d \boldsymbol{\omega} \tag{37}$$

If $\delta q_4 < 0$ then a positive feedback term is introduced which provides a shorter path to reach the desired equilibrium point. The control law in Eq. (5) is always preferred over the control law in Eq. (1) unless $\delta q_4 = 0$ exactly,

which is not likely for practical applications.
Global asymptotic stability for the control law in Eq.(5) is shown in Appendix B. Global asymptotic stability is described in Definition 3.6 in [21]:

"If asymptotic (or exponential) stability holds for any initial states, the equilibrium point is said to be asymptotically (or exponentially) stable in the large. It is also called globally asymptotically (or exponentially) stable."
The control law in Eq. (1) is in the linear state, but nonlinear control laws can also be used. In [7], the following nonlinear control law is suggested:

$$\mathbf{L} = -K_p \delta \boldsymbol{q}_{1:3} - K_d \left(1 \pm \delta \boldsymbol{q}_{1:3}^T \delta \boldsymbol{q}_{1:3}\right) \boldsymbol{\omega} \tag{38}$$

When the minus sign is used then by the quaternion unity constraint we have

$$1 - \delta \boldsymbol{q}_{1:3}^T \delta \boldsymbol{q}_{1:3} = \delta q_4^2 \tag{39}$$

The shortest distance nonlinear control law can be achieved by using the second control law presented in [7]:

$$\mathbf{L} = -K_p sgn(\delta q_4) \boldsymbol{q}_{1:3} - K_d \left(1 \pm \delta \boldsymbol{q}_{1:3}^T \delta \boldsymbol{q}_{1:3}\right) \boldsymbol{\omega} \tag{40}$$

The stability of control law in Eq.(12) is shown in Appendix B using Lyapunov's direct method, global asymptotic stability is also shown using LaSalle's theorem.

4.2.5 Simulation of the Linear Quaternion Feedback Controller: Small Satellite Case

The control law in eq. (37) was simulated in MATLAB for the Small Satellite with the gains $K_p = 0.258$, $K_d = 9079$. The quaternion errors are shown in Fig.6. Fig. 7 and 8 show that a stability of $0.001°/s$ is reached in 600 s but Fig. 9 and 10 show that a pointing accuracy of $0.01°$ is not reached. Though, a pointing accuracy of $0.04°$ is reached in 600 s. Fig. 11 is showing that the maximum angular momentum limit of the reaction wheels is not exceeded. The total cost of the maneuver is 0.1612.

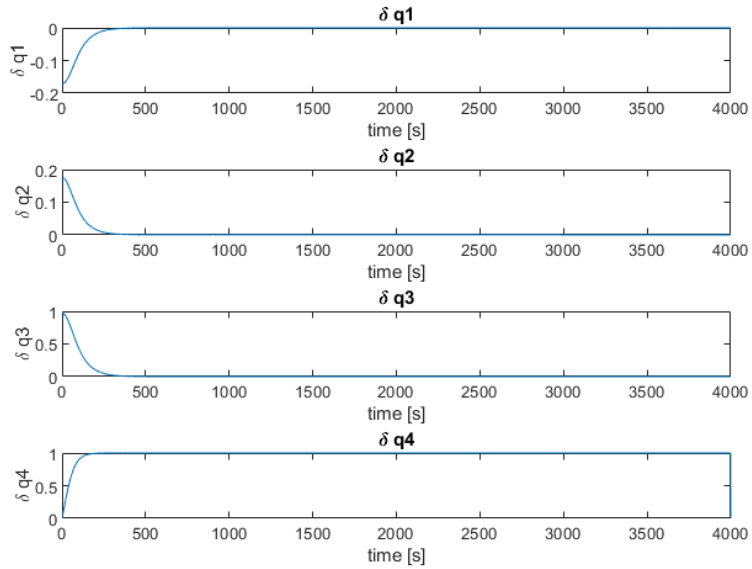

Figure 6: Quaternion errors for the Small Satellite simulation of the control law in Eq.(37).

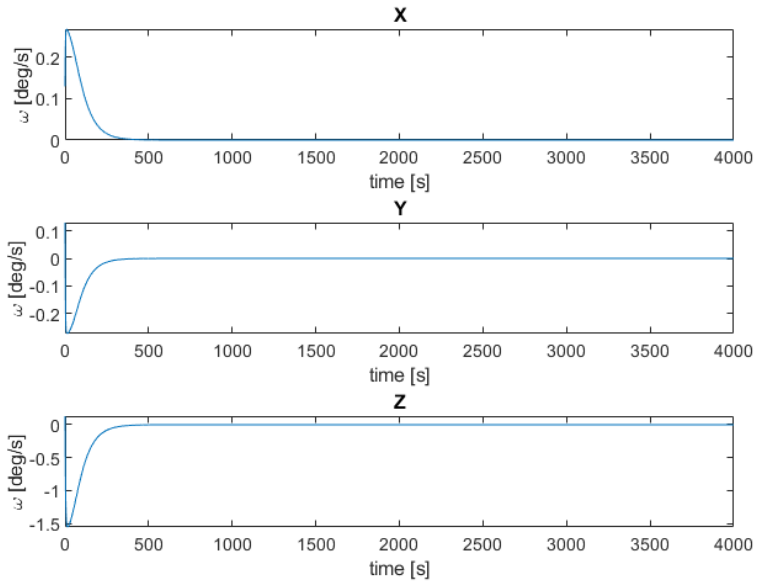

Figure 7: The angular rate for the Small Satellite simulation of the control law in Eq.(37).

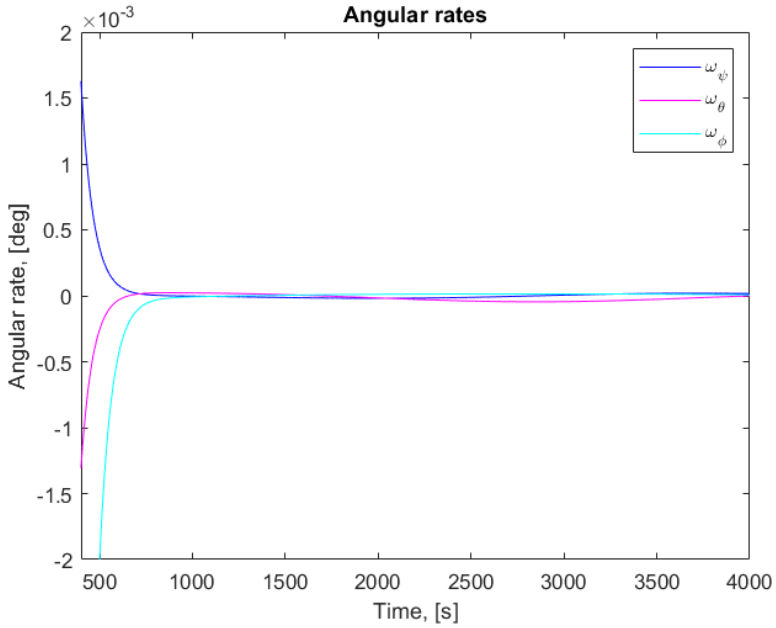

Figure 8: A zoom of the angular rate for the Small Satellite simulation of the control law in Eq.(37).

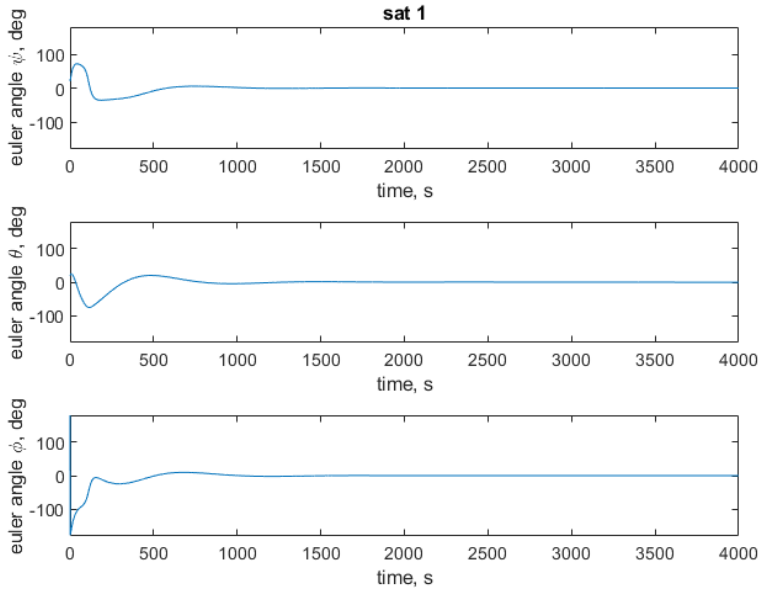

Figure 9: A plot of the Euler angles for the Small Satellite simulation of the control law in Eq.(37).

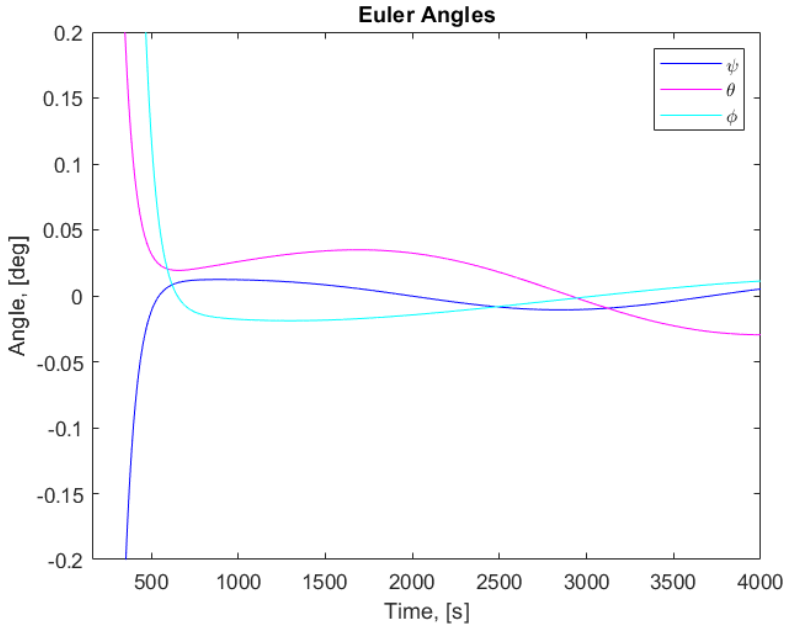

Figure 10: A zoom of the Euler angles for the Small Satellite simulation of the control law in Eq.(37).

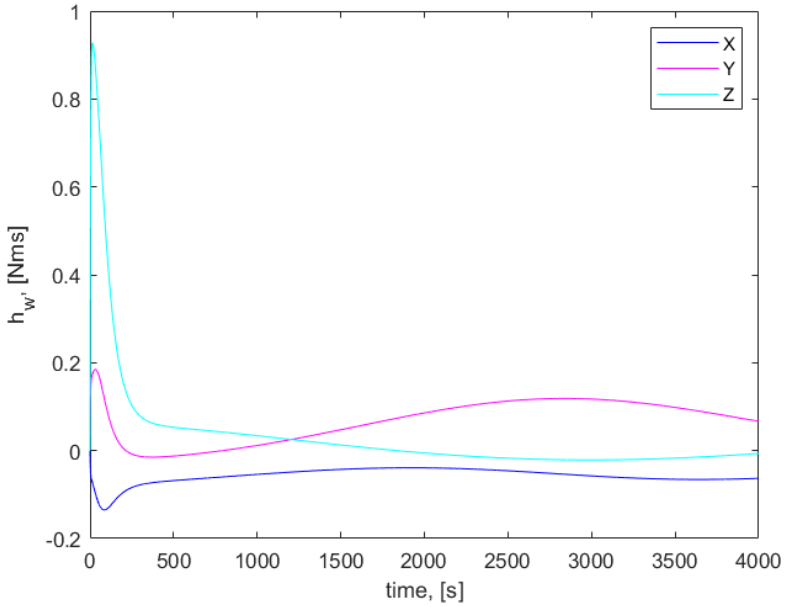

Figure 11: The reaction wheel angular momentum for the Small Satellite simulation of the control law in Eq.(37).

4.2.6 Simulation of the Linear Quaternion Feedback Controller: CubeSat Case

The control laws in Eq. (37) was also simulated in MATLAB for the CubeSat case with the gains $K_p = 0.0016$ and $K_d = 0.0035$. The quaternion errors are shown in Fig.12. Fig. 13 and 14 show that a stability of $0.001°/s$ in 120 s. Fig. 15 and 16 show that a pointing accuracy of $0.01°$ is reached in 120 s. Fig. 17 and 18 are showing that the maximum torque limit and the maximum angular momentum limit of the reaction wheels are not exceeded. In Fig. The total cost of the maneuver is $3.2765 * 10^{-6}$.

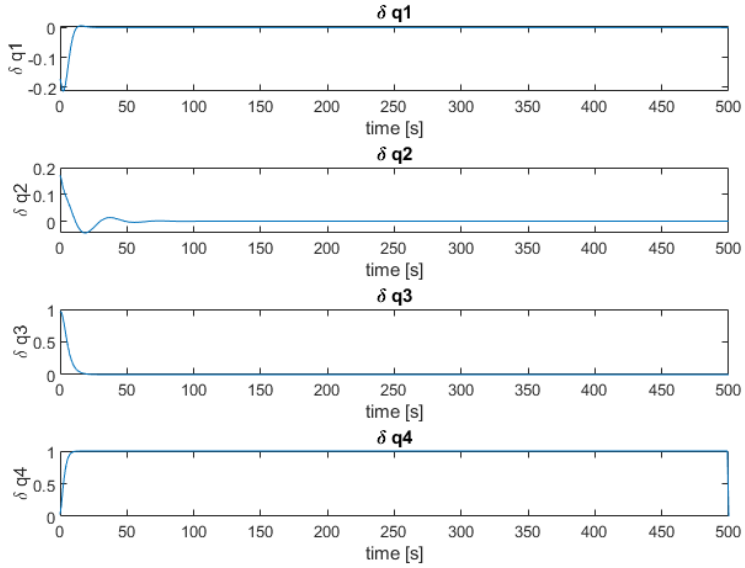

Figure 12: Quaternion errors for the CubeSat simulation of the control law in Eq.(37).

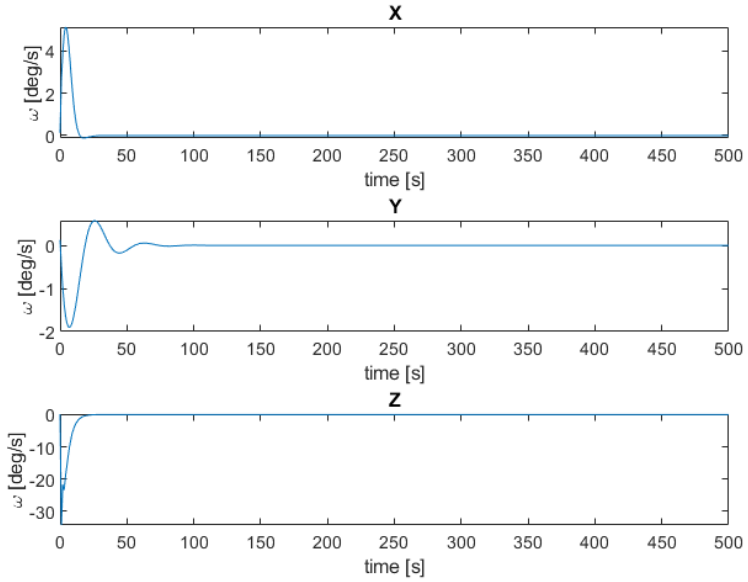

Figure 13: The angular rate for the CubeSat simulation of the control law in Eq.(37).

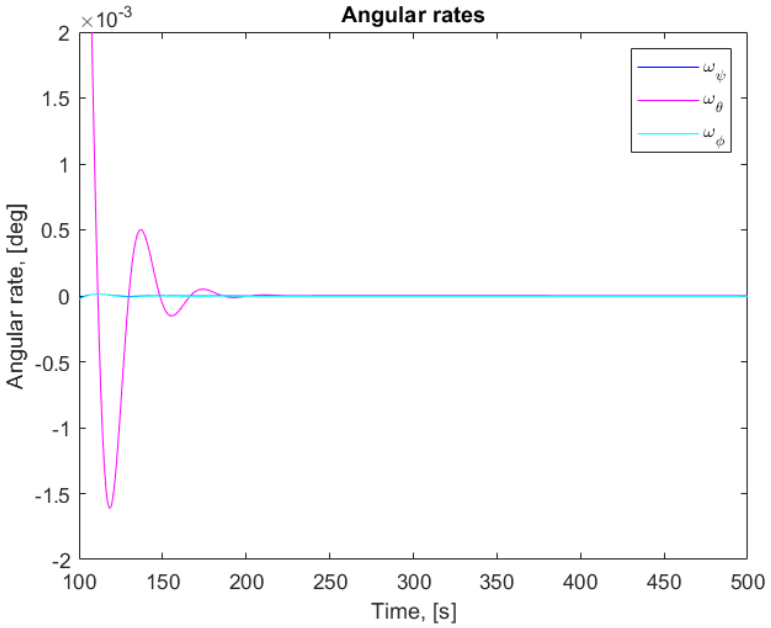

Figure 14: A zoom of the angular rate for the CubeSat simulation of the control law in Eq.(37).

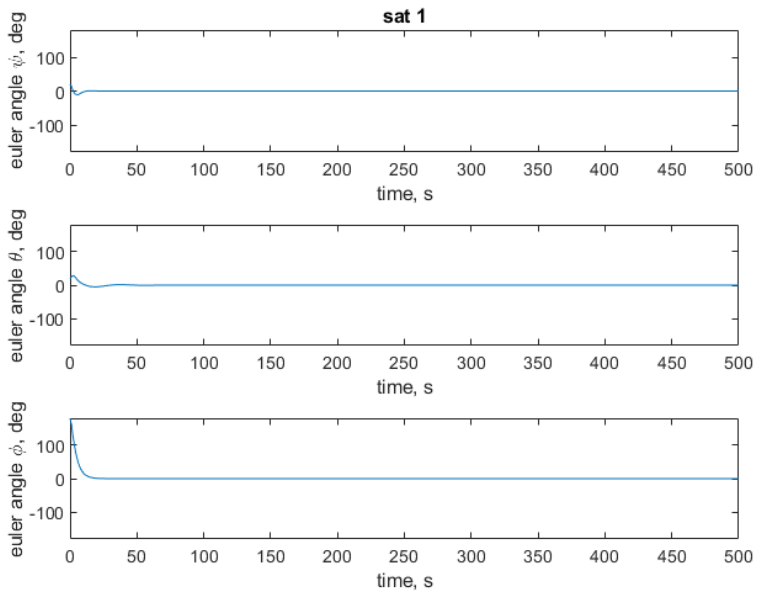

Figure 15: Euler angles for the CubeSat simulation of the control law in Eq.(37).

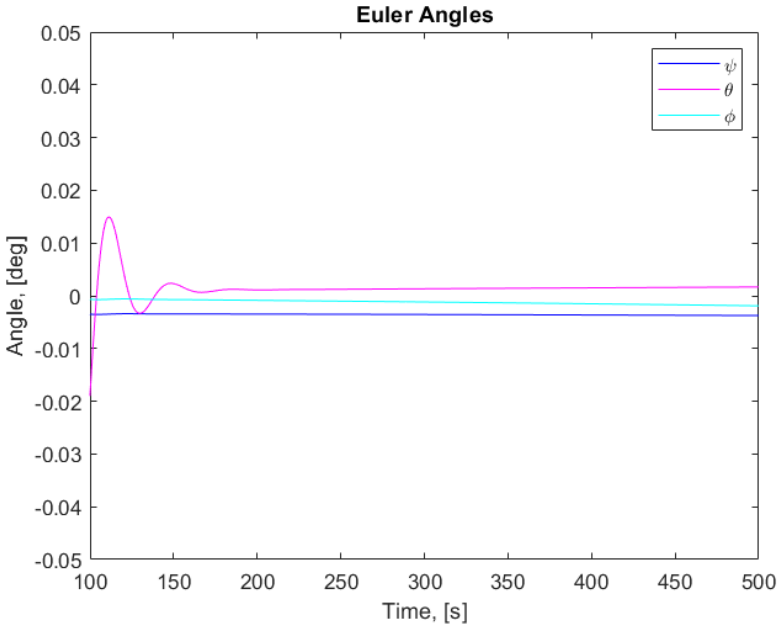

Figure 16: A zoom of the Euler angles for the CubeSat simulation of the control law in Eq.(37).

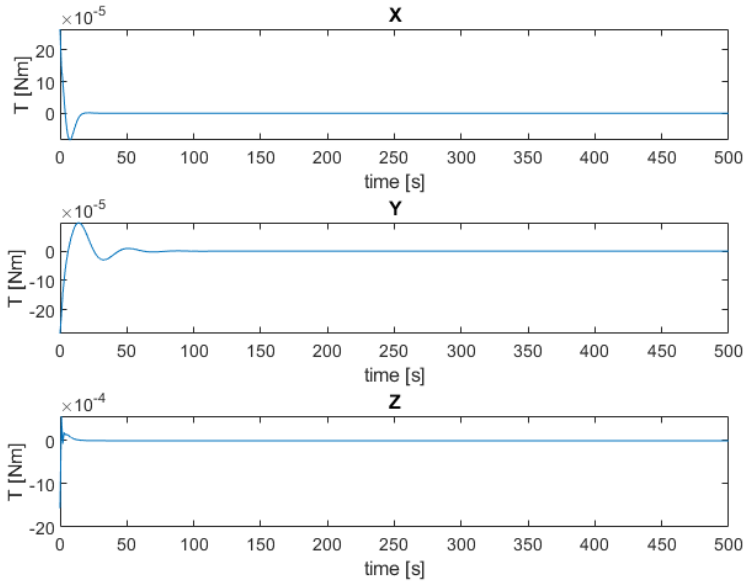

Figure 17: Control torques for the CubeSat simulation of of the control law in Eq.(37).

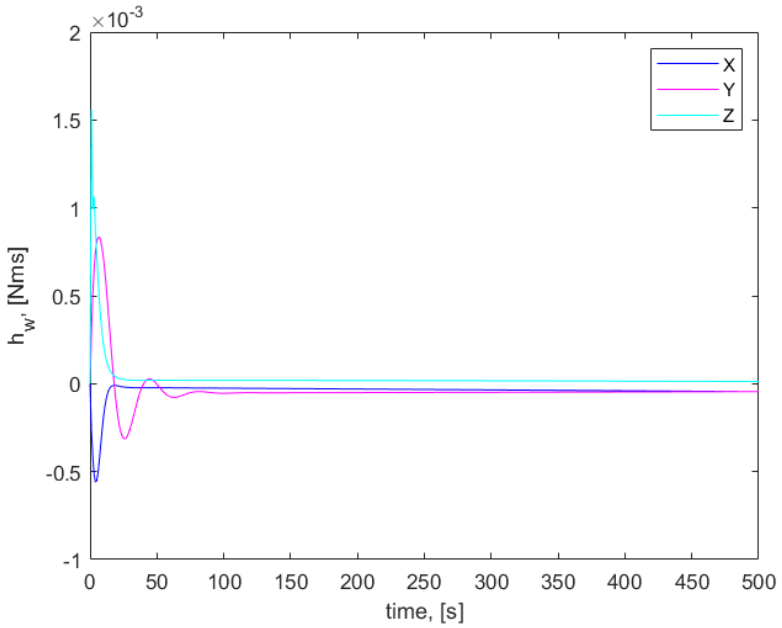

Figure 18: The reaction wheel angular momentum for the CubeSat simulation of the control law in Eq.(37).

4.2.7 Simulation of the Nonlinear Quaternion Feedback Controller: Small Satellite Case

The control law in eq. (40) was simulated in MATLAB for the Small Satellite with the gains $K_p = 0.318$ and $K_d = 8.129$. The quaternion errors are shown in Fig.19. Fig. 20 and 21 show that a stability of $0.001°/s$ is reached in 400s but Fig. 22 and 23 show that a pointing accuracy of $0.01°$ is not reached. Though, a pointing accuracy of $0.02°$ is reached in 400 s. Fig. 24 and 25 are showing that the maximum torque limit and the maximum angular momentum limit of the reaction wheels are not exceeded. The total cost of the maneuver is 0.1684.

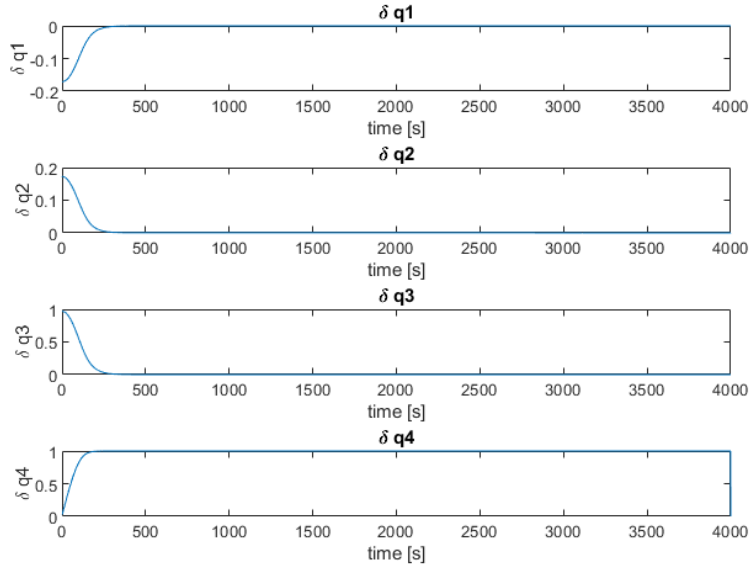

Figure 19: Quaternion errors for the Small Satellite simulation of the control law in Eq. (40).

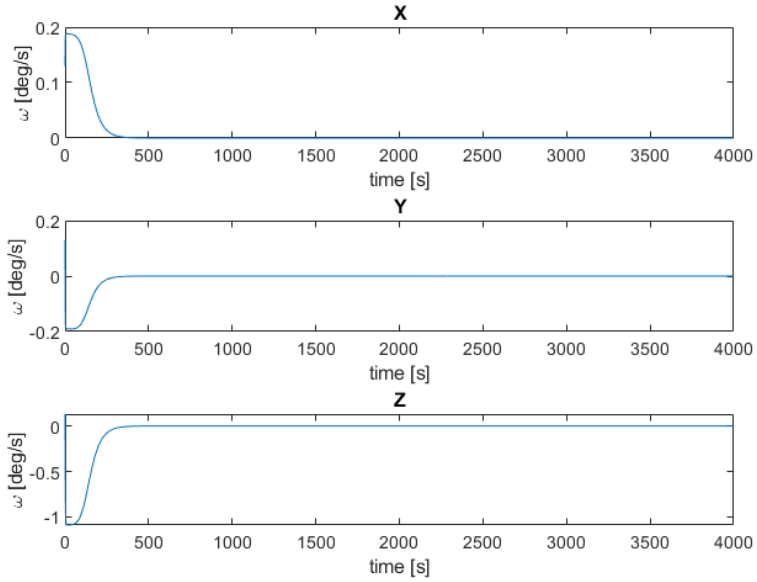

Figure 20: Magnitude of angular rate for the Small Satellite simulation of the control law in Eq. (40).

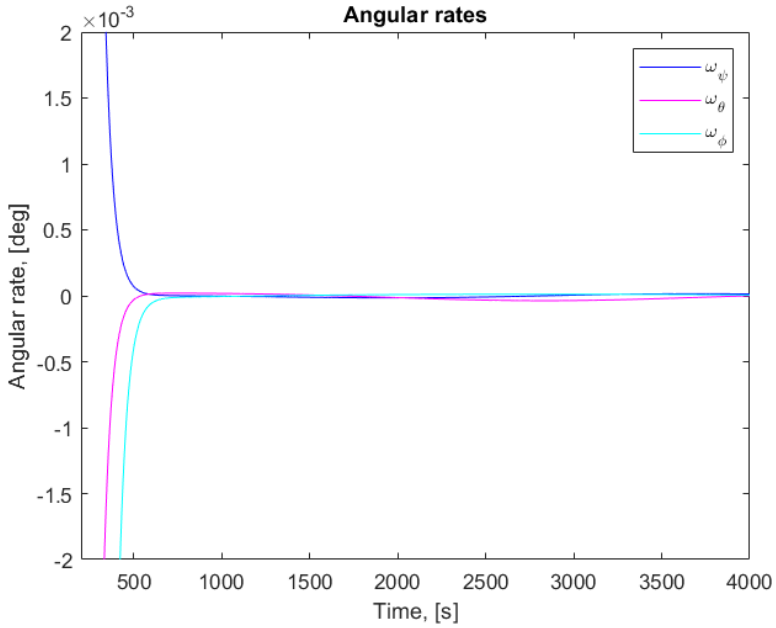

Figure 21: A zoom of angular rates for the Small Satellite simulation of the control law in Eq. (40).

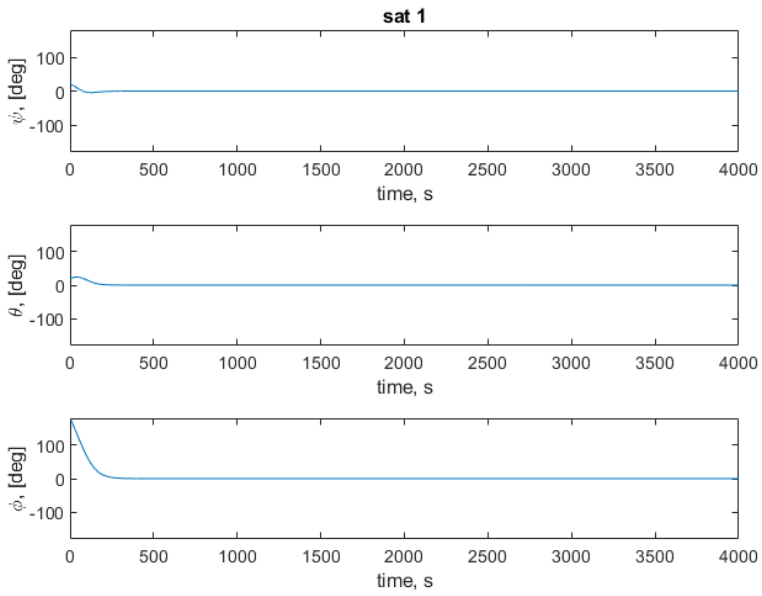

Figure 22: The Euler angles for the smallsat simulation of the control law in Eq.(40).

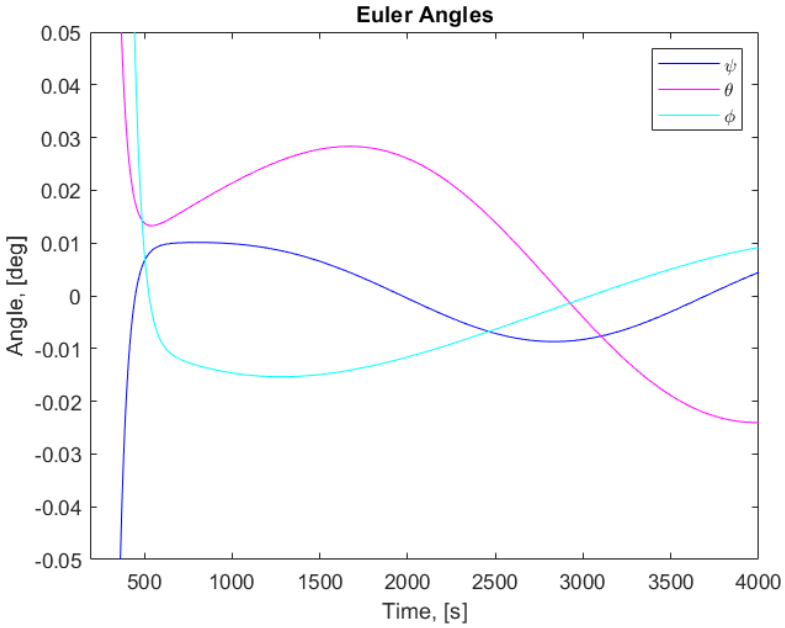

Figure 23: A zoom of the Euler angles for the smallsat simulation of the control law in Eq.(40).

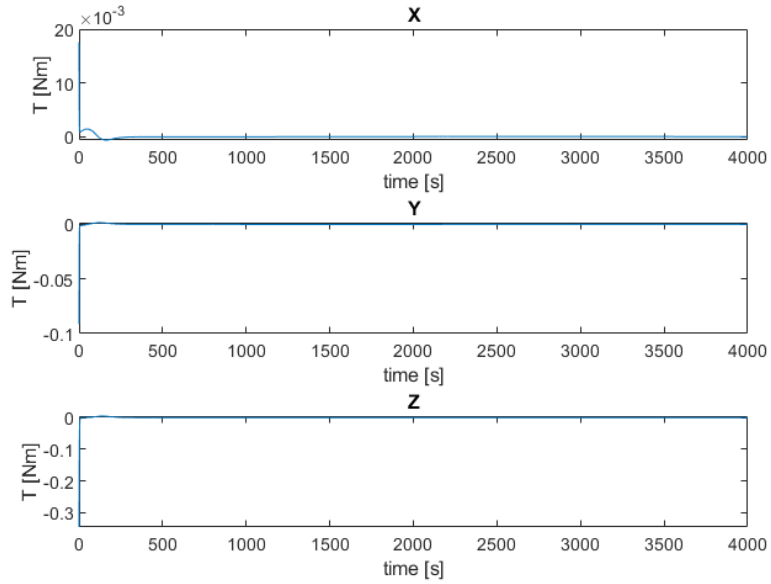

Figure 24: Control torques for the Small Satellite simulation of the control law in Eq. (40).

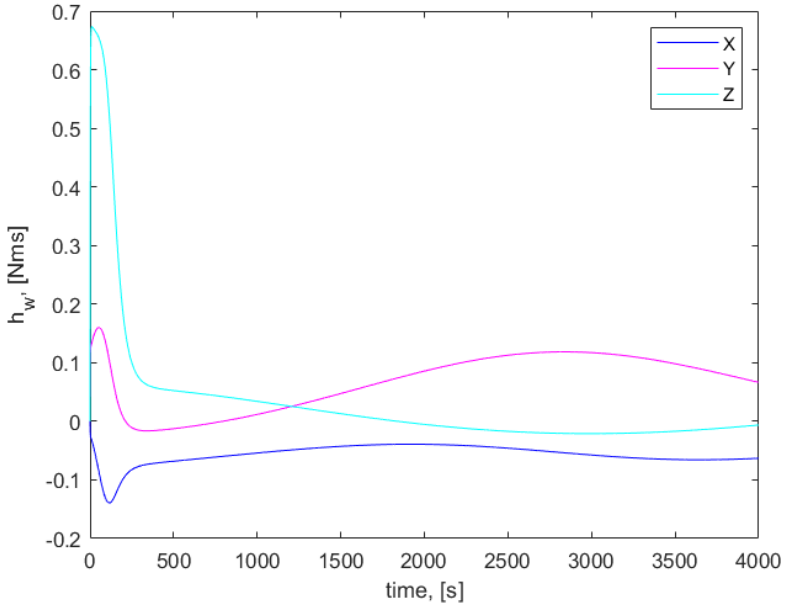

Figure 25: The reaction wheel angular momentum for the Small Satellite simulation of the control law in Eq. (40).

4.2.8 Simulation of the Nonlinear Quaternion Feedback Controller: CubeSat Case

The control laws in Eq. (40) was also simulated in MATLAB for the CubeSat case with the gains: $K_p = 0.0016$ and $K_d = 0.0025$. The quaternion errors are shown in Fig. 26. Fig. 27 and 28 show that a stability of $0.001°/s$ is reached in 200 s. Fig. 29 and and 30 show that a pointing accuracy of $0.01°$ is also reached in 200 s. Fig. 31 and 32 are showing that the maximum torque limit and the maximum angular momentum limit of the reaction wheels are not exceeded. The total cost of the maneuver is $7.2722 * 10^{-5}$.

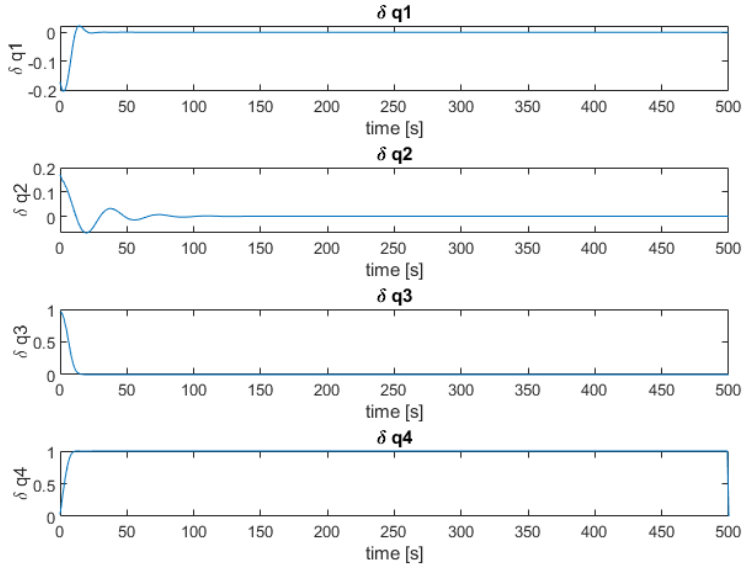

Figure 26: Quaternion errors for the CubeSat simulation of the control law in Eq.(40).

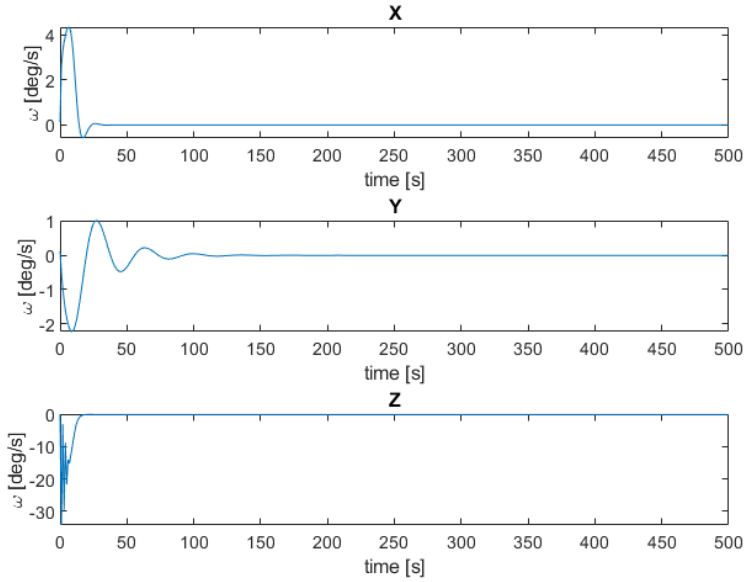

Figure 27: The angular rates for the CubeSat simulation of the control law in Eq.(40).

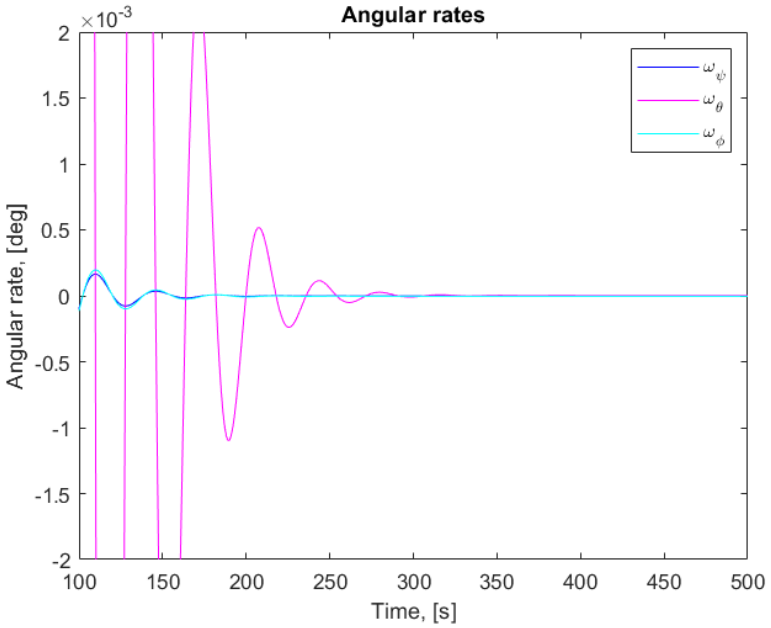

Figure 28: A zoom of the angular rates for the CubeSat simulation of the control law in Eq.(40).

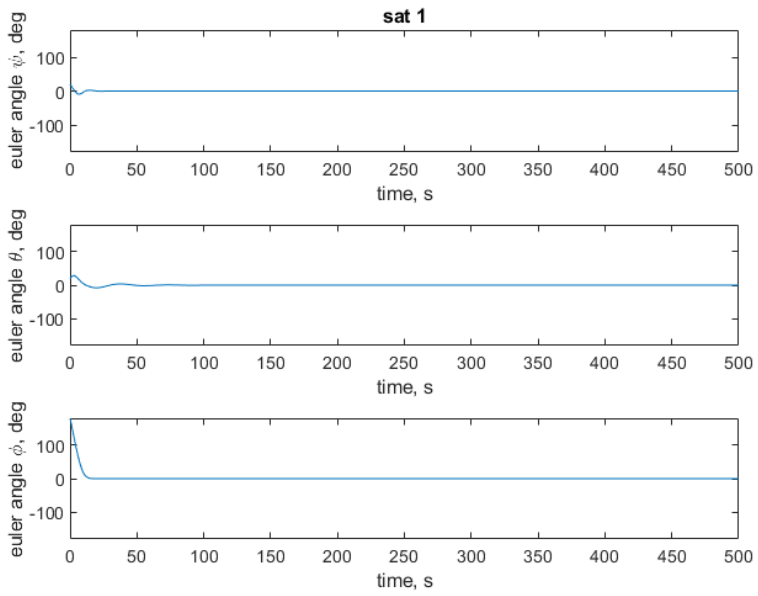

Figure 29: The Euler angles for the CubeSat simulation of the control law in Eq.(40).

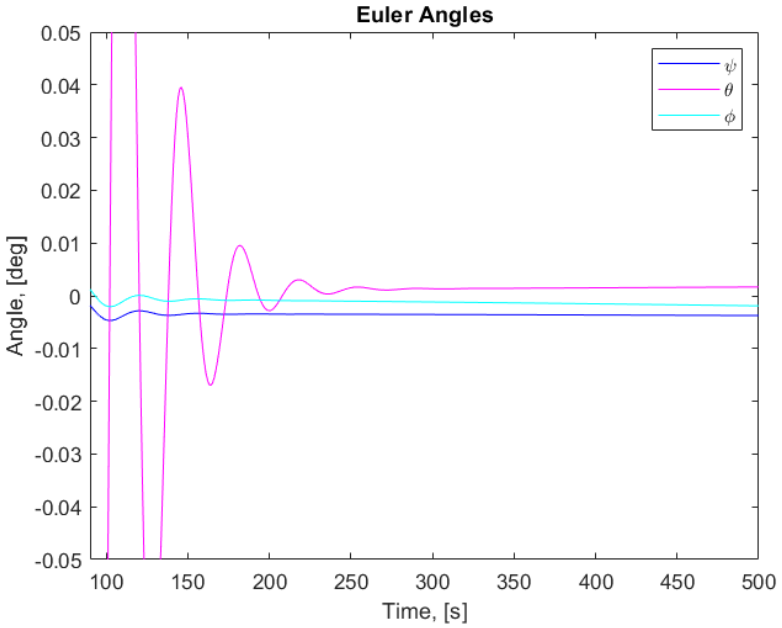

Figure 30: A zoom of the Euler angles for the CubeSat simulation of the control law in Eq.(40).

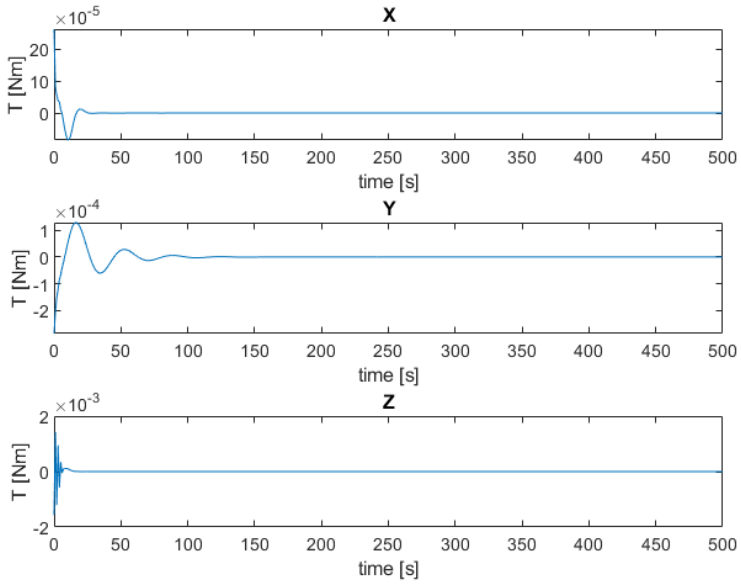

Figure 31: Control torques for the CubeSat simulation of the control law in Eq.(40).

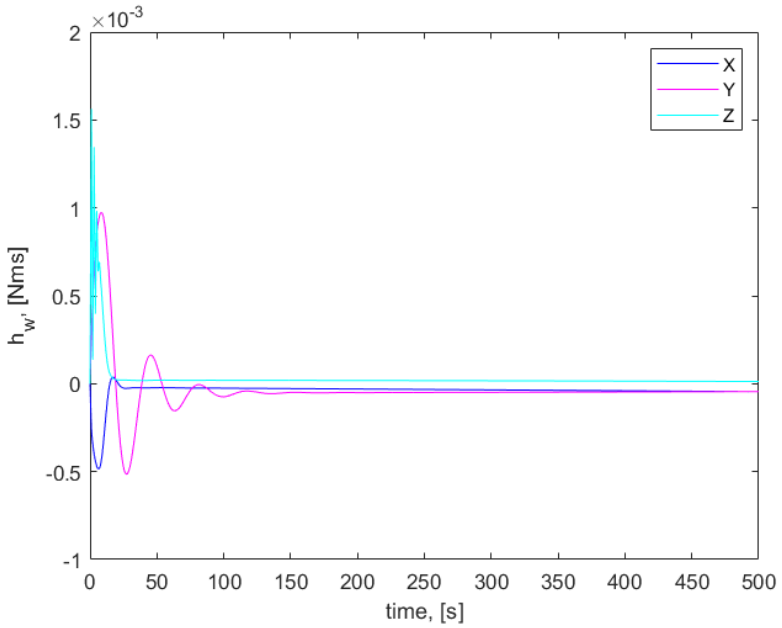

Figure 32: Reaction wheel angular momentum for the CubeSat simulation of the control law in Eq.(40).

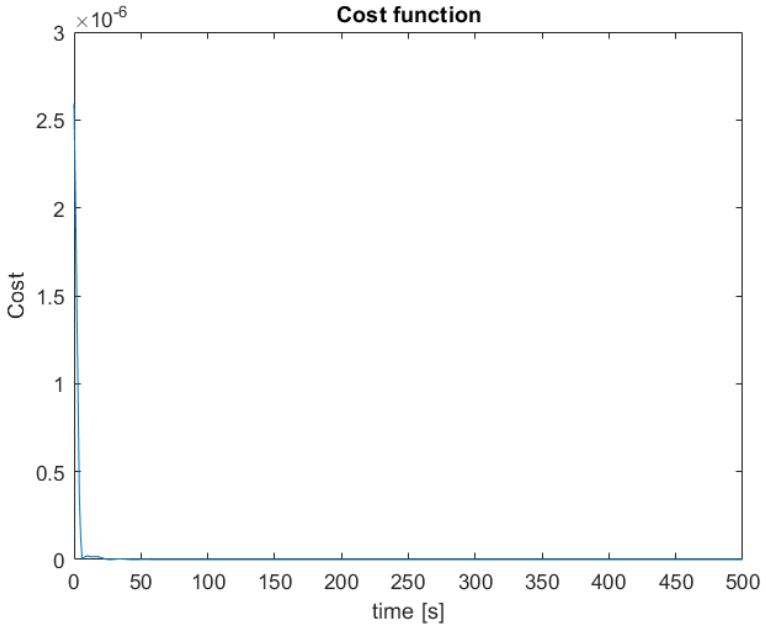

Figure 33: The cost function plotted as a function of time for the manoeuvre.

4.2.9 The Sliding Mode Controller

In [26] a time efficient maneuver of a rigid spacecraft is presented. A cluster of four reaction wheels is used to control the spacecraft and frequent unloading of the spacecraft of the reaction wheels is avoided using a maximum angular momentum torque distribution. A novel braking curve is designed here and a quaternion-based sliding mode control law is presented to track the braking curve strictly.
Here, three reaction wheels are used instead of four as in [26] and the maximum angular momentum distribution is not used here. Thus frequent unloading of the reaction wheels is probably needed. Based on the maximum allowable torque output and the angular momentum output of the reaction wheels an ideal braking curve from [26] can be calculated in real-time and tracked with a sliding-mode control law.
The sliding-mode controller from [26] is

$$\mathbf{L} = -\mathbf{T}_{emax} sgn(\omega) - kJ\mathbf{e} \tag{41}$$

where k is a scalar gain constant, $\mathbf{L}$ is the reaction wheel control torque command, $\mathbf{T}_{emax}$ is the maximum value of the external disturbance torques, J is the inertia of the satellite and $\mathbf{e}$ is the error between the real angular velocity and the ideal angular velocity:

$$\mathbf{e} = \omega - \omega_{ideal} \tag{42}$$

The ideal braking curve is given by

$$\omega_{ideal} = \begin{cases} \frac{-f_0}{J_{Tmax} r_h}\mathbf{q}_v, & \sqrt{\frac{2f_0}{J_{max} r_\tau}} > \frac{f_0}{J_{Tmax} r_h} \\ -\sqrt{\frac{2f_0}{J_{max} r_\tau}}\mathbf{q}_v, & \sqrt{\frac{2f_0}{J_{max} r_\tau}} \leqslant \frac{f_0}{J_{Tmax}} \end{cases}$$

where f_0 is a scalar constant, $\mathbf{q}_v$ is the vector part of the quaternion error, J_{Tmax} is the maximum value of the satellite's total moment of inertia, J_{max} is the maximum value of the satellites principle inertia.
When the attitude is far from the target, the satellite will rotate along its eigenaxis with the maximum velocity, which is determined by the maximum angular momentum and the mission requirements. When the attitude is near its target, the satellite decelerates to reach stability with the maximum deceleration which is determined by the maximum output torque. The total inertia of the satellite is given by

$$J_T = J + J_w(A_{rw}A_{rw}^T) \tag{43}$$

where J_w is the moment of inertia of each reaction wheel, J is the principal inertia of the satellite, and A_{rw} is the installation matrix of the reaction

wheels, here a 3x3 identity matrix for three orthogonal reaction wheels. The ratio r_h is defined by

$$r_h = \frac{\|\mathbf{H}\|_2}{J_{Tmax}\omega_{max}} \tag{44}$$

where $\mathbf{H}$ is the angular momentum of the control system and ω_{max} is the maximum allowable angular velocity of the satellite. The ration r_τ is given by

$$r_\tau = \frac{\|\mathbf{U_c}\|}{T_{max}} \tag{45}$$

where $\mathbf{U_c}$ is the command control torque and T_{max} is the maximum output control torque of the reaction wheels. The control law given by Eq.(13) is proven to be globally asymptotically stable in [26] using Lyapunov's second method.

4.2.10 Simulation of the Sliding Mode Controller: Small Satellite Case

The control law in Eq.(41) was simulated in MATLAB for the Small Satellite case with the parameters: $T_{max} = 0.1$ Nm, $\omega_{max} = 5°/\text{s}$, $h_{max} = 1.5$ Nms, $f_0 = 1$, $k = 0.2$ and $\mathbf{T}_{emax} = 10^{-5}$ Nm.

The quaternion errors are shown in Fig.34. A stability of $0.001°/s$ is reached in 200 s as Fig. 35 and 36 show. In Fig. 37 and 38 it is shown that a ponting accuracy of $0.01°$ is also reached in 200s. Fig. 39 and 40 are showing that the maximum torque limit and the maximum angular momentum limit of the reaction wheels are not exceeded. In Fig. 41 the cost function of the maneuver is shown, and the total cost of the maneuver is 0.1363.

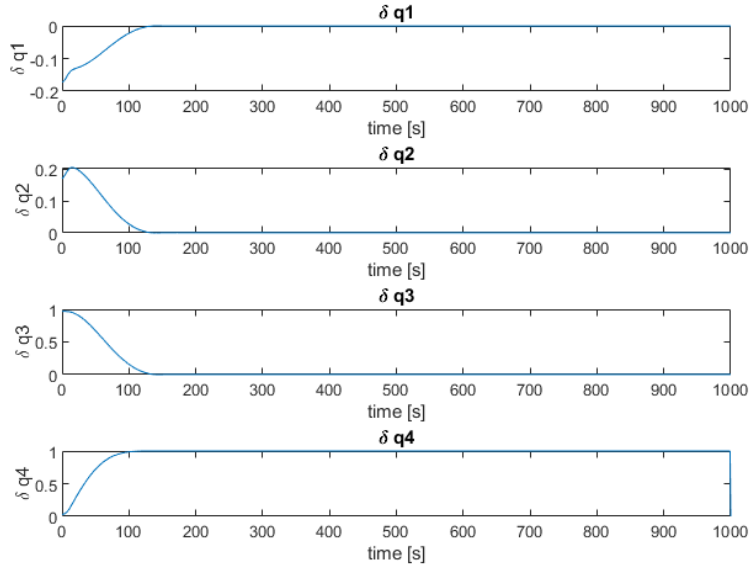

Figure 34: Quaternion errors for the Small Satellite simulation of the sliding mode controller.

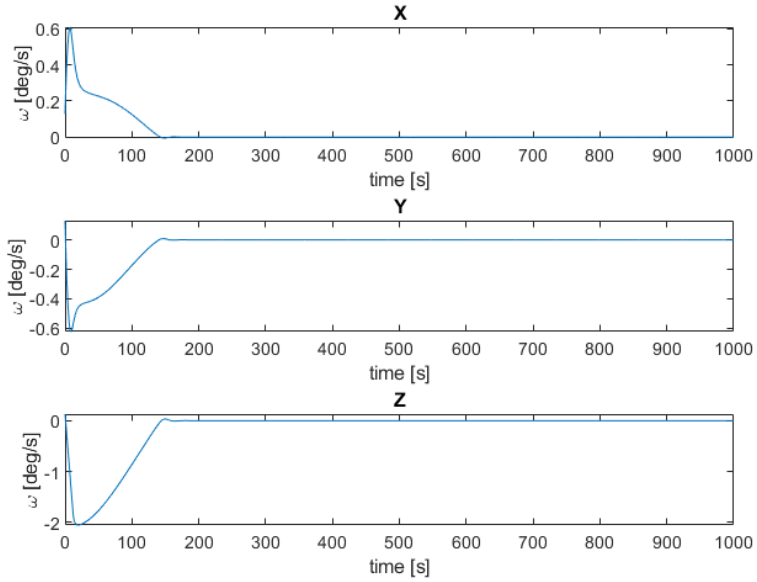

Figure 35: The angular rate of the Small Satellite simulation of the sliding mode controller.

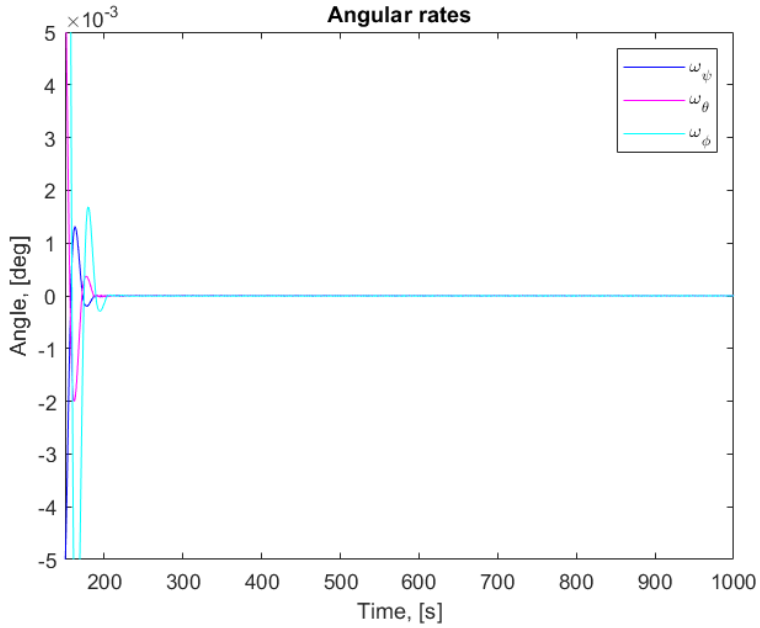

Figure 36: The angular rate of the Small Satellite simulation of the sliding mode controller.

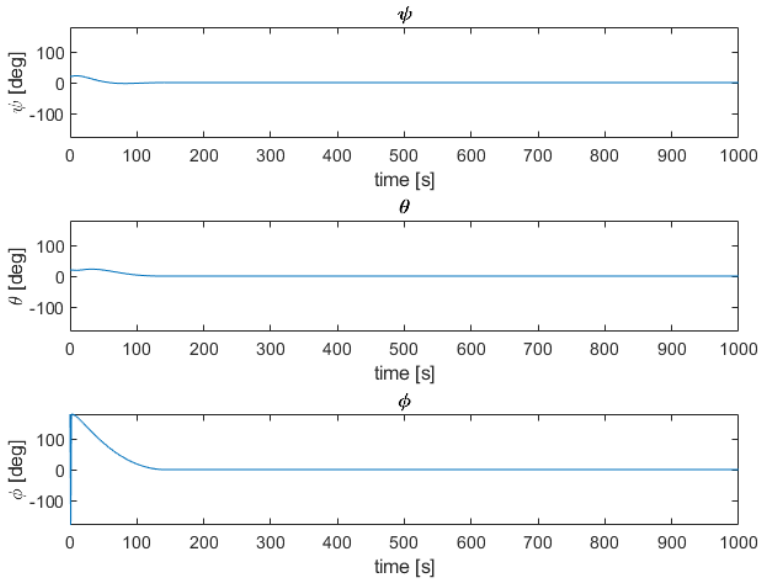

Figure 37: The Euler angles of the Small Satellite simulation of the sliding mode controller.

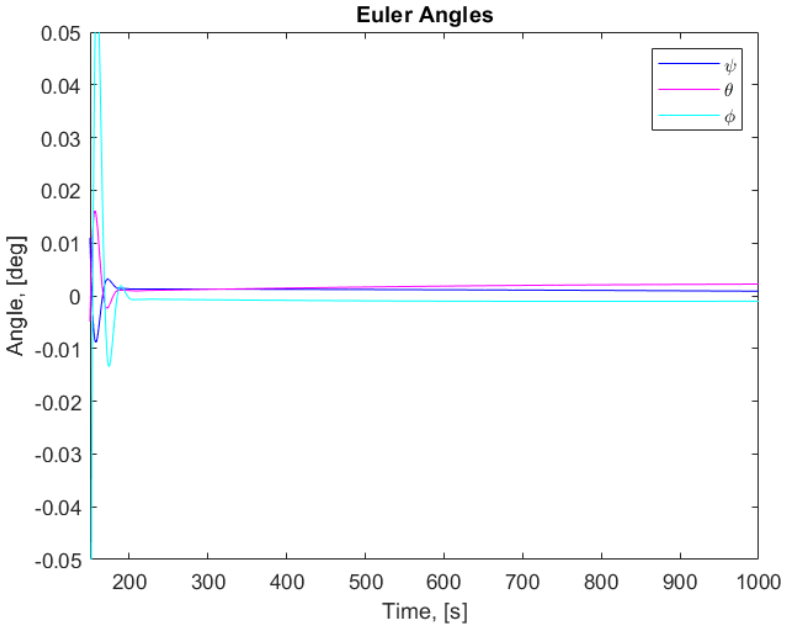

Figure 38: A zoom of the Euler angles of the Small Satellite simulation of the sliding mode controller.

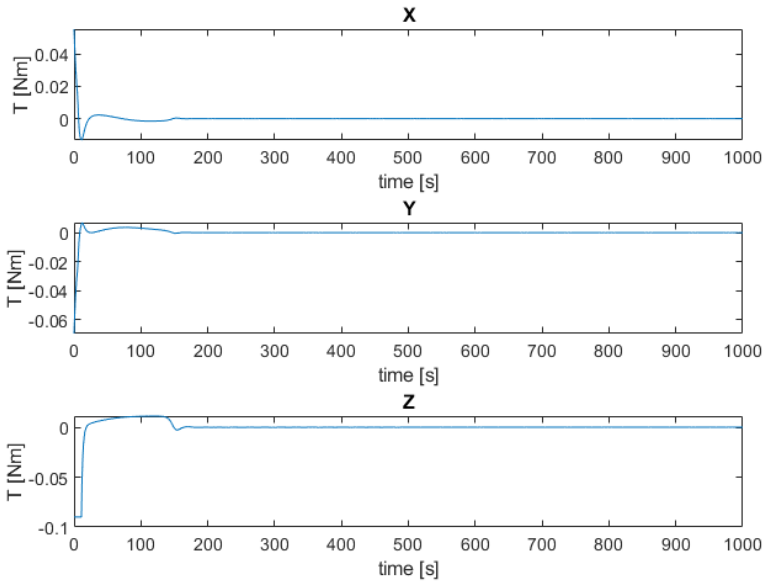

Figure 39: Control torques for the Small Satellite simulation of the sliding mode controller.

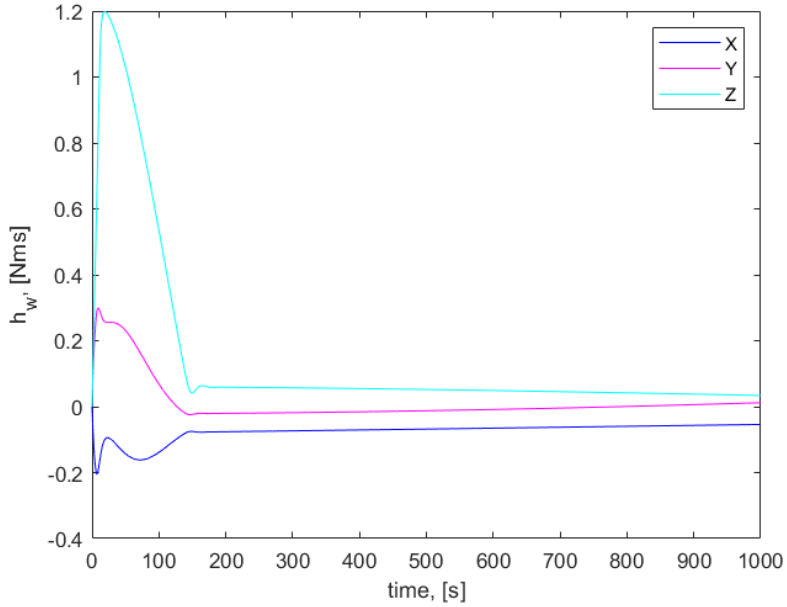

Figure 40: The reaction wheel angular momentum for the Small Satellite simulation of the sliding mode controller.

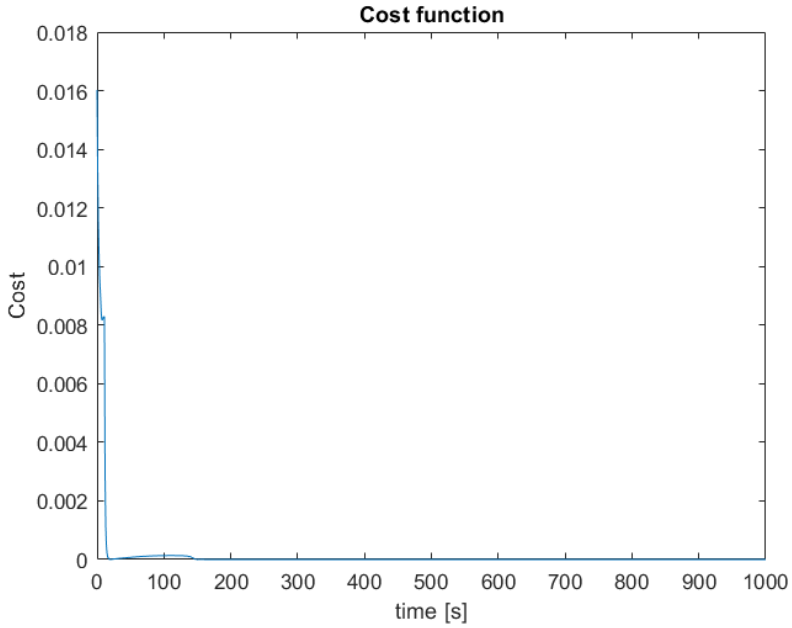

Figure 41: The cost function for the Small Satellite simulation of the sliding mode controller.

Simulation of the Sliding Mode Controller: CubeSat Case

The control law in Eq.(41) was also simulated in MATLAB for the CubeSat case with the parameters: $T_{max} = 1$ mNm, $\omega_{max} = 5°/\text{s}$, $h_{max} = 10.82$ mNms, $f_0 = 0.2830$, $k = 0.1135$ and $\mathbf{T}_{emax} = 10^{-6}$ Nm.

A stability of $0.001°/s$ is never reached as Fig. 43 and 44 show. In Fig. 45 and 46 it is shown that a pointing accuracy of $0.01°$ is never reached due to chattering. Though, as Fig. 47 and 48 show, the torque and momentum limits of the reaction wheels are not exceeded.

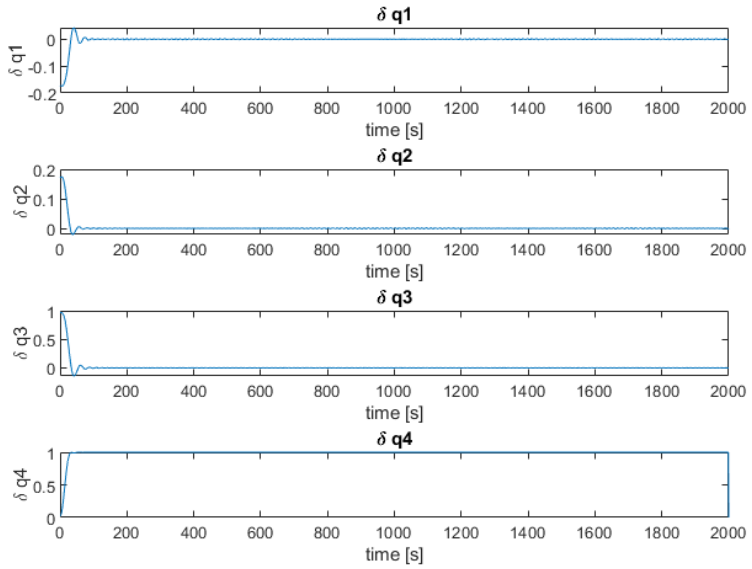

Figure 42: Quaternion errors for the CubeSat simulation of the sliding mode controller.

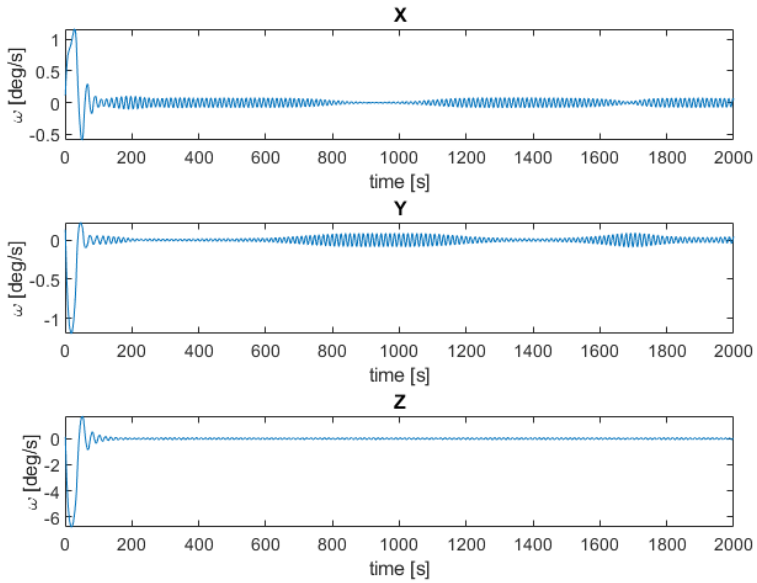

Figure 43: The angular rate of the CubeSat simulation of the sliding mode controller.

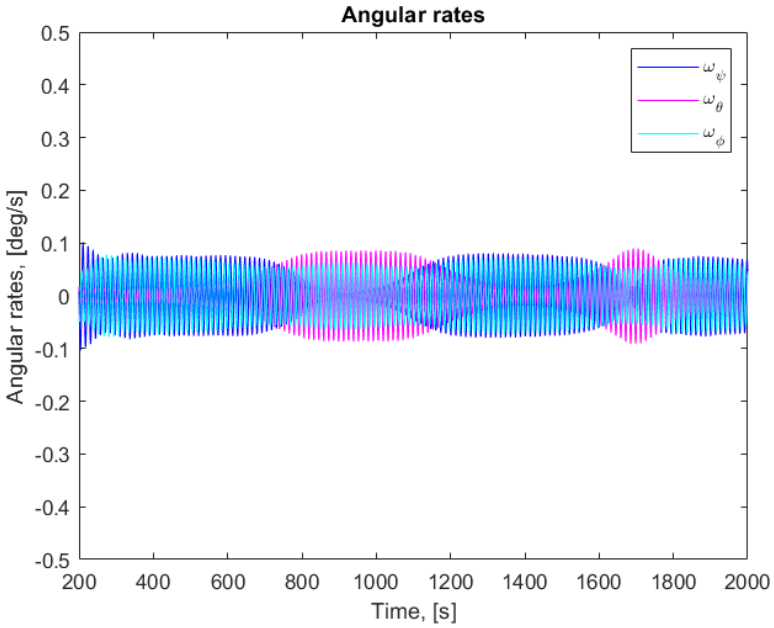

Figure 44: A zoom of the angular rate of the CubeSat simulation of the sliding mode controller.

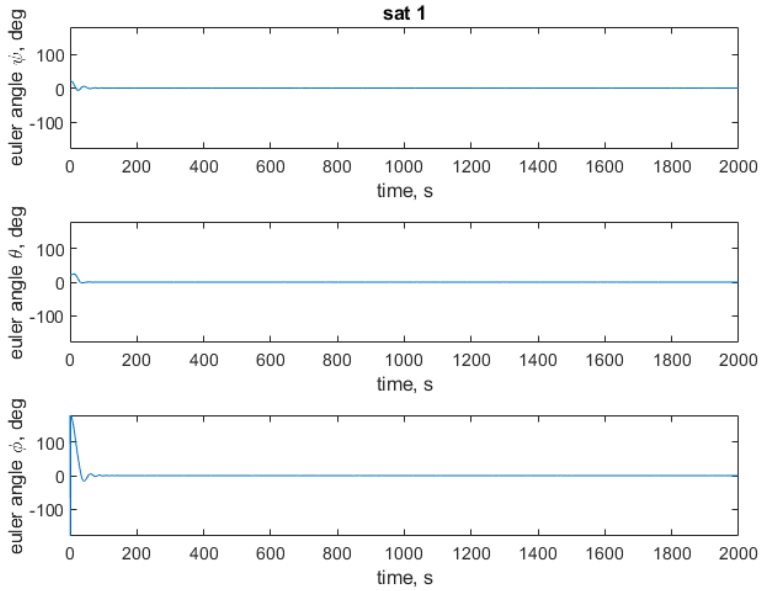

Figure 45: Euler angles for the CubeSat simulation of the sliding mode controller.

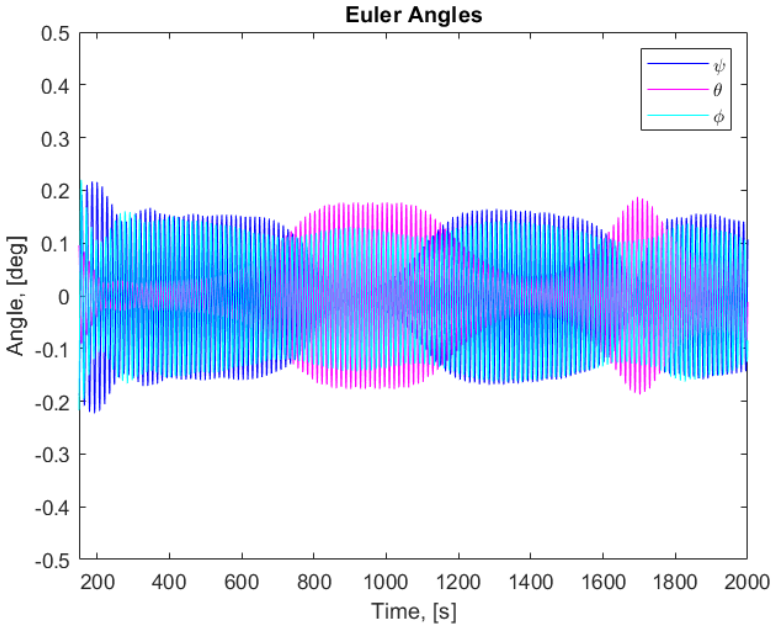

Figure 46: A zoom of the Euler angles for the CubeSat simulation of the sliding mode controller.

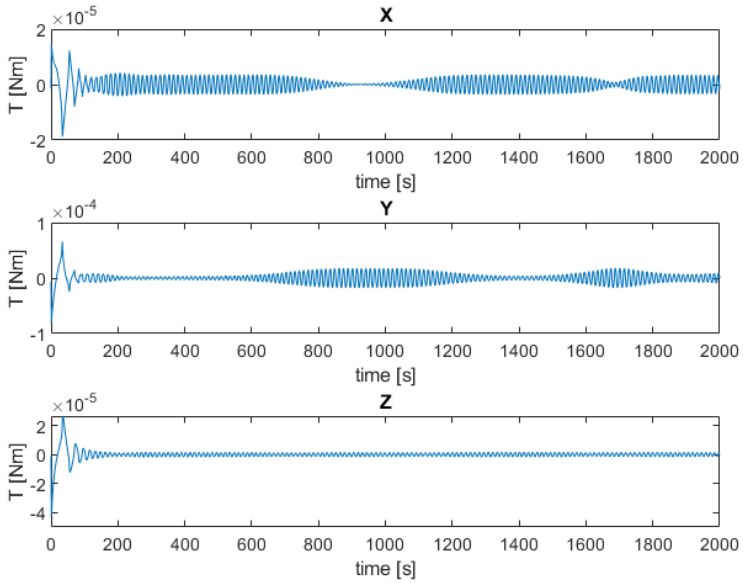

Figure 47: Control torques for the CubeSat simulation of the sliding mode controller.

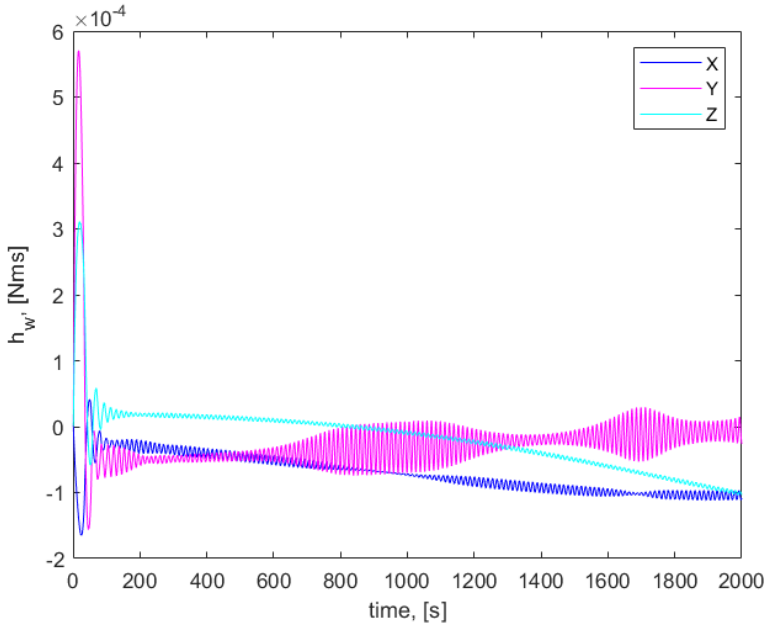

Figure 48: The reaction wheel angular momentum for the CubeSat simulation of the sliding mode controller.

5 Conclusion

A simulation environment for testing detumbling and nadir-pointing algorithms in low earth orbit was successfully built in MATLAB. The model includes an orbit and attitude propagator and takes environmental disturbances such as aerodynamic pressure, the gravity gradient and the solar pressure. The model also includes a model of Earths magnetic field. Though there are two significant limitations of this model:

- The orbit propagator does not take the effects of Earth's oblateness into account, this can result in notable errors accumulating.
- When modeling the disturbance torques, the change of attitude is not considered.

5.1 Detumbling

The b-dot controller was implemented in C, and successfully tested in the simulation environment. The implementation of the gateway between C and MATLAB, took some effort to develop as variables were to be sent, and put to the MATLAB workspace from the C-functions. However, writing the controller algorithms in C facilitates for future hardware-in-the-loop tests.

When a b-dot controller operates in real time on an OBC, a time stamp from the CPU for every measurement of the magnetic field is stored, and these these time stamps are used to calculate the time derivative of the magnetic field. Since the simulation is not executed in real time, another solution must be used here. The time step of the simulation, defined in set_int_param.m, is used to calculate the time derivative of the magnetic field.

5.2 Pointing

The pointing-controllers were meant to be implemented in C, but due to a limited time-span of this project it was decided that the pointing controllers should be implemented in MATLAB instead.
Three time-efficient nadir-pointing algorithms were chosen and tested for two satellites of different sizes in the simulation environment. Comparing the pointing algorithms for the small satellite case, by looking at the angular rates in Fig. 8, 21 and 44, it is clear that the sliding mode controller is the quickest one to achieve a stability of $0.001°/s$. Though, observing the Euler angles in Fig. 10, 23 and 46, only the sliding mode controller was capable of obtaining a pointing accuracy of $0.01°$. The second quaternion feedback controller could at best achieve an accuracy of $0.04°$ in 10 min and third

quaternion feedback controller could reach an accuracy of 0.03° in the same time. The sliding mode controller is here the quickest controller to perform a pointing maneuver for a small satellite, and also the best one when it comes to both stability and pointing accuracy. Comparing the control effort for the three different maneuver, the sliding mode controller requires the most control effort. However, the torque and angular momentum required using the sliding mode controller is still within the limits of the reaction wheels.

For the CubeSat case, using the second quaternion feedback controller, in Fig. 14, it is shown that a stability of $0.001°/s$ is reached in 130 s. Fig. 28 shows that the same stability is reached in 200 s using the third quaternion feedback controller, and Fig. 44 shows that the sliding mode controller never reaches stability. Looking at the Euler angles in Fig. 16 and 30, the quaternion feedback controllers manages to reach the accuracy of 0.01° within 130 s and 200 s respectively, but Fig. 46 shows that the sliding mode controller never reaches such accuracy due to chattering. Comparing the control effort of the quaternion feedback controllers, the second quaternion feedback controller requires less effort than the third quaternion feedback controller. Even though the sliding mode controller was the best one when simulating a small satellite, it did not work properly for a simulation of a 2u CubeSat. The second quaternion feedback controller was the best one looking at maneuver time, stability and cost.

References

[1] Bo Ørsted Andresen, Claus Grøn, Rasmus Hviid Knudsen, Claus Nielsen, Kresten Kjær Sørensen, and Dan Taagaard. *Attitude Control System for AAUSAT-II.* Technical Report,Institute of Electronic Systems, Aalborg University, 2005.

[2] Members of the Technical Staff Attitude Systems Operation Computer Sciences Corporation Ed. by James R. Wertz. *Spacecraft Attitude Determination and Control*, volume 73. Riedel Publishing Company, 1978.

[3] MATLAB. wrldmagm, https://se.mathworks.com/help/aerotbx/ug/wrldmagm.html, 2015.

[4] Marcel J. Sidi. *Spacecrafft Dynamics and Control-A Practical Engineering Approach, 1997.* Cambridge University Press.

[5] Jorge Pomares, Leonard Felicetti, Javier Pérez, and M.Reza Emami. Concurrent image-based visual servoing with adaptive zooming for non-cooperative rendezvous maneuvers. *Advances In Space Research*, 2017.

[6] Antonie Pignède. Detumbling of the ntnu test satellite, 2014.

[7] F. Landis Markley and John L. Crassidis. *Fundamentals of Spacecraft Attitude Determination and Control.* Microcosm Press and Springer, 2014.

[8] Sandra L. Scrivener and Roger C. Thompson. Survey of time-optimal attitude maneuvers. *Journal of Guidance, Control, and Dynamics*, Vol.17(No. 2), 1994.

[9] J.L. Junkins and S.R. Vadali. Optimal open-loop and stable feedback control of rigid spacecraft attitude maneuvers. *Journal of the Astronautical Sciences*, Vol.32(No.2), 1984.

[10] E. Hofer. Time-optimal semi-active attitude control for the pitch motion of the satellite. *Proceedings of the 22nd COngress of the International Astronautical Federation*, 1971.

[11] K. Pande, M. Davies, and V. Modi. Time-optimal pitch control of satellites using solar radiation pressure. *Journal of Spacecraft and Rockets*, Vol.11(No.8), 1974.

[12] W. Garrard. A method for sub-optimal stabilization of spacecraft angular velocity. *International Journal of Control*, Vol.8(No.3), 1968.

[13] J. Kranton. Minimum-time attitude maneuvers with control moment gyroscopes. *AIAA Journal*, Vol.8(No.8), 1970.

[14] E. Barbieri, S. Yuokvich, and U. Ozguner. Control of multiple-mirror/flexible-structures in slew maneuvers. *Proceedings of the AIAA Guodance, Navigation and Control Conference*, Vol.1, 1987.

[15] K. Benninghof and R.L. Boucher. An investigation of the time required for finite dimensional control of a flexible structure. *Journal of Guidance, Control and Dynamics*, Vol.2(No.6), 1989.

[16] Leonardo Mazzini. *Flexible Spacecraft Dynamics, Control and Guidance*. Springer, 2016.

[17] J.R. Etter. A solution of the time optimal euler rotation problem. *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 1989.

[18] K.D Bilimoria and B. Wie. Minimum-time large-angle reorientation of a rigid spacecraft. *AIAA Paper*, 1993.

[19] Z. Tan, P.M. Banium, and F. Li. Minimun-time large angle slew of an orbiting flexible shallow spherical shell. *Advances in the Astronautical Sciences*, Vol.75(Part 1), 1991.

[20] Bong Wie. *Space Vehicle Dynamics and Control*. American Institute of Aeronautics and Astronautics, Inc., 2008.

[21] Jean-Jaques E. Slotine and Weiping Li. *Applied Nonlinear Control*. Prentice Hall, Englewood Cliffs, 1991.

[22] J.L. Crassidis, S.R. Vadali, and F.L. Markley. Optimal variable-structure control tracking of spacecraft maneuvers. *Journal of Guidance, Control and Dynamics*, 2000.

[23] Ho-Nien Shou and Ying-Wen Jan. Nonlinear h_∞ for spacecraft control. *IFAC*, 2005.

[24] S.T. Venkataraman and S. Gulati. Control of nonlinear systems using terminal sliding modes. *American Control Conference*, vol.115(no.3), 2009.

[25] Hai-Tao Chen, Shen-Min Song, and Zhi-Bin Zhu. Robust finite-time attitude tracking control of rigid spacecraft under actuator saturation. *International Journal of Control, Automation and Systems*, 2018.

[26] Xibin Cao, Chengfei Yue, Ming Liu, and Baolin. Wu. Time efficient spacecraft maneuver using constrained torque distribution. *Acta Astronautica*, 2016.

[27] California Polytechnic State University. Cubesat design specification, 2004.

[28] David T. Gerhardt and Scott E. Palo. Passive magnetic attitude control for cubesat spacecraft. In *24th Annual/USU Conference on Small Satellites*. University of Colorado, Boulder.

[29] Blue Canyon Technologies. Microwheel rwp015, http://bluecanyontech.com/microwheel/, 2018.

[30] CubeSpace. Cubewheel medium, https://cubespace.co.za/cubewheel/, 2018.

[31] Blue Canyon Technologies. Reaction wheel rwp500, http://bluecanyontech.com/rwp500/, 2018.

[32] Blue Canyon Technologies. Reaction wheel rw1, http://bluecanyontech.com/rw1/, 2018.

Appendix A: Quaternion & math

The cross product matrix $[\omega\times]$ is defined below

$$[\omega\times] = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{A.1}$$

The quaternion cross product is defined as [7]

$$\overline{\mathbf{q}} \otimes \mathbf{q} = \begin{bmatrix} q_4\overline{\mathbf{q}}_{1:3} + \overline{q_4}\mathbf{q}_{1:3} - \overline{\mathbf{q}}_{1:3} \times \mathbf{q}_{1:3} \\ \overline{q_4}q_4 - \overline{\mathbf{q}}_{1:3} \cdot \mathbf{q}_{1:3} \end{bmatrix} \tag{A.2}$$

Apendix B: Stability Proofs

Stability proof of the second quaternion feedback controller

For the control law in Eq. (37) Euler's equation of motion is

$$\dot{\omega} = -J^{-1}\big([\omega\times]J\omega + K_p sign(\delta q_4)\delta\mathbf{q}_{1:3} + K_d\omega\big) \tag{B.1}$$

The time derivatives of the quaternion error is given in [7]:

$$\delta\dot{\mathbf{q}}_{1:3} = \frac{1}{2}[\dot{\mathbf{q}}_{1:3}\times]\omega + \frac{1}{2}\delta q_4\omega \tag{B.2}$$

$$\delta\dot{q}_4 = -\frac{1}{2}\mathbf{q}_{1:3}^T\omega \tag{B.3}$$

following candidate Lyaponov function is defined

$$V = \frac{1}{2}K_p\Big[\delta\mathbf{q}_{1:3}^T\delta\mathbf{q}_{1:3} + (1-\delta q_4)^2\Big] + \frac{1}{4}\omega^T J\omega \geqslant 0 \tag{B.4}$$

The scalar gain K_p must satisfy

$$K_p \geqslant -\frac{2\omega^T J\omega}{2[\mathbf{q}_{1:3}^T\delta\mathbf{q}_{1:3} + (1-\delta q_4)^2]} \tag{B.5}$$

taking the time derivative of V yields

$$\dot{V} = K_p\Big[\delta\mathbf{q}_{1:3}^T\delta\dot{\mathbf{q}}_{1:3} - (1-\delta q_4)\delta\dot{q}_4\Big] + \frac{1}{2}\omega^T J\dot{\omega} \tag{B.6}$$

Substituting Eqs. (18),(19) and (20) into (22) gives

$$\dot{V} = \frac{1}{2}K_p\delta\mathbf{q}_{1:3}^T\omega - \frac{1}{2}sign(\delta q_4)K_p\mathbf{q}_{1:3}^T\omega - \frac{1}{2}K_d\omega^T J\omega \leqslant 0 \tag{B.7}$$

The signum function gives three different cases:

- $\delta q_4 > 0$: Eq.(23) then becomes

$$\dot{V} = -\frac{1}{2}K_d\omega^T\omega \leqslant 0 \tag{B.8}$$

- $\delta q_4 = 0$: Eq.(23) becomes

$$\dot{V} = \frac{1}{2}K_p\delta\mathbf{q}_{1:3}^T\omega - \frac{1}{2}K_d\omega^T\omega \leqslant 0 \tag{B.9}$$

  where the gain K_d must satisfy

$$K_d \geqslant \frac{K_p\mathbf{q}_{1:3}^T\omega}{\omega^T\omega} \tag{B.10}$$

- $\delta q_4 < 0$: Eq.(23) becomes

$$\dot{V} = K_p \delta \mathbf{q}_{1:3}^T \omega - \frac{1}{2} K_d \omega^T \omega \leqslant 0 \tag{B.11}$$

where K_d must satisfy

$$K_d \geqslant \frac{2 K_p \mathbf{q}_{1:3}^T \omega}{\omega^T \omega} \tag{B.12}$$

The closed loop system of Eqs.(25) to (28) is stable when $\omega = 0$ and $\delta \mathbf{q}_{1:3}$ can be anything. Global asymptotic stability is proven using LaSalle's theorem . It must be checked that the system cannot remain in a state where $\dot{V} = 0$ while $\delta \mathbf{q}_{1:3} \neq 0$. Eqs.(24) to (26) show that $\lim_{t\to\infty} \omega = \mathbf{0}$. The closed loop dynamics in Eq.(18) shows that the asymptotic condition only holds if $\lim_{t\to\infty} \mathbf{q}_{1:3} = \mathbf{0}$ an the conditions for K_p and K_d are satisfied.[7].

Stability proof of the third quaternion feedback controller

For the control law in Eq.(8), Euler's equation of motion becomes

$$\dot{\omega} = -J^{-1}\big([\omega\times] J\omega + K_p sign(\delta q_4)\delta \mathbf{q}_{1:3} + K_d(1 \pm \delta \mathbf{q}_{1:3}^T \delta \mathbf{q}_{1:3})\omega \tag{B.13}$$

The candidate Lyapunov function is defined in Eq.(21). The condition for K_p in Eq.(22) must be satisfied here also. Substituting Eqs. (19),(20) and (30) into the time derivatide of the Lyapunov function in Eq. (23) gives

$$\dot{V} = \frac{1}{2} K_p \delta \mathbf{q}_{1:3}^T \omega - \frac{1}{2} sign(\delta q_4) K_p \mathbf{q}_{1:3}^T \omega - \frac{1}{2} K_d \underbrace{(1 \pm \delta \mathbf{q}_{1:3}^T \delta \mathbf{q}_{1:3})}_{\text{a}} \omega^T \omega \leqslant 0 \tag{B.14}$$

For Lyapunov stability, the term a in Eq.(31) must be greater than or equal to zero, we have two cases for a:

- $a = (1 - \delta \mathbf{q}_{1:3}^T \delta \mathbf{q}_{1:3}) = \delta q_4^2 \geqslant 0$, due to quaternion unity definition.
- $a = (1 + \delta \mathbf{q}_{1:3}^T \delta \mathbf{q}_{1:3}) \geqslant 0$

Thus $a \geqslant 0$.
The signum function gives three different cases:

- $\delta q_4 > 0$: Eq.(23) then becomes

$$\dot{V} = -\frac{1}{2} K_d (1 \pm \delta \mathbf{q}_{1:3}^T \delta \mathbf{q}_{1:3}) \omega^T \omega \leqslant 0 \tag{B.15}$$

- $\delta q_4 = 0$: Eq.(23) becomes

$$\dot{V} = \frac{1}{2} K_p \delta \mathbf{q}_{1:3}^T \omega - \frac{1}{2} K_d (1 \pm \delta \mathbf{q}_{1:3}^T \delta \mathbf{q}_{1:3}) \omega^T \omega \leqslant 0 \tag{B.16}$$

where the gain K_d must satisfy

$$K_d \geqslant \frac{K_p \mathbf{q}_{1:3}^T \omega}{(1 \pm \delta \mathbf{q}_{1:3}^T \delta \mathbf{q}_{1:3}) \omega^T \omega} \tag{B.17}$$

- $\delta q_4 < 0$: Eq.(23) becomes

$$\dot{V} = K_p \delta \mathbf{q}_{1:3}^T \omega - \frac{1}{2} K_d (1 \pm \delta \mathbf{q}_{1:3}^T \delta \mathbf{q}_{1:3}) \omega^T \omega \leqslant 0 \tag{B.18}$$

  where K_d must satisfy

$$K_d \geqslant \frac{2 K_p \mathbf{q}_{1:3}^T \omega}{(1 \pm \delta \mathbf{q}_{1:3}^T \delta \mathbf{q}_{1:3}) \omega^T \omega} \tag{B.19}$$

Here global asymptotic stability is also proven using LaSalle's theorem. The system is stable when $\omega = 0$ and $\delta \mathbf{q}_{1:3}$ can be anything. It must be checked that the system cannot remain in a state where $\dot{V} = 0$ while $\delta \mathbf{q}_{1:3} \neq 0$. Eqs.(32) to (35) show that $\lim_{t \to \infty} \omega = \mathbf{0}$ when the conditions for K_p and K_d are satisfied.. The closed loop dynamics in Eq.(30) shows that the asymptotic condition only holds if $\lim_{t \to \infty} \mathbf{q}_{1:3} = \mathbf{0}$ [7].